Algorithmique et complexité

(***éléments de correction supposant un minimum de maitrise du cours***)
Examen 2016-2017
Durée 2 h
Documents autorisés : une feuille A4 recto/verso manuscrite

Vous pouvez répondre aux questions dans n'importe quel ordre, prenez le temps de lire toutes les questions.
Les questions sont parfois détaillées sous forme de plusieurs interrogations pour vous aider à répondre, vous n'êtes pas obligés de répondre à chaque interrogation.
Vous répondrez sur la feuille dans les espaces encadrés.

Questions de cours/complexité (3 points)

Question préliminaire (1 point)

Soit une suite d'instructions I1,I2,I3, …, Ik
Soit C(I1) le cout en calcul de I1, etc
Quel est le cout de la suite ie, C(I1,I2,I3, …, Ik) en fonction de C(I1), C(I2), etc. ?

Cours introduction T 9

Mise en pratique et réflexion(2 points)

Nous avons vu dans le premier cours l'algorithme du tri par tas.
Une première version (algo 1) a été construite, puis une seconde (algo 2).

Algo 1

```
Soit t[1..n] de element les données

p<--0
Tant que p <n faire
        ajouter(t,p,t[p+1]) // on ajoute le nouveau
        p++
FinTantQue

Tant que p>1 faire
        min <--détruire(t,p)
        p--
        t[p+1]<--min
FinTantQue
```

Algo 2

Soit t[1..n] de element les données

Pour p de n div 2 à 1 faire
    Ordonner(t,p)
FinTantQue

p<--n

Tant que p>1 faire
    min <--détruire(t,p)
    p--
    t[p+1]<--min
FinTantQue

La classe de complexité de la partie grisée italique est en O(N $\log_2$ N) avec N la taille des données. Que peut-on dire des ***classes de complexités*** respectives de ces deux algorithmes (algo 1 et algo 2), comment les comparer (laquelle est plus faible que l'autre, etc.) ? A quelle condition le second sera plus performant que le premier ?

au mieux les algorithme auront une classe de complexité en O(N $\log_2$ N) à condition que la 1ère partie soit d'une classe de complexité inférieure. Il faut que Ordonner ait une complexité inférieure à ajouter

Approche force brute (10 points)

On souhaite résoudre le problème suivant (pas la classe des problèmes de ce type) avec une approche « force brute ».

DEUX+CINQ=SEPT

	D	E	U	X
+	C	I	N	Q
	S	E	P	T

X + Q|10| = T ?

?

Chaque chiffre est remplacé par une lettre, ce problème peut alors se ramener à un système de relations entre lettres sous la forme X+Q mod 10 = T, U+N+(X+Q) div10=P, etc
Attention rien ne dit qu'un même chiffre n'est pas associé à deux lettres différentes.

Il s'agit de trouver à quoi corresponde chaque lettre en terme de chiffre pour que la somme soit juste. Une solution évidente est que toutes les lettres correspondent à la valeur nulle.

On souhaite résoudre le problème en faisant appel à un algorithme « force brute ».

Questions introductives (3 points)

1) Qu'est-ce qu'un algorithme de force brute ? Quel est son principe général ?

Cf cours T 3et 4

2) Quelles sont les lettres représentant l'espace d'états (par opposition à celle se déduisant du choix de deux autres) ? Combien y en a t-il ? (Dit autrement quelles sont les inconnues du système)

DEUXCINQ (en toute généralité)

3) Combien de valeurs peuvent prendre chacune de ces lettres ?

10

4) Quelle est la taille de l'espace d'état, la complexité d'un algorithme de force brute sur le problème ?

10^8

Construction de l'algorithme (1+3+3 points)

5) On représente chaque lettre du problème à trouver par une variable du même nom. Par exemple la lettre I sera associé à une variable I (ou i) dans l'algorithme. Comment itérer sur cet espace d'états (cf questions 1 et 2) ?
Remarque : la manière la plus simple n'est peut être pas la meilleure mais si elle permet de répondre à la question elle est satisfaisante *ici*.

