Polyedges, Polyominoes and "The Digit" Game

Diplomarbeit
vorgelegt von
Alexander Malkis

nach einem Thema von Prof. Dr. Raimund Seidel
im Fachbereich 14, Informatik, der Universität des Saarlandes

Hiermit erkläre ich an Eides Statt, dass ich diese Arbeit selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Ferner habe ich die Arbeit noch keinem anderen Prüfungsamt vorgelegt.

Saarbrücken, im August 2003.

Acknowledgements

I'm indebted to many people who helped me to write this thesis.

Many thanks to my family and all of my friends in any case. Special thanks to Elina Koreshkovskaya for her support and understanding as well as for her hopeless attempts to improve my English.

Many thanks to my thesis advisor Prof. Dr. Raimund Seidel, who gave me an interesting topic, helped me with his advice and proof-read this "opus".

Many thanks to the members of the working group at his chair, whose pleasant working atmosphere I enjoyed. They provided computational resources, a working place, proved to be good dialog partners and often gave me a useful piece of advice. Great thanks to Dierk Johannes, our system administrator, and Ulrike Lemberg, our secretary.

I also have to thank a lot of people from the Usenet groups and "polyforms" discussion forum who helped me with lots of tiny questions when I got stuck.

Contents

Introduction

In this work, we'll dive into a very small part of the world of games and puzzles.

Polyominoes were discovered relatively late; puzzlers were talking about "fives" and "sixes" in the 1920s. Until 1953, such polyominoes were only known in the world of puzzlers. The task was to tile an $m \times n$ square with a single given polyomino. Due to Solomon Golomb, a more general definition of polyominoes has appeared. In 1960 Harary formulated the cell growth problem as a graph enumeration problem. This work initiated a big branch of frustrating attempts to compute the number of polyominoes with n cells. Eden (1961) and Klarner (1967) proved the main asymptotic results, that have remained in the history of polyominoes forever. As soon as the mathematicians understood its hardness, they switched over to tackling similar problems - and succeeded.

Plain enumeration was characterised by the development of different algorithms for it. Each new algorithm had a better asymptotic behaviour and was able to push the enumeration border a little further. The "intuitive" method to generate all polyominoes, called "extension" method, fills each computer's memory inevitably fast. Redelmeier (1981) applied the "rooted" method to get the run-time, proportional to the number of polyominoes; it uses negligibly few space. The run-time is still exponential in the degree. A. R. Conway (1994) applied the finite-lattice method associated with a transfer-matrix algorithm to cut the exponent in one half - and extended the enumeration up to degree 25. Until now, an optimised parallel solution of Jensen (2004) holds the record of counting all the polyominoes up to the degree of 56.

Another branch tried to get exact solutions for special classes of polyominoes. For some special classes generating functions have been discovered.

Klarner solved the task for the so-called row-convex polyominoes in 1967, and Dhar for the so-called directed polyominoes around 1982. Furthermore, enumeration by perimeter and connected problems have been studied.

Still others tried to improve the asymptotic results for polyominoes. We have to mention Klarner (1967), Klarner and Rivest (1972-73), Rands and Welsh (1981). Rands and Welsh were able to give a method to compute better asymptotic results; this method uses available enumeration data. Researchers applied their new enumeration data to make the asymptotic results more precise.

Polyominoes received attention in statistical physics, especially in the percolation theory. Nevertheless they are also captivating from a pure combinatorial point of view.

Polyedges - or graphs on the square lattice - we studied not as extensively as polyominoes, although some special cases such as self-avoiding walks, lattice trees and lattice graphs have been studied. Enumeration problem has only been touched, asymptotic results seem to be widely unknown, generating function could not be expressed either. Currently (2000-2004), Andrew Rechnitzer and Tony Guttmann are working on similar problems.

In this paper, we provide a mathematical framework for such graphs, formulate some problems and try to solve them. We explore the relations between such graphs and polyominoes and apply the methods from polyomino theory to our problem.

At the same time, we investigate a new board game, based on such graphs, formulate problems connected with this game, solve some of them and present an implementation of this game.

If a reader encounters a not introduced concept or an unknown symbol, he is directed to the appendix part B.

Chapter 1

Counting polyforms.

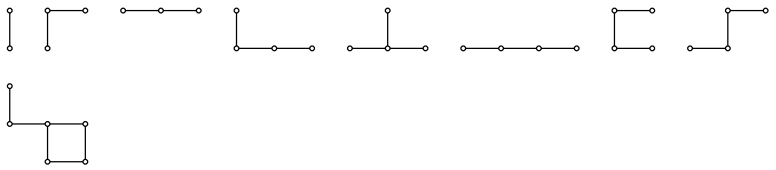
In this chapter, we'll work with our main object of interest, namely polyedges. We'll try to count them, explore their asymptotic behaviour, try to find some "well-behaved" subclasses of polyedges and look at their relations to some other "polyforms" such as polyominoes and polyplets.

1.1 Basic definitions.

Let $k \in \mathbb{N}, k \geq 2$ be fixed throughout this section. Consider infinite regular graphs (V, E), embedded into $\mathbb{R}^k$, i.e. with $V \subset \mathbb{R}^k$ and $E \subset \mathfrak{P}_2(V)$ (where $\mathfrak{P}_2(V)$ is the set of all 2-element subsets of V). The simplest example and at the same time our main point of interest is the *k-dimensional square lattice* $C_k := (\mathbb{Z}^k, \{\{a, b\} \in \mathfrak{P}_2(\mathbb{Z}^k) \mid |a - b| = 1\})$. Call its isometry group L_k. We can embed some undirected graphs into C_k. Consider the set $PR_k(m)$ of all possible embeddings of connected graphs with m edges in the k-dimensional square lattice and call its elements *k-dimensional m-edges*. Omitting the restriction on the number of edges gives PR_k, the set of *k-dimensional polyedges*. Each $PR_k(m)$ is infinite, since we can move an embedding in any direction infinitely often and get another embedding each time.

We regard two polyedges G and H as equivalent up to translation, if adding a constant integer vector to G (i.e. adding it to all node labels, changing the edge set appropriately) gives H. There are only finitely many equivalence classes $PS_k(m)$ in each $PR_k(m)$, we call them *fixed* k-dimensional m-edges. Similarly, we regard G and H as *k-dimensionally same*, if an operation from L_k transforms G into H. This also gives a finite set $PT_k(m)$ of

equivalence classes, we call them *free* k-dimensional m-edges. The elements of $PS_k := \bigcup_m PS_k(m)$ are called *fixed k-dimensional polyedges* and the elements of $PT_k := \bigcup_m PT_k(m)$ are called *free k-dimensional polyedges*.
Here the representatives of all free planar 1-, 2-, 3-edges and a single planar 6-edge are depicted:



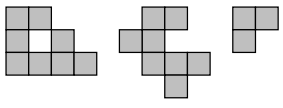
Now we define an important subset of polyedges. Consider the set $R_k \subset PR_k$ of all those k-dimensional polyedges, for which the following holds: whenever two nodes of a polyedge have the distance of a unit on the lattice (for the standard square lattice, this is equivalent to the fact that the Euclidean distance is one), they are connected; such polyedges are called *polyominoes*. Remark the difference: a k-dimensional *n-omino* has n nodes (not edges), their set is denoted by $R_k(n)$. Then make the same construction: define the equivalence relation "equality up to an additive constant from $\mathbb{Z}^k$ " on $R_k(n)$ and get the set of equivalence classes $S_k(n)$, the *fixed k-dimensional n-ominoes*; then we regard the set $T_k(n)$ of orbits over $R_k(n)$ under the operation of the isometry group L_k and call them *free k-dimensional n-ominoes*. Dropping the restriction on the number of nodes, we simply get *fixed* and *free k-dimensional polyominoes*, respectively, for which we write T_k and S_k. In the context of polyominoes, the term *cell* is equivalent to our term node, i.e. we can think of a cell as of a bordered filled square with the side length 1, its centre at the corresponding node and its sides parallel to the axes; if we draw cells for some polyomino, we detect that:

(a) There are finitely many cells;

(b) Each two cells have either exactly one common side, or one common corner, or don't overlap;

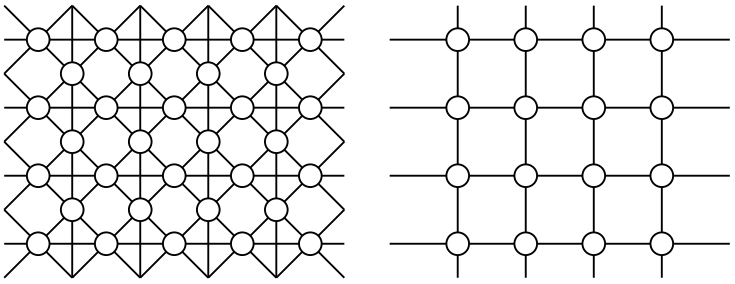
(c) The union of all cells without their corners is path-connected.

This represents the alternative definition of a (2-dimensional) polyomino. Here is an example of two 8-ominoes and a 3-omino:



Free and fixed polyominoes are defined by taking equivalence classes that arise from the operation of the isometry group of the lattice or of its subgroup of all translations, respectively.

Notice that polyedges can be also regarded as polyominoes if we change the underlying lattice from the standard square lattice (viewed as a graph) to its line graph (i.e. new nodes are former edges and a new edge is present between two new nodes if the underlying edges of these new nodes are directly connected). The right picture shows a part of the 2-dimensional square lattice, viewed as a graph with nodes and edges and the left picture shows its line graph.



However, this line graph of the lattice is no more planar and seems to be more complex than the underlying lattice.

In the following, we omit the prefix "k-dimensional", as far as k is clear from the context [1] .

The following table sums up the new symbols and introduces the corresponding numbers:

concept	"as-is"	fixed	free
k-dimensional m-edges	$PR_k(m)$	$PS_k(m)$	$PT_k(m)$
number of all k-dimensional m-edges		$ps_k(m)$	$pt_k(m)$
k-dimensional polyedges	PR_k	PS_k	PT_k
k-dimensional n-ominoes	$R_k(n)$	$S_k(n)$	$T_k(n)$
number of all k-dimensional n-ominoes		$s_k(n)$	$t_k(n)$
k-dimensional polyominoes	R_k	S_k	T_k

[1] Be careful, in present literature the term "polyX" usually refers to the planar, 2-dimensional version of X.

1.2 Known results.

At the time this manuscript is being written, the problem of enumerating 2-dimensional polyominoes, i.e. of giving the number of free (or fixed) planar polyominoes with n cells, remains unsolved, and generalisations of theory to arbitrary dimensions were not considered extensively (although 3-dimensional polyominoes have been counted for small number of cells). One of the strongest theoretical result given by Klarner in [6] proves the existence of the positive growth constant

$$c_o := \lim_{n\to\infty} t_2(n)^{1/n}$$

and thus a non-zero convergence radius of the generating function of 2-dimensional polyominoes. The practical results improve very fast, Jensen in [4] enumerated fixed planar polyominoes up to order 56. His estimation of the real value is $c_o \approx 4.0625696(5)$. It is known that $3.927378 \leq c_o \leq 4.649551$. Here we'll give a few known series of numbers for planar polyominoes and planar polyedges (see [8], [9], [10] and [11]); the numbers for fixed n-edges don't seem to be widely known or published before this work (although most probably counted), the algorithms were run on this problem by the author and independently by other groups at the same time:

	fixed n-ominoes	free n-ominoes	fixed n-edges	free n-edges
n \ c	$\approx 4.0625696(5)$	$\approx 4.0625696(5)$	≈ 5.20	≈ 5.20
1	1	1	2	1
2	2	1	6	2
3	6	2	22	5
4	19	5	88	16
5	63	12	372	55
6	216	35	1628	222
7	760	108	7312	950
8	2725	369	33466	4265
9	9910	1285	155446	19591
10	36446	4655	730534	91678
11	135268	17073	3466170	434005
12	505861	63600	16576874	2073783

Here c denotes the growth constant. We'll see in 1.3.1 that the distinction between fixed and free polyforms doesn't affect it. An interested reader is advised to read Golomb's article on polyominoes (see [2]). The "polyforms" Yahoo discussion group talks mainly about puzzles and tiling

problems involving polyforms.

On the other hand, polyedges were studied not so deeply: the only result is the enumeration of free planar polyedges up to order 17, given in [8], as well as the widely-assumed claim about the existence of a Klarner's constant for planar polyedges; proofs have never been thoroughly formulated. Attention can also be paid to the paper by Whittington and Soteros (see [16]), where they study *bond animals*, dependent on the number of nodes (and not on the number of edges, as we do here) and achieve similar results. The group around Tony Guttman is trying to characterise the generating function for different kinds of animals.

1.3 Counting polyedges

In this section we study polyedges a bit deeper. First we prove the existence of a Klarner's constant for polyedges of any dimension. Then we'll try to enumerate polyedges as far as possible and use these results to obtain a good estimation of the Klarner's constant for polyedges and a lower bound for the constant.

1.3.1 Theory on polyedges and polyominoes.

Here our aim is to prove the existence of a Klarner's constant for polyedges. In order to do that, we generalise and modify the proofs of Eden and Klarner. In a similar form, we can find this material in any paper which proves the existence of a growth constant for many polyforms. The new thing is the distinct treatment of polyedges in arbitrary dimension number and the treatment of polyedges and polyominoes together in the same context. Also more precise bounds on that constant for polyedges are given.

Remark and Definition 1.3.1. We define some notation for polyforms.

- We say that a k-dimensional polyedge is in *standard position* if
  (a) all its nodes have nonnegative coordinates and
  (b) for each hyperplane $H_i := \{x \in \mathbb{Z}^k \mid x_i = 0\}$ there is at least one node on it.

  Polyedges in standard position form the representative system of fixed polyedges.

- In order to get free k-dimensional polyedges, we notice that the isometry group of a k-dimensional polyedge is at most as big as the isometry group of a k-dimensional cube. The last one is known to be the wreath product $C_2 \wr_k \mathrm{Sym}_k$ of order $2^k k!$. So we have for all $n \in \mathbb{N}$:

$$\frac{1}{2^k k!} ps_k(n) \leq pt_k(n) \leq ps_k(n) \leq 2^k k! \cdot pt_k(n),$$

and since the same holds for polyominoes:

$$\frac{1}{2^k k!} s_k(n) \leq t_k(n) \leq s_k(n) \leq 2^k k! \cdot t_k(n).$$

We need some lemmas, from general mathematics as well as specific to the current situation.

Lemma 1.3.2 (due to Fekete)**.** If $(a_n)_{n\in\mathbb{N}}$ is a sequence of (positive) natural numbers such that $(a_n^{1/n})_{n\in\mathbb{N}}$ is bounded and $\forall n, m \in \mathbb{N} : a_n a_m \leq a_{n+m}$, then $\lim\limits_{n\to\infty} a_n^{1/n}$ exists.

That lemma will give us the finite answer. In order to apply it, we have to prove those two required conditions. To do that, we present some definitions and lemmas working in our special situation.

Remark and Definition 1.3.3. (a) Let $(G, +)$ be a semigroup, and let $F \subset G \ni v$. Set

$$v + F := \{v + f \mid f \in F\}.$$

(b) If we have a direct product of sets $\prod\limits_{i\in I} X_i$, write π_i for the projection map onto the ith component.

Next we reformulate the concatenation argument (which was used for planar polyominoes by Klarner and others) in a purely combinatorial way, that neither has anything to do with any polyforms nor requires any connectedness.

Lemma 1.3.4 (Translation lemma)**.** Let $k \in \mathbb{N}$ and $F, H \subset \mathbb{Z}^k$ both nonempty finite sets. Then

(a) $\exists\, v \in \mathbb{Z}^k : |(v + F) \cap H| = 1$.

(b) $\exists\, z \in \mathbb{Z}^k : (z + F) \cap H = \emptyset \wedge (\exists!\, a \in z + F, b \in H : |a - b| = 1)$

Proof. (a) Connect the lexicographically greatest element of F with the lexicographically smallest element of H.

