Aspects algorithmiques de la combinatoire
Algorithmical aspects of combinatorics

R. Cori, D. Rossin

October 19, 2006

Contents

1 Permutations

Definition 1.1. *A* permutation *of* $\{1 \dots n\}$ *is a bijection from* $\{1 \dots n\}$ *into itself.*

There are several ways to represent a permutation. The first and most natural one is given below:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \sigma(1) & \sigma(2) & \sigma(3) & \sigma(4) & \sigma(5) & \sigma(6) \end{pmatrix}$$

This representation is called the two-line representation. As the first line is always the identity one can forget its writing and the permutation is then given by its one-line representation:

$$\sigma = \begin{pmatrix} \sigma(1) & \sigma(2) & \sigma(3) & \sigma(4) & \sigma(5) & \sigma(6) \end{pmatrix}$$

The images of elements are written $\sigma(i)$ or σ_i throughout the lesson.

Another notation for permutations, the cyclic notation, will be given further in this lesson.

In the first part, we are going to use statistics on permutations to give some complexity results on sorting algorithms.

1.1 Inversion table

Definition 1.2. *(σ_i, σ_j) is an inversion in σ if and only if $\sigma_i > \sigma_j$ and $i < j$. We denote by $Inv(\sigma)$ the set of all inversions. $Inv(\sigma) = \{(\sigma_i, \sigma_j), \sigma_i > \sigma_j \text{ and } i < j\}$.*

The number of inversions in σ gives the number of elementary operations (transpositions) needed to transform σ into the identity element.

Definition 1.3. *The inversion table of a permutation is :*
$T_\alpha[i] = |\{j, (j,i) \in Inv(\sigma)\}|$

Example: $\alpha = (375248619)$

	1	2	3	4	5	6	7	8	9
T_α									

Note that $T_\alpha[i]$ is the number of elements j greater than i but before i in α's one-line notation. Thus $T_\alpha[n] = 0$ and $0 \leq T_\alpha[k] \leq n - k$.

Exercise 1.1.
1. *Prove that given a permutation σ you can compute in $\mathcal{O}(n^2)$ time its inversion table.*
2. *Prove that given a tabular T corresponding to a permutation σ (unknown), one can retrieve σ in $\mathcal{O}(n^2)$ time.*
3. *Can you make faster ?*

1.1.1 Enumeration and application to sorting algorithm analysis

Exercise 1.2. *How many inversions could a permutation have ?*

Let $I_{n,k}$ be the number of permutations of length n having k inversions. Note that:

$$\sum_{k=0}^{\frac{n(n-1)}{2}} I_{n,k} = n!$$

Note that $I_{n,k}$ is also the number of arrays of size n such that $0 \leq T[i] \leq n-i$ and the sum of all elements equals k. By deleting the first entry of the array we obtain a new array of size $n-1$ respecting all conditions such that the sum of all elements equals $k - T[0]$. Thus

$$I_{n,k} = \sum_{k'=k-n+1}^{k} I_{n-1,k'}$$

Exercise 1.3. *Fill the following array*

N. inversions	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
n=1	*1*										
n=2											
n=3											
n=4											
n=5											

Let $I_n(x)$ the generating function of inversions:

$$I_n(x) = \sum I_{n,k} x^k$$

Note that $I_n(x) = I_{n-1}(x)(1 + x + \ldots + x^{n-1})$
$I_n(x) = 1(1 + x)(1 + x + x^2)...(1 + x + \ldots + x^{n-1})$
$\bar{I}_n(x) = \frac{1}{n!} \sum_{k=0}^{\frac{n(n-1)}{2}} k I_{n,k}$
$\bar{I}_n(x) = \frac{I'_n(1)}{I_n(1)} = \frac{\partial ln(I_n(x))}{\partial x}(1)$

Theorem 1.1. *The average number of inversions in a permutation is* $\frac{n(n-1)}{4}$.

Another (simpler) proof is easily derived from studying the miror permutation with the permutation.

1.1.2 Sorting by selection

In this algorithm you first find the smallest element and then you put it in the first place.

```
for (int i = 0; i < n ; i++)
  for (int j = i+1; j < n; j++)
    if (a[i] > a[j]) swap(a[i],a[j]);
```

When performing a swap operation, the number of inversions decrease by 1. So, the number of swaps equal the number if inversions.

