L3-INFO Algorithmique et Modélisation

Quick – 18 février 2022 – durée 1 h

Sont interdits : les documents, les ordinateurs, les téléphones (incluant smartphone, tablettes,... tout ce qui contient un dispositif électronique).
Seuls les dictionnaires papier pour les personnes de langue étrangère sont autorisés.
Il sera tenu compte de la qualité de la rédaction et de la clarté de la présentation (2 pts).
Le barème **indicatif** : Exercice 1 : 9 pts ($\sim 25mn$) ; Exercice 2 : 6 pts ($\sim 20mn$), Exercice 3 : 3 pts ($\sim 10mn$)
Les 3 exercices sont indépendants.

I : Bogosort, un algorithme de tri randomisé

Le tableau T de taille n (indicé de 0 à $n-1$) contient des éléments distincts comparables deux à deux. On souhaite effectuer une permutation aléatoire des éléments de ce tableau. Pour cela on dispose d'un générateur aléatoire `Aléa` (a,b) qui génère uniformément un entier compris entre a et b inclus, les appels successifs à `Aléa` seront supposés indépendants.

On considère l'algorithme suivant opérant sur le tableau T.

```
Permuter (T)
  for i = n − 1 to 1
  |  j = Aléa ( 0 , i )
  |  Échanger (T,i,j)            // échange les valeurs T[i] et T[j]
```

I.1. Analyse de l'algorithme `Permuter`

(a) Montrer que `Permuter` (T) génère bien une permutation du tableau T.

(b) Combien existe-t-il de permutations différentes du tableau T ? (Justifier votre réponse)

(c) Prouver que l'algorithme `Permuter` génère une permutation selon une loi uniforme sur l'ensemble de toutes les permutations possibles. C'est à dire : étant donnée une permutation **spécifique** de T, la probabilité que `Permuter` (T) génère cette permutation spécifique est égale à 1/(le nombre total de permutations possibles).

(d) (Bonus) Réécrire la fonction `Permuter` sous forme **récursive**.

On envisage de trier le tableau T de taille n en le mélangeant jusqu'à ce qu'il soit trié avec la fonction `Bogosort` (.).

```
Bogosort (T)
  repeat
  |  Permuter (T)
  until Est_trié (T)
```

I.2. Analyse de `Bogosort`

(a) Au **minimum**, combien de fois la fonction `Bogosort` (T) appelle-t-elle la fonction `Permuter` (T) ?

(b) Au **maximum**, combien de fois la fonction `Bogosort` (T) appelle-t-elle la fonction `Permuter` (T) ?

(c) (Question de cours) **En moyenne**, combien de fois la fonction `Bogosort` (T) appelle-t-elle la fonction `Permuter` ?

(d) (Question de cours) Quelle est la probabilité que la fonction `Bogosort` (T) appelle **exactement** k fois la fonction `Permuter` (T) ?

(e) Que peut-on penser de cet algorithme ?

II : Avant La Reine des neiges et Le Roi lion

On dispose d'un tableau des films de Walt Disney dans l'ordre chronologique. Ce tableau contient des triplés (Rang,Titre,Année). On souhaite réorganiser le tableau en plaçant les films par rang.
Par exemple, pour les 13 premiers films, on a le tableau sur la gauche que l'on veut transformer en le tableau de droite.

Rang	Titre du film	Année
1	Blanche-Neige et les Sept Nains	1937
13	Dumbo	1941
3	Bambi	1942
4	Cendrillon	1950
12	Alice au pays des merveilles	1951
8	Peter Pan	1953
7	La Belle et le Clochard	1955
9	La Belle au bois dormant	1959
5	Les 101 Dalmatiens	1961
11	Merlin l'Enchanteur	1963
2	Le Livre de la jungle	1967
6	Les Aristochats	1970
10	Robin des Bois	1973

$\Rightarrow$

Rang	Titre du film	Année
1	Blanche-Neige et les Sept Nains	1937
2	Le Livre de la jungle	1967
3	Bambi	1942
4	Cendrillon	1950
5	Les 101 Dalmatiens	1961
6	Les Aristochats	1970
7	La Belle et le Clochard	1955
8	Peter Pan	1953
9	La Belle au bois dormant	1959
10	Robin des Bois	1973
11	Merlin l'Enchanteur	1963
12	Alice au pays des merveilles	1951
13	Dumbo	1941

Le tableau T est indicé de 1 à n ; on accède au rang du film de la ligne i par $T[i].rang$. Dans l'exemple ci-dessus, $T[13].rang$ vaut 10.

II.1. Écrire un algorithme **efficace** (en $\mathcal{O}(n)$) qui retourne un nouveau tableau contenant les éléments de T triés par rang. (Justifier)

L'algorithme ci-dessus utilise un espace mémoire supplémentaire au moins égal à la taille du tableau T, on souhaite réduire l'espace mémoire utilisé en réécrivant le tableau en place.

II.2. Écrire un algorithme qui opère directement sur le tableau (en place). Prouver cet algorithme. Donner sa complexité en temps et en mémoire.

III : SUM, conception d'algorithme

Soit $\mathcal{E}$ un ensemble de n éléments. À chaque élément i de $\mathcal{E}$, on associe une valeur v_i. Pour éviter les cas particuliers, on supposera les v_i distincts.
On cherche tous les couples (i,j) avec $i \neq j$ tels que :

$v_i + v_j = S$ pour un S donné.;

Ce problème a été vu en ALGO5 ; où un algorithme naïf en $\mathcal{O}(n^2)$ testait tous les couples (i,j) possibles. Une version améliorée basée sur un pré-calcul (tri du tableau) permettait d'améliorer la complexité en $\mathcal{O}(n \log(n))$.

III.1. Proposer un algorithme qui résout le problème en $\mathcal{O}(n)$ opérations. (Justifier avec soin la réponse)