(b) We determine z by setting $\operatorname{lmax}(z+F)+e_1 = \operatorname{lmin}(H)$ (where $e_1 = (1,0,...,0) \in \mathbb{Z}^k$), i.e. by moving F until $\operatorname{lmax}(z+F)+e_1$ is on the same place as $\operatorname{lmin}(H)$. Let $a \in z+F$ and $b \in H$. Then:

$$a \leq_{lex} \operatorname{lmax}(z+F) \text{ and } b \geq_{lex} \operatorname{lmin}(H) \Rightarrow$$

$$\pi_1(a) \leq \pi_1(\operatorname{lmax}(z+F)) = \pi_1(\operatorname{lmin}(H)) - 1 \leq \pi_1(b) - 1. \quad (*)$$

Let's additionally suppose $|a-b| = 1$. Since a and b are integer vectors, they differ in at most one component. But they already differ in the first component, so they are equal in other components. Thus we have equalities in $(*)$:

$$\pi_1(a) = \pi_1(\operatorname{lmax}(z+F)) = \pi_1(\operatorname{lmin}(H)) - 1 = \pi_1(b) - 1.$$

By the definition of lexicographic order, we get first for $i = 2$:

$$\pi_i(a) \leq \pi_i(\operatorname{lmax}(z+F)) = \pi_i(\operatorname{lmin}(H)) \leq \pi_i(b) \quad (**)$$

From $\pi_i(a) = \pi_i(b)$ we get

$$\pi_i(a) = \pi_i(\operatorname{lmax}(z+F)) = \pi_i(\operatorname{lmin}(H)) = \pi_i(b) \quad (***)$$

If $n = 2$, we are done; otherwise we conclude inductively that $(**)$ and $(***)$ hold for $i = 3, ..., n$. So actually $a = \operatorname{lmax}(z+F), b = \operatorname{lmin}(H)$. □

We need one more definition.

Definition 1.3.5. If we have two k-dimensional polyedges $X = (V, E)$ and $X' = (V', E')$, so that $V \cap V' \neq \emptyset$, let their *union* be the k-dimensional polyedge $(V \cup V', E \cup E')$.

The following result represents the first requirement of Fekete lemma.

Lemma 1.3.6 (Superadditivity[2] of polyedges and -ominoes)**.** Assume $k, m, n \in \mathbb{N}, k \geq 2$. Then

(a) $ps_k(m) \cdot ps_k(n) \leq ps_k(n+m)$;

[2]In fact, not the numbers themselves, but their logarithms are superadditive.

(b) $s_k(m) \cdot s_k(n) \leq s_k(n+m)$.

Proof. (a) It suffices to construct a one-to-one correspondence

$$PS_k(m) \times PS_k(n) \hookrightarrow PS_k(n+m).$$

Let α_k be a map from the previous translation lemma 1.3.4a, i.e. a map which for two given point sets $F, H \subset \mathbb{Z}^k$ yields a $v \in \mathbb{Z}^k$ so that $|(v+F) \cap H| = 1$. Regard the map

$$\alpha_{k,n,m} : \text{ fixed } k\text{-dim } n\text{-edges } \times \text{ fixed } k\text{-dim } m\text{-edges} \to \mathbb{Z}^k,$$

$$(F, H) \mapsto \alpha_{k,n,m}(V(F), V(H)).$$

Let X and Y be representative elements from equivalence classes in $PS_k(n)$ and $PS_k(m)$, respectively. Let $v := \alpha_{k,n,m}(X, Y)$. Since $v + V(X)$ and $V(Y)$ have exactly one node in common, the union of the shifted (by v) polyedge X and Y is connected and has $n+m$ edges. This is the correspondence we were looking for. To see that it is one-to-one, notice that each $(n+m)$-edge R in its range has a unique node, which partitions R in two sub-polyedges: one (lexicographically smaller) with n and the other (lexicographically greater) with m edges.

(b) Analogously. Consider a map from 1.3.4b, which for finite nonempty $F, H \subset \mathbb{Z}^k$ yields a $v \in \mathbb{Z}^k$ so that $(v+F) \cap H = \emptyset$ and there is exactly one pair $(a, b) \in (v+F) \times H$ with $|a - b| = 1$. That means that if X and Y were polyominoes with vertex sets F and H with n and m nodes, respectively, then the graph embedding, which consists of X, shifted by v, of Y, and of the edge $\{a, b\}$ is connected as a graph and thus a polyomino itself with $n+m$ nodes. This correspondence in one-to-one, since each polyomino with $m+n$ nodes in its range has (seen as a graph) a unique edge $\{a, b\}$ which partitions it into two connected subgraphs with n and m nodes, where the first one is lexicographically smaller than the second.

□

Definition 1.3.7. An (n, m)-edge is a polyedge with n nodes and m edges.

The natural question would ask about the relationship between the number of edges and the number of nodes of a polyedge. We give a trivial result, the upper bound may still be improved.

Theorem 1.3.8. For a k-dimensional (n, m)-edge holds: $n - 1 \leq m \leq nk$.

Proof. Since the polyedge (call it X) is a connected graph, we have the first inequality. The second follows from $2m = \sum_{v \in V(X)} \deg v \leq \sum_{v \in V(X)} 2k$. □

The following theorem represents the basic result on polyominoes and polyedges. For planar square and hexagonal polyominoes, it is already known and better bounds for these two planar classes exist. For polyedges, the result is new, although not astonishing.

Theorem 1.3.9 (Polyforms inequalities)**.** Let $k \in \mathbb{N}$, $k \geq 2$. Then:

(a) $\forall\, n \geq 2 : s_k(n) \leq ps_k(n-1)$;

(b) $\forall n \geq 2, m \geq 1$: The number of k-dimensional (n,m)-edges is bounded from above by $\binom{(2k-1)(n-2)+k}{m} = \binom{(2k-1)n-3k+2}{m}$;

(c) $\forall m \geq 1$:

$$ps_k(m) \leq \binom{(2k-1)m-k+1}{m} < (2k-1)\left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}\right)^{m-1};$$

(d) $\forall\, n \geq 2$:

$$s_k(n) \leq \binom{(2k-1)n-3k+2}{n-1} < (2k-1)\left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}\right)^{n-2}.$$

Proof. Before we start proving all inequalities, let's attach to every dimension k some fixed $2k$-cycle $\sigma \in \mathrm{Sym}(\{e_i, -e_i \mid i \in \mathbb{N}_k\})$, which will determine the order of directions.

(a) We construct a one-to-one correspondence $S_k(n) \hookrightarrow PS_k(n-1)$. Given a k-dimensional n-omino, use the lexicographic order on the node set to find the smallest node. Then do a breadth-first search starting with this node. In order to choose the order in which the search takes the successor nodes of a given node v, regard the direction, from which v was first discovered, as a standard unity vector e_i or $-e_i$ for some i. If v is the very first node, assume this direction to be e_1 (there is no node at position $v - e_1$). Then look for the successors of v in the directions $\sigma(e_i), \sigma^2(e_i), ..., \sigma^{2k-1}(e_i)$ and recurse into the actually present successors, which were not discovered yet, in this order.
The corresponding breath-first-search tree (which consists only of forward edges) has n nodes and thus $n-1$ edges, fits in the lattice (as a

subgraph of a polyomino), so it's an $(n-1)$-edge.
The construction is reversible: given an $(n-1)$-edge, which is a tree, we know it has n nodes. We can place a polyomino cell at each node and get an n-omino.
Since the constructed tree is spanning, (i.e. contains all nodes) we cannot have two different polyominoes (which can be represented only by their labelled node set, the labels are from $\mathbb{Z}^d$) with same node sets. So the correspondence is one-to-one.

(b) We construct a one-to-one correspondence from the set of fixed k-dimensional (n,m)-edges to a set of binary strings of a special kind, which we'll write as matrices with partially occupied first line. To each (n,m)-edge X in standard position we assign such a matrix $W(X)$ from the set

$$\{\begin{pmatrix} a_{11} & \dots & a_{1k} & & & \\ a_{21} & \dots & a_{2k} & a_{2(k+1)} & \dots & a_{2(2k-1)} \\ a_{31} & \dots & a_{3k} & a_{3(k+1)} & \dots & a_{3(2k-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nk} & a_{n(k+1)} & \dots & a_{n(2k-1)} \end{pmatrix} \mid a_{ij} \in \{0,1\} \wedge \sum_{i,j} a_{ij} = m\}.$$

First of all we orient the edges of X. Let $s \in V(X)$ be the smallest node respective to the lexicographic order. Consider some edge $\{w,v\}$. We give it the orientation (v,w) if $d(s,v) < d(s,w)$ (where d is the graph distance map). Remark that the case $d(s,v) = d(s,w)$ cannot occur. (Imagine the k-dimensional chess board which cells are around our nodes. Each two neighbour nodes have different colours. Let s be white, then $\{t \mid d(s,t) = 1\}$ is black, then $\{t \mid d(s,t) = 2\}$ is white, etc. If $d(s,v) = d(s,w)$, then v and w have equal colour in contradiction to the fact that they are joined.)
Start with s and go with BFS (with a working list of nodes at depth j) through X as follows (almost the same as in part (a)).
Before we start, the working list is empty. At s, regard the directions $e_1, ..., e_k$ in this order. If an edge is present in the direction e_i, set $a_{1i} := 1$, otherwise $a_{1i} := 0$. In the same order, append the present neighbours of s to the working list.
Let $v \neq s$ be the lth node we discovered (the front element of the list is taken). Let w be the node from which v was discovered for the first time and $r := w - v$ the corresponding direction. Regard the directions $\sigma(r), \sigma^2(r), \sigma^3(r), ..., \sigma^{2k-1}(r)$ in this order. If an outgoing edge is present in the direction $\sigma^i(r)$, set $\alpha_{li} = 1$, otherwise (i.e. if an

in-going edge is present or no edge at all in that direction) $\alpha_{li} = 0$. Append those present neighbours of v, which are discovered for the first time, in that order to the working list and pop the front element v from the list.
The correspondence W is one-to-one, since we can reconstruct the polyedge from the matrix. In order to do that, we maintain the working list, which contains coordinates of the nodes that are not yet drawn. We put the first node to any place on the lattice, appending its neighbours according to a_{1i} $(i = 1, ..., k)$ to the empty list and drawing them at the same time on the lattice with the corresponding edges. Then we pop the first element of the list, and draw it's outgoing edges according to a_{2i} $(i = 1, ..., k)$. From the neighbour nodes we draw only those which were not yet drawn and append these "new" nodes at the end of the work-list. Then we pop the its first element, and repeat the whole thing... Notice that each time we pop a new node from the top of the list, we know its coordinates and hence we are able to compute the coordinates of its neighbours. We proceed until the list is empty.
Remark that the last line of each matrix has only zeros, since the last node regarded during BFS has no outgoing edges. So we have to choose m positions of 1 in the first line with k entries together with the other $n-2$ lines with $2k-1$ entries. This gives the upper bound of $\binom{(2k-1)(n-2)+k}{m} = \binom{(2k-1)n-3k+2}{m}$.

(c) By theorem 1.3.8, the number of nodes n satisfies $n \leq m+1$. This allows us to modify the map W from part (b) to be a one-to-one map from the set $PS_k(m)$ to the matrix set $\mathrm{Mat}((m+1) \times (2k-1), \mathbb{Z}/2\mathbb{Z})$. Here, each matrix is filled from above exactly the way it was done in part (b). The matrix rows which were not defined since there were not enough nodes to describe them during BFS, are filled with zeroes.
The proof of one-to-one property is the same, through reconstruction of a polyedge X from its representation $W(X)$. This reconstruction continues until the working list gets empty, the resting lines of $W(X)$ which were not considered during reconstruction should only contain zeroes.
We count the upper bound on the number of different matrices $W(X)$ for a k-dimensional m-edge the same way. For each such matrix holds:

- The last line is always empty, since the deepest node during BFS has no outgoing edges;
- $A_{1(k+1)} = ... = A_{1(2k-1)} = 0$;

- Altogether the "1" occurs m times.

This gives the bound of $\binom{(2k-1)(m-1)+k}{m} = \binom{(2k-1)m-k+1}{m}$.
For the second inequality, we remark that $\binom{(2k-1)m-k+1}{m} < \binom{(2k-1)m}{m}$ and prove the inequality (in the "lower equal" version) for the last binomial by induction.
$m = 1$. $\binom{2k-1}{1} = 2k-1 = (2k-1) \cdot 1^{m-1}$.
$m \geq 2$. Assume

$$\forall\, 1 \leq n \leq m : \binom{(2k-1)m}{m} \leq (2k-1) \left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}} \right)^{m-1}.$$

We are going to prove

$$\binom{(2k-1)(m+1)}{m+1} < (2k-1) \left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}} \right)^{m}.$$

Proof:

$$\frac{\binom{(2k-1)(m+1)}{m+1}}{\binom{(2k-1)m}{m}} = \frac{((2k-1)m+2k-1)!\, m!\, ((2k-2)m)!}{(m+1)!\, ((2k-2)m+2k-2)!\, ((2k-1)m)!} =$$

$$= \frac{((2k-1)m+1) \cdot \ldots \cdot ((2k-1)m+2k-1)}{(m+1)((2k-2)m+1) \cdot \ldots \cdot ((2k-2)m+2k-2)} =$$

$$= \frac{((2k-1)m+2k-1)}{m+1} \prod_{i=1}^{2k-2} \frac{(2k-1)m+i}{(2k-2)m+i} =$$

$$= (2k-1) \prod_{i=1}^{2k-2} \frac{(2k-1)(m+\frac{i}{2k-1})}{(2k-2)(m+\frac{i}{2k-2})} =$$

$$= (2k-1) \left(\frac{2k-1}{2k-2} \right)^{2k-2} \prod_{i=1}^{2k-2} \frac{m+\frac{i}{2k-1}}{m+\frac{i}{2k-2}}. \qquad (1.1)$$

Since for all $i > 0, m > 0, k \geq 2$ holds

$$2k-1 > 2k-2 \Rightarrow \frac{i}{2k-1} < \frac{i}{2k-2} \Rightarrow$$

$$m + \frac{i}{2k-1} < m + \frac{i}{2k-2} \Rightarrow \prod_{i=1}^{2k-2} \frac{m+\frac{i}{2k-1}}{m+\frac{i}{2k-2}} < 1,$$

the last term in (1.1) is smaller than

$$(2k-1) \left(\frac{2k-1}{2k-2} \right)^{2k-2} = \frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}.$$

(d) Let's take a k-dimensional n-omino in standard position for some $n \geq 2$. Regarded as a polyedge by means of the map described in the proof of (a), it is an $(n, n-1)$-edge. Part (b) and the proof from the part (c) give the bound

$$\begin{aligned}\binom{(2k-1)(n-2)+k}{n-1} &= \binom{(2k-1)n-3k+2}{n-1} = \\ = \binom{(2k-1)(n-1)-k+1}{n-1} &< \binom{(2k-1)(n-1)}{n-1} \leq \\ &\leq (2k-1)\left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}\right)^{n-2}.\end{aligned}$$

□

And now we finally get to our goal and prove the existence of this growth constant.

Corollary 1.3.10 (Klarner's constant)**.** The growth constants for k-dimensional polyedges and polyominoes exist and the following inequalities hold:

$$\lim_{n\to\infty} s_k(n)^{1/n} = \lim_{n\to\infty} t_k(n)^{1/n} \leq \frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}} \ ,$$

$$\lim_{m\to\infty} ps_k(m)^{1/m} = \lim_{m\to\infty} pt_k(m)^{1/m} \leq \frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}} \ .$$

Proof. Polyforms inequalities (1.3.9) tell us, that the sequences $(s_k(n)^{1/n})_n$ and $(ps_k(m)^{1/m})_m$ are bounded by $\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}$. Apply the superadditivity lemma (1.3.6) and the Fekete lemma (1.3.2) to get the result. The limits for the free versions of polyedges and -ominoes exist and are equal to the limits of the fixed versions because the corresponding counts differ at most by a constant factor $2^k k!$. □

Remark 1.3.11. The results of Klarner and Rivest from [7] are generalisable to higher dimensions leading to smaller upper bounds for polyominoes. There exists a procedure which allows the human to shrink the upper bound on the polyominoes. The amount of work even for two dimensions is considered to be excessive, however. For polyedges, the mentioned generalisation is unlikely to work. But it's not excluded that some similar methods may exist.

