Université Paris Descartes - UFR de mathématiques et informatique - L2

Algorithmique et structures de données

Examen final, 8 janvier 2020 - Durée 1h30 - 5 pages

- Documents et appareils électroniques interdits
- Le barème est donné à titre indicatif et pourra être légèrement modifié.
- Les exercices peuvent être abordés dans n'importe quel ordre, mais les réponses à un même exercice ne doivent pas être dispersées dans la copie (risque de non-correction).
- Le sujet étant décliné en plusieurs versions, il est risqué de s'inspirer de la copie de son/sa voisin.e. La fraude est détectable.

Exercice 1 : Questions de cours (3 points)

a) En notant n la taille des données à traiter, soient les complexités suivantes :

$$4n\,;\quad n^2\,;\quad \log_2(n)+5\,;\quad n-1\,;\quad 2^n\,;\quad n(n-1)$$

Ordonnez ces complexités selon leur ordre de grandeur asymptotique (sans expliquer). Vous utiliserez les signes $<$ et $=$.

b) Donner (sans démonstration) les ordres de grandeur asymptotiques de la complexité moyenne des algorithmes suivants en nombre de comparaisons. Vous préciserez bien le nom des algorithmes sur votre copie.
— recherche séquentielle dans un vecteur trié
— tri fusion
— tri par insertion séquentielle
Pour ce dernier tri, indiquez un moyen de réduire la complexité (on ne vous demande pas d'écrire le nouvel algorithme mais d'expliquer en français le principe général).

Exercice 2 : Représentations d'arbres (*exécution* - 4 points)

Dans cet exercice, pour chaque question, vous donnerez uniquement le résultat final.

a) Transformez l'arbre général de la figure 1 en arbre binaire.

b) Soit un arbre binaire dont les sommets sont étiquetés par des nombres à 1 chiffre et dont les parcours préfixe et symétrique sont respectivement 2918304 et 9123084. Dessinez cet arbre.

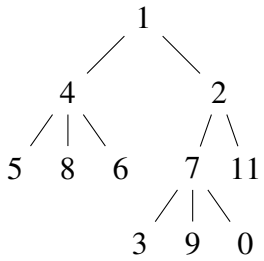

FIGURE 1 –

Exercice 3 - Transformation d'un algorithme itératif en récursif (*conception* - 3 points)

Écrivez une version récursive de l'algorithme 1. Votre algorithme aura 4 arguments en entrée : V, x, inf et sup, en considérant que l'appel initial se fait avec $inf = 1$ et $sup = n$.

Algorithme 1 : Recherche dichotomique de la dernière occurrence dans un vecteur trié

```
début
    /* ENTRÉES : Un vecteur V de taille n, un élément x */
    /* SORTIE : i si x apparait au rang i de V, 0 si x ∉ V */
    inf ← 1, sup ← n, i ← 0
    tant que inf < sup faire
        med ← (inf + sup + 1) div 2
        si x < V(med) alors sup ← med − 1
        sinon inf ← med
    si V(inf) = x alors
        retourner inf
    sinon
        retourner 0
fin
```

Exercice 4 - Démonstration d'une propriété des ABR (*analyse* - 5 points)

On rappelle qu'un arbre binaire de recherche (ABR) est un arbre binaire dont **tout nœud** x vérifie :

- les éléments du sous-arbre gauche de x sont inférieurs ou égaux à x
- les éléments du sous-arbre droit de x sont supérieurs à x

L'arbre vide est considéré comme un ABR.

L'algorithme 2 affiche les nœuds d'un arbre binaire selon le parcours symétrique. Pour tout arbre binaire dont toutes les étiquettes des nœuds sont distinctes, démontrer que si cet algorithme affiche les nœuds de l'arbre en ordre croissant, alors cet arbre est un ABR. La démonstration peut se faire en utilisant le principe d'induction avec la relation d'ordre "est un sous-arbre de" [1].

Algorithme 2 : parcoursSym(A)

```
début
    /* ENTRÉES : un arbre binaire A */
    /* SORTIE : affichage des nœuds selon le parcours symétrique */
    si Non(est_vide(A)) alors
        parcoursSym(G(A))
        affiche(racine(A))
        parcoursSym(D(A))
fin
```

1. En notant $\leq$ cette relation, $A \leq B$ ssi $A = B$ ou $A \leq G(B)$ ou $A \leq D(B)$

Exercice 5 - Complexité d'un algorithme récursif (*analyse* - 5 points)

L'enveloppe convexe d'un ensemble de points du plan est le plus petit polygone convexe contenant ces points, comme illustré par l'exemple de la figure 2.