1.1.3 Sorting by insertion

```
for (int i = 1; i < n; i++)
  for (int j = i; j !=0 && a[j]<a[j-1]; j--)
    swap(a[j],a[j-1]);
```

Exercise 1.4. *The number of tests is :*

1.2 Cycles and smallest element

Exercise 1.5. *Let α be the following permutation : $\alpha = 372159648$.*

1. *Draw the digraph (directed graph) where each vertex represents a number of the permutation and there exists an arc between i and j if and only if $j = \alpha(i)$.*
2. *Write the cycles of this graph. This is called the cyclic notation of the permutation.*
3. *Let $C_{n,k}$ be the number of permutations of size n with k cycles. Give the first values for $n \leq 5$.*
4. *Show that $C_{n+1,k} = nC_{n,k} + C_{n,k-1}$.*
5. *Let $C_n(x) = \sum C_{n,k}x^k$. Prove that*

$$C_n(x) = \Pi_{i=0}^{n-1}(x+i)$$

6. *Prove that the average number of cycles in a permutation of size n is $H_n = \sum_{i=1}^{n} \frac{1}{i}$.*

Definition 1.4. *Let σ be a permutation. $\sigma(i)$ is a partial minimum if $\sigma(i)$ is tricly less than all $\sigma(j), j < i$.*

The following algorithm gives the number of partial minima in a permutation.

```
min = a[0];
for (int i = 0; i < n ; i++)
  if (a[i] < min) min = a[i];
```

Exercise 1.6. *Show that number of changes of partial minimal is equal to $C_{n,k}$.*

We can give a bijective proof of this result using Foata transformation.

1.3 Descents and excedences

Definition 1.5. *Let α be a permutation of size n.*

- *α_i is a descent if $\alpha_i > \alpha_{i+1}$*
- *α_i is an excedence (or weak excedence) if $a_i \geq i$*
- *α_i is a strict excedence if $a_i > i$*

Fill the following array:

	Nb descents	Nb excedences	Nb strict excedences
123			
132			
213			
231			
312			
321			

We denote by:

- $D_{n,k}$ the number of permutation of size n having k descents.
- $E_{n,k}$ the number of permutation of size n having k excedences.
- $F_{n,k}$ the number of permutation of size n having k strict excedences.

Note that for $n = 3$ we have $D_{n,k} = F_{n,k} = E_{n,k+1}$.

Proposition 1.1. $D_{n,k} = D_{n,n-k-1}$.

Proof. Use $\tilde{\alpha}$ the mirror permutation. □

Proposition 1.2. $\alpha^{-1}(i) = j \Leftrightarrow \alpha(j) = i$, $|E(\alpha)| + |F(\alpha^{-1})| = n$. *Thus* $E_{n,k} = F_{n,n-k}$

Proof.
- Prove that if $i \leq \alpha(i)$ then $\alpha^{-1}(\alpha(i))$ is not a strict excedent for α^{-1}.
- Prove that if $i > \alpha(i)$ then $\alpha^{-1}(\alpha(i))$ is a strict excedent for α^{-1}.

□

Proposition 1.3. $D_{n,k} = E_{n,n-k}$

Proof. We only sketch the proof. Let α be a permutation with $n-k$ excedents. We rewrite the permutation α in the cyclic notation, with the greatest element of each cycle in the first place and every maxima in increasing order like in Foata transformation. We obtain a permutation with k descents.

Exercise 1.7. *Make an example of this transformation and prove the property above.*

□

Exercise 1.8. *With the three last propositions, prove the claim result* $D_{n,k} = F_{n,k} = E_{n,k+1}$

Proposition 1.4. $D_{n,k} = (k+1)D_{n-1,k} + (n-k)D_{n-1,k-1}$

Proof. Choose where you insert the greatest element. □

Exercise 1.9. *Let* $D_n(x) = \sum D_{n,k}x^k$. *Prove that* $D_n(x) = (1 + (n-1)x)\, D_{n-1}(x) + (x - x^2)D'_{n-1}(x)$.

2 Permutation pattern

2.1 Greatest increasing subsequence

Definition 2.1. *The longest increasing subsequence of a permutation σ is the largest value p such that there exist $i_1, \ldots, i_p$ and $a_{i_1} < a_{i_2} < \ldots < a_{i_p}$ with $i_1 < i_2 < \ldots < i_p$.*

Exercise 2.1. *Let $\sigma = (7, 9, 12, 2, 11, 3, 8, 5, 6, 1, 4, 10)$. Give the longest increasing subsequence.*

Proof. Build iteratively all the longest increasing subsequences.

$$\begin{array}{lll} 7 & \leftarrow 9 \leftarrow & \begin{cases} 12 \\ 11 \end{cases} \\ 2 & \leftarrow 3 \leftarrow & \begin{cases} 8 \\ 5 \leftarrow 6 \leftarrow 10 \\ 4 \end{cases} \\ 1 & & \end{array}$$

□

This leads to the following dynamic programming algorithm where you fill in two different arrays:

- $BEST[i] = j$ if the longest increasing subsequence of size i ends by j and j is the smallest value possible.
- $PRED[k] =$ precessor of k in the largest increasing subsequence ending with k.