En itérant sur chaque lettre sur les 10 valeurs

6) On suppose l'existence d'une fonction OK qui, étant données les valeurs de ces lettres (cf point 1) représentant l'espace d'états, retourne un booléen indiquant si ces valeurs sont solutions du problème ou non.
Donner l'algorithme de force brute qui utilise cette méthode/fonction

```
Pour d de 0 à 9
    pour e de 0 à 9
    pour u de 0 à 9
    pour x de 0 à 9
```

```
pour c … etc
        pour q de 0 à 9
        si ok(d,e,u,x,c,i,n,q) alors memoriser(d,e,u,x,c,i,n,q))

fpour ...
```

7) Ecrire la méthode ok (définir les paramètres et leur type, le type du résultat et l'algorithme associé, ...)

méthode ok(d,u,x,c,i,n,q : entier) : booleen

Retourner X+Q mod 10 = T et U+N+(X+Q) div10=P et …. (cf l'addition)

Algorithme glouton (7 points)

Questions préliminaires (3 points)

1) Pour quel(s) type(s) de problème utilise t-on un algorithme glouton ? Quelle est son principe général ?

Cours T2

2) Un algorithme glouton peut être considéré comme construit en 2 parties, la première effectue un traitement sur les données selon un certain critère, la seconde construit la solution; donnez cet algorithme (choisissez la façon de l'écrire qui vous semble la plus opportune[1])

cours T13

1 L'algorithme suivant n'est pas une réponse satisfaisante (on considère A comme les données)
traiter(A)
s<-- contruireSolution(A)

3) Que pouvez vous dire de la complexité d'un algorithme glouton en général ?

Cours T14

Application à un problème

Enoncé

On considère le problème du sac à dos formulé ainsi :

soit

- un ensemble de n items de poids w_i (w=weight) et de cout p_i (p=profit)
- C la « capacité » maximale du sac à dos
- x_i une variable valant 1 si le $i^{ème}$ item est pris (0 sinon)

L'objectif est de maximiser ou de minimiser une fonction (par exemple $\sum_{i=1}^{n} p_i x_i$ on veut le maximum de profit sous la contrainte $\sum_{i=1}^{n} c_i x_i < C$)

Soit les données suivantes :

C = 100

i	**Wi**	**Pi**
1	100	40
2	50	35
3	45	18
4	20	4
5	10	10
6	5	2

Résoudre le problème consiste donc à fournir pour chaque item i une valeur de x_i et vérifier si les contraintes sont respectées.
Une solution potentielle du problème est par exemple

	Wi	Pi	Solution quelconque
1	100	40	1
2	50	35	1
3	45	18	0
4	20	4	0
5	10	10	1
6	5	2	1
Total poids (wi)			100+50+10+5 =165
Total coût (pi)			40+35+10+2 = 87

Mais cette solution ne respecte pas la contrainte sur C et n'est donc pas bonne.

Application(s) (2+2 points)

	Wi	Pi	Selon Wi	Selon Pi
1	100	40		
2	50	35		
3	45	18		
4	20	4		
5	10	10		
6	5	2		
Total poids (wi)				
Total coût (pi)				

Il vous est demander d'appliquer à ce problème une approche gloutonne. Attention, nous appliquons strictement l'approche gloutonne (nous verrons après si l'optimum est atteint).

Nous allons considérer successivement deux critères pour cette mise en œuvre : p_i (plus fort d'abord) puis w_i (plus faibles poids d'abord).

4) On considère une approche gloutonne qui cherche à maximiser le cout(profit) en prenant les items les plus profitables (donc p_i le plus fort d'abord)

   Si l'ensemble initial, A, des données se décrit par les indices i dans l'ordre suivant

(1,2,3,4,5,6) quel sera l'ordre des indices après traiter(A) ? commentaire?

Il s'agit d'appliquer le cours T13

Construisez alors la solution en appliquant l'algorithme glouton en complétant le tableau ci-dessus avec les valeur de xi pour chaque item et les valeurs totales

5) On considère maintenant une approche gloutonne qui cherche à mettre un maximum d'items dans le sac en prenant les plus légers d'abord (donc w_i le plus faible)

   Si l'ensemble initial, A, des données se décrit par les indices i dans l'ordre suivant (1,2,3,4,5,6) quel sera l'ordre des indices après traiter(A)? commentaire?

Il s'agit d'appliquer le cours 13

Construisez alors la solution en appliquant l'algorithme glouton en complétant le tableau ci-dessus avec les valeur de xi pour chaque item et les valeurs totales

Question Bonus (max 2 points) :

On peut remarquer que l'optimum n'est pas atteint, pourriez vous (tenter d') expliquer pourquoi ?

La structure sous jacente n'est pas un matroïde