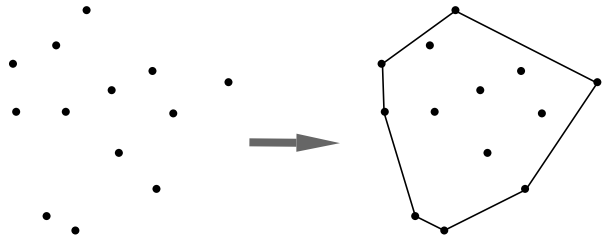

FIGURE 2 – Un exemple d'enveloppe convexe d'un ensemble de points.

Pour le calcul des enveloppes convexes, nous nous intéressons à un algorithme récursif de type diviser pour régner. Si l'ensemble contient un seul point, alors l'enveloppe convexe est ce point. Sinon, on divise l'ensemble des n points en deux presque-moitiés, on calcule l'enveloppe convexe de chacune puis on fusionne les deux enveloppes convexes en une seule.

Plus précisément, les étapes de l'algorithme 3 pour $n > 1$, illustrées par la figure 3, sont les suivantes :

a) On divise l'ensemble P en 2 sous-ensembles : P_1 contenant les $\lfloor n/2 \rfloor$ points de plus petite abscisse ; P_2 contenant les $\lceil n/2 \rceil$ points de plus grande abscisse [2].

b) On calcule l'enveloppe convexe de chacune des presque-moitiés P_1 et P_2, par un appel récursif à l'algorithme.

c) On relie les deux enveloppes convexes ainsi formées par deux segments tangents et on élimine les segments intérieurs inutiles.

Algorithme 3 : enveloppe

```
début
    /* ENTRÉE : Un ensemble P de n points */
    /* SORTIE : l'enveloppe convexe de P */
    si n = 1 alors
        retourner P
    sinon
        (P1, P2) ← divise(P)
        E1 ← enveloppe(P1)
        E2 ← enveloppe(P2)
        retourner fusion(E1, E2)
fin
```

Pour un ensemble de taille n,

— la fonction *divise* requiert au maximum αn comparaisons, avec $\alpha \in \mathbb{R}$;

— la fonction *fusion* requiert au maximum βn comparaisons, avec $\beta \in \mathbb{R}$;

2. On suppose pour simplifier que tous les points ont des abscisses distinctes. Rappel : $\lfloor x \rfloor$ désigne la partie entière inférieure de x, $\lceil x \rceil$ désigne la partie entière supérieure de x.

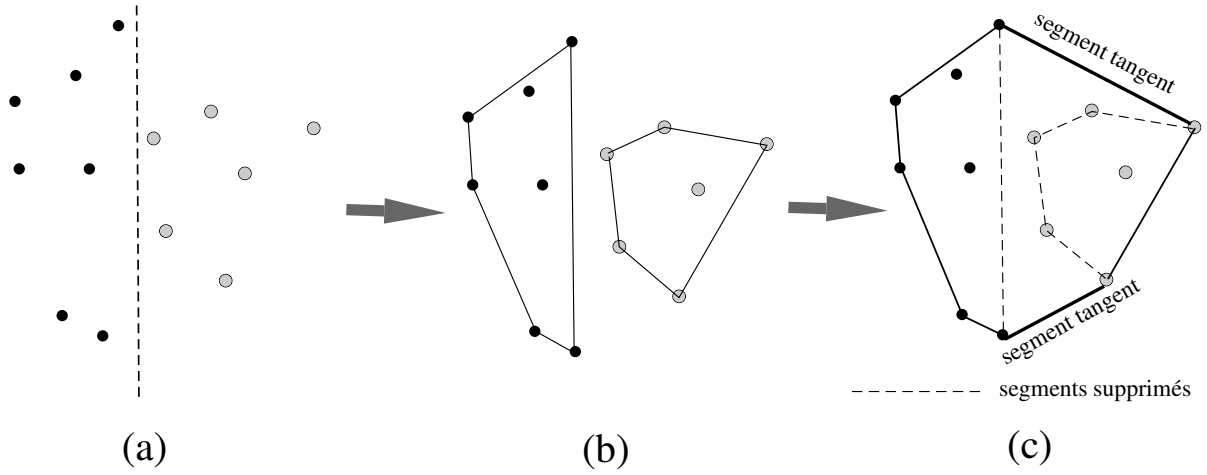

FIGURE 3 – Illustration de l'algorithme récursif "diviser pour régner" de construction de l'enveloppe convexe.

Questions :

a) Exprimer la complexité maximale en nombre de comparaisons, $C(n)$, en fonction de $C\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$, $C\left(\left\lceil \frac{n}{2} \right\rceil\right)$ et n.

b) En approchant n par un nombre de la forme 2^p et en posant $x(p) = C(2^p)$, exprimer $x(p)$ en fonction de p puis $C(n)$ en fonction de n. Il peut être utile de passer par les séries génératrices. On rappelle les formules suivantes :

$$\sum_{p\geq 0} a^p z^p = \frac{1}{1-az}$$

$$\sum_{p\geq 0} (p+1) a^p z^p = \frac{1}{(1-az)^2}$$

Rappel : tous vos calculs doivent être expliqués en français.