Exercise 2.2. *Give the algorithm for computing the longest increasing subsequence.*

2.2 Permutation pattern

Definition 2.2. *Let σ and π be two permutations of size n and p respectively with $p \leq n$. We say that π is a pattern of σ whenever there exists $i_1 < i_2 < \ldots < i_p$ such that $\sigma_{i_k} < \sigma_{i_l}$ whenever $\pi_k < \pi_l$ for all $1 \leq k \neq l \leq p$.*

For example, we say that 132 *occurs* in $\bar{1}23\bar{6}4\bar{5}$. But we say that 543216 *avoids* 132. .

2.3 One-stack sortable permutations

A stack is an ordered set, with two operations, pop and push such that:

- *pop* returns the last element inserted and deletes it from the set.

- *push* add an element to the set.

Let S be a stack. A permutation $\sigma_1\sigma_2\ldots\sigma_n$ is said to be *one-stack sortable* if it can be transformed to identity by the following algorithm.

1. $i \leftarrow 1$, S is the empty stack
2. Either:
   - push the element σ_i in the stack and increment i by one.
   - or pop an element from the stack and write it out.
3. Return to step 2.

The output is the list of elements popped from the stack.

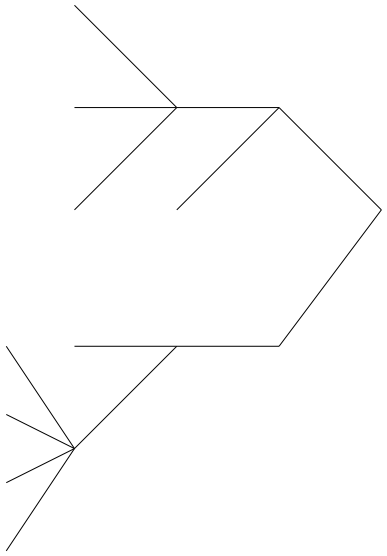
Exercise 2.3. *1. Give the one-stack sortable permutations of size* 1, 2, 3 *and* 4.

2. Find a characterization for these permutations and prove it..

Definition 2.3. *A plane tree of size n is a tree with n edges embedded in the plane and rooted on an edge.*

Exercise 2.4. *1. How many tree are there of size* 1, 2 *and* 3.

Exercise 2.5. *Let T be the following tree:*



1. Number the edges through a postfix depth first traversal of the tree

2. Read the tree through a prefix depth first traversal of the tree

3. Notice that the resulting permutation avoids 231. *Prove it.*

Definition 2.4. *A unary-binary tree is a plane rooted tree where each vertex has arity* 1 *or* 2 *(i.e. degree* 2 *or* 3*). Vertices of arity* 1 *have either a right or a left son. A binary tree is a plane rooted tree where each vertex has arity* 2*.*

