L3-INFO Algorithmique et Modélisation

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Examen – session 2 – 22 juin 2016 – durée 3 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 : Problème 1 : 6 pts ; Problème 2 : 6 pts ; Problème : 6 pts.

Problème 1 : Mise en boite

Soit O un ensemble de n objets. À chaque objet o_i de $\mathcal{E}$ on associe une taille $t(o_i)$. On souhaite ranger les objets dans une boite de taille T.

Cependant la boite est trop petite pour contenir tous les objets, on cherche donc à mettre un maximum d'objets dans la boite. Annabelle propose l'algorithme suivant :

Algorithme `Remplir` (O)
Données : Un ensemble $O = \{o_1, \cdots, o_n\}$ de n objets avec leur taille
Une taille de boite T
Résultat : Un ensemble maximal d'éléments entrant dans la boite

```
1 if O est non vide
2     i = argmin{t(o_i)}        // o_i est un objet de taille minimale
3     if t(o_i) ⩽ T
4         choisir (o_i)         // o_i est dans la solution proposée par
                                   Anabelle
5         Remplir (O\{o_i}, T − t(o_i))
```

Algorithme 1 : Algorithme de remplissage

Question 1 : Itératif

Mettre cet algorithme sous forme itérative (avec les commentaires et les justifications). On pourra également proposer un pré-traitement.

Question 2 : Optimalité

Démontrer que l'algorithme glouton proposé par Annabelle produit bien une solution maximisant le nombre d'objets dans la boite. Pour cela on exprimera un invariant de l'itération et on détaillera la preuve complète.

Bertrand demande maintenant de trouver un sous-ensemble d'objets qui remplissent au mieux la boite (il n'y a pas d'objet qui dépasse), c'est à dire que la somme des tailles des objets dans la boite soit la plus proche possible de T.

Question 3 : Non optimal pour la place occupée

Montrer que l'algorithme proposé par Annabelle de donne pas une solution optimale pour la boite la mieux remplie.

Question 4 : Énumération

En vous inspirant de l'algorithme d'énumération des parties, proposer un algorithme qui fournit une solution optimale. On justifiera les choix associés aux branchements.

Question 5 : Preuve

Faire la preuve de votre algorithme.

Exercice 2 : Inversions

Une inversion dans un tableau T de taille n est un couple d'indices (i, j) vérifiant $i < j$ et $T[i] > T[j]$, on supposera que les éléments du tableau sont distincts et deux à deux comparables.

Question 1 : Nombre d'inversions

Donner pour un tableau de taille n le nombre minimum d'inversions et fournir un exemple de tableau donnant cette valeur minimale.
Donner pour un tableau de taille n le nombre maximal d'inversions et fournir un exemple de tableau donnant cette valeur maximale.

Question 2 : Algorithme bulldozer

Écrire un algorithme qui calcule le nombre d'inversions en $\mathcal{O}(n^2)$.

Question 3 : Raffinement

Écrire un algorithme, le prouver et évaluer sa complexité, qui calcule le nombre d'inversions en $\mathcal{O}(n \log n)$ (on pourra s'inspirer du principe de diviser pour régner tel qu'il est utilisé dans l'algorithme de tri partition/fusion et on justifiera avec soin la complexité obtenue).

Problème 3 : Chemins minimaux

On se donne un graphe $\mathcal{G} = (X, A)$ avec X un ensemble de n sommets et A un ensemble de m arcs. On suppose que les arcs sont valués, si (i, j) est un arc de $\mathcal{G}$ on note $d_{i,j}$ sa valeur et on suppose que $d_{i,j} \geqslant 0$. La valeur d'un chemin du graphe est la somme des valeurs des arcs composant le chemin.

On se donne s et t deux sommets du graphe.

Question 1 : Algorithme de Dijkstra

Modifier l'algorithme de Dijkstra, donné en annexe, pour obtenir un chemin de s à t de plus petite valeur (on précisera les conditions sous lesquelles un tel chemin existe). Calculer le coût de cet algorithme.

Question 2 : Chemins minimaux

Montrer sur un exemple bien choisi qu'il peut exister plusieurs chemins de valeur minimale.

Question 3 : Compter les chemins minimaux

Modifier l'algorithme de Dijkstra afin d'obtenir tous les chemins de longueur minimale de s à t.

Algorithme de Dijkstra

```
Dijkstra (x : sommet)
   y, z : sommet
   py, pz : entiers positifs
début
   pour tout sommet s : P(s) ← +∞
   F ← FileVide
   Insérer (F, x, 0)
   P(x) ← 0 -- le poids d'un chemin de x à x est 0
   tant que non Estvide(F)
      Extraire (F, y, py) -- (2)
      pour tout successeur z de y
         pz ← py + poids(y,z)   -- poids du chemin passant par y
         si pz < P(z) alors
            P(z) ← pz   -- on fait passer le chemin par y
            Insérer (F, z, pz)
         fin si
      fin pour -- (1)
   fin tant que
fin
```