Examen du 12 décembre 2023

Consignes.
L'examen dure 2 heures. Aucun document n'est autorisé.
Le sujet est constitué de trois exercices indépendants. Toute question non résolue peut être admise dans la suite.
Le barème total est sur 23 points.
Les réponses doivent être justifiées et correctement rédigées.

Exercice 1 (sur 5 pts). *C'est la clé !*
Soit $C = \{1, 2, 5, 6, 7, 9, 10, 12, 15, 17\}$ un ensemble de clés.

1. Dessiner un arbre binaire de recherche (ABR) de hauteur minimale contenant les clés de C et *justifier qu'il n'en existe pas de hauteur plus petite.*
2. Dessiner l'ABR obtenu après suppression de la racine de l'ABR précédent avec l'algorithme du cours.
3. Représenter un tas contenant les éléments de C, sous forme de tableau et sous forme d'arbre binaire.
4. Représenter le tas obtenu après insertion d'un nouvel élément 20 dans le tas précédent avec l'algorithme du cours, sous forme de tableau et sous forme d'arbre binaire.
5. On insère les clés de C par ordre croissant dans un table de hachage de taille $m = 5$ avec la fonction de hachage $h(x) = ((3x + 2) \bmod 29) \bmod 5$, en résolvant les collisions par chaînage. Donner l'état de la table après toutes les insertions.

Exercice 2 (sur 8 pts). *Somme maximale*
Étant donné un tableau T d'entiers, on cherche à trouver deux indices $0 \le i < j \le n$ tel que la somme $\Sigma_{[i,j[} = \sum_{k=i}^{j-1} T_{[k]}$ soit la plus grande possible. Par exemple, si $T = [2, 18, -22, 20, 8, -6, 10, -24, 13, 3]$, on doit renvoyer les indices $i = 3$ et $j = 8$, qui font une somme de $20 + 8 - 6 + 10 = 32$.

1. **Algorithme quadratique.**
   i. On fixe j. Écrire un algorithme qui, étant donné T et j, renvoie le couple $(i, \Sigma_{[i,j[})$ où $\Sigma_{[i,j[}$ est maximale. *L'algorithme doit être de complexité linéaire.*
   ii. En déduire un algorithme de complexité quadratique qui renvoie le triplet $(i, j, \Sigma_{[i,j[})$ qui maximise $\Sigma_{[i,j[}$.
2. **Diviser pour régner.**

i. On fixe un indice de passage m : on cherche la somme maximale $\Sigma_{[i,j[}$ où $i < m < j$. Écrire un algorithme qui, étant donné T et m, renvoie le triplet $(i, j, \Sigma_{[i,j[})$ où $i < m < j$ et $\Sigma_{[i,j[}$ est maximale. *L'algorithme doit être de complexité linéaire.*

ii. En déduire un algorithme de type « diviser pour régner » qui, étant donné T, calcule le triplet $(i, j, \Sigma_{[i,j[})$ qui maximise $\Sigma_{[i,j[}$. *Indication. Si on fixe un indice m, soit $i < j \leq m$, soit $m \leq i < j$, soit $i < m < j$. Déterminer une valeur de m intéressante et traiter les trois cas.*

iii. Analyser la complexité de l'algorithme.

3. Programmation dynamique.

i. Pour $j = 1$ à n, on note $M_j = \max\{\Sigma_{[i,j[} : 0 \leq i < j\}$. Montrer que $M_j = \max(T_{[j-1]}, T_{[j-1]} + M_{j-1})$.

ii. En déduire un algorithme de programmation dynamique qui, étant donné T, renvoie la valeur de la somme maximale $\Sigma_{[i,j[}$ et analyser sa complexité. *L'algorithme doit utiliser le moins d'espace supplémentaire possible.*

iii. Modifier l'algorithme précédent pour qu'il renvoie également les indices i et j qui maximisent la somme $\Sigma_{[i,j[}$.