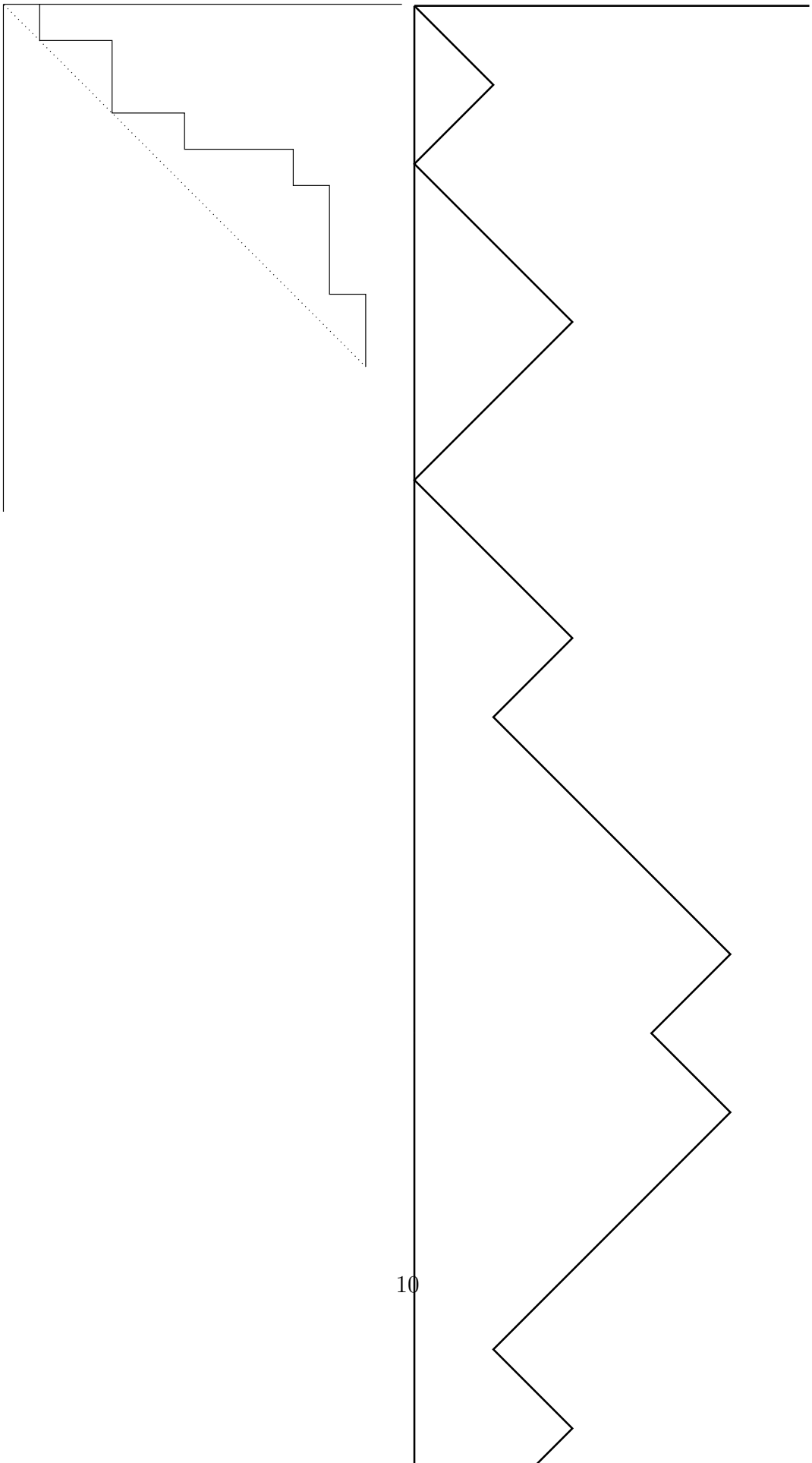
Exercise 2.6.

1. *How many unary-binary trees are there of size* 1*,* 2*,* 3 *and* 4 *?*
2. *Notice that a vertex of a plane tree is either the leftmost child of another node or the brother of a node. Deduce a correspondance between unary-binary trees and plane trees.*
3. *How many binary trees are there with* 2*,* 3 *and* 4 *leaves ?*
4. *Find a correspondance between those objects*

2.4 Dyck paths

Definition 2.5. *A Dyck path of length* n *is a path in the plane starting at* $(0, 0)$*, ending at* (n, n) *and made of* n *horizontal and* n *vertical steps that never goes under the line* $y = x$*. A Dyck path of length* n *is a path in the plane starting at* $(0, 0)$*, ending at* $(2n, 0)$ *made of* n $(1, 1)$ *steps and* n $(1, -1)$ *steps which never goes below the* x *axis.*

Exercise 2.7. *Considering a left-right depth first traversal of a plane tree, show that Dyck paths of length* n *are in one-to-one correspondence with plane trees of size* n*.*



2.5 Enumeration

We want to enumerate the number of plane trees with n edges or the number of Dyck path of size n or the number of unary-binary trees with $n-1$ edges or the number of binary trees with $n+1$ leaves.

2.5.1 Enumeration of binary trees

A binary tree is either a leaf or an ordered set of two binary trees. Thus :

$$B(x) = x + B^2(x)$$

2.5.2 Enumeration of Dyck paths

A Dyck path is either an empty one or an ordered set of two Dyck paths. Thus:

$$D(x) = xD^2(x) + 1$$

2.5.3 Direct proof on Dyck path

Note that a Dyck path is a word which contains n letters a and n letters b. Get a Dyck path and add a dwon step at the end. These paths are in bijection with Dyck paths. Then consider all words with n letters a and $n+1$ letters b. Notice that only one rotation of this word corresponds to such a path.

