Aspects algorithmiques de la combinatoire
Algorithmical aspects of combinatorics

R. Cori, D. Rossin

September 27, 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:

$$\sigma = \left(\begin{array}{cccccc} \sigma(1) & \sigma(2) & \sigma(3) & \sigma(4) & \sigma(5) & \sigma(6) \end{array} \right)$$

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

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.

$$7 \quad \leftarrow 9 \leftarrow \begin{cases} 12 \\ 11 \end{cases}$$

$$2 \leftarrow 3 \leftarrow \begin{cases} 8 \\ 5 \leftarrow 6 \leftarrow 10 \\ 4 \end{cases}$$

$$1$$

□

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.

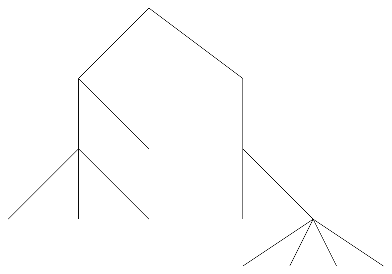
Exercise 2.3. *1. Give the one-stack sortable permutations of size* 1, 2, 3 *and* 4.

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*.*

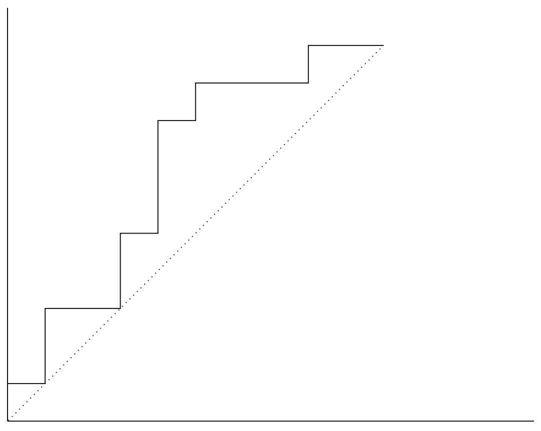
Exercise 2.6.

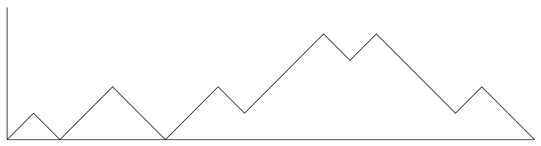
1. *How many unary-binary trees are there of size* 1*,* 2*,* 3 *and* 4 *?*
2. *Notice that a vertex of a plane tree is either the leftmost child of another node or the brother of a node. Deduce a correspondance between unary-binary trees and plane trees.*
3. *How many binary trees are there with* 2*,* 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 $lim_(n \rightarrow \infty)[F(n,\sigma)]^{(1/n)}$ 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.

Index