Algorithmique avancée – Examen Réparti 2
UPMC — Master d'Informatique —
Janvier 2014 – durée 2h

Les seuls documents autorisés sont les polys de cours, ainsi que la copie double personnelle.

1 QCM [10 points]

Dans ce QCM, pour chaque question vous devez donner 1 seule réponse et expliquer votre choix par une ligne de texte ou une figure. Le barême sera le suivant : **Réponse correcte et correctement argumentée : 1 point. Réponse incorrecte ou correcte mais non argumentée : 0 point.** *Il n'y a pas de points négatifs.*

Q1. Le parcours infixe donne la liste des clés en ordre croissant
A) dans un tournoi binomial
B) dans un arbre digital
C) dans un AVL
Sol : C : un AVL est un ABR et c'est dans les ABR que cette prop est vérifiée

Q2. La récurrence $T(n) = 2T(n/2)+n$ a pour solution (et donner un exemple d'algorithme qui a cette complexité)
A) $T(n) = \Theta(n \log n)$
B) $T(n) = \Theta(n)$
C) $T(n) = \Theta(\log n)$
Sol :A : quicksort

Q3. L'arbre binomial $B_1 = o - o$, a pour hauteur 1. Le nombre de feuilles de l'arbre binomial de hauteur k est
A) $2k + 3$
B) 2^{k-1}
C) k^2
Sol : B : par récurrence

Q4. On travaille sur un ensemble ordonné de 10^9 éléments, sur lequel on veut faire de la recherche par intervalle (i.e. rechercher tous les éléments entre 2 bornes). Quelle structure de données utiliser ?
A) arbre AVL
B) hachage dynamique
C) arbre B
Sol : C : beaucoup d'éléments donc recherche externe ; le hachage ne conserve pas les relations entre les éléments, donc arbres B pour la propriété de recherche par intervalle.

Q5. Pour un arbre 2-3-4 contenant n clés. L'algorithme permettant de calculer la hauteur de l'arbre est au pire cas (en nombre de noeuds traversés)
A) $\Theta(n)$
B) $\Theta(\log n)$
C) $\Theta(1)$
Sol : B : il suffit de calculer la hauteur d'une branche

Q6. On considère une table de hachage de taille $2N$ dans laquelle on insère $N/8$ clés, avec une fonction de hachage uniforme. Le nombre moyen de clés qui ont la même valeur de hachage i fixée est
A) 1/16
B) 1/4
C) 1/2

*Sol : A : pour tout élément la proba qu'il soit haché sur i est 1/2N. Le nombre moyen d'éléments hachés sur i est donc N/8 * 1/2N = 1/16*

Q7. Quelle est la longueur du texte compressé par Huffmann statique, pour un texte composé de 5 symboles différents, chacun apparaissant 3 fois
A) 36
B) 40
C) 48
Sol : A : 3x2 + 3x2 + 3x2 + 3x3 + 3x3

Q8. Quel est l'algorithme de compression le plus efficace (quelle que soit l'entrée), par rapport au gain en espace ?
A) Shannon-Fano
B) Huffman statique
C) Huffman dynamique

Q9. La borne inférieure du nombre de comparaisons d'un algorithme calculant le contour positif de l'enveloppe convexe d'un ensemble de n points en procédant par comparaisons est
A) $\Theta(n)$
B) $\Theta(n \log n)$
C) $\Theta(n^2)$
Sol : B : car sinon on peut trier en moins de n log n comparaisons (points sur un cercle et ordre polaire)

Q10. Soit un polygone P dont on sait que l'enveloppe convexe E contient n sommets
A) le polygone P a $2n$ sommets
B) tout point à l'intérieur de l'enveloppe convexe E est à l'intérieur du polygone P
C) tout sommet de l'enveloppe convexe E est un sommet du polygone P

2 Problème [10 points]

Soit S un ensemble de N points du plan. Le but de l'exercice est de donner un algorithme efficace pour calculer le *diamètre* de S, c'est à dire la distance maximale entre 2 points de S. Une paire $\{p, q\}$ de points de S est appelée *diamètre* de S si p et q sont deux points de S à distance maximale. (La distance entre 2 points se calcule en $O(1)$ opérations arithmétiques.)

Q1. Donner un algorithme quadratique pour le calcul du diamètre d'un ensemble de N points.

