Travaux pratiques 3a

CCINP 2019 I

Option informatique MP1, MP2 & MP3

Défibitions et notations

- Pour tout $n \in \mathbb{N} \setminus \{0\}$, une permutation de taille n est une bijection de l'ensemble $E_n = \{0, 1, \cdots, n-1\}$ dans lui-même.
- Dans la suite, l'ensemble des permutations de taille n est noté $\mathfrak{S}_n$.
- Étant donné une permutation σ de taille n, on la représente sous la forme $s_0, s_1, \ldots, s_{n-1}$ où : $s_i = \sigma(i)$ pour tout $i \in E_n$.
  Par exemple, 1032 représente la permutation envoyant 0 sur 1, 1 sur 0, 2 sur 3 et 3 sur 2.
- Soit $\sigma \in \mathfrak{S}_n$. On dit que $(i, j) \in E_n^2$ est une inversion de σ si $\sigma(i) > \sigma(j)$ et $i < j$.
- Si A est un ensemble fini, $|A|$ désigne le cardinal de l'ensemble A.
- Pour tout entier $n \geq 1$, on pose $\widehat{E}_n = \prod_{k=0}^{n-1} E_{n-k}$.
  Par exemple $\widehat{E}_3 = \{0, 1, 2\} \times \{0, 1\} \times \{0\}$.

Le sujet original demandait une écriture en python, on donnera les solution en python et en OCaml.

I Tri à bulles

Exercice 1
Déterminer l'ensemble des inversions de la permutation $\sigma = 140253$.

Le tri à bulles est un tri en place défini par l'algorithme

```
Entree : une liste d'entiers l
n = longueur(l)
pour i allant de n - 1 à 1 faire
    pour j allant de n - 1 à n - i faire
        si L[j] < L[j-1]
        alors echanger L[j-1] et L[j]
        finsi
    finpour
finpour
```

Dans la suite, on admet que l'algorithme du tri à bulles est bien un algorithme de tri.

Exercice 2 -
Écrire une fonction `tri_bulle` qui prend en argument un tableau d'entiers et qui trie cette liste à l'aide du tri à bulles. À l'issue de la fonction, la liste est triée.

Exercice 3 -
Montrer que si on effectue exactement un échange dans une liste lors du tri à bulles, la nouvelle liste a exactement une inversion en moins.

Exercice 4 -
En déduire une fonction `nombre_inversions` qui prend en argument un tableau correspondant à une permutation et renvoyant le nombre d'inversions de celle-ci.
Cette fonction sera une légère modification du tri à bulles.

II Table d'inversions

Définition 1 : table d'inversions

La table d'inversions, $\text{inv}(\sigma)$, de $\sigma \in \mathfrak{S}_n$ est le n-uplet $(\alpha_0, \cdots, \alpha_{n-1})$ tel que α_i est le nombre d'inversions de σ de la forme (i, j) avec $0 \leq i < j < n$.

$$\alpha_i = \left|\left\{j \in \ ; \ \sigma(i) > \sigma(j)\right\}\right|$$

Exercice 5 -
Déterminer la table d'inversions de la permutation 140253.

Exercice 6 -
Écrire une fonction `permutation_vers_table` qui prend en argument un tableau représentant une permutation et qui renvoie la table d'inversions correspondante.

Exercice 7 -
Montrer que pour toute permutation $\sigma \in \mathfrak{S}_n$, $\text{inv}(\sigma)$ est un élément de $\widehat{E}_n$.

Exercice 8 -
Montrer que $\sigma(0) = \alpha_0$ pour tout permutation σ où α_0 est le premier terme de $\text{inv}(\sigma)$..

Exercice 9 -
Montrer que pour tout $n \geq 1$, l'application $\sigma \mapsto \text{inv}(\sigma)$ est une bijection de $\mathfrak{S}_n$ vers $\widehat{E}_n$.

Exercice 10 -
Écrire une fonction `table_vers_permutation` qui prend en argument un tableau qui correspond à une table d'inversions et qui renvoie la permutation qui lui est associée.