2.6 Enumeration of pattern-avoiding permutations

The first question which arises in this definition is given a pattern π, how many permutations of size n avoids π ?

Stanley and Wilf conjectured (Bona 1997, Arratia 1999), that for every permutation pattern σ, there is a constant $c(\sigma) < \infty$ such that for all n, $F(n, \sigma) \leq [c(\sigma)]^n$.

A related conjecture stated that for every σ, the limit $lim_(n \to \infty)[F(n, \sigma)]^{(1/n)}$ exists and is finite.

Arratia (1999) showed that these two conjectures are equivalent. The conjecture was proved by Marcus and Tardos (2004). In fact Marcus and Tardos prove The Füredi-Hajnal conjecture on matrix containment.

See article from Marcus and Tardos for the proof.

3 Trees

Given n numbered vertices, how many trees can you draw ?

Exercise 3.1.

- *Try* $n = 2, 3, 4$
- *Guess a formula*

3.1 Cayley formula and Prüfer code

Definition 3.1. *A tree si a set of $n-1$ pairs of vertices (i,j) such that the graph obtained is acyclic.*

Theorem 3.1. *The number of trees (with n vertices) equals n^{n-2}.*

Note that n^{n-2} is also the number of words of length $n-2$ on the alphabet $1, 2 \ldots, n$.

3.1.1 Prüfer code

The prüfer code of a tree is given by the following algorithm:

1. Let l the leave with the greatest label.
2. Write the label of the neighbor of l.
3. Delete l from the tree.
4. If the number of remaining vertices is greater than 2 goto 1.

Remark 3.1. *The leaves of the original tree are the numbers which does not appear in the code.*

Exercise 3.2. *Prove it*

Exercise 3.3. *Given the Prüfer code of a tree $a_1, a_2, \ldots, a_{n-2}$ give an algorithm to find the associated tree*

Another proof of the enumeration of trees can be given via the matrix tree theorem.

Theorem 3.2 (Matrix Tree)**.** *Let $G = (V, E)$ be a graph. The number of spanning trees of this graph is given by the determinant of any of $(n-1) \times (n-1)$ submatrices of the following matrix :*

$$\begin{pmatrix} d_1 & -E(1,2) & \ldots & -E(1,n) \\ -E(2,1) & d_2 & \ldots & -E(2,n) \\ \ldots & \ldots & \ldots & \ldots \\ -E(n,1) & -E(n,2) & \ldots & -d_n \end{pmatrix}$$

3.1.2 Trees with prescribed degree (proof from Wikipedia)

Another proof of the Cayley formula comes from the following Lemma:

Lemma 3.1. *The number of trees on vertices $1, \ldots, n$ with degree $d_1, d_2, \ldots, d_n, (\sum d_i = 2n-2)$ is:*

$$\frac{(n-2)!}{(d_1-1)!(d_2-1)!\ldots(d_n-1)!}$$

Proof. We prove this formula by induction on the number of vertices. For $n = 1$ and $n = 2$ the proposition holds trivially. So assume the proposition holds for $n - 1$. Since $\sum d_i = 2n - 2$ there exists k such that $d_k = 1$. Suppose that $d_n = 1$ without loss of generality. For $i = 1, 2, \ldots, n-1$ let $\mathcal{B}_i$ be the set of trees on the vertex set $\{v_1, v_2, \ldots v_{n-1}\}$ such that:

$$d(v_j) = \begin{cases} \mathrm{d}_j, & \text{if } j \neq i \\ \mathrm{d}_j - 1, & \text{if } j = i \end{cases}$$

Trees in $\mathcal{B}_i$ correspond to trees in $\mathcal{A}$ with the edge v_i, v_n, and if $d_i = 1$, then $\mathcal{B}_i = \phi$. Since v_n must have been connected to some node in every tree in $\mathcal{A}$, we have that

$$|A| = \sum_{i=1}^{n-1} |\mathcal{B}_i|$$

Further, for a given i we can apply either the inductive assumption (if $d_i > 1$) or our previous note (if $d_i = 1$, then $\mathcal{B}_i = \phi$) to find $|\mathcal{B}_i|$:

$$|\mathcal{B}_i| = \begin{cases} 0, & \text{if } d_i = 1 \\ \frac{(n-3)!}{(d_1-1)!\cdots(d_i-2)!\cdots(d_{n-1}-1)!}, & \text{otherwise} \end{cases} for i = 1, 2, \ldots, n-1$$