1.3.2 Computational enumeration

Polyomino enumeration served as a kind of benchmark for new enumeration algorithms and new computers. It also served as CPU/memory testing procedure[3].

Remark 1.3.12 (Counting methods for polyedges, overview)**.** We only concentrate on the 2-dimensional versions.
In order to determine the growth constant, more terms from any of the series - fixed or free - are needed, while counting the fixed version takes far less time. There are several methods of counting.

Extension Generate and store all the m-edges, then, based on them, all the $m+1$ edges, etc... The main problem is that one has to check whether a $m+1$-edge is not generated twice as a child of different m-edges. This is only helpful if you really need all these polyedges and takes $O(ps_2(m))$ space and $O(ps_2^2(m))$ time in the worst case. The two in the exponent may be diminished a bit. We could classify fixed polyedges according to their length and width, first node (leftmost node in the lowest row), number of horizontal edges, number of nodes with different degrees (from 1 to 4 in the planar case), etc. This allows us to compare new polyedges only within the group with the same parameters, while searching for this group may be realized via a balanced tree, if the number of invariants increases so that fixed-size arrays get too big. Counting all 9-edges takes a few days (without pruning).

Rooted with depth-first-search. This method, sometimes called Redelmeier's algorithm (see [3]), was widely used for polyominoes and is extendible to polyedges as well as to many other "polyforms". It allows you to do without any comparisons at all. It is also highly parallelisable also for polyedges. The single problem is that it's not "incrementable", that means that it's not possible to count all m-edges first and only after that all $m+1$-edges, each enumeration of $m+1$-edges involves the enumeration of $m, m-1, m-2, ...,$ 1-edges. We need time $O(ps_2(m))$. The space is almost constant: $O(m)$ for the place to store one single polyedge (on which we operate) and the same for the stack which supports recursion. A parallel version also takes space proportional to

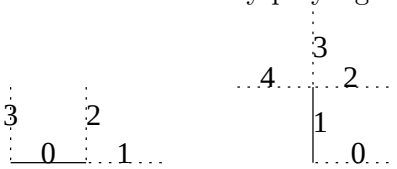
[3]Three of four machines the author used for testing turned out to dislike these CPU- and memory-intensive processes and finished with a "Bus error" message after a week of computation, thus upsetting the author terribly.

the number of processes. Here are the results of single-computer runs (these results are not systematic, since the computers were also doing other work in parallel thus affecting speed):

Maximal number of edges	Run-time in seconds
10	80 ms.
11	400 ms.
12	2 s.
13	9 s.
14	23 s.
15	2 min.
16	10 min.
17	47 min.
18	3 h 40 min.
19	20 h.
20	2 d. 20 h.
21	13 d. 22 h.

Much better methods (finite-lattice with a matrix-transfer algorithm, see [4]) exist, which give better results but were not implemented for polyedges yet.

Remark 1.3.13 (Sketch of the rooted method)**.** We use the simple edge extension to create an $m+1$-edge from an m-edge. The main purpose is to avoid redundancy. We do that by assigning numbers to the present edges and placeholders for them on a grid. The assigned numbers always form a set $\{0,..,a\}$ for some $a \in \mathbb{N}_0$, that means, they have no "holes". We start with two single polyedges, which are located at the leftmost position in the bottom row of every polyedge we generate; we call them "roots":



The numbers are present only on existing edges and on those placeholders for the edges which either can become neighbours or were able to become neighbours in the past (but didn't). The biggest number of all present edges is the delimiter between these two classes. It's 0 in the first case and 1 in the second. Then we extend each of these polyedges, using the following rules:

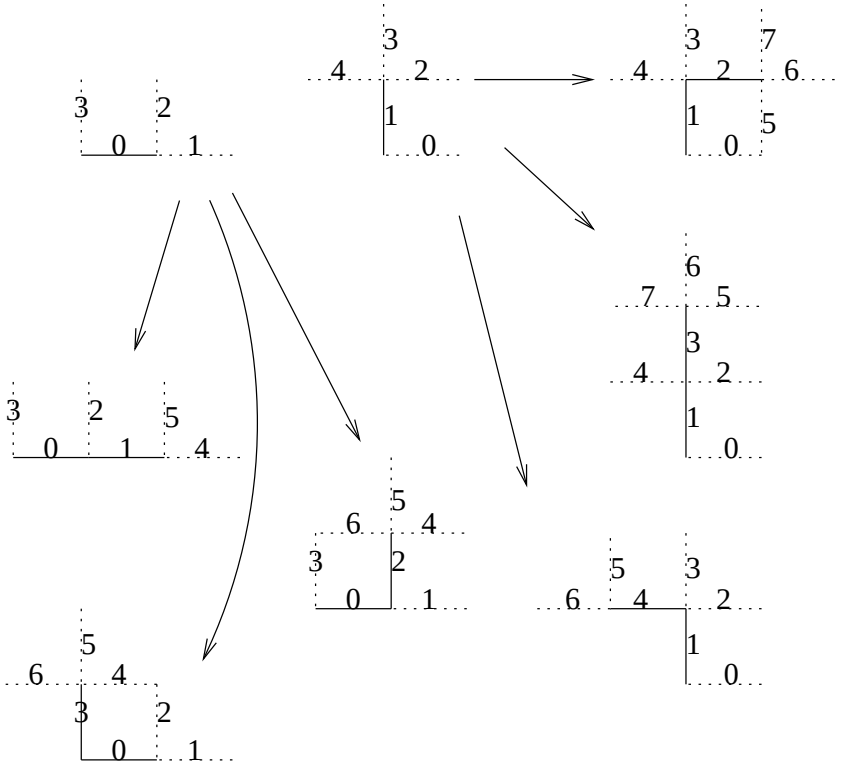
(a) For each number on the grid greater than the biggest number of all

present edges, extend the present polyedge to a new polyedge by putting the edge on the placeholder for that number;

(b) For each of these new polyedges, give all the unnumbered placeholders, which are adjacent to the last created edge, the numbers directly following the numbers on the grid.

(c) Avoid numbering placeholders which are in the bottom row and to the left of the roots or which are below the roots so that the roots remain the leftmost edges in the bottom row.

The following forest symbolises this process:



The m-th level of the forest contains all the fixed m-edges exactly once each. The two trees can be generated (and traversed) either depth- or breadth-first. The breadth-first version exhausts memory too quickly to be useful, and the depth-first version can be done on a single grid, avoiding memory costs and time needed for copying. On the other side, "incremental" enumeration is impossible during the depth-first version. Of course, parallel "hybrids" of both versions are implementable.

Until now, nothing really new has happened. In what follows, we describe a real - although small - improvement.

Remark 1.3.14 (Optimised implementation of the rooted method)**.** For the lack of necessary resources, the author implemented the version for a two-CPU-machine, which gave a speedup factor of 2: each processor enumerated one of the two trees of the forest. Further, some improvements were made, which reduced the runtime by a multiplicative constant, which we describe now.
The idea is that for counting fixed polyedges, we don't need to know its real form at the recursion leaves. Even more, we can organise the data structures in such a way, that tracking the actual polyedge form becomes superfluous. All the coordinates of edges with numbers are additionally maintained in a list (implemented as an array, whose indices are the corresponding numbers), and the grid only contains boolean variables, telling us whether each edge has been numbered or not. Two list pointers (array indices) are maintained, with the first one pointing to the end of the list, i.e. to the element which has the biggest number and the second one pointing to the element containing coordinates of the recently occupied edge. The counts of m-edges are stored in an extra array. Given an m-edge, the algorithm does the following:

(a) Add 1 to the counter for m-edges;

(b) If we have to enumerate only direct children of the current m-edge (and not all the descendants), add the difference between the "last-used" and the "last-occupied" index to the count for $m+1$-edges;

(c) Otherwise

  (i) Back up the two indices "last-used" and "last-occupied";

  (ii) Increase the edge count;

  (iii) Take the coordinates of the first unoccupied edge and increase the "last-occupied" index;

  (iv) Determine the neighbours of this edge which were not numbered yet and are legal to be numbered (take this information from the grid);

  (v) Append these neighbours to the end of the list (incrementing the "last-number" index) and mark their presence on the grid;

  (vi) Recurse;

(vii) Remove the marks of these new neighbours from the grid and remove them from the list (by restoring the value of the stored "last-used" index);

(viii) Go over to state (ciii) (with the next unoccupied edge);

(ix) Restore the "last-occupied" index;

(x) Decrease the edge count.

All the optimisations, which depend on the architecture of the machine, were left over to the compiler (the impact of it's automated optimisation should not be underestimated, the speedup achieved more than the factor of two).

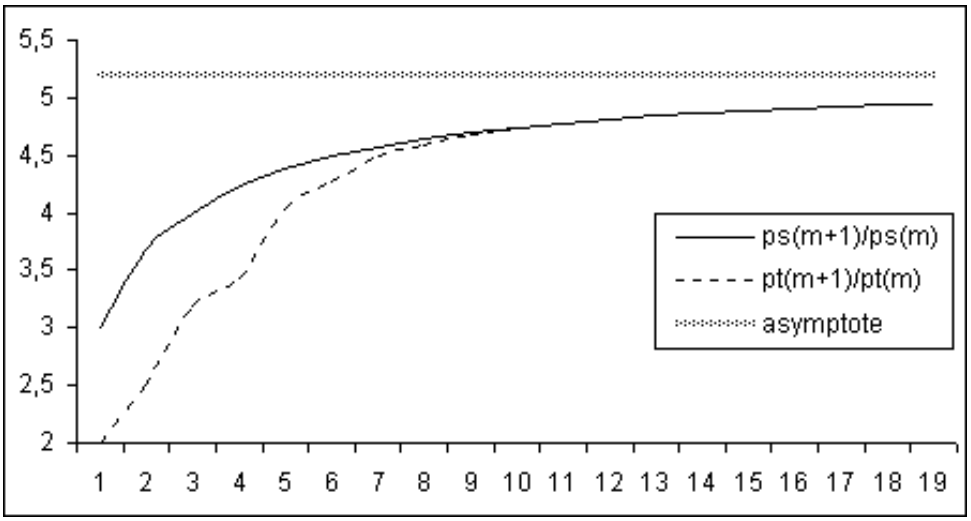
Problem 1.3.15. Give tight upper and lower time bounds for counting $ps_2(m)$.

1.3.3 Growth constant estimation for planar polyedges.

In this part we attempt to estimate the growth constant for two-dimensional polyedges. Our analysis doesn't prove anything, but only suggests the approximate value of the constant.

Problem 1.3.16. Find the Klarner's constant for planar polyedges. Or give a polynomial-time approximation algorithm for it.

Remark 1.3.17 (Simple estimation of the planar polyedges growth constant)**.** We use the available numerical data for polyedges to estimate the borders. First, we note that if $\lim_{n\to\infty} a_n^{1/n}$ exists, then it is equal to $\lim_{n\to\infty} \frac{a_{n+1}}{a_n}$. The last term converges even faster in our case. Substituting the free and fixed planar polyedges counts $pt_2(m)$ and $ps_2(m)$ for a_m gives the following graphs of a_{m+1}/a_m:



We have drawn the asymptote more or less arbitrarily at 5.208 (later we'll see why we have chosen this value). The graphs resemble the graphs of the function $c - \frac{a}{b+m}$ where m is the argument and a, b and c are some constants. Through each three points on the graph we can draw such a curve and read the value of c. Under the assumption that the asymptotic behaviour of $b_m := a_{m+1}/a_m$ is of the upper form, the value of c will converge to our growth constant if we shift all three points (through which we draw the curve) to the right. During estimation, we take b_m, b_{m+1}, b_{m+2} and shift m to the right as long as numerical data from counting algorithms is available. To our great relief, the estimation for the next value of b_{m+3} coincides with the real value quite well, so we can be quite sure about the usefulness of the method. We get the following values for c, if we increase m (left - fixed version, right - free version):

m	fixed b_m	estimated c	free b_m	estimated c
1	3	5	2	-1
2	3.666667	5.428571	2.5	3.91892
3	4	5.09375	3.2	2.65031
4	4.227273	5.384819	3.4375	4.85375
5	4.376344	5.155838	4.03636	7.40016
6	4.491400	5.244146	4.27928	4.90087
7	4.57686	5.203446	4.48947	5.6
8	4.644893	5.213746	4.59343	4.97499
9	4.6996	5.208979	4.6796	5.20667
10	4.744707	5.210987	4.73401	5.07614
11	4.782476	5.208272	4.77825	5.19637
12	4.814584	5.209203	4.81235	5.15294
13	4.84219	5.208530	4.84131	5.20135
14	4.866186	5.208418	4.86573	5.19084
15	4.887231	5.208265		
16	4.905838	5.208198		
17	4.922406	5.208094		
18	4.937253	5.208043		

So we can estimate the growth constant somewhere around 5.20(8). At the same time, the values of b_m remained under this border and this is likely to continue. Unfortunately, no proof is known that b_m is growing for big m or at least approaching c from below.

Remark 1.3.18 (Another estimation attempt)**.** We could allow a more general formula for b_m, for example, $c - \frac{a}{(b+m)^d}$, where $d \in \mathbb{R}^+$. In order to determine a, b, c, d, we need four values: b_m, b_{m+1}, b_{m+2} and b_{m+3}. At the same time, our system of four equations with four unknowns is no longer exactly solvable and so we have to rely on numerical methods. After having pre-scanned a 4-dimensional interval around the expected values of a, b, c, d (we assume that $0.7 < d < 1.3$ and so expect a, b, c to be near the values from the first case), we found some starting points for Newton iteration and were able to solve the system with the accuracy of order 10^{-15}. Here are some results for big m:

m	a	b	c	d
11	6.128449055	2.70523696	5.19920904	1.057549038
12	5.381170198	2.34816225	5.20672157	1.010671737
13	5.471303735	2.39617404	5.205796278	1.016587831
14	5.34061428	2.32350665	5.207041898	1.008150116
15	5.432905309	2.3769732	5.206221205	1.014018491
16	5.332355777	2.31658752	5.207054461	1.007733673

We see that $d \approx 1, c \approx 5.20$. At the same time, the solutions don't seem to converge fast, this may give a hint that our formula is not perfect yet. This poses the following question.

Problem 1.3.19. For $b_m := ps_2(m+1)/ps_2(m), c := \lim_{m\to\infty} b_m$ describe the asymptotic behaviour of $b_m - c$. Is $b_m - c \sim \frac{a}{(b+m)^d}$ for some constants $a, b, d \in \mathbb{R}^+$? If the answer is positive, is $d = 1$?

1.3.4 Lower bounds for the growth constant for planar polyedges.

In this part we apply theoretical results, that were developed by Rands and Welsh, in order to establish a lower bound on the polyedge growth constant for the square lattice. First of all we'll summarise their results by citing necessary definitions and theorems, and then apply the theory to our case. We'll omit the proofs of those theorems, but if an eager reader insists, he can read them in the original paper (see [13] and [14]).

Definition 1.3.20. We define reciprocal and pseudo renewal sequences.

(a) Let $\mathbf{u} := (u_n)_{n\in\mathbb{N}_0}$ be a sequence of real numbers with $u_0 = 1$. Its *reciprocal sequence* is $\mathbf{f} := (f_n)_{n\in\mathbb{N}^+}$ is defined by the relations

$$u_n = u_0 f_n + u_1 f_{n-1} + \ldots + u_{n-1} f_1, \quad n \geq 1.$$

This uniquely defines the sequence $\mathbf{f}$ and one can check that the corresponding generating functions U and F are related by the formal identity

$$U(x) = U(x)F(x) + 1.$$

(b) A sequence $\mathbf{a} := (a_n)_{n\in\mathbb{N}_0}$ with $a_0 = 1$ is called a *pseudo renewal sequence*[4] if its reciprocal sequence $\mathbf{g}$ satisfies

$$g_n \geq 0, \quad 1 \leq n < \infty, \qquad \text{and}$$
$$\exists\, K < \infty : \forall\, n : a_n < K^n.$$

[4]The reader should not be confused by the long name: renewal sequences exist also, but they are not introduced here, since unnecessary.