Exercice 3 (sur 10 pts). *Remplissage de boîtes*

Étant donné un tableau de n entiers positifs T et une *taille de boîte* b, on cherche à répartir les éléments de T dans k boîtes de taille b chacune, en minimisant k. Une boîte peut contenir autant d'entiers qu'on souhaite, mais leur somme ne doit pas dépasser b.

Une *répartition* des entiers de T dans k boîtes est représentée par un tableau R d'entiers entre 0 et $k-1$ tel que $R_{[i]} = j$ signifie « l'objet i est placé dans la boîte j ». On note $\Sigma_T = \sum_{i=0}^{n-1} T_{[i]}$ et pour tout j, on note $\Sigma_j = \sum_{i:R_{[i]}=j} T_{[i]}$ la somme des éléments en boîte j. L'objectif est donc de calculer une répartition R *valide* (c'est-à-dire telle que $\Sigma_j \leq b$ pour tout j) en minimisant la valeur de k.

Exemple Soit $T = [5,7,2,6,1,8]$ et $b = 10$. Une répartition possible est $[0,1,0,2,1,3]$ qui correspond aux boîtes $\{5,2\}$, $\{7,1\}$, $\{6\}$ et $\{8\}$.

1. Existe-t-il une répartition en trois boîtes ? *Justifier.*

Résolution optimale Pour résoudre le problème de manière optimale, on utilise un algorithme de recherche exhaustive en parcourant toutes les répartitions R possibles.

2. Combien y a-t-il de répartitions avec k boîtes pour un tableau de taille n, si on ne tient pas compte de la borne b ?
3. L'algorithme de recherche exhaustive est le suivant. On essaie toutes les répartitions avec k boîtes, pour des valeurs de k croissantes. On teste la validité de chaque répartition et on s'arrête dès qu'on en a trouvé une valide.
   - **i.** Quelle est la complexité du test de validité d'une répartition dans k boîtes ?
   - **ii.** Soit k_0 la valeur finale de k, telle qu'on a trouvé une répartition de taille k_0. Montrer que la complexité de l'algorithme complet de recherche exhaustive est $O(nk_0^{n+1})$.

Résolution approchée On propose maintenant un algorithme glouton pour résoudre le problème. On parcourt les éléments de T dans l'ordre, et on affecte l'objet i à la première boîte dans laquelle il rentre.

RÉPARTITION(T, b):

```
R, Σ ← tableaux de taille n = #T, initialisés à 0
Pour i = 0 à n − 1 :
  j ← 0
  Tant que Σ[j] + T[i] > b : j ← j + 1
  Σ[j] ← Σ[j] + T[i] ; R[i] ← j
Renvoyer R
```

4. i. Appliquer l'algorithme sur l'entrée $T = [5, 2, 4, 3, 4]$ avec $b = 9$. Fournit-il une solution optimale ?
   ii. Quelle est la complexité de RÉPARTITION ?
5. Soit R la solution renvoyée par RÉPARTITION, et $k = \max_i R_{[i]}$ le nombre de boîtes. On note k_{OPT} le nombre de boîtes dans une solution optimale.
   i. Montrer qu'il y a au plus une boîte j telle que $\Sigma_j < \frac{b}{2}$.
   ii. Montrer que $k_{\text{OPT}} \geq \Sigma_T / b$.
   iii. En déduire que $k \leq 1 + 2k_{\text{OPT}}$.
6. On suppose que $T_{[i]} \leq \frac{b}{2}$ pour $0 \leq i < n$. On garde les notations R, k et k_{OPT}.
   i. Montrer qu'il y a au plus deux boîtes j telles que $\Sigma_j \leq \frac{2b}{3}$. *Indication. Montrer que si $\Sigma_j \leq \frac{2b}{3}$, les boîtes $\ell > j$ ne contiennent que des entiers $> \frac{b}{3}$.*
   ii. En déduire une borne sur k en fonction de k_{OPT}.