

EA4 – Éléments d’algorithmique

TD n° 7 : tris, fin ?

Exercice 1 : décomposition en produit de transpositions

On rappelle qu’une *transposition* une permutation ayant un unique cycle de longueur 2 (et donc $n - 2$ points fixes). Le but de cet exercice est de montrer plusieurs manières de décomposer une permutation en produit de transpositions, sur l’exemple de la permutation^[1]

$$\sigma = 4 \ 9 \ 7 \ 2 \ 1 \ 3 \ 10 \ 8 \ 5 \ 6 = (1 \ 4 \ 2 \ 9 \ 5)(3 \ 7 \ 10 \ 6).$$

À partir de la représentation en cycles disjoints

1. Calculer les produits $(i_1 \ i_2) \circ (i_2 \ i_3) \circ (i_3 \ i_4)$ et $(i_1 \ i_2) \circ (i_1 \ i_3) \circ (i_1 \ i_4)$.
2. En déduire deux factorisations différentes du cycle $(3 \ 7 \ 10 \ 6)$ en produit de 3 transpositions, puis (au moins) une factorisation de σ .

À partir de la représentation linéaire

3. Calculer $\sigma_1 = \sigma \circ (1 \ 5)$ et $\sigma'_1 = (1 \ 4) \circ \sigma$, puis comparer σ_1 et σ'_1 à σ . Quel algorithme de tri procède de cette manière ?
4. Calculer itérativement des transpositions τ_i telles que $\sigma \circ \tau_1 \circ \tau_2 \circ \dots \circ \tau_\ell = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$.
5. Que peut-on en déduire pour $\tau_1 \circ \tau_2 \circ \dots \circ \tau_\ell$? En déduire une décomposition de σ en produit de ℓ transpositions.
6. Calculer τ'_2 telle que $\sigma_2 = \tau'_2 \circ \sigma_1$, puis itérativement des τ'_i telles que $\tau'_\ell \circ \dots \circ \tau'_2 \circ \tau'_1 \circ \sigma = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$. En déduire une autre décomposition de σ en produit de ℓ transpositions.
7. Au lieu de chercher à trier σ , donc à partir de σ pour obtenir la permutation identité, on peut faire l’inverse : partir de $1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10$ pour *engendrer* σ en plaçant d’abord le bon élément en position 1, puis celui en position 2, etc. Calculer de cette manière une décomposition de σ en produit de transpositions. Que remarquez-vous ?

Exercice 2 : déroulement du tri rapide

On considère le tableau T suivant :

7	1	5	6	2	3	10	8	0	9	4
---	---	---	---	---	---	----	---	---	---	---

Décrire le déroulement des variantes suivantes du tri rapide sur le tableau T :

1. variante avec mémoire auxiliaire, en prenant le premier élément comme pivot ;
2. *idem*, mais en choisissant toujours pour pivot l’élément médian.
3. variante « en place », le pivot étant toujours le premier élément.

Dans chaque cas, compter précisément le nombre de comparaisons effectuées.

Exercice 3 : complexité du tri rapide

1. Évaluer le nombre de comparaisons d’éléments effectuées par le tri rapide (avec pivot $T[0]$) dans les cas suivants :

- $C[n] = [1, 2, \dots, n-1, n]$;
- $D[n] = [n, n-1, \dots, 2, 1]$;
- $M[k] = [2**k] + M[k-1] + [i+2**k \text{ for } i \text{ in } M[k-1]]$ (avec $M[0] = [1]$).

1. il n’est pas interdit de vérifier qu’il s’agit bien de deux représentations de la même permutation

2. Donner d'autres exemples de tableaux de taille 7 pour lesquels la complexité du tri rapide est la même que pour $C[7]$. Même question pour $M[2]$.
3. Quel est le nombre de sommets de l'arbre de récursion du tri rapide pour un tableau de longueur n dont tous les éléments sont distincts ? Que peut-on en conclure concernant sa hauteur minimale ? et sa hauteur maximale ?
4. Donner un encadrement aussi fin que possible du nombre de comparaisons effectuées, en cumulé, pour les appels récursifs de profondeur p .
5. En déduire la complexité en temps du tri rapide dans le meilleur et dans le pire cas (en supposant toujours que tous les éléments sont distincts).

Exercice 4 : hauteur de pile du tri rapide

Comme tout algorithme récursif, le tri rapide est sujet à des débordements de la pile si le nombre d'appels récursifs devient trop grand.

1. On considère l'implémentation « naïve » du tri rapide avec copies de tableaux :

```
def triRapide(T) :
    if len(T) <= 1 : return T
    pivot = T[0]
    gauche = [elt for elt in T[1:] if elt <= pivot]
    droite = [elt for elt in T[1:] if elt > pivot]
    return triRapide(gauche) + [pivot] + triRapide(droite)
```

Quelle hauteur atteint la pile dans le pire cas ? dans le meilleur cas ?

2. Le deuxième appel récursif est terminal^[2] et peut donc être dérécursivé. Quelle hauteur atteint alors la pile sur l'entrée $C[n] = [1, \dots, n]$? Et sur l'entrée $D[n] = [n, \dots, 1]$?
3. Proposer une solution permettant de garantir une hauteur de pile maximale en $O(\log n)$ dans tous les cas (où n est la longueur du tableau à trier).
4. Quel est alors le meilleur cas en ce qui concerne la hauteur de pile ?