Observing that

$$\frac{(n-3)!}{(d_1-1)!\cdots(d_i-2)!\cdots(d_{n-1}-1)!} = \frac{(n-3)!(d_i-1)}{(d_1-1)!\cdots(d_{n-1}-1)!}$$

it becomes clear that, in either case, $|\mathcal{B}_i| = \frac{(n-3)!(d_i-1)}{(d_1-1)!\cdots(d_{n-1}-1)!}$.
So we have

$$\begin{aligned} |\mathcal{A}| &= \sum_{i=1}^{n-1} |\mathcal{B}_i| && (1) \\ &= \sum_{i=1}^{n-1} \frac{(n-3)!(d_i-1)}{(d_1-1)!\cdots(d_{n-1}-1)!} && (2) \\ &= \frac{(n-3)!}{(d_1-1)!\cdots(d_{n-1}-1)!} \sum_{i=1}^{n-1} (d_i - 1) && (3) \end{aligned}$$

And since $d_n = 1$ and $\sum_{i=1}^{n} d_i = 2n - 2$, we have:

$$|\mathcal{A}| = \frac{(n-3)!}{(d_1-1)!\cdots(d_n-1)!}(n-2) = \frac{(n-2)!}{(d_1-1)!\cdots(d_n-1)!}$$

which proves the lemma.

We have shown that given a particular list of positive integers $d_1, d_2, \ldots, d_n$ such that the sum of these integers is $2n - 2$, we can find the number of trees with labeled vertices

of these degrees. Since every tree on n vertices has $n-1$ edges, the sum of the degrees of the vertices in an n-vertex tree is always $2n-2$. To count the total number of trees on n vertices, then, we simply sum over possible degree lists. Thus, we have:

$$|T_n| = \sum_{d_1+d_2+\cdots+d_n=2n-2} \frac{(n-2)!}{(d_1-1)!\cdots(d_n-1)!}$$

If we reindex with $k_i = d_i - 1$ for $i = 1, 2, \ldots, n$, we have:

$$|T_n| = \sum_{k_1+k_2+\cdots+k_n=n-2} \frac{(n-2)!}{k_1!\cdots k_n!}$$

Finally, we can apply the multinomial theorem to find:

$$|T_n| = n^{n-2}$$

as expected. □

3.1.3 Cycles and transpositions

Let α be a cycle : $\alpha = (13758246)$. α can be written as a product of transpositions:

$$\alpha = (13)(37)(75)(58)(82)(24)(46)$$

Note the you can always write a cycle as a product of $n-1$ transpositions.

Theorem 3.3. *The number of ways to write a cycle of length n as a product of $n-1$ transpositions is n^{n-2}.*

For example: $(132) = (12)(23) = (13)(12) = (23)(13)$

Lemma 3.2. *The product $\tau_1\tau_2\ldots\tau_{n-1}$ is a cycle if and only if the graph $G = (V, E)$ where $V = 1, 2, \ldots, n$ and there is an edge between i and j if and only if (ij) is a transposition that appears in the product.*

Proof. Note first that a graph made of n vertices and $n-1$ edges is either a tree (if it is conected) or it has several connected components.

Then, note that if σ is a permutation that has $k \geq 2$ cycles then $\sigma\tau$ is a permutation with $k-1$ cycles if $\tau = (ij)$ and i and j are in different cycles.

As there is n^{n-2} different trees, and for each tree there are $(n-1)!$ different ways of making the product we have a surjection from trees to cycles. But each cycle is obtained the same number of times by symmetry so that for each cycle there is n^{n-2} ways of writing it as a product of transpositions.

□

3.2 Parking function

Definition 3.2. *A serie $a_1, \ldots, a_n$ is a Parking serie if the following algorithm give a position to all cars $1, \ldots, n$.*

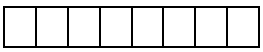
1. *For i from 1 to n do*
2. *i takes the first empty position k such that $k \geq a_i$*

For example $111, 123, 132, 231, 213, 312, 321, 112, 121, 211, 113, 131, 311, 122, 212, 221$ are all Parking series of size 3.