Solutions

Solution de l'exercice 1
$\text{inv}(\sigma) = \{(0,2),(1,2),(1,3),(1,5),(4,5)\}$.

Solution de l'exercice 2 -

```
let echanger t i j =
   let temp = t.(j) in
   t.(j) = t.(i);
   t.(i) = temp;;

let tri_bulle t =
   let n = Array.length t in
   for i = (n-1) downto 1 do
      for j = (n-1) downto (n-i) do
         if t.(j) > t.(j-1)
         then echanger t j (j-1) done done;;
```

```
def echanger(L, i, j):
    temp = L[j]
    L[j] = L[i]
    L[i] = temp

def tri_bulle(L):
    n = len(L)
    for i in range(n-1, 0, -1):
        for j in range(n-1, n-i-1, -1) :
            if L[j] < L[j-1]:
                echanger(L, j, j-1)
```

Solution de l'exercice 3 -
On note $l_0, l_1, \ldots, l_{n-1}$ la liste lors de l'échange.
L'algorithme échange l_j et l_{j-1} lorsqu'on a $l_j << l_{j-1}$, cela supprime une inversion.
Pour $i \notin \{j-1, j\}$, (i,j) est un inversion après l'échange si et seulement si $(i, j-1)$ était une permutation avant l'échange et $(i, j-1)$ est un inversion après l'échange si et seulement si (i,j) était une permutation avant l'échange.
Les permutations faisant intervenr deux entiers hors de $\{j-1, j\}$ ne changent pas.
Ainsi le nombre total de permutation est diminué de 1.

Solution de l'exercice 4 -

```
let nombre_inversions t =
   let n = Array.length t in
   let inv = ref 0 in
   for i = (n-1) downto 1 do
      for j = (n-1) downto (n-i) do
         if t.(j) > t.(j-1)
         then (echanger t j (j-1); incr inv) done done;
   !inv;
```

```python
def nombre_inversions(L) :
    inv = 0
    n = len(L)
    for i in range(n-1, 0, -1):
        for j in range(n-1, n-i-1, -1) :
            if L[j] < L[j-1]:
                echanger(L, j, j-1)
                inv = inv + 1
    return inv
```

Solution de l'exercice 5 -
$(1, 3, 0, 0, 1, 0)$

Solution de l'exercice 6 -

```
let permutation_vers_table t =
   let n = Array.length t in
   let inv = Array.make n 0 in
      for i = 0 to (n-1) do
         for j = (i+1) to (n-1) do
            if t.(j) < t.(i)
            then inv.(i) <- inv.(i) + 1 done done;
   inv;;
```

```python
def permutation_vers_table(L):
    n = len(L)
    inv= [0]*n
    for i in range(n):
        alpha = 0
        for j in range(i+1, n):
            if L[j] < L[i]:
                alpha = alpha + 1
        inv[i] = alpha
    return inv
```

Solution de l'exercice 7 -
Pour tout i, α_i est le cardinal d'un sous-ensemble de $\{i+1, \dots, n-1\}$ qui comporte $n-i-1$ éléments donc $\alpha_i \in E_{n-i}$.

Solution de l'exercice 8 -
On note $\sigma(0) = k$. Pour tout $i \in \{0, 1, \dots, k-1\}$ il existe un unique $j \in \{1, 2, \dots, n-1\}$ tel que $\sigma(j) = i$ car σ est une bijection. Ces entiers définissent donc k inversions $(0, j)$. Pour les autres avaleurs $j' > 1$ on a $\sigma(j') > k$: $(0, j')$ n'est pas une inversion.
Ainsi, il y a exactement k inversions de la forme $(0, j)$: $k = \alpha_0$.

