Bloc 2: DM d'Algorithmique

Exercice 1. **Calculs de O**

✎ Pour les paires de fonctions (f, g) suivantes, est-ce que $f(n) = O(g(n))$? Et $g(n) = O(f(n))$?

a. $f(n) = n-2$ et $g(n) = n+3$ **b.** $f(n) = \sqrt{n}$ et $g(n) = n^{3/4}$ **c.** $f(n) = \sqrt{n}$ et $g(n) = \sqrt{\log n}$
d. $f(n) = n^{1,01}$ et $g(n) = n\log^2 n$ **e.** $f(n) = 2^n$ et $g(n) = 3^n$ **f.** $f(n) = n^3$ et $g(n) = 8^{\log n}$

Exercice 2. **Combien de temps ?**

✎ Établir la complexité en temps des trois algorithmes suivants (les opérations élémentaires ont été omises). Vous justifierez rapidement vos réponses et pourrez utiliser le 'Master Theorem' si besoin.

```
1 Algorithme : ALGO1(n)
2 pour i de 0 à n−1 faire
3     pour j de 0 à n−1 faire
4         <op elem>
5 pour i de 0 à n−1 faire
6     <op elem>
```

```
1 Algorithme : ALGO2(n)
2 <op elem>
3 tant que n > 1 faire
4     <op elem>
5     n ← n/2;
```

```
1 Algorithme : ALGO3(n)
2 si n = 0 alors return
    val;
3 <op elem>
4 pour i de 0 à n−1 faire
5     <op elem>
6 ALGO3(n−1);
7 <op elem>
```

```
1 Algorithme : ALGO4(n)
2 <op elem>
3 pour i de 0 à 4 faire
4     ALGO4(⌈n/2⌉)
5 <op elem>
```

Exercice 3. **Binci Autoroutes**

On se donne un ensemble de villes $V = \{v_1, \ldots, v_8\}$ ainsi que le coût de construction $c(v_i, v_j)$ d'une portion d'autoroute entre les villes v_i et v_j, pour $1 \leq i \leq j \leq 8$. Ces coûts, en dizaines de millions d'euros, sont résumés dans la matrice suivante :

$$(c(v_i, v_j))_{1\leq i\leq j\leq 8} = \begin{pmatrix} 0 & 7 & 5 & 11 & 9 & 7 & 9 & 10 \\ . & 0 & 8 & 6 & 7 & 11 & 11 & 12 \\ . & . & 0 & 5 & 4 & 10 & 12 & 10 \\ . & . & . & 0 & 7 & 5 & 12 & 9 \\ . & . & . & . & 0 & 7 & 12 & 10 \\ . & . & . & . & . & 0 & 11 & 11 \\ . & . & . & . & . & . & 0 & 8 \\ . & . & . & . & . & . & . & 0 \end{pmatrix}$$

✎ Comment construire un réseau autoroutier le moins cher possible permettant de rallier n'importe quelle ville depuis n'importe quelle autre ? Modéliser le problème sous forme de graphe puis appliquer un algorithme pour le résoudre en explicitant (rapidement) les étapes du déroulement de l'algorithme sur cet exemple.

Exercice 4. **Enregistrement de fichiers**

On souhaite enregistrer sur un disque externe de capacité L un ensemble de fichiers $(P_1, \ldots, P_n)$. Chaque fichier P_i a une taille a_i. On suppose que $\sum_{i=1}^{n} a_i > L$ (on n'a pas la place suffisante pour enregistrer tous les fichiers sur le disque) et aussi qu'il existe i tel que $a_i \leq L$ (au moins un fichier tient sur le disque dur).

Dans un premier temps, on veut une stratégie *glouton1* qui va maximiser le nombre de fichiers sauvegardés sur le disque.

1. On suppose que a_1 est la plus petite taille de fichier parmi $\{a_1, \ldots, a_n\}$.
   (a) Montrer qu'il existe une solution optimale qui contient P_1.
   (b) En déduire qu'il existe une solution optimale qui est formée de P_1 et d'une solution optimale au problème consistant à stocker $(P_2, \ldots, P_n)$ sur un disque de capacité $L - a_1$.

2. Écrire un algorithme glouton pour résoudre le problème. On pourra commencer par trier les fichiers $(P_1, \ldots, P_n)$ par taille croissante.
3. À l'aide de la question 1.(a), prouver la validité de votre algorithme. On pourra utiliser un résultat du cours.
4. Calculer une borne sur la complexité en temps de votre algorithme.

On veut maintenant changer d'objectif et minimiser la place inutilisée sur le disque.

5. Donner un exemple sur lequel l'algorithme de la question précédente ne donne pas la meilleure solution.
6. On veut essayer une autre approche gloutonne, nommée *glouton2*, pour ce problème : on traite les fichiers par ordre de taille décroissante, et pour chaque i dans cet ordre, on ajoute P_i à la solution si il reste assez d'espace pour celà. Donner le pseudo code de l'algorithme correspondant.
7. Montrer sur un exemple que cette stratégie n'est pas non plus la meilleure possible.
8. *(un peu dur...)* Prouver par contre que la stratégie *glouton2* permet de remplir au moins la moitié de l'espace que prendrait une solution optimale. Pour cela, on pourra considérer une solution optimale $\mathcal{O}$ ainsi que le premier fichier sélectionné par par la stratégie *glouton2* qui n'appartient pas à $\mathcal{O}$.
9. *(un peu moins dur mais tout de même...)* Donner un exemple dans lequel la stratégie *glouton2* donne à peine mieux que la moitié d'une solution optimale ('à peine mieux' voulant dire 'aussi petit qu'on le souhaite').

Exercice 5. **Élément majoritaire**

On considère un tableau T de taille n contenant des entiers. On dit que l'élément p de T est *majoritaire dans* T si il est contenu dans strictement plus de $\lfloor n/2 \rfloor$ cases de T. Le but de l'exercice est d'écrire un algorithme qui retourne l'indice d'un élément majoritaire de T si celui-ci en contient un et -1 sinon.

1. On veut d'abord un algorithme simple pour résoudre le problème.
   (a) Écrire un tel algorithme : pour chaque indice i de T avec $0 \leq i \leq n-1$ on comptera le nombre d'éléments de T qui sont égaux à $T[i]$, puis on concluera.
   (b) Borner la complexité en temps de votre algorithme.
2. On veut maintenant élaborer un algorithme de type 'diviser-pour-régner'. On note par T_1 le tableau formé par les $\lfloor n/2 \rfloor$ premières cases de T et par T_2 le tableau formé par les cases restantes de T, c'est-à-dire $T_1 = T[0, \lfloor n/2 \rfloor - 1]$ et $T_2 = T[\lfloor n/2 \rfloor, n-1]$
   (a) Montrer que si T a un élément majoritaire p, alors p est aussi élément majoritaire de T_1 ou élément majoritaire de T_2.
   (b) En supposant que l'algorithme aît été appliqué correctement sur les deux sous-tableaux T_1 et T_2, expliquer comment trouver un élément majoritaire pour T si il en possède un.
   (c) En déduire l'écriture d'un algorithme récursif pour le problème.
   (d) Écrire l'équation de récurrence vérifiée par le temps mis par l'algorithme à s'exécuter.
   (e) À l'aide du Master Theorem, borner la complexité en temps de votre algorithme.