Theorem 3.4. *The number of Parking functions is $(n+1)^{n-1}$.*

Another equivalent definition is :

Definition 3.3. *$a_1, \ldots, a_n$ is a Parking serie if and only if there exists a permutation $\alpha = \alpha_1, \alpha_2, \ldots, \alpha_n$ such that $\forall i, a_i \leq \alpha_i$.*

Exercise 3.4. *Is the serie* 63122766 *a Parking serie ?* 

The following java algorithm determine if a serie is a Parking serie/
Data: $n$the number of cars
Data: u an array where $u[i]$ is the choice of car i
Data: *place* an empty array of size n
Result: Returns true if all cars have found an empty place and false otherwise

```
for i = 1 to n do
    j ← u[i];
    while j ≤ n && place[j]! = 0 do
        j ← j + 1;
        if j == n + 1 then
            return false;
        else
            place[j] ← i;
return true;
```

To prove the enumeration formula consider a Parking with $n+1$ places and a serie of n integers between 1 and $n+1$.

4 Chromatic polynomial

Let ϕ be a function such that $\phi_G(z)$ counts the number of ways to color G with exactly z colors.

Exercise 4.1. *Show that $\phi_G(z)$ is a polynomial by considering the coloring of G, $G - e$ and G/e.*

Definition 4.1. *The chromatic polynomial of a graph G is a polynomial $\pi_G(z)$ which counts the number of ways to color G with exactly z colors.*

If e is an edge of a graph G then $G - e$ represents the graph G where the edge e has been deleted and G/e represents the graph where G has been contracted.

Exercise 4.2. *Show that $\pi_G(z)$ is a polynomial by considering the coloring of G, $G - e$ and G/e.*

Prove that :

$\pi(x) = \sum_{F \subset E} (-1)^{|F|} x^{|V|-r(F)}$ *where the sum is over all subsets F of E, and $r(F)$ denotes the rank of F in G, i.e. the number of elements of any maximal cycle-free subset of F. Compute the chromatic polynomial of T_n a tree with n vertices, K_n and C_n, .*

The chromatic polynomial of a disconnected graph is the product of the chromatic polynomials of its connected components. The chromatic polynomial of a graph of order n has degree n, with leading coefficient 1 and constant term 0. Furthermore, the coefficients alternate signs, and the coefficient of the $(n-1)$st term is $-m$, where m is the number of edges.

5 Tutte polynomial

5.1 Definitions

The Tutte polynomial T_G of the graph $G = (V, E)$ could be defined by the two following recursive formula:

$$T_G = \begin{cases} xT_{G/e} & \text{If e is an isthmus} \\ yT_{G-e} & \text{If e is a loop} \\ T_{G-e} + T_{G/e} & \text{Otherwise} \end{cases}$$

or

$$T_G = \sum_{F \subset E} (x-1)^{r<E>-r<F>} (y-1)^{n<F>}$$

where $r < E >= |G| - \kappa(E)$ et $n < E >= |E| - |G| + \kappa(G) = |E| - r < E >$ (edges-vertices+connected components). r is called the rank and n the nullity.

Note that the second definition yields a polynomial but the first one could give many polynomials. First, we can prove that the first definition is correct i.e. that every sequence of edge contraction/deletion yields to the same polynomial.

An other proof is to repark that both definitions are equivalent so that the first is a definition !!!

6 Evaluations of the Tutte polynomial for a connected graph

6.1 Evaluation in $(1,1)$

Give an interpretation of $T_G(1,1)$.

6.2 $T_G(2,1)$

Give an interpretation of $T_G(2,1)$.

6.3 $T_G(1,2)$

Give an interpretation of $T_G(1,2)$.

6.4 $T_G(2,2)$

Give an interpretation of $T_G(2,2)$.

6.5 Towards a new Definition of Tutte polynomial

$$T_G(x,y) = \sum_{i,j} t_{ij} x^i y^j$$

where t_{ij} is the number of spanning forests of G with internal activity i and external activity j.

The proof is by induction on the number of edges of $G = (V, E)$. The formula is true for $G = (V, \varnothing)$. Suppose now that the formula is true till $m-1$ edges and take a graph G with n vertices and m edges $e_1, \ldots, e_m$.

Let $G' = G - e_m$ and $G'' = G/e_m$. By induction $T_{G'} = \sum_{ij} t'_{ij} x^i y^j$ et $T_{G''} = \sum_{ij} t''_{ij} x^i y^j$
We now separate different cases depending on the nature of e_m.

- If e_m is an isthmus (or bridge), then e_m belongs to any spanning forest of G. Moreover, e_m is internal active in every forest. So F is a (i,j) spanning forest if and only if $F - e_m$ is a $(i-1,j)$ spanning forest of $G - e_m$. Indeed e_m does not modify the activity of any other edges.