Solution de l'exercice 9 -
On a $\left|\widehat{E}_n\right| = \left|\prod_{k=0}^{n-1} E_{n-k}\right| = \prod_{k=0}^{n-1} |E_{n-k}| = \prod_{k=0}^{n-1} (n-k) = n! = \left|\mathfrak{S}_n\right|$.
Compte tenu de l'égalité des cardinaux, il suffit de montrer l'injectivité pour prouver la bijectivité..
On suppose qu'on $(\alpha_0, \dots, \alpha_{n-1}) = \text{inv}(\sigma) = \text{inv}(\sigma') = (\alpha'_0, \dots, \alpha'_{n-1})$ pour $\sigma, \sigma' \in \mathfrak{S}_n$.
On a alors $\sigma(0) = \alpha_0 = \alpha'_0 = \sigma'(0)$. On note $k = \sigma(0)$.
On compose alors les permutations par le cycle $k \to 0 \to 1 \to \cdots \to k-1 \to k$ c'est-à-dire par la permutation $c = 123 \cdots k0(k+1) \cdots (n-1)$: $\pi = c \circ \sigma$ et $\pi' = c \circ \sigma'$.

- On a $\pi(0) = c\big(\sigma(0)\big) = c(k) = 0$ et, de même $\pi'(0) = 0$.
- Pour $i \geq 0$ on a $\pi(i) = \sigma(i) + 1$ si on avait $\sigma(i) < k$ et $\pi(i) = \sigma(i)$ sinon ; ainsi $\pi(i) \in \{1, 2, \ldots, n-1\}$.
- Pour $\sigma(j) < \sigma(i)$ avec $1 \leq i$ et $1 \leq j$, il y a 3 cas :
  - $\sigma(j) < \sigma(i) < k$, alors $\pi(j) = \sigma(j) + 1 < \sigma(i) + 1 = \pi(i)$,
  - $\sigma(j) < k < \sigma(i)$, alors $\pi(j) = \sigma(j) + 1 \leq k < \sigma(i) = \pi(i)$,
  - $k < \sigma(j) < \sigma(i)$, alors $\pi(j) = \sigma(j) < \sigma(i) = \pi(i)$.

  Dans tous les cas $\pi(j) < \pi(i)$. Ainsi les inversions de la forme (i, j) pour $1 \leq i < j$ sont conservées entre σ et π et il n'y en pas de nouvelles. La même démonstration prouver que σ' et π' ont les mêmes inversions.
- Si on se restreint à $\{1, 2, \ldots, n-1\}$, stables par π et π', on obtient deux permutations d'un ensemble à $n-1$ éléments qui ont la même table d'inversions.

On peut alors faire une démonstration par récurrence sur le cardinal de l'ensemble permuté : $\pi = \pi'$ impliqueσ $= c^{-1} \circ \pi = c^{-1} \circ \pi' = \sigma'$.
Ainsi la fonction est injective donc bijective.

Solution de l'exercice 10 -
Maintenant qu'on a prouvé la bijectivité on peut raisonner sur la permutation σ dont la table d'indices et $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$.
On va définir $\sigma(k)$ dans l'ordre croissant de k.
Pour calculer $\sigma(k)$, on sait qu'il est une des valeurs de $E_n \setminus \big\{\sigma(0), \sigma(1), \ldots, \sigma(k-1)\big\} = I_k$. De plus il y a α_k inversion de la forme (k, j) avec $k < j$ donc il y a exactement α_k éléments dans l'ensemble I_k plus petits que $\sigma(k)$. On en déduit que $\sigma(k)$ est le $\alpha_k + 1$-ième élément de I_k.
Pour définir I_k on peut utiliser un tableau de booléens qui marque les valeurs déjà prises.

```
let table_vers_permutation inv =
   let n = Array.length inv in
   let sigma = Array.make n (-1) in
   let pris = Array.make n false in
   for i = 0 to (n-1) do
      let pos = ref (-1) in
      let j = ref (-1) in
      while !pos < inv.(i) do
          incr j;
          if not pris.(!j) then incr pos done;
      sigma.(i) <- !j;
      pris.(!j) <- true done;
   sigma;;
```

Une autre stratégie est de maintenir la liste ordonnée des éléments restants, on peut le faire avec la méthode `.pop(i)` qui enlève le i-ième élément.

```
def table_vers_permutation(inv):
    n = len(inv)
    sigma = [0]*n
    reste = list(range(n))
    for i in range(n):
        sigma[i] = reste[inv[i]]
        reste.pop(inv[i])
    return sigma
```