L3 Math-Info, Algorithmique: examen terminal

7 décembre 2021

Une page recto-verso manuscrite autorisée.
Téléphones portables et autres outils de télécommunication interdits.
Le barème sur 29 produira la note sur 20, il n'est donc pas nécessaire de traiter tout le sujet pour avoir la note maximale.
La qualité de la rédaction sera évaluée au même niveau que l'exactitude des réponses.
Durée : 2h.

Exercice 1. Tri par tas (4 points)

On considère le tableau : $(7, 3, 9, 1, 2, 6)$. On rappelle qu'à tout tableau correspond un unique arbre binaire presque complet, tel que le fils gauche de `T[i]` est `T[2i+1]` et le fils droit de `T[i]` est `T[2i+2]`.

a. (1pt) Représenter l'arbre binaire presque complet correspondant à ce tableau

b. (1pt) Tracer l'exécution de ConsTas transformant cet arbre binaire presque complet en un tas.

c. (2pt) Tracer l'exécution de l'algorithme de tri par tas sur ce tas.
Pour ces deux questions, on ne donnera à chaque étape que le tas et l'état courant du tableau.

Exercice 2. Algorithme mystère (6 pts)

On a retrouvé l'algorithme suivant où l'entrée x et la sortie z sont des éléments d'un anneau $\mathbb{A}$, mais on ne sait plus bien ce qu'il fait.

Algorithm 1 `AlgoMystere`

Entrée: $x \in \mathbb{A}$
Entrée: $n \in \mathbb{N}$
Sortie: $z \in \mathbb{A}$

```
if n = 0 then
    return 1
else
    y ← AlgoMystere(x, n/2)
    z ← y × y
    if n mod 2 = 1 then
        z ← z × x
    end if
    return z
end if
```

a. (1.5pt) Que fait-il ? Quel est son nom ?

b. (1.5pt) Prouver qu'il fait bien cela.

c. (1.5pt) Quel est son coût en nombre d'opérations dans $\mathbb{A}$?

d. (1.5pt) Si $\mathbb{A} = \mathbb{Z}$, quel est son coût binaire ?

Exercice 3. Recherche Dichotomique (3 points)

On considère un cache de taille Z avec des lignes de longueur L, tel que Z soit un multiple de L.

a. Quel est le nombre de défauts de cache de l'algorithme de recherche dichotomique dans un tableau trié de n entiers ? On pourra distinguer le cas où le tableau tient en cache ($n < Z$) du cas où il dépasse du cache ($n > Z$).

Exercice 4. Programmation dynamique (8 points)

Un distributeur automatique de boissons doit rendre la monnaie avec la contrainte de donner un nombre minimal de pièces à chaque fois. On supposera que le distributeur dispose toujours d'une quantité suffisante de pièces de chaque type.

On rappelle dans le tableau suivant les différentes pièces existantes pour les trois principales monnaies du moment.

euro €	0,01	0,02	0,05	0,10	0,20	0,5	1	2
dollar $	0,01	0,05	0,10	0,25				
brousouf ß	1	10	17	31	42			

On rappelle qu'une stratégie gloutonne consiste à donner d'abord le plus de pièces ayant la plus forte valeur, puis de celles de valeur immédiatement inférieure, etc.

a. (1.5pt) Pour chacune des ces monnaies, dire si une stratégie gloutonne permet de rendre la monnaie de façon optimale. Donner un argument pour les cas positifs, et un contre-exemple pour les cas négatifs.

Pour le ou les cas où la solution par algorithme glouton n'existe pas, on se propose de résoudre ce problème par programmation dynamique. On se placera désormais dans un cas général : il y a k types de pièces, de valeur croissante : $v_1 < v_2 < \cdots < v_k$. On notera s la somme que doit rendre le distributeur. On cherche donc à calculer des quantités de chaque type pièces à rendre $n_1, n_2, \ldots, n_k$ tels que $\sum_{i=1}^{k} n_i$ soit minimal et $\sum_{i=1}^{k} n_i v_i = s$. On notera $N(s,k)$ ce nombre minimal de pièces à rendre pour atteindre la valeur s avec les k types de pièces les plus faibles.

b. (1,5pt) Justifier que

$$N(s,k) = \min_{0 \leq i \leq \lfloor s/v_k \rfloor} (i + N(s - iv_k, k-1)), \quad N(0,k) = 0, \; N(s,0) = +\infty$$

c. (2pt) En déduire un algorithme récursif avec mémoïsation calculant la valeur de $N(s,l)$.

d. (2pt) Proposer une variante itérative calculant $N(s,k)$ ainsi que les informations nécéssaire à reconstruire la solution.

e. (1pt) Donner un algorithme calculant les nombres de pièces $n_1, \ldots, n_k$ d'une solution optimale.

Exercice 5. Transposition de matrices (8 points)

On s'intéresse au coût en défauts de cache des algorithmes calculant la transposée d'une matrice : $B \leftarrow A^T$ où A est de dimensions $m \times n$. Le modèle de cache est celui vu en cours : on suppose un cache à 1 niveau, de taille Z avec des lignes de longueur L et L divise Z.

Les matrices sont stockées en mémoire dans un tableau unidimensionnel par lignes : pour une matrice de dimensions $m \times n$, le coefficient (i,j), avec $i \in \{0 \ldots m-1\}$ et $j \in \{0 \ldots n-1\}$ est stocké en position $i \times n + j$ du tableau.

a. (0,5pt) Combien de lignes de cache sont occupées par les entrées-sorties ? En déduire une borne inférieure du coût en défauts de cache pour cette opération.

b. (0,5pt) Proposer un algorithme (naïf) traitant la matrice A ligne à ligne.

c. Donner son coût en défauts de cache selon que

1. (0.5pt) les deux matrices tiennent dans le cache ;
2. (1pt) une ligne de A et une colonne de B tiennent dans le cache (peuvent être chargées sans donner lieu à une eviction LRU) ;
3. (1pt) une colonne de B ne tient pas dans le cache (son chargement entraine au moins une éviction LRU) ;

Pour chaque cas, préciser comment s'exprime cette condition comme une relation entre n, L et Z.

d. (1,5pt) Proposer un algorithme cache aware, basé sur une découpe en blocs $K \times K$. On supposera que K est suffisament petit pour que 2 blocs $K \times K$ tiennent dans le cache.

e. (1pt) Analyser son coût en défauts de cache.

f. (1pt) Proposer une variante récursive cache-oblivious.

g. (1pt) Quel est son coût en nombre d'affectations ?

h. (Bonus 1pt) Analyser son coût en défaut de cache.