L3-INFO Algorithmique et Modélisation

UGA UFR IM²AG Université Grenoble Alpes

Examen session 1 – mai 2021 – 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.
En cas de doutes sur l'énoncé, préciser les choix que vous faites sur votre copie et continuer.
Il sera tenu compte de la qualité de la rédaction et de la clarté de la présentation (2 pts).
Le barème indicatif : Exercice 1 : 4 points ; Exercice 2 : 5 points ; Exercice 3 : 4 points ; Exercice 5 : points

---

Exercice 1 : Trouver la route *(∼ 40 mn)*

Au moyen-âge les commerçants transportaient leurs marchandises pour aller d'une ville où ils achetaient de la marchandise à une ville où ils livraient la marchandise. Le transport s'effectuait en plusieurs étapes, le convoi s'arrêtant dans des villes avant de repartir pour l'étape suivante. Les villes, situées à des grands carrefour de routes, exigeaient des taxes dont le montant était fixé par la ville.

L'objectif du commerçant est de trouver une route (suite de villes étapes) minimisant ses dépenses.

1. Modéliser ce problème dans un formalisme de graphe.
2. Proposer un algorithme permettant la résolution de ce problème (*indication : on pourra transformer le problème pour se ramener à un problème connu et utiliser un(des) algorithme(s) classique(s)*)
3. Quel itinéraire doit-on choisir pour transporter des marchandises de Arelatae (Arles) à Portus Namnetus (Nantes) sur le réseau des routes romaines de commerce ?

Ville	Taxe	Villes "voisines"
Aginnum	9	Augustoritum, Burdigala, Tolosa
Arelatae	10	Lugdunum, Narbo
Augustoritum	13	Aginnum, Avaricum, Limonum, Lugdunum
Avaricum	6	Augustoritum, Caesarodunum, Lugdunum
Burdigala	7	Aginnum, Limonum
Caesarodunum	9	Juliomagus, Avaricum, Limonum
Juliomagus	3	Caesarodunum, Portus Namnetus
Limonum	17	Augustoritum, Burdigala, Caesarodunum, Portus Namnetus
Lugdunum	14	Arelatae, Augustoritum, Avaricum
Narbo	8	Arelatae, Tolosa
Portus Namnetus	20	Juliomagus, Limonum
Tolosa	16	Aginnum, Narbo

Exercice 2 : Un tri récursif... *(∼ 50 mn)*

Soit l'algorithme de tri

Tri_2_tiers (T, i, j)
Données : T tableau de n éléments distincts comparables, i, j indices du tableau $1 \leqslant i \leqslant j \leqslant n$
Résultat : Les éléments du tableau T entre les indices i et j sont triés

```
1 if T[i] > T[j]
2     Échange (T, i, j)
3 if j-i+1 >2
4     t = (j − i + 1) div 3   a        // commentaire 0
5     Tri_2_tiers (T,i,j-t)            // commentaire 1
6     Tri_2_tiers (T,i+t,j)            // commentaire 2
7     Tri_2_tiers (T,i,j-t)            // commentaire 3
```

a. **div** est l'opérateur de division entière

Algorithme 1 : Algorithme de tri par tranches de $\frac{2}{3}$

1. Faire un dessin pour illustrer cet algorithme et donner les 4 commentaires manquants.
2. Quels sont les points à démontrer pour prouver la correction d'une procédure récursive (question de cours) ?
3. Faire la preuve que l'exécution de **Tri_2_tiers** $(T, 1, n)$ trie le tableau T
4. Démontrer que le coût de l'algorithme sur un tableau de taille n ne dépend que de n et pas de l'ordre des éléments des éléments dans le tableau (on prendra comme coût de l'algorithme le nombre d'appels à la foncton **Tri_2_tiers**).
5. Établir une équation de complexité vérifiée par $C(n)$ et résoudre cette équation.
6. Comparer cet algorithme avec des algorithmes de tri plus classiques.

Exercice 3 : Trouver les arrêts *(∼ 40 mn)*

Le Professeur Midas conduit une voiture entre Amsterdam à Lisbonne sur l'Européenne E10. Son réservoir d'essence, quand il est plein, contient assez d'essence pour faire D kilomètres, et sa carte lui donne les distances entre les stations-service sur la route. Le professeur souhaite faire le moins d'arrêts possible pendant le voyage.

1. Proposer un modèle associé à ce problème.
2. Donner une méthode efficace grâce à laquelle le Professeur Midas peut déterminer les stations-service où il peut s'arrêter.
3. Démontrer que cette méthode aboutit à une solution optimale. Calculer la complexité de cette méthode

Exercice 4 : La lutte contre l'algorithme de Dijkstra *(∼ 50 mn)*

On considère l'algorithme de Dijkstra, dans une version simplifiée telle que vue en cours, associant à chaque sommet x du graphe $\mathcal{G}$ la valeur minimale des chemins de s à x (vecteur d).

On rappelle les notations utilisées $\mathcal{G} = (\mathcal{X}, \mathcal{A}, \mathcal{V})$ avec $\mathcal{X}$ l'ensemble des sommets, $\mathcal{A}$ l'ensemble des arcs et $\mathcal{V}$ une valuation positive des arcs de $\mathcal{G}$. $v(x_i, x_j) \geqslant 0$ est la valeur de l'arc (x_i, x_j). Par convention on pourrait prendre $v(x_i, x_j) = +\infty$ si x_j n'est pas voisin de x_i.

Algorithme_Dijkstra($\mathcal{G}$,s)
Données : Un graphe $\mathcal{G}$ valué et un sommet s de $\mathcal{G}$
Résultat : Visite tous les sommets du graphe accessibles à partir de $s$et calcule la valeur minimale des chemins de s à tout autre sommet du graphe

$d(s) = 0$
$S \leftarrow \{s\}$
`// ensemble de sommets (chemins) dont la valeur minimale` d `est établie`
repeat
  `Choisir` $y \notin S$ minimisant

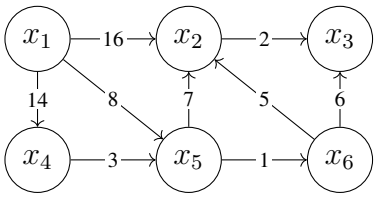
$$\pi(y) = \min_{x \in S, (x,y) \in \mathcal{A}} \{d(x) + v(x, y)\}$$

  $S \leftarrow S \cup \{y\}$
  $d(y) = \pi(y)$
until *il existe* $y \notin S$ *et* $x \in S$ *tels que* $(x, y) \in \mathcal{A}$

Algorithme 2 : Version simplifiée de l'algorithme de Dijkstra

Pour implémenter cet algorithme, on propose d'utiliser une file à priorité $\mathcal{F}$, codée par un tas, pour que la sélection du min et l'insertion soient plus efficaces. La file à priorité est munie de ses opérateurs standards `vide()`, `est_vide(f)`, `inserer(f,e)` et `extraire(f)`.

1. Quels sont les éléments que l'on va insérer dans $\mathcal{F}$ (avec quelle priorité), on justifiera ce choix ?
2. Transformer l'algorithme proposé en utilisant la file à priorité.
3. Donner l'état de la file à priorité et du vecteur d lorsque l'on exécute pas à pas cet algorithme sur l'exemple ci-dessous à partir du sommet source x_1.



4. Calculer le coût au pire de cet algorithme en nombre d'insertion et d'extraction dans la file à priorité.
5. Donner un exemple de pire cas pour $n = 5$, c'est à dire trouver un graphe ayant 5 sommets arcs faisant un maximum de manipulations de la file et de mises à jour.