(c) For a pseudo renewal sequence **a** with the reciprocal sequence **g** define polynomials p_n for $n \geq 1$ by

$$p_n(x) = x^n - (g_1 x^{n-1} + g_2 x^{n-2} + \ldots + g_{n-1}x + g_n).$$

One can show an analogue of our superadditivity lemma ($a_{m+n} \geq a_m a_n$ for $m, n \geq 1$) for pseudo renewal sequences, which implies the existence of the growth constant $a = \lim_{n\to\infty} a_n^{1/n}$ for every pseudo renewal sequence **a**. The following theorem is essential to formulating an algorithm which establishes a lower bound.

Theorem 1.3.21. For a pseudo renewal sequence **a** with the growth constant a, the reciprocal sequence **g** and the corresponding polynomials $(p_n)_n$ holds:

(a) Each p_n has a unique non-negative zero α_n;

(b) $\forall\, n \geq 1 : \alpha_n \leq \alpha_{n+1}$;

(c) $p_n'(x) > 0$ for $x \geq \alpha_n$;

(d) $\lim_{n\to\infty} \alpha_n$ exists and is equal to a.

That theorem will be used to give a lower bound for a by determining α_n for the largest possible n. The next claim proves that the new result can give a better lower bound than $a_n^{1/n} \leq a$.

Proposition 1.3.22. Suppose that **a** is a pseudo renewal sequence and $(a_n^{1/n})_n$ converges monotonically to a. Then for each value of n,

$$a_n^{1/n} \leq \alpha_n \leq a.$$

The proof in [13] uses the monotonicity in a special way: if we want to establish the inequality for some fixed n, then instead of the monotonicity requirement for the whole sequence, we could require only $a_n^{1/n} \geq a_k^{1/k}$ for $0 < k < n$.
The next proposition will help making lower bounds a bit more tight.

Proposition 1.3.23. Suppose that $\mathbf{a} = (a_n)_{n\in\mathbb{N}_0}$ is a pseudo renewal sequence with the reciprocal sequence **g**, the values α_n are defined as above and that the sequence $\mathbf{g}'$ satisfies $0 \leq g_n' \leq g_n$ for $n \geq 1$. If β_N is a positive zero of

$$x^N - g_1' x^{N-1} - g_2' x^{N-2} - \ldots - g_N',$$

then $\beta_N \leq \alpha_N$.

We use this theorem by estimating those α_N from below, for computing which we don't have enough numerical data.

Now we apply these theoretical results to our problem the same way it has been applied by Rands and Welsh to polyominoes. Let's fix the dimension number $k \geq 2$. The following proposition was formulated in [13] for polyominoes with almost the same proof.

Proposition 1.3.24. The sequence $(ps_k(m))_{m\in\mathbb{N}_0}$ with $ps_k(0) := 1$ is a pseudo renewal sequence.

Proof. Following Rands and Welsh, we say that a polyedge $A \in PS_k(m)$ is *constructible*, if it can be split into two polyedges $B \in PS_k(r)$ and $C \in PS_k(m-r)$ for some $0 < r < m$ by using the one-to-one map (call it Z here)

$$Z\colon PS_k(r) \times PS_k(m-r) \hookrightarrow PS_k(m)$$

from 1.3.4, which connects the lexicographically greatest element of the node set of B with the lexicographically smallest element of the node set of C. If such splitting is impossible we call this polyedge *inconstructible*. Let

$$\Delta_k(m) := \{A \in PS_k(m) \mid A \text{ is inconstructible } \} \quad \text{and}$$

$$\delta_k(m) := |\Delta_k(m)|.$$

Each polyedge $A \in PS_k(m)$ is either inconstructible or we can split it (maybe in more than one way) into $B \in PS_k(r)$ and $C \in PS_k(m-r)$ for some $0 < r < m$. In the latter case, we choose the split possibility with the smallest r. Then B is inconstructible and $A \in Z(\Delta_k(r) \times PS_k(m-r))$. Thus

$$PS_k(m) = \Delta_k(m) \dot\cup \left(\bigcup_{r=1}^{m-1} Z(\Delta_k(r) \times PS_k(m-r)) \right).$$

By definition of Z and r, the union on the right hand side is disjoint. Since Z is one-to-one, we have

$$ps_k(m) = \delta_k(m) + \sum_{r=1}^{m-1} \delta_k(r) ps_k(m-r), \quad m \geq 1. \qquad (*)$$

So the inconstructible polyedge numbers form the reciprocal sequence for $(ps_k(m))_m$. Since $\delta_k(m) \geq 0$ and from inequalities in 1.3.9 follows that $ps_k(m) \leq \left(\frac{(2k-1)^{2k-1}}{(2k-2)^{2k-2}}\right)^m$, we have proven that $(ps_k(m))_m$ with $ps_k(0) = 1$ is a pseudo renewal sequence by definition. □

Now we set $k = 2$ and can determine iteratively $\delta_2(m)$ from $(*)$ and from the direct enumeration results of planar polyedges until $M := 21$. We use the theorem 1.3.21 and determine the unique non-negative zeros α_m of the corresponding polynomials p_m. These zeros are lower bounds for the polyedge growth constant. Hereby we get a little better bound if we remark that $\delta_2(m+1) \geq \delta_2(m)$ and use the proposition 1.3.23, where the numbers $\delta_2(m)$ for $m \geq M$ are set to $\delta_2(M)$. It turns out that extending the sequence of inconstructible polyedge numbers this way doesn't give much improvement. Let's view the results:

m	$\delta_2(m)$	α_m
1	2	2
2	2	2.73205
3	6	3.20701
4	20	3.51508
5	76	3.73736
6	304	3.9042
7	1280	4.03544
8	5538	4.14125
9	24550	4.22863
10	110766	4.30207
11	507050	4.36473
12	2348546	4.41886
13	10985660	4.46614
14	51815378	4.5078
15	246144560	4.54482

m	$\delta_2(m)$	α_m
16	1176536280	4.57793
17	5654171948	4.60774
18	27302650348	4.63473
19	132398319130	4.65929
20	644477607218	4.68174
21	3147863722202	4.70234
22	the same (extended)	4.7064
23	...	4.70725
24	...	4.70743
25	...	4.70747
26	...	4.70748
27	...	4.70748
28	...	4.70748
29	...	4.70748
...	...	≈ 4.70748

We should mention, that all the values printed here are rounded and that $p'_m(\alpha_m)$ gets very high with growing m, so that even a small rounding of the argument $x \approx \alpha_m$ produces value $p_m(x)$ which is far away from zero. We can summarise now:

Theorem 1.3.25 (Bounds on the planar polyedge growth constant)**.**

$$4.70747 < \lim_{m\to\infty} ps_2(m) \leq 27/4$$

Now we have a procedure for improving the lower bound, which depends on knowing polyedge counts, which can be reached (nowadays) only through a super-polynomial-time enumeration algorithm. Note that these lower bounds converge to our limit, guaranteed by the theorem 1.3.21. For the upper bound, we don't see anything similar at the moment.

Problem 1.3.26. Give a procedure for improving an upper bound for the number of planar m-edges.

1.4 Related polyforms

If the main problems concerning polyedges (e.g. generating function, asymptotic behaviour) are hard to solve exactly, we could try to explore similar "polyforms", hoping to get more exact results, or hoping to find out whether there is a relationship between polyedges and the other "polyforms".

1.4.1 Trees with candles

Here we introduce a new type of planar polyedges, for which we are able to find exact formulas and establish asymptotic properties relatively easy.

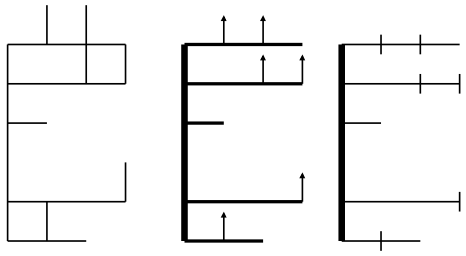
Definition 1.4.1 (Chopped Tree With Candles)**.** A *chopped tree with candles*[5] is a two-dimensional m-edge with the following properties:

- It contains a vertical line segment (of any length between 0 and m), which we call "trunk"; the trunk is to the left of all other edges.
- It contains horizontal line segments, which we call "branches", which lie to the right of the vertical segment. Their leftmost point is on the trunk.
- Each branch is allowed to have "candles", which are vertical edges of length 1, whose lower node is on a branch.

A fixed chopped tree with candles is an equivalence class under the group of translations. However, we are going to speak only about fixed chopped trees, so we omit the prefix "fixed".

Example 1.4.2. Here is an example of a chopped tree with candles. We can view it either as a polyedge (left picture), or as a tree with candles on each branch (middle picture) or as a histogram (right picture):

[5] A more precise name would be "one-sided histogram", but "imagination is more important than knowledge"(A. Einstein).



Before we count the "chopped trees with candles", we need to list some known facts for preparation.

Definition and Proposition 1.4.3 (Pell Numbers)**.**

(a) An n-th Pell number p_n is defined by

$$\begin{aligned} p_0 = 1, p_1 = 2, \\ p_n = 2p_{n-1} + p_{n-2}, \quad n \geq 2 \end{aligned} \tag{1.2}$$

The sequence starts with 1, 2, 5, 12, 29, 70, ...

(b) For all $n \in \mathbb{N}_0$ we have:

$$p_n = \sum_{\substack{0 \leq l,d,k \\ l+2k+d=n}} \frac{(l+d+k)!}{d!k!l!} = \sum_{\substack{0 \leq l,d,k \\ l+2k+d=n}} \binom{n-k}{d,k,l},$$

where the last notation uses multinomial coefficients.

(c) For all $n \in \mathbb{N}_0$ holds:

$$p_n = \frac{(1+\sqrt{2})^{n+1} - (1-\sqrt{2})^{n+1}}{2\sqrt{2}}.$$

(d) The generating function is $\sum_{n \geq 0} p_n x^n = \frac{1}{1-2x-x^2}$.

(e) $\lim_{n \to \infty} p_n^{1/n} = 1 + \sqrt{2}$.

For an extensive summary on Pell numbers, we direct the reader to the Encyclopedia of Integer Sequences (see [12]).

Proof. Since all this is known, we only sketch the ideas behind the proof. For the proof of part (b) from the recursive definition, see a paper by Tewodros Amdeberhan (available at [1]), where he exploits a software package and achieves the result very fast.

In fact, the whole thing can also be proven as follows: we assume the definition through multinomial coefficients, devise a closed form of the generating function $\sum_{d,k} \binom{n-k}{d,k,n-2k-d} y^d z^k$ (which is quite big) and get the closed form for p_n from (c) by setting $y = z = 1$. The latter can be checked to satisfy the recursive formula very quickly. We can multiply the recursive equation (1.2) by x^n and sum on n, thus getting an equation for the generating function of the sequence $(p_n)_n$. Solving it gives $1/(1 - 2x - x^2)$. This rational function has two singularities $-1 \pm \sqrt{2}$, so the convergence radius of the power series is $\sqrt{2}-1$ (the smallest absolute value), and Cauchy-Hadamard formula gives us $\lim_{n\to\infty}(p_n)^{1/n} = 1/(\sqrt{2}-1) = \sqrt{2}+1$. □

We need another standard combinatorial lemma. We speak of numbered (distinguishable) objects $x_1, x_2, ...$ if we count the pairs (x_i, x_j) and (x_j, x_i) as two different pairs for all i, j. If we count them as one, we speak about unnumbered (indistinguishable) objects.

Lemma 1.4.4. The number of ways to distribute b unnumbered balls into c numbered boxes is $\binom{b+c-1}{b}$.

Proof. See any good book on combinatorics (e.g. [15], page 42). In short: Any such distribution can be written in form of $xx|x||xxx|||x$, where x denotes a ball and $|$ separates different boxes. Vice versa, each finite string from the alphabet $\{x, |\}$ can be interpreted as a distribution. The symbol x occurs b times, the symbol $|$ occurs $c-1$ times. □

Now to our main result.

Theorem 1.4.5. The number of chopped trees "without candles", that have m edges overall, is 2^m. The number of chopped trees with candles, that have m edges overall, is p_m.

Proof. We give two different proofs.

(a) If a chopped tree has d edges in its trunk, it has $d+1$ positions where branches can be attached to. If the branches have s edges, they can be distributed in $\binom{s+(d+1)-1}{s} = \binom{s+d}{s}$ ways onto these positions according to the previous lemma. For each such distribution, we can place k candles in $\binom{s}{k}$ ways, since each branch edge can support at most one candle independently on other branch edges. So the whole number of chopped trees that have m edges is

$$\sum_{\substack{0\le s,d,k \\ s+d+k=m}} \binom{s+d}{s}\binom{s}{k} = \sum_{\substack{0\le d \\ 0\le k\le s \\ s+d+k=m}} \frac{(s+d)!s!}{s!d!k!(s-k)!} \overset{l=s-k}{=} \sum_{\substack{0\le l,d,k \\ l+2k+d=m}} \frac{(l+k+d)!}{d!k!l!}.$$

Choosing only summands with $k = 0$ leads to $\sum_d \binom{m}{d} = 2^m$, which completes the proof.

(b) We check immediately that there are two trees with candles with one edge, 5 trees with candles with two edges, 2 trees without candles with one edge and 4 trees without candles with 2 edges. In order to prove the recurrence relations, we note that each tree with candles with overall $m + 2$ edges is formed either by putting an edge on the top of a tree with $m + 1$ edges, or by lengthening the topmost branch of a tree with $m + 1$ edges, or by lengthening the topmost branch of a tree with m edges and putting a candle on its end.[6] This process generates all trees with or without candles in a unique way. It leads to the recurrence relation $c_{m+2} = 2c_{m+1}$ for trees without candles (if c_m denotes their number) and to $p_{m+2} = 2p_{m+1} + p_m$ for trees with candles.

□

The growth constant for chopped trees with candles is $1 + \sqrt{2}$, so a very small part of polyedges has this special form. We could ask ourselves, what happens if we don't chop the left side of the tree, whether we get a bigger growth constant. This leads to the following definition.

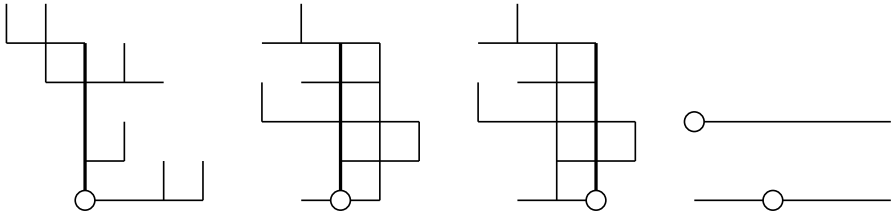
Definition 1.4.6 (Tree with Candles)**.** A (fixed) *tree with candles* is a two-dimensional m-edge with the following properties:

- It contains a line segment of any length from 0 to m on the vertical axis, call it the "trunk"; its lowest node is at the origin.
- It contains horizontal line segments; each of them intersects the trunk. The parts to the left and to the right of the trunk are called "branches".
- Each branch is allowed to have "candles", which are vertical edges of length 1, whose lower node is on a branch.

We define q_m as the number of all (fixed) trees with candles that have overall m edges. By convention, $q_0 = 1$. The sequence starts as follows: $1, 3, 10, 31, 96, 296, 912, ...$

Example and Remark 1.4.7. All chopped trees with candles are also trees with candles. Here are some nontrivial trees with candles, where the origin (the root) is marked with a big dot:

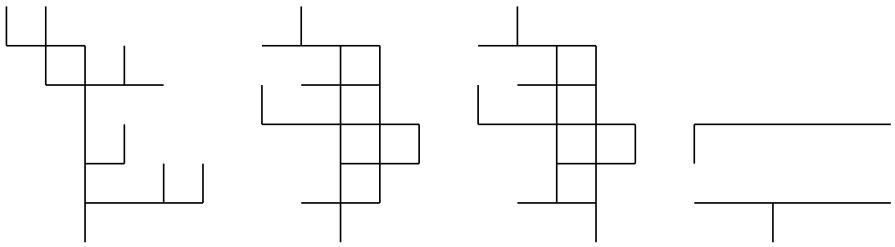
[6] A trivial idea thanks to Mitch Harris.



