Université Paris 7 - Licence d'informatique - L2

EA4 – Partiel du 2 mars 2013 - durée : 2h30

La qualité de la rédaction sera prise en compte dans la notation. Justifiez toutes vos réponses et expliquez les fondements de vos algorithmes en français avant de les rédiger en pseudo-code compréhensible et commenté. Le barème est donné seulement à titre indicatif.

Les algorithmes donnés sans commentaires ou sans explications ne seront pas pris en compte par le correcteur. Recopier les algorithmes vus en cours sur la feuille de copie ne vous donnera aucun point.

Exercice 1 (2 points) :

N.	secondes
3125	0.23
15625	9.50
78125	393.02
390625	16038.64
1953125	663526.98

Soit un programme qui utilise une entrée entière N. En observant le temps d'exécution de ce programme pour différentes valeurs de N on obtient le tableau ci-contre. En utilisant ces données, il faut estimer l'ordre de grandeur du temps d'exécution du programme en fonction de N. Pour cela, on suppose que le temps d'exécution est une fonction $T(N)$ de la forme $a \times N^b + c$ avec a, b, c des constantes réelles.

1. Calculez une valeur exacte pour b.
2. Au vu des données du tableau ou de la valeur de b calculée au points précédent, donnez la valeur de vérité les affirmations ci-dessous.

a) $T(N) \in O(N^4)$ b) $T(N) \in \Theta(N)$ c) $T(N) \in \Omega(N^2)$ $T(N) \in O(1/N)$

Exercice 2 (6 points) :

Pour les deux programmes ci-dessous, calculez les classes de complexité en temps $O(f_\ell(N))$ et $\Omega(g_\ell(N))$ pour $\ell \in \{a, b\}$. Justifiez votre réponse.

(a)

```
int t[N], j = 0, i = 1;
while (i <= N) {
   t[j] = i;
   j++;
   i = i * 2;
}
```

(b)

```
int sum = 0;
for (int i = 0; i < N; i++)
    for (int j = i+1; j < N; j++)
        for (int k = i+1; k < N; k++)
            sum++;
```

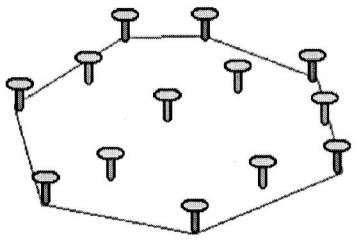
Exercice 3 (4 points) :

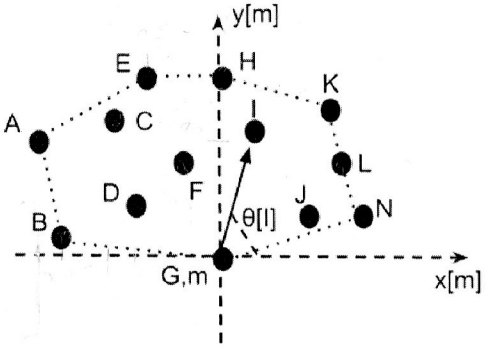
Simuler l'algorithme de `TriFusion` sur le tableau $T = [15, 66, 78, 82, 85, 97, 89, 28, 34, 51]$.

Problème 4 (8 points) :

Etant donnée un ensemble P de points dans un plan, le polygone convexe ayant la plus petite surface et qui entoure ces points s'appelle *"l'enveloppe convexe"* de l'ensemble, qu'on notera $E(P)$. On remarque que les coins de ce polygone sont des points de l'ensemble. Le calcul de cette enveloppe a beaucoup d'applications : trouver les deux villes les plus éloignés sur un carte, calculer le t d'un robot qui doit contourner un obstacle, etc.

Pour calculer l'enveloppe convexe, il faut déterminer quels sont les points de l'ensemble forment les coins du polygone $E(P)$. Un algorithme "mécanique" pour calcul $E(P)$ entoure une corde l'ensemble de points, comme à gauche de la Fig. 1 : les coins de $E(P)$ sont les point





i	0	1	2	3	4	5	m=6	7	8	9	10	11	12
$P[i].nom$	A	B	C	D	E	F	G	H	I	J	K	L	N
$P[i].x$	0,0	0,5	20,5	21,5	23,0	24,0	26,5	26,5	28	32	33	33,5	34
$P[i].y$	20,0	0,5	17,5	7,5	23,0	12,0	0,0	23	17,5	8	18	16	8

FIGURE 1 –

touchent la corde. Un algorithme informatique pour calculer $E(P)$ est l'algorithme de Graham. Dans ce problème, vous allez coder quelques procédures utilisées dans cet algorithme. **Pour chaque procédure, on vous demande de donner le pseudo-code et la complexité asymptotique en temps et en espace.**

Les exemples donnés ci-dessous font référence à l'ensemble de points P à droite de la Fig. 1. Cet ensemble est décrit par un tableau P dont les éléments sont des enregistrements ayant trois champs : le `nom` du point (un caractère), sa coordonnée x (un réel) et sa coordonnée y (un réel). On notera $P[i].x$ pour parler de la coordonnée x du i-ème point en P, et N sera la taille de P.

1. Le tableau P présenté en bas de la Fig. 1 est-il trié par rapport au champ `nom`? Si non, pourquoi. Si oui, dans quel ordre ? La même question pour les champs x et y.
2. La première étape de l'algorithme de Graham calcule l'indice m du point dans le tableau P qui a la plus petite ordonnée y.
   Coder la procédure `plusPetitY` qui prend comme argument le tableau P et renvoie l'indice m ayant la propriété ci-dessus. Donner sa complexité.
3. La deuxième étape remplit un tableau T de même taille que P tel que pour tout point i dans P, $T[i]$ est une valeur dans l'intervalle $[0, 180[$ qui représente l'angle formé par l'axe des x et la droite qui passe par $P[m]$ et $P[i]$. Cet angle est obtenu en appelant une fonction (donnée) $\theta(P, m, i)$; $\theta(P, m, m)$ renvoie 0.
   Coder une procédure `anglesP` qui prend en argument le tableau P et renvoie le tableau T. Donner sa complexité en sachant que le coût en temps de θ est en $\Theta(1)$.
4. Dans l'étape suivante, l'algorithme de Graham trie le tableau T en ordre croissant en changeant de place les enregistrements dans le tableau P tel que, à la fin du tri, pour tout i, le point $P[i]$ a la valeur $T[i]$ qu'il avait avant le tri. Par exemple, si $T = [0, 45, 90, 135, 180, 225, 270, 315]$ et $P = [A, B, C, D, E, F, G, H]$, après le tri, T est en ordre croissant et $P = [A, E, H, C, B, D, F, G]$.
   Coder une procédure `triP` qui prend en argument le tableau P et le tableau T et les trie comme indiqué ci-dessus. Donner sa complexité.
   Indication : Vous devez adapter un algorithme de tri vu en cours. Dans votre choix, privilégiez la simplicité à la performance de l'algorithme !
   Avec quelle condition pour P et T a été déplacé l'enregistrement qui avait initialement l'indice m ?
5. (**) La dernière étape de l'algorithme de Graham parcourt le tableau P en partant de 0 pour sélectionner les points qui constituent les coins de $E(P)$. Le parcours du tableau P est fait tel qu'un point est visité une seule fois et pour chaque point est fait un calcul en $\Theta(1)$. Quelle est la complexité globale de l'algorithme de Graham ?