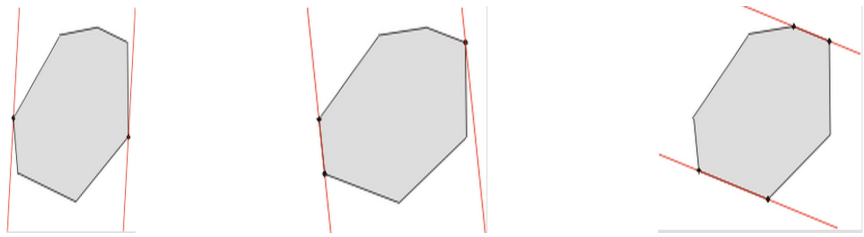
Pour trouver un algorithme efficace on va rechercher un diamètre de l'enveloppe convexe de S.

Q2. Montrer que si une paire $\{p, q\}$ de points de S est un *diamètre* de S, alors p et q sont des sommets de l'enveloppe convexe de S.

Une paire de points $\{p, q\}$ de S est dite *antipodale* s'il existe 2 droites parallèles passant respectivement par p et q et telles que la bande du plan entre ces deux droites contient (au sens large) tous les points de S.
La figure suivante montre des paires de points antipodales sur un polygone convexe P : à gauche les droites parallèles intersectent le polygone en 2 sommets qui forment une paire de points antipodale ; au milieu les droites parallèles intersectent le polygone en 1 sommet et 1 côté (d'où 2 paires de points antipodales) ; et à droite les parallèles intersectent le polygone en 2 côtés (d'où 4 paires de points antipodales).

Q3. Montrer qu'une paire de points antipodale d'un polygone convexe P n'est pas forcément un diamètre.

Q4. Montrer que le diamètre d'un polygone convexe P est égal à la plus grande distance entre deux points d'une paire antipodale de P.



L'algorithme suivant permet de trouver toutes les paires de points antipodales d'un polygone convexe P de $n+1$ sommets. Il prend en entrée la liste des points $C = (p_0, p_1, \ldots, p_n)$ d'un contour positif (sens trigonométrique) de P, et retourne la liste L des paires de points antipodales de P. On utilise la primitive $succ(q)$, qui retourne le point qui est à la suite de q dans le contour direct de P, ainsi que la primitive $ajout(p, q, L)$, qui ajoute la paire de points $\{p, q\}$ à la liste L. On utilise aussi la primitive $Aire(p, q, r)$, qui calcule, en $O(1)$ opérations arithmétiques (et aucune comparaison), l'aire du triangle (p, q, r).

Algorithme : PairesAntipodales(C)

L := ListeVide
$p := p_n$; $q := p_0$
while $Aire(p, succ(p), succ(q)) > Aire(p, succ(p), q)$ **do**
    $q := succ(q)$
end while
$q_0 := q$
while $q \neq p_0$ **do**
    $p := succ(p)$
    $ajout(p, q, L)$
    while $Aire(p, succ(p), succ(q)) > Aire(p, succ(p), q)$ **do**
        $q := succ(q)$
        if $(p, q) \neq (q_0, p_0)$ **then**
            $ajout(p, q, L)$
        else
            Return
        end if
    end while
    if $Aire(p, succ(p), succ(q)) = Aire(p, succ(p), q)$ **then**
        if $(p, q) \neq (q_0, p_n)$ **then**
            $ajout(p, succ(q), L)$
        else
            $ajout(succ(p), q, L)$
        end if
    end if
end while
Return L

La première boucle **Tant Que** permet de trouver le premier point (noté q_0) le plus éloigné de la droite $(p_n p_0)$ quand on parcourt le polygone dans le sens trigonométrique depuis p_0. En effet, d'une part l'aire du triangle pqr est proportionnelle à la distance de p à la droite (qr) ; et d'autre part quand on parcourt le contour positif $(p_0, p_1, \ldots, p_n)$ d'un polygone convexe les aires des triangles $p_0p_1p_2, p_0p_1p_3, \ldots, p_0p_1p_n$ forment une suite qui est d'abord croissante puis décroissante.

Q5. La seconde boucle **Tant Que** permet de trouver toutes les paires de points antipodales : expliquez le code, en particulier les parcours des pointeurs p et q, et la détection des paires antipodales.

Q6. Quelle est la complexité de l'algorithme précédent, comptée en nombre de comparaisons.

On admettra que le nombre maximal de paires antipodales dans la liste L est $2n$.

Q7. Utiliser les résultats précédents pour écrire un algorithme qui calcule le diamètre d'un ensemble S de N points en $O(N \log N)$ comparaisons.