The problem is that some fixed m-edges correspond to more than one fixed tree with candles having m edges. But if we append a single vertical edge to the origin from below, then we will get a one-to-one correspondence

$$\text{trees with candles with } m \text{ edges overall} \hookrightarrow ps_2(m+1).$$

Here the upper examples become unique fixed polyedges:



We could as well let this additional edge go into the definition, but for the sake of simplicity during calculations we renounced to do that.

To our great astonishment we see now that there exists an exact formula for the number of trees with candles with m edges.

Theorem 1.4.8. The number of trees with candles with m edges is

$$q_m := \sum_{\substack{0\le s,d,k \\ s+d+k=m}} \binom{s+2d+1}{s}\binom{s}{k}.$$

Proof. If such a tree has d edges in its trunk, it has $2d+2$ positions to attach branches. If the branches have s edges, they can be distributed in $\binom{s+(2d+2)-1}{s}$ ways onto these positions according to the lemma 1.4.4, since the edges are indistinguishable and the positions are distinguishable. For each such distribution, we can place k candles onto s edges we used for branches in $\binom{s}{k}$ ways, since each branch edge can carry at most one candle independently on other branches. For each triple of numbers (s,d,k) we get $\binom{s+2d+1}{s}\binom{s}{k}$ possibilities. □

This result is not very easy to deal with. So our next task would be to find a generating function for this term, a recursive formula and a closed form for q_m - if any of the kind exists. We show our solutions for all three problems. We start with two simple lemmas. [7]

Lemma 1.4.9. For all $n \in \mathbb{N}_0$:

$$\sum_{k\geq 0} \binom{n-k}{k} z^k = \frac{(1+\sqrt{1+4z})^{n+1} - (1-\sqrt{1+4z})^{n+1}}{2^{n+1}\sqrt{1+4z}}.$$

Proof. [8] We use the normal convention that $\binom{a}{b} \neq 0$ iff $0 \leq b \leq a$.

$$\sum_{n\geq 0}\sum_{k\geq 0} \binom{n-k}{k} z^k x^n = \sum_{k\geq 0} z^k \sum_{n\geq 0} \binom{n-k}{k} x^n \overset{n-k=l}{=} \sum_{k\geq 0} z^k \sum_{l\geq 0} \binom{l}{k} x^{k+l} =$$

$$= \sum_{l\geq 0} x^l \sum_{k\geq 0} \binom{l}{k} (zx)^k = \sum_{l\geq 0} x^l (1+zx)^l = \sum_{l\geq 0} (x+zx^2)^l = \frac{1}{1-x-zx^2} =$$

$$= \frac{-1}{z\left(x - \frac{-1+\sqrt{1+4z}}{2z}\right)\left(x - \frac{-1-\sqrt{1+4z}}{2z}\right)} =$$

$$= \frac{-1}{z}\left(\frac{z/\sqrt{1+4z}}{\left(x - \frac{-1+\sqrt{1+4z}}{2z}\right)} - \frac{z/\sqrt{1+4z}}{x - \frac{-1-\sqrt{1+4z}}{2z}}\right) =$$

$$= \frac{1}{\sqrt{1+4z}}\left(\frac{1}{\frac{-1+\sqrt{1+4z}}{2z} - x} - \frac{1}{\frac{-1-\sqrt{1+4z}}{2z} - x}\right).$$

[7] One could use a software package, for example Maple, to prove lemmas, but the correctness proof for Maple is still missing.

[8] Thanks to Rob Johnson for showing a trick to shorten the proof.

Since $\frac{1}{a-x} = \frac{1}{a\left(1-\frac{x}{a}\right)} = \sum_{n\geq 0} a^{-n-1}x^n$ for $a \neq 0$, we have

$$\frac{1}{\sqrt{1+4z}}\sum_{n\geq 0}\left(\left(\frac{2z}{-1+\sqrt{1+4z}}\right)^{n+1} - \left(\frac{2z}{-1-\sqrt{1+4z}}\right)^{n+1}\right)x^n =$$

$$= \sum_{n\geq 0}\frac{(2z)^{n+1}\left(\left(-1-\sqrt{1+4z}\right)^{n+1} - \left(-1+\sqrt{1+4z}\right)^{n+1}\right)}{\sqrt{1+4z}\left(\left(-1+\sqrt{1+4z}\right)\left(-1-\sqrt{1+4z}\right)\right)^{n+1}}x^n =$$

$$= \sum_{n\geq 0}\frac{(2z)^{n+1}(-1)^{n+1}\left(\left(1+\sqrt{1+4z}\right)^{n+1} - \left(1-\sqrt{1+4z}\right)^{n+1}\right)}{\sqrt{1+4z}\left(1-(1+4z)\right)^{n+1}}x^n =$$

$$= \sum_{n\geq 0}\frac{\left(1+\sqrt{1+4z}\right)^{n+1} - \left(1-\sqrt{1+4z}\right)^{n+1}}{2^{n+1}\sqrt{1+4z}}x^n.$$

□

Another lemma gives us the generating function of our sequence $(q_m)_m$.

Lemma 1.4.10.

$$\sum_{m\geq 0}\sum_{\substack{0\leq s,d,k\\ s+d+k=m}}\binom{s+2d+1}{s}\binom{s}{k}x^m = \frac{1}{x^4+2x^3-x^2-3x+1}.$$

Proof.

$$\sum_{m\geq 0}\sum_{\substack{0\leq s,d,k\\ s+d+k=m}}\binom{s+2d+1}{s}\binom{s}{k}x^m = \sum_{s\geq 0}x^s\sum_{m\geq 0}\sum_{\substack{0\leq d,k\\ d+k=m-s}}\binom{s+2d+1}{s}\binom{s}{k}x^{m-s}.$$

Substituting $l = m - s$ and noticing that the inner sum is zero for $m < s$ gives

$$\sum_{s\geq 0}x^s\sum_{l\geq 0}\sum_{\substack{0\leq d,k\\ d+k=l}}\binom{s+2d+1}{s}\binom{s}{k}x^l =$$

$$= \sum_{s\geq 0}\left(\sum_{d\geq 0}\binom{s+2d+1}{s}x^d\right)\left(\sum_{k\geq 0}\binom{s}{k}x^k\right)x^s =$$

$$= \sum_{s\geq 0}(1+x)^s\sum_{d\geq 0}\binom{s+d+d+1}{s}x^{d+s} \overset{d+s=n}{=} \sum_{s\geq 0}\sum_{n\geq s}(1+x)^s\binom{2n+1-s}{s}x^n =$$

$$= \sum_{n\geq 0}x^n\sum_{s\geq 0}\binom{2n+1-s}{s}(1+x)^s.$$

We know that the generating function from the previous lemma 1.4.9 is a polynomial, so substitution of $1+x$ for z in it is well-defined. That lemma gives us:

$$\sum_{n\geq 0} x^n \frac{\left(1+\sqrt{5+4x}\right)^{2n+2} - \left(1-\sqrt{5+4x}\right)^{2n+2}}{2^{2n+2}\sqrt{5+4x}} =$$

$$= \frac{1}{\sqrt{5+4x}} \sum_{n\geq 0} x^n \left[\left(\left(\frac{1+\sqrt{5+4x}}{2} \right)^2 \right)^{n+1} - \left(\left(\frac{1-\sqrt{5+4x}}{2} \right)^2 \right)^{n+1} \right] =$$

$$= \frac{1}{\sqrt{5+4x}} \sum_{n\geq 0} \left[\left(\frac{6+4x+2\sqrt{5+4x}}{4} \right) \left(\frac{6+4x+2\sqrt{5+4x}}{4} x \right)^n - \right.$$

$$\left. - \left(\frac{6+4x-2\sqrt{5+4x}}{4} \right) \left(\frac{6+4x-2\sqrt{5+4x}}{4} x \right)^n \right] =$$

$$= \frac{1}{\sqrt{5+4x}} \left(\frac{3+2x+\sqrt{5+4x}}{2\left(1 - \frac{2x^2+3x+x\sqrt{5+4x}}{2}\right)} - \frac{3+2x-\sqrt{5+4x}}{2\left(1 - \frac{2x^2+3x-x\sqrt{5+4x}}{2}\right)} \right) =$$

$$= \frac{1}{\sqrt{5+4x}} \left(\frac{3+2x+\sqrt{5+4x}}{2-2x^2-3x-x\sqrt{5+4x}} - \frac{3+2x-\sqrt{5+4x}}{2-2x^2-3x+x\sqrt{5+4x}} \right)$$

We use that $\frac{a+b}{c-d} - \frac{a-b}{c+d} = \frac{2(bc+ad)}{c^2-d^2}$ and get

$$\frac{2\left(\sqrt{5+4x}(2-2x^2-3x) + (3+2x)x\sqrt{5+4x}\right)}{\sqrt{5+4x}(4+4x^4+9x^2-8x^2-12x+12x^3-5x^2-4x^3)} =$$

$$= \frac{2(2-2x^2-3x+3x+2x^2)}{4x^4+8x^3-4x^2-12x+4} = \frac{1}{x^4+2x^3-x^2-3x+1}.$$

□

Now we could try to determine whether there is any recursive definition of the numbers q_m. And it turns out that a fourth-order linear recursive formula exists indeed. Before we formulate it, we introduce the notation $\sum_k a_k x^k$ for the formal sum from $k=-\infty$ till $k=+\infty$. If we write $\sum_m q_m x^m$ but q_m is only defined for non-negative m, we assume $q_m = 0$ for all negative m.

Proposition 1.4.11. Assume that for the generating function of a sequence $(a_l)_{l\in\mathbb{N}_0}$ and a polynomial $p(x) = \sum_{j=0}^{n} p_j x^j$ holds:

$$\sum_i a_i x^i = \frac{1}{p}.$$

Then $p_0 \neq 0$, the recurrence relations

$$a_0 = \frac{1}{p_0},$$
$$a_l = \frac{1}{p_0} \sum_{j=1}^{n} (-p_j) a_{l-j} \quad \text{for } l \neq 0$$

hold and the recursion is linear of degree $\deg p$.

Proof. Multiplying the given equation by the polynomial in the denominator gives

$$\sum_l \left(\sum_{j=0}^{n} p_j a_{l-j} \right) x^l = 1,$$

so $a_0 p_0 = 1$ and for $l \neq 0$ the equation implies $\sum_{j=0}^{n} p_j a_{l-j} = 0$. □

This simple proposition allows us to read the recurrence for the number of trees with candles having m edges right from the generating function and we get

Corollary 1.4.12. $\forall\, m \geq 1:\quad q_m = 3q_{m-1} + q_{m-2} - 2q_{m-3} - q_{m-4}$ (with $q_{-1} = q_{-2} = q_{-3} = 0$).

Now we derive an explicit formula for the number of trees with candles that have m edges. Before that, we need to mention a well-known result as a lemma.[9]

Lemma 1.4.13. Let K be a field, $p \in K[x], n \in \mathbb{N}^+, p = \prod_{i=1}^{n}(x - r_i)$ with pairwise different $r_i \in K$ for $i = 1, \ldots, n$.

(a) Then

$$\frac{1}{p} = \sum_{i=1}^{n} \frac{a_i}{x - r_i},$$

where

$$a_i = \prod_{\substack{1\leq j\leq n \\ j\neq i}} \frac{1}{r_i - r_j}.$$

[9] Thanks to Robert Israel for driving my attention towards it.

(b) If $\forall\, i \in \{1,...,n\} : r_i \neq 0$, then the m-th coefficient of the formal power series $1/p$ is

$$-\sum_{i=0}^{n} \frac{a_i}{r_i^{m+1}}.$$

Proof.

$$\sum_{i=1}^{n} \frac{a_i}{x-r_i} = \sum_{i=1}^{n} \frac{\left(\prod\limits_{\substack{1\leq j\leq n\\ j\neq i}} \frac{1}{r_i-r_j}\right)\left(\prod\limits_{\substack{1\leq j\leq n\\ j\neq i}} (x-r_j)\right)}{(x-r_i)\prod\limits_{\substack{1\leq j\leq n\\ j\neq i}} (x-r_j)} =$$

$$= \frac{\sum\limits_{i=1}^{n} \prod\limits_{\substack{1\leq j\leq n\\ j\neq i}} \frac{x-r_j}{r_i-r_j}}{\underbrace{\prod\limits_{j=1}^{n}(x-r_j)}_{p}}.$$

The numerator is a polynomial of degree at most $n-1$ and on n positions $r_k \in \{r_1,\ldots,r_n\}$ it takes the value

$$\left(\sum_{\substack{1\leq i\leq n\\ i\neq k}} \prod_{\substack{1\leq j\leq n\\ j\neq i}} \frac{r_k-r_j}{r_i-r_j}\right) + \underbrace{\prod_{\substack{1\leq j\leq n\\ j\neq k}} \frac{r_k-r_j}{r_k-r_j}}_{1} =$$

$$= 1 + \sum_{\substack{1\leq i\leq n\\ i\neq k}} \left(\prod_{\substack{1\leq j\leq n\\ k\neq j\neq i}} \frac{r_k-r_j}{r_i-r_j}\right) \underbrace{\left(\frac{r_k-r_k}{r_i-r_k}\right)}_{0} = 1.$$

So the numerator is a constant polynomial and the first part is proven.
The second part follows from the first part and from

$$\sum_{i=1}^{n} \frac{a_i}{x-r_i} = \sum_{i=1}^{n} -a_i \sum_{m\geq 0} r_i^{-m-1} x^m.$$

□

The proof of the following proposition establishes an explicit formula for the number of candle trees with m edges.

Corollary 1.4.14. (a)

$$q_m = -\sum_{i=0}^{n} \frac{1}{r_i^{m+1} \prod\limits_{\substack{1\le j\le n \\ j\neq i}} (r_i - r_j)},$$

where $r_1 = 1$, $r_2 = \frac{a}{6} + \frac{2}{a} - 1$, $r_3 = -\frac{a}{12} - \frac{1}{a} - 1 + i\sqrt{3}\left(\frac{a}{6} - \frac{2}{a}\right)$, $r_4 = -\frac{a}{12} - \frac{1}{a} - 1 - i\sqrt{3}\left(\frac{a}{6} - \frac{2}{a}\right)$ with $a = (108 + 12\sqrt{69})^{1/3}$.

(b) The growth constant of candle trees is $\frac{1}{|r_2|} = \frac{6a}{a^2-6a+12} \approx 3.0795956....$

Proof. One can verify that the r_i are the zeros of $p = x^4+2x^3-x^2-3x+1$ and apply the previous lemma to get the exact formula. In order to compute the growth constant, we remark that our generating function $1/p$ should have a singularity on the border of the convergence ball. The smallest absolute zero of p is $|r_2| \approx 0.3247...$ □

We have answered all the posed questions concerning trees with candles. What is still to be done is the "bijection" proof of the recurrence relation for q_m. For the chopped trees with candles, it was not hard. For the trees with candles in general, the author doesn't see how to do it at the moment.

1.4.2 Connections among polyominoes, polyedges, polyplets.

We could ask ourselves also about the relationships to other more natural "polyforms" than candle trees. We concentrate only on the two-dimensional case in this section. Let's first define these other polyforms.

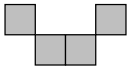
Definition 1.4.15. A *polyplet* is a finite set of 2-dimensional square cells so that:

(a) Each cell has side length 1, its centre has integer coordinates, its sides are parallel to the axes;

(b) If two cells overlap, then their intersection is either one point or one common side;

(c) The union of all cells is path-connected.

