Feuille d'exercices n°11 - Schémas algorithmiques 1/2

Notions abordées

- Algorithmes gloutons optimaux, ensemble dominants, preuve par argument d'échange, algorithme gloutons d'approximation
- Algorithmes de programmation dynamique : calcul d'une valeur optimale, calcul d'une solution optimale, réduction de la complexité spatiale si possible
- Algorithmes "diviser pour régner", cas où l'on traite tous les sous-problèmes, cas où l'on en traite qu'un, coût de scission et coût de fusion.

Lorsqu'on demande de modéliser le problème comme un problème d'optimisation il faut identifier :

→ quels objets mathématiques représentent au mieux les données du problème ;

→ quel objet mathématique permet de représenter une éventuelle solution ;

→ à quelles conditions un tel objet représente une solution valide, *i.e.* identifier les contraintes ;

→ quel serait le coût d'une telle solution, autrement dit identifier la fonction objectif.

Exercice 1 Playlist en espace mémoire limité

On cherche à créer une playlist qui occuperait un espace mémoire restreint (CD, MP3, place restant sur un smartphone après installation de tout un tas d'applications...). On a à notre disposition une grande bibliothèque de morceaux dont on connaît la taille en mémoire, et la durée en minutes. On ne suppose pas que ces deux grandeurs sont corrélées, car certains morceaux peuvent être compressés, d'autres au contraire en haute définition...

Question 1
On cherche d'abord à créer une playlist avec un maximum de morceaux. Formuler ce problème comme un problème d'optimisation.

Question 2
Proposer un algorithme de complexité pire cas en $\Theta(n \log n)$ pour résoudre ce problème.

Question 3
Justifier que l'algorithme proposé est correct.

Question 4
On cherche maintenant à créer une playlist de durée maximale. Formuler ce problème comme un problème d'optimisation.

Question 5
L'algorithme précédent est-il encore correct ? Peut-il être adapté ? Si oui dire comment, sinon dire pourquoi.

Exercice 2 Chercher une pièce dans un lot de pièces

On cherche à retrouver une pièce défectueuse parmi n pièces d'aspects identiques. La pièce défectueuse n'est identifiable que par sa masse, en effet elle est plus légère que les autres. On sait qu'il n'y a qu'une seule pièce défectueuse. On dispose d'une balance de type Roberval, c'est-à-dire une balance à plateaux qui permet de comparer les masses de deux ensembles d'objets disjoints (en effet on ne peut pas mettre le même objet des deux côtés de la balance à la fois...).

Question 1
On considère deux ensembles disjoints de pièces A et B. Quelles sont les issues possibles d'une comparaison de A et B grâce à la balance Roberval ? En supposant que A et B sont de même cardinal, que peut on déduire concernant la pièce défectueuse dans chacun des cas.

Question 2
En utilisant les remarques faites à la question précédentes, proposer un algorithme efficace pour retrouver la pièce défectueuse. *On attend une explication ou un pseudo-code de l'algorithme.*

Question 3
Quelle est la complexité temporelle de l'algorithme proposé ? Justifier.

Exercice 3 Nombre de combinaisons

Question 1
Pour $(k, n) \in \mathbb{N}^2$, rappeler ce que représente $\binom{n}{k}$.

Question 2
Donner une définition récursive des valeurs $\binom{n}{k}$ pour $(k, n) \in \mathbb{N}^2$.

Question 3
En itérant cette définition récursive sans simplifications, combien de fois est calculée la valeurs $\binom{2}{2}$ pour calculer $\binom{5}{3}$.

Question 4
Proposer un algorithme efficace pour le calcul d'un coefficient $\binom{n}{k}$ pour deux entiers naturels n et k. Préciser de quel paradigme relève cet algorithme, ainsi que ses complexités spatiale et temporelle.

Question 5
Si la complexité spatiale de l'algorithme proposé à la question précédente est de l'ordre de n^2, proposer une variante ayant la même complexité temporelle mais une complexité spatiale de l'ordre de n.

Exercice 4 Immeubles au bord d'un lac

Le service de l'urbanisme d'une cité lacustre décide que tous les immeubles de la rive devraient avoir vue sur le lac depuis le dernier étage au moins. Pour cela il faut détruire certains immeubles existants (il n'est pas possible de supprimer seulement quelques étages à un immeuble). On s'intéresse à un quartier où les immeubles sont alignés les uns derrière les autres, et de hauteurs $(h_i)_{i\in[1..n]}$ en s'éloignant du lac. On connaît de plus $(c_i)_{i\in[1..n]}$ les coûts de destruction de ces immeubles. On cherche à déterminer quels immeubles détruire et quels immeubles garder afin de respecter les directives du service de l'urbanisme, et ce à moindre coût.

Question 1
Modéliser le problème comme un problème d'optimisation.

Question 2
Si ce n'est pas déjà le cas, reformuler le problème comme un problème de **maximisation**. *On peut penser que l'on essaye de maximiser les économies réalisées par rapport au scénario où l'on détruirait tous les immeubles, c'est-à-dire de maximiser les coûts de destruction des immeubles qu'on ne détruit pas.*

Question 3 ✍
Pour $k\in[1..n]$, définir E_k la valeur maximale d'un sous-ensemble d'immeubles de $[1..k]$ ayant tous vue sur le lac et contenant l'immeuble k. *On attend une définition formelle non récursive.*

Question 4 ✍
En quoi les valeurs E_k pour $k\in[1..n]$ permettent-elles de résoudre le problème ?