  Hence:

$$\sum_{i,j} t_{ij} x^i y^j = \sum_{i,j} t'_{i-1,j} x^i y^j = x \sum_{i,j} t'_{i-1,j} x^{i-1} y^j = x T_{G-e_m}(x,y) = T_G(x,y)$$

- If e_m is a loop, then e_m belongs to no spanning forest. A subgraph F is a spanning foret if and only if it is a spanning forest of $G - e_m$. Moreover, e_m is external active in any spanning forest. F is a (i, j) spanning forest iff it is a $(i, j-1)$ spanning forest of $G - e_m$.
- Otherwise notice that we can separate the spanning forests of G into two parts: those which contain e_m and the others. The first ones are in bijection with spanning forests of $G - e_m$ and the others with G/e_m.

This new definition allow to prove that the number of spanning trees whose internal activity and external activity is given do not depend on the numbering of the edges.

6.6 Hardness of computation

Computing $T_G(a, b)$ for a graph G is $\sharp P$-hard everywhere except for $(1, 1)$, $(-1, -1)$, $(0, -1)$, $(-1, 0)$, $(i, -i)$, $(-i, i)$, (j, j^2), (j^2, j) where the evaluation is polynomial.

7 Words

7.1 Exhaustive generation

The goal is to give an algorithm to compute all words of size n on an alphabet with m letters.

The first and naive algorithm consists in counting from 0000000 to $(m-1)(m-1)(m-1)\ldots$ by adding 1 at each step.

The algorithm becomes:

```
Data: n the size of words
Data: m the size of the alphabet, here m = 2
Result: Compute all words of size n
while (true){ do
    print(f);
    j = n-1;
    while (f[j] == 1) do
        f[j] = 0;
        j–;
    if (j == −1) then
        Exit;
    else
        f[j] = 1;
```

Proposition 7.1. *The number of comparisons is* $2^{n+1} - 1$.

7.2 Gray Code

We want to compute every word but changing only one bit between two words.

Exercise 7.1.
- *Find an ordering of words of size n verifying the above property. Try for $n = 1, 2, 3$.*
- *Write the form of the serie.*
- *Write the example for $n = 4$.*
- *Guess an algorithm to compute all words using this Gray code.*

7.3 Factors of a word

We want to find a word such that all words of size n are factors of this word. For example 00110 for $n = 2$ and 0001011100 for $n = 3$ are such words. This words are called De Bruijn words. Their length is $2^n + n - 1$.

8 Lyondon words

Definition 8.1. *A word is a Lyndon word iff $\forall f = f'f'', f' \neq \epsilon$ we have $f < f''$ for the lexicographic order.*

For example 010 is not a Lyndon word because $0 < 010$.

Exercise 8.1. *Give all Lyndon words of size $1, 2, 3, 4$. Prove that every word (except 0) begins with 0 and ends with 1.*

Proposition 8.1. *Every word which is not a power of another word has a unique conjugate which is a Lyndon word.*

8.1 Enumeration

Theorem 8.1. *Let $L_m(d)$ the number of Lyndon words of size d on alphabet $\{0, 1, \ldots, m-1\}$.*

$$\sum_{d|n} dL_m(d) = m^n$$

Theorem 8.2. *The Mbius function is a number theoretic function defined by* $\mu(n) = \begin{cases} 0 if n has one or more repeated prime factors; \\ 1 if n == 1; \\ (-1)^k if n is a product of k distinct primes, \end{cases}$

The moebius inversion formula is the transform inverting the sequence $g(n) = sum_(d|n) f(d)$ into $f(n) = \sum_{(d|n)} \mu(d) g(n/d)$

where the sums are over all possible integers d that divide n and $\mu(d)$ is the Mbius function.

Theorem 8.3.

$$L_m(n) = 1/n \sum_{d|n} \mu(d) m^{n/d}$$

8.2 Fundamental Theorem

Theorem 8.4. *Every word f can be written in a unique way as $f = \lambda_1 \lambda_2 \dots \lambda_k$ such that λ_i is a Lyndon word and $\lambda_i \geq \lambda_{i+1}$*

Proof. Existence by induction by taking the largest suffix that is a Lyndon word. Unicity : If $\lambda_1 \dots \lambda_k = \lambda'_1 \dots \lambda'_{k'}$ suppose that $\lambda'_k = u\lambda_k$. □

Theorem 8.5. *For all n, the word obtained by taking all Lyondon words of size $d|n$ followed by its $n-1$ first letters is a De Bruijn word. The Lyondon words are taken in lexicographic order.*

Example, for $n = 4$ the word $0, 0001, 0011, 01, 0111, 1, 000$ has length 19 and each factor of length 4 is distinct.

Index