Example 1.4.16. Here is an example of a polyplet:



Remark 1.4.17 (Klarner's constant for polyplets and other polyforms)**.** We could redo the proofs from the previous chapters for polyplets. Since each polyplet cell has 8 neighbours (and not 4, as a polyomino), the number of fixed n-plets can be bounded by $\binom{7n}{n} \leq 7\left(\frac{7^7}{6^6}\right)^{n-1}$, so the Klarner's constant for polyplets exists too and is lower than $7^7/6^6$. It is already known, that for polyominoes on any lattice with bounded degree k the growth constant exists and is lower than $(k-1)^{k-1}/(k-2)^{k-2}$, so there is nothing really new.

Remark 1.4.18 (Relations between polyedges and polyplets)**.** The interesting thing about polyplets is that there are at least as many fixed n-plets as the half number of the fixed n-edges! We can see it by constructing a special map[10]

$$f_n \colon PS_2(n) \to \text{fixed n-plets.}$$

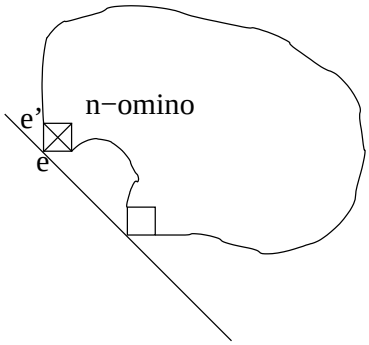
If X is a polyedge, construct a square around each stick of this polyedge so that this stick is its diagonal. The "overlapping" and "connectedness" properties are satisfied. In order to get the right side length as well as the right centres, we scale the polyplet (or the underlying lattice) by the factor $1/\sqrt{2}$ and rotate it by $\pi/4$ and shift it a bit.
In order to see that each fibre has at most two elements, notice that each polyplet-cell has only two diagonals, and choosing a diagonal in a single cell automatically determines the diagonals of its neighbours. So there are at most two different polyedges which can be constructed from a single polyplet.
This proves that asymptotically, there are no less polyplets than polyedges, i.e. the Klarner's constant for polyedges is a lower bound for the Klarner's constant for polyplets.

Remark 1.4.19 (Relations between polyominoes and polyedges)**.** We try to explore the relations between polyominoes and polyedges in order to get more information about the Klarner's constant for polyominoes.

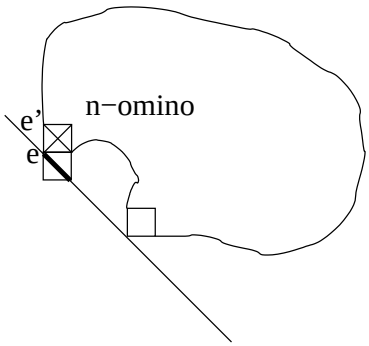
(a) All planar polyominoes are certainly polyplets. More than that, if we take any diagonal of some cell of a (2-dimensional) n-omino, and, starting from it, draw the diagonals of the adjacent cells that touch the former diagonal at the ends, we get an n-edge. So, polyominoes are in the range of the maps f_n. Let $A_n := f_n^{-1}(\text{ fixed } n\text{-ominoes })$. If

[10] I'm indebted to Günter Stretenbrück for this simple construction.

we take a fixed n-omino P, then $|f_n^{-1}(P)| \leq 2$, since there are at most two ways to construct a polyedge from it, since each square has two diagonals. Assume that some polyomino P has only one corresponding polyedge, i.e. $f_n^{-1}(P) = \{E\}$ (it can't be the empty set anyway) for some polyedge E. Let e be the leftmost node in the bottom row of E. We can determine the previous position of e if we imagine the line which is tilt by $\pi/4$ to the left (i.e. which goes in the SE-NW-direction) and let the line approach P from south-west until it touches P. Among the touching points, the one which lies further to north-west, is the point, which is mapped to e. On the other hand, there should also exist another (translationally equivalent) version E' of E which we can get by taking the perpendicular diagonals, here the furthest vertex in the north-west direction is e'. Both e and e' belong to the same cell (otherwise, e.g. if e' were further left, it would be an end-point of a diagonal of a cell, whose second diagonal would touch the line).



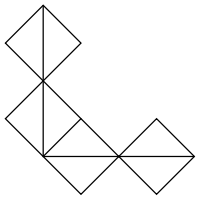
Since both E and E' are translationally equivalent and thus e' lies on e in the standard position, we know that the leftmost node in the bottom row of both E and E' has both vertical and horizontal edges. So, the drawn n-omino looks in fact like this:



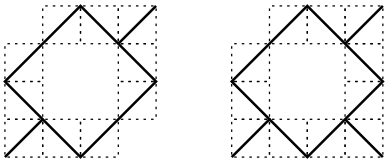
This contradicts to the way the NW-SE-line was drawn.

We have proven that each fixed n-omino has exactly two corresponding fixed n-edges.

(b) Nevertheless not all polyedges have corresponding polyominoes. Even if we allow each cell of the polyomino to have 6 neighbours (instead of 4), for example, if we allow them to have neighbours in the north-west and south-east directions, there would still exist polyedges without the corresponding counterpart, as here:



(c) In order to achieve some results about the Klarner's constant for polyominoes, we could work with the sets A_n, which were defined above, and use the equation $|A_n| = 2s_2(n)$. At the same time, the n-edges from A_n are not classifiable exactly enough through some local topological properties, for example the second polyedge is in A_{11} and the first is not in A_{10}:



But a very big class, contained in A_n, which is much better classifiable through local properties, is

$$B_n := \{p \in PS_2(n) \mid p \text{ contains no “}-\,-\text{”}\}.$$

This means that in any member of B_n, there exists no node of degree two, whose two adjacent edges are both horizontal or both vertical. Let's count its members:

n	1	2	3	4	5
$\lvert B_n\rvert$	2	4	12	38	120

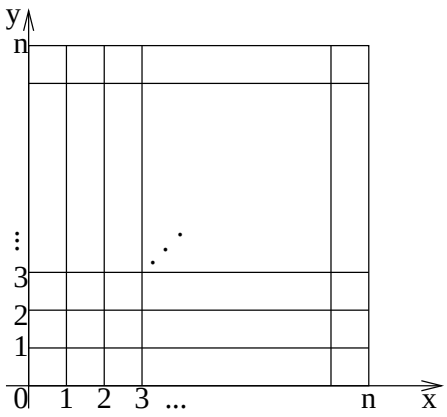
We see that although the growth constant c for the sets B_n is lower than that for polyominoes (and for the sets A_n), B_n has seemingly more members, so for smaller n we get a better lower bound for c. We have to take into account that we get bigger numbers for the same n.

1.4.3 A relaxed problem.

We could ask what will happen if we relax some properties of the polyedges and count the resulting objects.

Definition 1.4.20. A *quasi-polyedge* is an embedding of a simple undirected graph into the 2-dimensional square lattice (remark that connectedness is no more required here). A *free quasi-polyedge* is an equivalence class of quasi-polyedges under the isometry group of an $n \times n$ square.

Now we could reflect on the number of quasi-polyedges with m edges which fit into a $n \times n$ box:



Clearly, since this box has n edges on each side, it has $(n+1)n$ horizontal edges and thus $2(n^2+n)$ edges overall. So the number of quasi-m-edges in an $n \times n$-box is $\binom{2(n^2+n)}{m}$. What happens if we count free quasi-m-edges fitting into the $n \times n$ bounding box, is answered by the special case of Polya's counting theorem. We are going to introduce it.

Theorem 1.4.21 (Special case of Polya's Theorem)**.** Let G be a group of permutations of a finite set D, let R be the set of l colours and $C(D,R)$ be the set of different colourings of elements of D using colours from R.
Then the number of equivalence classes in $C(D,R)$ induced by the operation of G is given by

$$\frac{1}{|G|}\sum_{p\in G} l^{\mathrm{cyc}(p)}$$

where $\mathrm{cyc}(p)$ is the number of cycles of the permutation p in it's unique standard representation.

For the proof see books on enumerative combinatorics (e.g. [15], pages 281–318).

In principle, this theorem suffices to count all quasi-polyedges by letting D be the set of all labelled nodes within the $n \times n$-box, and R be the set $\{\text{present}, \text{absent}\}$ with $l = 2$. Then G contains following permutations of the set of nodes in the $n \times n$-box in the standard position:

- identity
- rotations $\pi/2, -\pi/2$
- flip over centre
- flip over the main diagonal and flip over the second diagonal
- flip upside-down and flip left-right

In order to count the m-quasi-edges separately from, say, $(m+1)$-quasi-edges, we have to proceed more generally than we proceed while computing the total number of all quasi-edges that fit into the $n \times n$ square.

Definition 1.4.22. Let G be a group of permutations. For $\pi \in G$, let $c_i(\pi)$ be the number of cycles of length i in the standard representation of π and let $k := \max\{i \in \mathbb{N} \mid \exists \pi \in G : c_i(\pi) \neq 0\}$ be the length of the longest cycle of all permutations in G. Then the *cycle index* $P_G \in \mathbb{N}_0[x_1, ..., x_k]$ is defined as

$$P_G(x_1, x_2, ..., x_k) = \frac{1}{|G|} \sum_{\pi \in G} x_1^{c_1(\pi)} x_2^{c_2(\pi)} \cdots x_k^{c_k(\pi)}.$$

Computing the cycle index of groups (embedded into some permutation groups) is a hard task in general. Fortunately, our case can be done by hand. Theorem 1.4.21 can be restated as follows

Corollary 1.4.23. Suppose that G is a group of permutations of a finite set D and that $C(D, R)$ is the set of colourings of elements of D using colours in R, a set of l elements. Then the number of inequivalent colourings in $C(D, R)$ is given by $P_G(l, l, ..., l)$.

This corollary gives us the direct answer to the number of all quasi-edges which fit into the $n \times n$ box, by using the cycle index. At the same time, the theorems introduced so far are not enough for count quasi-polyedges with different edge counts separately. To do this, we need to invest a little more work.

Remark and Definition 1.4.24. Let D be a finite set, G a group of permutations on D, let R be the set of l colours. Distinguish the colours by assigning a *weight* $w(r)$ to each colour r. The *weight of a colouring* of D is then defined as the product of weights of the colours assigned to the elements of D. The *inventory* of a set of colourings is the sum of all weights of colourings from this set.
The good thing about colourings is that if they are equivalent (under the operation of some group of permutations G), then they have the same weight. So we can speak about the weight of the whole equivalence class (defined as the weight of the representatives), let's call it the *weight of a pattern*; and we can speak about *pattern inventory*, defined as the sum of weights of the patterns in the set.

Given a pattern inventory as a polynomial in colour weights $r_1, ..., r_l$, we can read the coefficients of monomials $r_1^{q_1} \cdots r_l^{q_l}$. They would give the number of patterns with q_1 colours r_1, ..., q_l colours r_l. That means that the task is to compute such polynomials, which is done by the following theorem.

Theorem 1.4.25 (Polya's Theorem)**.** Suppose that G is a group of permutations on a finite set D and $C(D, R)$ is the collection of all colourings of D using colours from R. If w is a weight assignment on R, then the pattern inventory of colourings in $C(D, R)$ is given by

$$P_G(\sum_{r\in R} w(r), \sum_{r\in R} w^2(r), ..., \sum_{r\in R} w^k(r)).$$

Now let's go over to our initial problem. First we compute the cycle index.

Theorem 1.4.26. The cycle index of the $n \times n$-box symmetry group, embedded into $\mathrm{Sym}_{2(n^2+n)}$, is

$$\frac{1}{8}\left(x_1^{2(n^2+n)} + 3x_2^{n^2+n} + 2x_4^{(n^2+n)/2} + 2\begin{cases} x_1^n x_2^{n^2+n/2}, & \text{if } n \text{ is even} \\ x_2^{n^2+n}, & \text{if } n \text{ is odd} \end{cases}\right).$$

Proof. Since each of the permutations in the group (which were listed above) has the order 1,2 or 4, we have $k = 4$ variables. We'll examine how much each permutation contributes to the index.

(a) Identity. All cycles have length 1. Contribution $x_1^{2(n^2+n)}$.

(b) Rotation $\pi/2$. Each edge returns to its place after exactly four rotations. No edge gets to its place earlier. Contribution $x_4^{(n^2+n)/2}$.

(c) Rotation $-\pi/2$. The same.

(d) Flip over the centre (the same as rotate by π). Each edge has order two. Contribution $x_2^{n^2+n}$.

(e) Flip over the main diagonal. Each edge has order 2. Contribution $x_2^{n^2+n}$.

(f) Flip over the second diagonal. The same applies.

(g) Flip left-right.
If n is odd, then there are $n+1$ edges which remain invariant: they intersect the symmetry line. All the other edges (there are $2n^2+2n-n-1$ $= 2n^2+n-1$) have the order two and the contribution is $x_1^{n+1}x_2^{n^2+(n-1)/2}$.
If n is even, then there are n edges on the symmetry line which are mapped to themselves, other $2n^2+n$ edges have the order two. Contribution $x_1^n x_2^{n^2+n/2}$.

(h) Flip upside-down. The same holds.

Let's sum it up:

$$|G|P_G(x_1,x_2,x_3,x_4) =$$

$$= x_1^{2(n^2+n)} + 3x_2^{n^2+n} + 2x_4^{(n^2+n)/2} + 2\begin{cases} x_1^n x_2^{n^2+n/2}, & \text{if } n \text{ is even} \\ x_1^{n+1}x_2^{n^2+(n-1)/2}, & \text{if } n \text{ is odd} \end{cases}$$

□

Corollary 1.4.27. The number of free quasi-polyedges in an $n \times n$-box is

$$\frac{1}{8}\left(4^{n^2+n} + 3\cdot 2^{n^2+n} + 2\cdot 2^{(n^2+n)/2} + 2\begin{cases} 2^{n^2+3n/2}, & \text{if } n \text{ is even} \\ 2^{n^2+(3n+1)/2}, & \text{if } n \text{ is odd} \end{cases}\right).$$

Proof. Follows from 1.4.23 and 1.4.26 with number of colours $l = 2$. □

We observe that the contribution of permutations other than identity is asymptotically meaningless, the asymptotic behaviour is dominated by $2^{2(n^2+n)}$. That means that asymptotic results of "free" quasi-polyedges and "normal" quasi-polyedges are equal.

Corollary 1.4.28. The number of free quasi-m-edges in an $n \times n$-box is

$$\frac{1}{8}\left(\binom{2(n^2+n)}{m} + 3\begin{Bmatrix} 0, & \text{if } m \text{ odd} \\ \binom{n^2+n}{m/2}, & \text{if } m \text{ even} \end{Bmatrix}\right) + \frac{1}{4}\begin{Bmatrix} 0, & \text{if } m \neq 0 \mod 4 \\ \binom{(n^2+n)/2}{m/4}, & \text{if } m = 0 \mod 4 \end{Bmatrix} + \frac{1}{4}\begin{Bmatrix} \sum\limits_{\substack{0\leq t\leq n \\ m-t \text{ even}}} \binom{n}{t}\binom{n^2+n/2}{(m-t)/2}, & \text{if } n \text{ even} \\ \sum\limits_{\substack{0\leq t\leq n+1 \\ m-t \text{ even}}} \binom{n+1}{t}\binom{n^2+(n-1)/2}{(m-t)/2}, & \text{if } n \text{ odd} \end{Bmatrix}$$

(with the convention that $\binom{x}{y} = 0$ for $y < 0$ or $y > x$).

Proof. We assign two weights: a for present edges and b for absent edges. Then we compute the cycle count from Polya's theorem as a polynomial in standard form and read the coefficient of $a^m b^{2(n^2+n)-m}$.
The polynomial is

$$\frac{1}{8}\left((a+b)^{2(n^2+n)} + 3(a^2+b^2)^{n^2+n} + 2(a^4+b^4)^{(n^2+n)/2}\right) + \frac{1}{4}\begin{Bmatrix} (a+b)^n(a^2+b^2)^{n^2+n/2}, & \text{if } n \text{ is even} \\ (a+b)^{n+1}(a^2+b^2)^{n^2+(n-1)/2}, & \text{if } n \text{ is odd} \end{Bmatrix}.$$

The coefficients in front of $a^m b^{2(n^2+n)-m}$ sum up to our result. □

Chapter 2

Game "Digit" and connected problems

In this chapter we'll talk about the game "Digit". We present different models, talk about the representation of the game in computer's memory, explore the complexity of our game. Furthermore, we discuss an implementation of a 2-player version.