Question 5 ✍
Donner une définition récursive de E_k pour $k\in[1..n]$.

Question 6 ✍
Proposer un algorithme de complexité temporelle quadratique permettant de résoudre ce problème. Préciser sa complexité spatiale.

Question 7 ✍
Pourrait-on réduire la complexité spatiale de cet algorithme sans dégrader sa complexité temporelle ? Si oui, cette amélioration est-elle encore envisageable si l'on cherche non pas la valeur optimale mais une solution optimale ?

Question 8 ✍
Grâce à une structure de données vue en cours, pourrait-on réduire la complexité temporelle de cet algorithme ? Si oui, cette amélioration est-elle encore envisageable si l'on cherche non pas la valeur optimale mais une solution optimale ?

Exercice 5 Réservation de salle

Une mairie disposant d'une salle polyvalente cherche à attribuer l'occupation de la salle à diverses associations qui la demandent. Les associations demandent la salle sur un ou plusieurs créneaux qui sont caractérisés par leur date de début et leur date de fin. Dans un premier temps, la mairie cherche simplement à satisfaire le plus de demandes possibles.

Question 1 ✍
Formuler ce problème comme un problème d'optimisation.

Question 2 ✍
Proposer un algorithme de complexité pire cas en $\Theta(n \log n)$ pour résoudre ce problème. *On attend un pseudo-code.*

Question 3 ✍
Justifier que l'algorithme proposé est correct.

Question 4 $*$
La mairie souhaite maintenant prendre en compte la taille et l'importance des associations dans l'attribution de la salle. Comment cela peut-il être modélisé ? Est-ce que cela remet en cause l'algorithme proposé ?

Exercice 6 Sous-tableau de somme maximale

Étant donné un tableau d'entiers, le problème du sous-tableau maximum consiste à trouver un ensemble d'indices consécutifs (i.e. un intervalle d'indices) non vide qui maximise la somme des éléments du tableau. Par exemple, le sous-tableau maximum du tableau `[-3, 4, 5, -1, 2, 3, -6, 4]` est le tableau `[4, 5, -1, 2, 3]` de valeur 13.

Question 1 ✍
Formaliser ce problème comme un problème d'optimisation.

Question 2 ✍
Dans un premier temps, on considère l'algorithme de résolution ci-contre. Quelle est la complexité de cet algorithme ?

```
A_max = max A[1], ..., A[n]
pour i allant de 1 à n faire
    pour j allant de i à n faire
        sum ← 0
        pour k allant de i à j faire
            sum ← sum + A[j]
            si sum > A_max
            alors A_max ← sum
retourner A_max
```

Question 3 ✍
Proposer une variante en $\Theta(n^2)$ de cet algorithme.

Dans la suite de l'exercice, on s'intéresse à des algorithmes de type diviser pour régner. Pour un tableau A indicé par $[1..n]$ avec $n > 1$, on note $A_1 = A[1..m]$ et $A_2 = A[m+1..n]$ les deux sous-tableaux définis par $m = \left\lfloor \frac{n+1}{2} \right\rfloor$. On note alors $stm_1(A) = stm(A_1)$ et $stm_2(A) = stm(A_2)$. On introduit aussi $stm_3(A)$ la valeur maximum d'un sous-tableau de A qui couvre à la fois les indices m et $m+1$.

Question 4 ✍
Comment calculer $stm_3(A)$ efficacement ? Quelle est la complexité en fonction de n la taille de A ?

Question 5 ✍
Connaissant $stm_1(A)$, $stm_2(A)$ et $stm_3(A)$, quelle est la valeur de $stm(A)$? Dans le cadre d'une méthode diviser pour régner qui calcule $stm(A)$ en calculant récursivement et $stm_1(A)$ et $stm_2(A)$ en quoi consisterait l'opération de fusion ? Quelle serait la complexité de cette opération ?

Question 6 ✍
Soit T_n le nombre d'opérations élémentaires (additions et comparaisons d'éléments du tableau) que réalise cette méthode pour un tableau de taille n. Donner l'équation de récurrence satisfaite par T_n. En déduire la complexité de la méthode.

On s'intéresse maintenant à une autre approche diviser pour régner afin de réduire encore la complexité. Pour ce faire, on définit, pour un tableau A indicé par $[1..n]$ les valeurs suivantes :

- $pref(A) = \max\{\sum_{k=1}^{i} A[k] | i \in [1..n]\}$
- $suff(A) = \max\{\sum_{k=i}^{n} A[k] | i \in [1..n]\}$
- $tota(A) = \sum_{k=1}^{n} A[k]$

Autrement dit, $pref(A)$ (resp.. $suff(A)$) est la valeur maximum d'un sous-tableau préfixe (resp. suffixe) de A, et $tota(A)$ est la somme des valeurs de A.

Question 7 ✍
Expliquer comment calculer ces valeurs pour un tableau A ?

Question 8 $*$
Quelle est la complexité de l'algorithme diviser pour régner associé ? Justifier.