Exercice 5 : tri de crêpes

Dans cet exercice, on s'intéresse au tri d'une pile de crêpes, de manière à ce que les crêpes soient empilées de la plus grande à la plus petite. Un seul type d'opération est autorisé pour manipuler la pile : insérer une spatule à un endroit de la pile et retourner d'un coup toutes les crêpes qui se trouvent au-dessus de la spatule. On s'intéresse au nombre de retournements $p(n)$ qu'on doit effectuer pour arriver à une pile triée quand il y a n crêpes.

1. On numérote les crêpes en fonction de leur taille – la plus petite reçoit le nombre 1, la plus grande le nombre n . Trouver le nombre p dans les cas suivants : $[3, 2, 1], [2, 3, 1], [1, 4, 3, 2]$.
2. Proposez un algorithme qui trie le tas de crêpes. On essaiera de minimiser le nombre de retournements effectués. *Indice : on peut s'intéresser à la crêpe la plus grande du paquet.*
3. Montrer qu'un tas de n crêpes peut être ordonné à l'aide d'au plus $2n - 1$ manipulations.
4. Le cuisinier, maladroit, a brûlé une face de chacune de ses crêpes. Peut-on trier le tas de crêpes de façon à cacher les faces brûlées ? (toujours en utilisant les retournements)

2. enfin... presque... faisons comme si c'était le cas !

Ex 1

$$\begin{aligned} \sigma &= 4\ 9\ 7\ 2\ 1\ 3\ 10\ 8\ 5\ 6 \\ &= (1\ 4\ 2\ 9\ 5)(3\ 7\ 10\ 6) \end{aligned}$$

$$\frac{(i_1, i_2) \circ (i_2, i_3) \circ (i_3, i_4)}{(i_1, i_2) \circ (i_1, i_3) \circ (i_4 \circ i_4)}$$

$$(i_1, i_2) \circ (i_2, i_3) \circ (i_3, i_4)$$

$$= (i_1, i_2, i_3, i_4)$$

$i_4 \rightarrow i_4$

$$(i_1, i_2, i_3) \circ (i_1, i_3) \circ (i_1, i_4)$$

$$= (i_1, i_4, i_3, i_2)$$

$i_4 \text{ devient } i_4$

$i_4 \text{ devient } i_4$

$$b_7 = (3 \ 6) \circ (3 \ 10) \circ (3 \ 7)$$

$$= (3 \ 7 \ 10 \ 6)$$

$$(3 \ 7) \circ (7 \ 10) \circ (10 \ 6)$$

$$= (3 \ 7 \ 10 \ 6)$$

$$c_7 b_6 = (1 \ 4 \ 2 \ 9 \ 5)(3 \ 7 \ 10 \ 6)$$

$$6 \circ (1,5) = (1 \ 4 \ 2 \ 9 \ 5)(3 \ 7 \ 10 \ 6)$$

$$\circ (1 \ 5)$$

$$(1)(5 \ 4 \ 2 \ 9)(3 \ 7 \ 10 \ 6)$$

$$(1 \ 4) \circ (1 \ 4 \ 2 \ 9 \ 5)$$

$$(3 \ 7 \ 10 \ 6)$$

$$= (1)(2 \ 9 \ 5 \ 4)(3 \ 7 \ 10 \ 6)$$

$$G = \begin{pmatrix} 4 & 9 & 1 & 3 & 10 & 8 & 5 & 6 \end{pmatrix}$$

$$6 \circ (15) = (19 \quad 2 \quad 4 \quad 3 \quad 10 \quad 8 \quad 5 \quad 6)$$

$$\begin{pmatrix} 1 \\ 4 \end{pmatrix} \circ G$$

$$d_7: 6 \circ T_0 \circ T_2 \dots \circ T = \text{id}_{123\dots 910}$$

↳ rechtsseitig
bei id

$$T_1 = (15)$$

$$6 \circ (15) \circ (45) \circ (25)$$

$$\circ (35) \circ (36) \circ (76)$$

$$\circ (106) = \text{id}$$

$$T_1 \circ T_2 \dots \circ T_l = 6^{-1}$$

$$5 \gamma: 6 = (T_1 \circ T_2 \circ \dots \circ T_l)^{-1}$$

$$= \frac{(T_l^{-1} \circ T_{l-1}^{-1} \circ \dots \circ T_2^{-1} \circ T_1^{-1})}{(15)^{-1}} = (15)$$

$$(15)^{-1} = (15)$$

$$6 \circ 6_2 = T_1^{-1} \circ 6_1$$

$$T_1^{-1} = (1\ 4)$$

$$T_1^{-1} \circ \dots \circ T_2^{-1} \circ T_1^{-1} \circ 6 = \text{id}$$

$$\underbrace{(10\ 6) \circ (7, 10) \circ (3\ 7) \circ (9\ 5)}_{(2\ 3) \circ (4\ 2) \circ (1\ 4) \circ 6} = \text{id}$$

$$\alpha^{-1} = 6$$

$$6 = ((\) \circ (\) \circ (\))^{-1}$$

$$= (1\ 4) \circ (4\ 2) \circ (2\ 9) \circ (3\ 5) \circ (3\ 7) \\ \circ (7\ 10) \circ (10\ 6)$$

Ex 2 :-

15 6 2 3 10 8 0 9 4

0 1 2 3 4 5 6 7 8 9 10

$P = f$

1 5 6 2 3 0 4

10 8 9

$P = 1$ / 0 1 2 3 4 5 6

$P = 10$ / 8 9 10

0 5 6 2 3 4

$P = 5$ 2 3 4 5 6

$P = 8$ / 8 9

2 3 4 6

$P = 2$ / 2 3 4
∅ 3 4

∅ 8 9

$P = 3$ / 3 9
∅ 4 ↑

7 1 5 6 2 3 10 8 0 9 4