2.1 Rules

There are several possible variants and definitions of the game. Some of them proved to be interesting, while the others turned out to be dull.

Definition 2.1.1. • Let's consider the following game between p players who share a common board, which is our 2-dimensional infinite plane lattice (almost the same goes for k-dimensional case). Before the game starts, each player possesses a hand of c cards, c is some constant natural number. Each card has a single 2-dimensional m-edge drawn on it. On the board we also have a single m-edge, which we imagine composed of matches. All the polyedges present on the cards and on the board are drawn from the set of all m-edges randomly (suppose that m is big enough so that the set contains at least c different polyedges).

The players play in turn. To do that, we assume that for each player a unique next player is known. (We can imagine the players sitting on the circle line, so that the next player is the one in the clockwise-order). Each move is done the following way: the player who is on

turn takes a match from the board and puts in onto another place so that the resulting graph on the board is still a valid m-edge, i.e. connected. Then the player throws out those of his cards which coincide with the resulting figure on the board. If he has no more cards, he wins. Then the other players (in the given order) throw out their cards which coincide with the polyedge on the board. The first player who gets rid of all of his cards, wins.

- There are different questions which the former definition leaves open. The answers to these questions represent possible variations of the game.
  If the players see each other's cards, we call the game variant *open*. If they don't, we call it *closed*.
  Repetitions of the game state may be treated differently. A game *state* includes the position on the board and may or may not include the cards of all players. Repetitions may either be allowed or prohibited or the first player who moves into a repeated state loses or, as in chess, after a state has been repeated a predetermined number of times, the game is considered a draw. Passes may either be allowed or prohibited or after two consecutive passes the game ends, after which the players with the least number of cards win and all the others lose.
  The coincidence of cards on the hand with the card on the board may either be checked the "fixed" way, that means each polyedge is drawn, say, in standard position, and we can tell where its upper, lower, left and right side is. Or it can be "free", that means that we can rotate and flip two polyedges (in our mind) to determine whether they coincide in any "rotated" or "flipped" way. Or it can be done "isomorphic", that means that two polyedges are equal if they are isomorphic as graphs. We don't treat the isomorphic case and concentrate ourselves only on the first two cases.
  Furthermore, the card on the board and the cards on the hands may either be all different at the beginning or they are allowed to coincide (at least partially).

2.2 Complexity results.

Now we'll try to identify some problems connected with this game. Assume that all the polyedges we talk about are planar.

Definition and Remark 2.2.1. (a) For a natural number m, define a

fixed (free) transition graph for m-edges $TT(m)$ ($TS(m)$) as an undirected labelled graph with the label set of all fixed (or free, dependent on the game variant) m-edges. Two vertices are connected by an edge in this graph, if the underlying polyedges coincide up to a single edge, i.e. if we can get one polyedge from the other by removing an edge from one place and putting it onto another place.

(b) The complexity of our game can be measured in terms of this graph. As follows from the previous chapter, the number of nodes grows asymptotically as $\approx 5.20^m$, i.e. it's not possible to store the transition graph even for relatively small m "as is", e.g. as an incidence matrix or as an adjacency list. The number of edges cannot be determined trivially.

Lemma 2.2.2 (Upper bound on the number of transitions)**.** For the fixed transition graph $TT(m)$, each fixed (n, m)-edge is connected to at most $2mn$ other fixed polyedges in the transition graph. So each m-edge is connected to at most $2(m + 1)m$ other polyedges.

Proof. First of all we count the number of free positions in a polyedge, where we can put an additional edge.
For $n = 1$ we have 4 free positions, for $n = 2$ we have 6, and for $n = 3$ we have 8 different free positions in a polyedge.
If we add a node to a polyedge, we have to connect it to the polyedge by an edge. By placing an edge onto one of free positions, we reduce their number by one, the edge itself carries at most three new. So adding a node increases the number of the free positions up to maximally two.
Since we can also change the number of edges without adding nodes (which doesn't increase the number of free positions), we state, that the number of free positions of an (n, m)-edge is at most $2n + 2$. If we have an (n, m)-edge and remove an edge, we have two cases. In the first one, the number of nodes is decreased, then we have at most $2(n - 1) + 2 = 2n$ free positions to put the edge to, minus the one place where the edge has been before, thus giving an upper border of $2n - 1$ choices. In the second one, no node is removed, so we have exactly two places of the new polyedge where the edge cannot be put into, thus giving us at most $2n + 2 - 2 = 2n$ choices.
So the total number of possibilities to move any edge in an (n, m)-edge is $\leq 2mn$. Theorem (1.3.8) gives $2m(m + 1)$. □

It's harder to obtain a tight lower bound, since removing an edge might destroy connectedness, thus diminishing the number of possible places, where the edge can be placed to. Nevertheless the previous lemma gives us:

Corollary 2.2.3. Storing the transition graph for m-edges takes $O(m^3 \cdot pt_2(m))$ place, if we store edge and node lists "as is", assuming that storing a label (polyedge) takes $O(m)$ place.

Remark 2.2.4. Here we count the total number of edges in the transition graphs (repeating the node counts we already know) for free and fixed kinds:

m	free, nodes	free, edges	fixed, nodes	fixed, edges
1	1	0	2	1
2	2	1	6	14
3	5	8	22	142
4	16	71	88	934
5	55	529	372	5552
6	222	3424	1628	31906
7	950	20628	7312	180928
8	4265	119872	33466	1017830
9	19591	680189	155446	not counted

Now we simplify the game and consider the following problem: If a single player moves and all the others don't disturb him (e.g. always pass instead of moving), what is the minimal number of steps he needs to win the game? To see that this is non-trivial, we reformulate the problem as a graph problem.

Definition and Remark 2.2.5. We introduce a functional and a decision variant.

(a) Suppose that $G = (V, E)$ is a simple connected undirected graph, $s \in V$ and $W \subset V$. Find a path p in G of minimal length, so that p starts in s and goes through each node from W at least once.

(b) MIN-SUBSET-STARTING-PATH[1] $= \{\langle V, E, W, v, k\rangle \mid G = (V, E)$ is a connected undirected graph, $W \subset V$, $k \in \mathbb{N}$, $v \in V$, there exists a path in G which starts in v and goes at least once through at least all nodes in $W \subset V$, so that its length $\leq k\}$.

In both definitions the minimal path is also allowed to go through the nodes in $V \setminus W$.
We can "translate" these decision problems into the game language, if we

[1]The task of finding new names for NP-hard problems is getting hard itself: the author tried several naming variants and was greatly astonished to notice that they were all already used to describe other graph problems. Neither guarantee nor warranty is provided that the chosen name is otherwise unused.

think of (V, E) as of a transition graph, W contains cards from a hand of the player who moves, v is the polyedge on the board. Since our transition graph is a very special structure and $|W|$ is in practice relatively small compared to $|V|$, our special problem could actually be easier than the general problem, but intuitively speaking, even a polynomial-time algorithm operating on the whole graph wouldn't help much, since the length of our input, namely the size of the transition graph, is already exponential in m. Of course, since the whole graph cannot be stored in general, an implementation has to generate it locally on need.

Theorem 2.2.6. The decision problem MIN-SUBSET-STARTING-PATH is NP-complete.

Proof. First we notice that the language is in NP, since we can construct the prover which accepts a path as a proof and checks, whether it's in the graph, whether it touches all the nodes in the given subset and starts with the given node and computes its length - and all this in polynomial time.
In order to prove NP-hardness, we construct a polynomial reduction from the decision problem UNDIR-HAMCYCLE. The last one is a problem whether an undirected graph possesses a Hamiltonian cycle. By convention, the trivial graph on a single node is considered to posses a Hamiltonian cycle, but the connected graph on two nodes is not. We assume that some fixed alphabet Σ with at least two symbols and a reasonable encoding function are given. We give a polynomial-time reduction $f : \Sigma^* \to \Sigma^*$ with $\forall\, x \in \Sigma^*$:

$$x \in \text{UNDIR-HAMCYCLE} \Leftrightarrow f(x) \in \text{MIN-SUBSET-STARTING-PATH}.$$

Let $x \in \Sigma^*$ be an input word. If x is not a valid encoding of an undirected graph $G = (V, E)$, then some fixed word $y \notin$ MIN-SUBSET-STARTING-PATH is returned. Otherwise construct another graph $G' = (V', E')$ from G as follows:

(a) If $|V| = 1$, return some fixed word $w \in$ MIN-SUBSET-STARTING-PATH. If G is a connected graph on two nodes, return y. Otherwise:

(b) Choose a single node $u \in V$ and duplicate it with its edges and call the new node u';

(c) Insert two new nodes v, v' and two new edges $\{u, v\}, \{u', v'\}$;

(d) Return $\langle V', E', V', v, |V| + 2\rangle$.

This transformation is polynomial-time.

Let $x \in \Sigma^*$. We show
$x \in$ UNDIR-HAMCYCLE $\Leftrightarrow f(x) \in$ MIN-SUBSET-STARTING-PATH.

"$\Rightarrow$". Let $x = \langle G \rangle$, $G = (V, E)$ is an undirected graph with a Hamiltonian cycle h. If there is only one node, then $f(x)$ is in MIN-SUBSET-STARTING-PATH per construction. Otherwise G is connected (via h), h is a cycle with at least three edges and h passes through u. In the transformed graph, there exists a path from u to u' of length $|V|$. So in the transformed graph there exists a path from v to v' of length $|V| + 2$.

"$\Leftarrow$". Let $x \in \Sigma^*$, $f(x) \in$ MIN-SUBSET-STARTING-PATH. If $x = w$, then we already have $x \in$UNDIR-HAMCYCLE, otherwise $x = \langle G \rangle$ for some graph $G = (V, E)$ with at least two nodes, $f(x) = \langle V', E', V', v, |V| + 2 \rangle$ for V', E', v as in algorithm. Let h' be a path which starts in v and goes at least once through all nodes in V'. Then h' goes also through v'. Since $\deg v' = 1$, the path ends in v'. The only nodes connected to v and v' are u and u', respectively: $h' = (v, u, ..., u', v')$. Throwing out the first and the last node we get a shorter path $h = (u, ..., u')$, which touches neither v nor v' (because $\deg v = \deg v' = 1$). It has length $\leq |V|$ and goes through $|V| + 1$ nodes, so it is simple (i.e. without cycles). Let u'' be the neighbour of u' on this path. So G has a path from u to u'' which goes through each node of G exactly once. Since $\{u, u''\} \in E$, we can append u also to the end of the path: $(u, ..., u'', u)$. This is our Hamiltonian cycle. It's longer that two, since otherwise G would be a connected graph on two nodes which could not lead to $f(x)$ by construction. □

At the same time there exists a non-trivial algorithm for solving this problem. It requires exponential time, so it can be applied only to small graphs.

Algorithm 2.2.7 (Solve functional MIN-SUBSET-STARTING-PATH)**.** We present a procedure[2] which works for small W and V. (All variable names have the same meaning as in 2.2.5a.) W.l.o.g. let the starting point s be in W.

For each $v \in V, k \in \mathbb{N}$ we maintain the set

[2]I'm indebted to Raimund Seidel for this algorithm.

$X(v,k) = \{(U,l) \in W \times \mathbb{N} \mid \exists$ path p of length $\leq k : p \cap W = U$ and $p = (s,...,l,v)\}$.

Here we denote by $p \cap W$ the set of all nodes from W which lie on p.
We construct these sets inductively as follows:
$\underline{k = 1}$: $\forall\, v \in V$:

$$X(v,1) = \begin{cases} \{(\{s\},s)\}, & \text{if } v \text{ is a neighbour of } s \text{ and } v \notin W; \\ \{(\{s,v\},s)\}, & \text{if } v \text{ is a neighbour of } s \text{ and } v \in W; \\ \emptyset, & \text{if } v \text{ is not a neighbour of } s. \end{cases}$$

$\underline{k > 1}$: $\forall\, v \in V$:

$$X(v,k{+}1) = \begin{cases} \{(U \cup \{v\}, v') \mid \{v,v'\} \in E \text{ and } \exists\, l \in V : (U,l) \in X(v',k)\}, & v \in W \\ \{(U,v') \mid \{v,v'\} \in E \text{ and } \exists\, l \in V : (U,l) \in X(v',k)\}, & v \notin W \end{cases}$$

We work until we get $(W,l) \in X(s,k)$ for some $k \in \mathbb{N}, l \in V$. Then there exists a path which starts in s and takes k steps until all W is touched. We can reconstruct the path by going via the second component l to that node and checking $X(l,k-1)$, proceeding recursively until we reach $X(s',1)$ where s' is some neighbour of s.
Space used: $O(V \cdot 2^{|W|}) + O$(place to store the graph);
Time used: $O(k \cdot |E| \cdot 2^{|W|})$.

2.3 Implementation.

Here we discuss problems which are necessary to be solved in order to create a playable version of the game. At last we talk about coding and user interface.

The author implemented two versions of the game:

(a) An arbitrary number of computers that play randomly against an arbitrary number of human players.

(b) Computer against computer or computer against a human.

Both variants are "open-cards", repetitions are allowed, each variant can be run either with the "fixed" or with the "free" way of comparing polyedges. As soon as the polyedge on the board is changed, the players are allowed to throw out their cards, starting with the player who made the move. As

soon as some player has no more cards, all the others don't come to turn. The first variant served as a test platform for the interface and is interesting only from this point of view. The second variant tested the algorithm.

2.3.1 Key problems and algorithms.

The implemented algorithm is an alpha-beta search with fixed depth, at which a heuristic function is called. It turns out, that it's quite difficult to find a good one. One possible attempt is the "shortest-distance-heuristic", which has its roots in approximation algorithms for the travelling-salesman problem. In a simple TSP heuristic algorithm, the town with the shortest distance from the current location is targeted. It's the simplest way for the salesman to determine, which way he goes next.

In our situation, the polyedge on the board is the starting town and the other towns are the polyedges on the hand. The distance is measured in the minimal number of edge moves which have to be undertaken in order to transfer one polyedge into another. It corresponds with the graph distance between two polyedges in the transition graph. So the problem is how to determine this distance, or at least how to find an approximation algorithm for it. We assume that both polyedges have the same edge count.

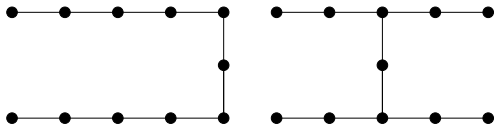
Algorithm 2.3.1 (Computing approximation from below on the distance of two polyedges)**.** Let P_1, P_2 be two input polyedges ("stiff", i.e. with a determined location of the lattice), not necessarily in standard position. Put one on top of the other. Then some edges of P_1 may coincide with some edges of P_2 and some edges of P_1 may stay alone and some edges of P_2 may stay alone. Subtract the number of coinciding edges from the total edge count to get the number of all edges that don't coincide and divide by two.

The intention is that if we take an edge e_1 from P_1 and an edge e_2 from P_2 so that neither e_1 nor e_2 have partners from the other polyedge, as P_1 and P_2 are laid on each other, then we can make them partners by moving, for instance, e_1 in P_1 to the place where e_2 is placed in another polyedge. Each move brings an edge onto "its place", thus making P_1 and P_2 more "similar", so this algorithm computes a lower bound on the distance.

The extension from these "stiff" polyedges to polyedges modulo translation (fixed polyedges) is not problematic - shift one polyedge along another in both directions and each time determine the number of coinciding edges.

As soon as the maximum of coinciding edges is reached, we subtract it from the total edge count and divide by two. If we want to handle free polyedges, we have to compare all the eight images of the first polyedge with the second in a "translation-invariant" way.

The major problem is that only a lower bound on the distance is computed, although it may coincide (and coincides in most cases) with the actual distance. Here is an example where the distance is three and the algorithm computes two:



Nevertheless, this approximation turns out to work quite well.

Definition and Remark 2.3.2. We define the *nearest hand distance* as the minimal number of consecutive moves a single player has to make in order to get rid of a card from his hand. In other words, it is $\mathrm{dist}(b, H) := \min\{\mathrm{dist}(b, c) | c \in H\}$ where b is the polyedge on the board and $\mathrm{dist}(b, c)$ denotes the graph distance between the polyedges b and c in the transition graph and H is the set of cards in the hand.

The implementation of the heuristic function is given below. The situation is before our own move; better positions get higher values. Zero means neutral.

Algorithm 2.3.3 (Heuristic function)**.** Let M be the hand of "our" player, H be the hand of the opponent.

if H is empty **then**
  return $-\infty$ (the opponent has just moved and cleared first his hand)
else
  if M is empty **then**
    return $+\infty$ (the opponent has just moved and cleared our hand)
  else
    return $\mathrm{dist}(b, H) - \mathrm{dist}(b, M)$ (both hands are not empty)
  end if
end if

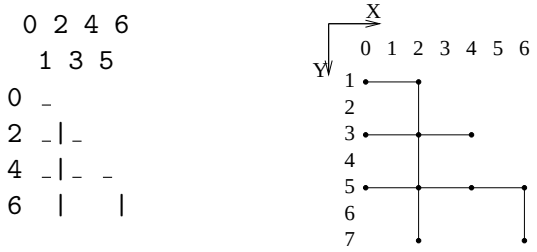
The intuition tells us, that during a game with allowed repetitions, a player can always prevent the state where the opponent gets rid of all his

cards; in order to do this, he just has to undo each move of the opponent. In fact, the described algorithm actually does that: if we run two instances of this algorithm against each other, either one of them wins in the first moves or they both run forever, undoing each other's moves. So this version of a game turns out to be quite dull. At the same, it's quite hard to find a good heuristic for the version with prohibited repetitions of game states (maybe, except for the game state we have at the start, since it is not a result of a move, but of a random card distribution).
In principle, the same heuristic could do the job, where the distance between cards is the distance between the underlying polyedges in the transition graph, but it gets harder to compute now. In order to see that, remark, that each move changes the transition graph, removing nodes from it which could lie on a shortest path between two other nodes. So making too many moves could make some distances longer or disconnect the transition graph at all.

2.3.2 Concrete coding issues.

During programming, the author decided to leave the user interface as simple a possible, and to concentrate on the inner problems and computer player algorithms. The game was implemented under Unix, using the GNU C++ 3 compiler. It can be run under each console and doesn't require any X window system. So the user interface needs getting used to. The program source can be downloaded from the homepage of the Computational Geometry chair of the Saarland university: http://www-tcs.cs.uni-sb.de/alexmalk/soft2.tar.gz

Remark 2.3.4 (Polyedge representation on the screen and addressing individual edges)**.** Each card has a polyedge drawn on it. In order to move different edges on the polyedge on the board from one place onto another, we address them by coordinates. (The coordinate system here is only for the user's commodity, it has nothing to do with the coordinate system of the underlying lattice.) To the left we see an example of a polyedge, drawn in its coordinate system by the program, and to the right we see the same "finely-drawn" polyedge.



Single edges are addressed through their coordinates. The edge on the top of the polyedge from the example has coordinates $(1, 1)$, the second edge, counting from above, has coordinates $(2, 2)$, the leftmost edge in the bottom row has coordinates $(2, 6)$. The move "`1,1-5,3`" , which moves the edge on $(1, 1)$ to the position $(5, 3)$, gives the following result:

```
 0 2 4 6
  1 3 5
0
2 _|_ _
4 _|_ _
6  |   |
```

Remark 2.3.5 (Starting games)**.** In order to start the game itself, we use the program `comp_man`. Here is how to use it:

```
comp_man [{H|C} {H|C} <edges> <cards>]
where
H means human, C means computer, they play in the given order.
At least one computer should be involved.
<edges> is the number of edges of each figure,
<cards> is the number of cards each player possesses.
By default the human starts and plays against the computer.
```

So in order to start the game with a single computer against a single human, where a computer starts and where each side possesses 2 cards à 3 edges, we type:

```
comp_man C H 3 2
```

The cards of both sides well be displayed and you'll be asked to type in a move. Then you see what changed on the "play-field" through your move, the computer does its own move, you'll be asked to give in your move again, etc. Repetitions of game states mean that the best move for a computer is to go back to the previous position, so (since there is no draw in this versions)

the only possibility you have is to terminate the program or play forever.

Another program you can try is

```
try_game [<random_players> <human_players> <edges> <cards>]
where
<random players> is the number of random players involved,
<human_players> is the number of human players involved,
<edges> is the number of edges of each figure,
<cards> is the number of cards each player possesses.
```

For instance, "`try_game 1 1 5 2`" gives you a chance to simulate a game against a drunken person, where each has two cards à 5 edges. Started without arguments, "`try_game`" lets a single drunken player play against itself with three cards à three edges. Thus you can simulate a random walk through the transition graph, which continues until a random subset of nodes has been passed through.

Summary and Discussion

In this work, we explored the borders of the very captivating world of animals.

We looked at such species as site animals (also known as polyominoes) and paid a special attention to bond animals (polyedges). We were able to observe and prove that the second behaves almost in the same way as the first, at least on the (k-dimensional) surface. They both grow asymptotically constantly, although at different rates, we were even able to estimate their growth rate and give provable upper and lower bounds on it. We counted bond animals by high-tech methods (and it turned out that they are so far from extinction, that it's not possible to count all of them with size 100 edges by current methods).

We tried to observe subspecies of bond animals in hope to find out more about them. We found candle trees and were happy to count them with relatively simple methods.

We also looked at some other animals (polyplets) in attempt to find some relationships among polyplets, site animals and bond animals and succeeded.

Next we have relaxed the bond animals by allowing them to be unconnected and tried to determine their number in restricted regions ($n \times n$ square).

At last we played with bond animals by involving them into the so-called "Digit game". The game turned out to be too hard for a human or even a computer to be solved (NP-completeness result), but nevertheless we programmed a computer to play with such animals against another human or against another computer.

Still there exist problems to be solved.

Nothing is known about the generating function of free or fixed polyedges. It's not known how to compute their number in polynomial time. Approximation algorithms were not found either. The same holds for the growth constant. Even its first digit is not provably known. A polynomial-time approximation algorithm is unknown either. Even a super-polynomial-time approximation from above is missing. Sub-dominant asymptotic behaviour is uncertain. General framework uniting different polyforms is still absent (possibly, it cannot be created meaningfully at all). What happens if we allow "parallel" edges? How about hexagonal and triangular lattices? The list may be extended.

If we speak about the game "Digit", there is little to be done by the future generations. Changing the underlying lattice has not been covered. Determining exact graph distance on transition graphs or on their subgraphs is nontrivial. Graphic user interface could make the game more attractive. There exists a variety of unsolved problems in this area, too.

Appendix A

Series listing.

- Fixed planar polyominoes, see [11]. This result however, was taken from Iwan Jensen's page
  http://www.ms.unimelb.edu.au/ iwan/animals/series/square.site.ser
  1 1
  2 2
  3 6
  4 19
  5 63
  6 216
  7 760
  8 2725
  9 9910
  10 36446
  11 135268
  12 505861
  13 1903890
  14 7204874
  15 27394666
  16 104592937
  17 400795844
  18 1540820542
  19 5940738676
  20 22964779660
  21 88983512783
  22 345532572678
  23 1344372335524

24 5239988770268
25 20457802016011
26 79992676367108
27 313224032098244
28 1228088671826973
29 4820975409710116
30 18946775782611174
31 74541651404935148
32 293560133910477776
33 1157186142148293638
34 4565553929115769162
35 18027932215016128134
36 71242712815411950635
37 281746550485032531911
38 1115021869572604692100
39 4415695134978868448596
40 1749811117283831298 2542
41 69381900728932743048483
42 275265412856343074274146
43 1092687308874612006972082
44 4339784013643393384603906
45 17244800728846724289191074
46 685577626663451654101687 38
47 272680844424943840614538634
48 1085035285182087705685323738
49 43193315093445654875552706 60
50 17201460881287871798942420736
51 68530413174845561618160604928
52 27312666001651914329332002 6256
53 10889336855593503008200959900 30
54 43429974696239331559427538990 00
55 17326987021737904384935434351490
56 69150714562532896936574425480218

- Free planar polyominoes, see [10]:

  1, 1, 2, 5, 12, 35, 108, 369, 1285, 4655, 17073, 63600, 238591, 901971, 3426576, 13079255, 50107909, 192622052, 742624232, 2870671950, 11123060678, 43191857688, 168047007728, 654999700403, 2557227044764, 9999088822075, 39153010938487, 153511100594603

- Free planar polyedges, see [8]:
  1, 2, 5, 16, 55, 222, 950, 4265, 19591, 91678, 434005, 2073783, 9979772, 48315186, 235088794, 1148891118, 5636168859

- Fixed planar polyedges. The starting terms were counted by the author first via the simple extension method, then it was abandoned for the sake of the rooted method. Brendan Owen's program (used to count other kinds of polyforms) was used to compare the results until $m = 16$, it also gives a good support for the results till $m = 19$ (unfortunately, at the time this paper is being written, his program has integer overflow at m=16, so the results coincide only modulo 2^{32}). The author managed to reduce the counting time up to a factor of 5, but nevertheless 21 needed two weeks on a 2.4 MHz Intel Linux machine to complete:
  1 2
  2 6
  3 22
  4 88
  5 372
  6 1628
  7 7312
  8 33466
  9 155446
  10 730534
  11 3466170
  12 16576874
  13 79810756
  14 386458826
  15 1880580352
  16 9190830700
  17 45088727820
  18 221945045488
  19 1095798917674
  20 5424898610958
  21 26922433371778

Appendix B

Notation and abbreviations.

- $\exists!$ ... means "there exists exactly one ..."
- $\wedge$ means "and", $\vee$ means "or"
- Natural numbers and its subsets: $\mathbb{N}^+ = \mathbb{N} = \{1, 2, 3, ...\} \not\ni 0$, $0 \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $\mathbb{N}_k = \{1, ..., k\} = [1, k] \cap \mathbb{N}$ for $k \geq 1$
- If X is a set, $\mathfrak{P}(X)$ denotes the power set of X and $\mathfrak{P}_i(X)$ denotes the set of all subsets of X with i elements.
- For a set X let id_X be the identity map. For a map $f : X \to X$, define inductively the powers of f through $f^0(x) := x, f^{i+1}(x) := f(f^i(x))$
- A (labelled) graph is a pair (V, E) with vertex set V (which is by default finite, if nothing else mentioned) and edge set E. If the graph is undirected, then $E \subset \mathfrak{P}_2(V)$; if the graph is directed, then $E \subset V \times V \setminus \{(v, v) \mid v \in V\}$. We write $d(v, w)$ for the graph distance between two nodes (length of the shortest path from v to w). A path is a tuple $(v_0, ..., v_k) \in V^{1+k}$ for $k \geq 1$ so that $\{v_i, v_{i+1}\} \in E$ for $0 \leq i < k$. A simple path contains only pairwise different nodes. A cycle is a path $(v_0, ..., v_k, v_0)$ with $k \geq 2$ so that $v_0, ..., v_k$ are pairwise different. A graph is called regular if the degree of each node is constant. The abbreviations BFS and DFS refer to the breadth- and depth-first-searches.
- Let's call $\mathfrak{G}(X)$ the set of all undirected graphs on the node set X. A map
$$F : \mathfrak{G}(X) \to \mathfrak{G}(\mathbb{Z}^k)$$

is called an embedding map if there exists a one-to-one map

$$H_F : X \to \mathbb{Z}^k,$$

so that $F((X, E)) = (H_F(X), \{(H_F(v), H_F(w)) \mid \{v, w\} \in E\})$.
An embedding of a graph G into $\mathbb{Z}^k$ is $F(G)$ for some embedding map F.

- Let $(X, <)$ be an ordered set, $k \in \mathbb{N}$. The lexicographic order on the set X^k is given by:

$$(x_1, ..., x_n) <_{lex} (y_1, ..., y_n) \text{ iff}$$

$$\exists\, i \in \mathbb{N}_k : \forall\, j < i : y_j = x_j \wedge x_i < y_i$$

  The lexicographic largest and the lexicographic smallest elements of a finite nonempty set $A \subset X^k$ are denoted by $\mathrm{lmax}(A)$ and $\mathrm{lmin}(A)$.

- For $k \in \mathbb{N}$ and a ring R with 1 define the standard unit vectors e_i of R^k as $e_i = (0, ..., 0, 1, 0, ..., 0)$ where the 1 stands on the ith position.

- Sym_k is the permutation group of $\mathbb{N}_k$, $\mathrm{Sym}(X)$ is the permutation group of the set X.

- $\mathrm{Mat}(m \times n, X)$ is the set of all matrices with m rows and n columns over the set X.

- During formulation of decision problems, we denote the encodings of graphs, paths, lists, Turing machines et cetera via the triangle brackets "$\langle$" and "$\rangle$", though it doesn't matter what alphabet Σ and what encoding exactly we use. We should have $|\Sigma| \geq 2$ and an encoding should preserve the asymptotic lengths of the encoded objects. Example: $\langle G \rangle$ is an encoding of the graph G.

- w.l.o.g. is an abbreviation for "without loss of generality"

Bibliography

[1] Amdeberhan Tewodros: *Solution to Problem #10663 Proposed by E. Deutch*, http://www.math.tulane.edu/~tamdeberhan/solutions/10663.PDF

[2] Solomon W. Golomb, David A. Klarner: *Polyominoes*, Handbook of Discrete and Computational Geometry, 2nd edition, 2004 by Chapman & Hall/CRC, 331–352.

[3] Kevin Gong: *Applying Parallel Programming to the Polyomino Problem*, UC Berkeley CS 199 Fall 1991, http://www.kevingong.com/Polyominoes/ParallelPoly.html.

[4] I. Jensen: *Counting polyominoes: A parallel implementation for cluster computing*, Lecture Notes in Computer Science, 2659, 203–212 (2003).

[5] M. Jünger, G. Reinelt, G. Rinaldi: *The Traveling Salesman Problem*, in Ball, Magnanti, Monma and Nemhauser (eds.), Handbook on Operations Research and the Management Sciences North Holland Press, 225–330.

[6] D. A. Klarner: *Cell growth problems*, Van J. Math. 19 (1967), 851–863.

[7] D. A. Klarner, R. Rivest: *A procedure for improving the upper bound for the number of n-ominoes*, Canad. J. Math. 25 (1973), 585–602.

[8] On-Line Encyclopedia of Integer Sequences, sequence A019988: *Number of ways of embedding a connected graph with n edges in the square lattice*, http://oeis.org/A019988.

[9] On-Line Encyclopedia of Integer Sequences, sequence A096267: *Number of fixed polyedges with n edges*, http://oeis.org/A096267.

[10] On-Line Encyclopedia of Integer Sequences, sequence A000105: *Number of polyominoes (or square animals) with n cells*, http://oeis.org/A000105.

[11] On-Line Encyclopedia of Integer Sequences, sequence A001168: *Number of fixed polyominoes with n cells*, http://oeis.org/A001168.

[12] On-Line Encyclopedia of Integer Sequences, sequence A000129: *Pell Numbers*, http://oeis.org/A000129.

[13] B. M. I. Rands, D. J. A. Welsh: *Animals, Trees and Renewal Sequences*, IMA J. Appl. Math. (1981) 27, 1–17.

[14] B. M. I. Rands, D. J. A. Welsh: *Animals, Trees and Renewal Sequences: Corrigendum*, IMA J. Appl. Math. (1982) 28, 107.

[15] Fred S. Roberts: *Applied Combinatorics*, Copyright 1984 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 281–318 and 42.

[16] S. G. Whittington and C. E. Soteros: *Lattice animals: rigorous results and wild guesses*, A Volume in Honour of John M. Hammersley, Oxford University Press, 1990, 323–335.