Université de Savoie, UFR SFA
INFO626, L3 STIC ISC

11 avril 2017
Durée : 1h30

Examen, INFO626

Documents autorisés : tous documents du cours/td/tp, notes manuscrites (nb : pas de livres)

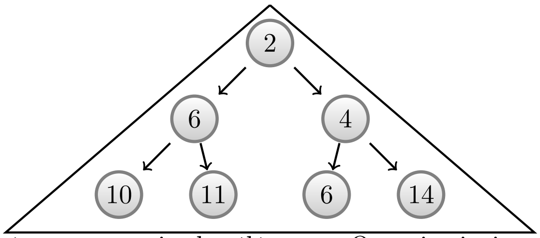
Les exercices sont indépendants. Le barème est indicatif. Il dépasse volontairement 20 pour que vous ayez le choix dans les exercices.

Exercice 1. Tas (/10)

On rappelle qu'un *tas* est un tableau dans les éléments sont organisés de manière spécifique, qui imite un arbre binaire. D'une part, le plus petit élément est à l'indice 1 du tableau et est appelé *racine* (l'indice 0 n'est pas utilisé). D'autre part, un élément à l'indice i du tableau est plus petit que l'élément à l'indice $2i$ et que l'élément à l'indice $2i+1$, s'ils existent. On dit que l'élément à l'indice $2i$ est le *fils gauche* de l'élément à l'indice i, et que l'élément à l'indice $2i+1$ est le *fils droit* de l'élément à l'indice i. Enfin, l'élément à l'indice i est le *père* des éléments aux indices $2i$ et $2i+1$. Notons que la dernière ligne peut être incomplète à droite, si le nombre d'éléments n n'est pas de la forme $2^k - 1$.

Par exemple, le tas ci-dessous stocké sous forme de tableau ainsi à gauche, correspond à la structure hiérarchique à droite :

indices	1	2	3	4	5	6	7	...
valeurs	2	6	4	10	11	6	14	...



Les tas sont très utiles pour faire des files à priorités ou pour trier les éléments. On voit ainsi que le plus petit élément est toujours à l'indice 1. Pour faire un tri, il suffit de sortir cet élément, de mettre le dernier élément du tas à la place, puis de le pousser dans une branche jusqu'à ce que la propriété de croissance soit respectée partout.

1. Si $n = 2^k - 1$ est le nombre d'éléments, combien y a-t-il de lignes dans l'arbre représentant le tas ? Combien y a-t-il de valeurs sur la dernière ligne ?
2. On s'intéresse à la création du tas lorsque l'on insère un nouvel élément dedans (ici des entiers). On peut écrire la procédure Inserer ainsi :

```
// Insère un entier dans une position valide.
Action Inserer( ES T : Tas, E e : entier);
Var : i : entier;
début
    T.dernier ←T.dernier+1;
    T.elems[ T.dernier ] ←e;
    i ←T.dernier /* i est la position courante de e */ ;
    Tant Que i > 1 et T.elems[i] < T.elems[ i/2 ] Faire
        // On le remonte dans le tas en l'échangeant avec son père.
        Echange( T.elems[i], T.elems[ i/2 ] );
        i ←i / 2;
fin
```

Quel est la complexité en pire cas d'une insertion d'un nouvel élément dans un tas qui a déjà m éléments ? Justifiez brièvement.

3. En déduire la complexité en pire cas de la création d'un tas à $n = 2^k - 1$ éléments au total, obtenu en insérant progressivement ces n éléments par n appel de Inserer.
4. On va utiliser une autre stratégie pour l'insertion, qui ne peut marcher que si on connait *a priori* le nombre d'éléments que l'on veut insérer. On l'écrit ainsi avec les actions ConstruitTas et InsèreFin.

`// Crée un tas T formé des` n `éléments du tableau P en les insérant par la fin`

Action CONSTRUITTAS*(S T : Tas, E P[1..n] : Tableau d'entiers, E n : entier);*
Var : j :entier;
début
  CRÉERTASVIDE(T,n) /* Crée un tas vide avec assez d'espace mémoire pour n éléments */
  Pour j *de* n *à* 1 *par pas de* -1 **Faire**
    INSÈREFIN(T, P[j], j, n);
fin

`// Insère` v `dans le tas T en position` i`, en supposant que les éléments entre` $i+1$ `et` n `sont déjà insérés.`

Action INSÈREFIN*(ES T : Tas, E v : entier, E i : entier, E n : entier);*
début
  T.elems[i] $\leftarrow v$;
  Tant Que $2*i+1 <= n$ *et* $v >$ *min(T.elems[*$2*i$*], T.elems[* $2*i+1$ *])* **Faire**
    si $v >$ *T.elems[*$2*i$*]* **alors** $j \leftarrow 2*i$;
    sinon $j \leftarrow 2*i+1$;
    Echange(T.elems[i], T.elems[j]);
    $i \leftarrow j$;
fin

5. On suppose que l'on appelle CONSTRUITTAS sur le tableau $P[] = \{14, 7, 12, 3, 0, 20, 9\}$. Que vaut le tas après construction ? (i.e. les quelles sont les valeurs dans le tableau T.elems ?)
6. Dans la suite $n = 2^k - 1$. Quelle est la complexité en pire cas d'un appel à INSÈREFIN ?
7. Néanmoins, quelle est la complexité en pire cas d'un appel à INSÈREFIN(T,v,i,n) lorsque $\frac{n}{2} < i \leq n$?
8. De même, quelle est la complexité en pire cas d'un appel à INSÈREFIN(T,v,i,n) lorsque $\frac{n}{4} < i \leq \frac{n}{2}$? Et plus généralement lorsque $\frac{n}{2^j} < i \leq \frac{n}{2^{j-1}}$?
9. Si maintenant vous sommez les complexités précédentes pour tous les éléments, vous obtiendrez la complexité amortie totale des n appels à INSÈREFIN, donc de CONSTRUITTAS. Quelle est-elle ? Quelle est alors la complexité amortie de chaque appel à INSÈREFIN ? Fait-on mieux que la première méthode ?
   NB : Faites la somme des complexités par groupes (d'abord les éléments de n à $\frac{n}{2}$, puis ceux de $\frac{n}{2}$ à $\frac{n}{4}$, etc).
   NB2 : Utilisez le fait que $\frac{1}{2} + \frac{2}{2^2} + \frac{3}{2^3} + \frac{4}{2^4} + \cdots \leq 2$.

Exercice 2. Notations O, Ω, Θ (/5)

Complétez le tableau ci-dessous en indiquant si f est un grand O, un grand Ω ou un grand Θ de la fonction g (pour n tendant vers l'infini). Attention, Juste = $+0,5$, Faux = $-0,5$, Rien = $+0$, ceci pour éviter que vous ne répondiez au hasard.

		g			
		$n \log n$	$3n^2 + 10n$	$n2^n$	$n\sqrt{n}$
f	$3n$				
	$2n^2 + \log n$				
	$\frac{1}{4} n \log n$				

Exercice 3. Polygones simples, convexité et points intérieurs (/10)

Si $Q = (q_i)_{i=0..n-1}$ est un polygone simple, il existe une façon relativement simple de savoir si un point p est à l'intérieur de Q. On trace un rayon $[p, r)$ à partir de p, où r est un point différent de p, et on compte le nombre de fois où $[p, r)$ intersecte le bord de Q. Si ce nombre est impair alors le point est à l'intérieur, sinon le point est à l'extérieur (voir illustration page suivante).

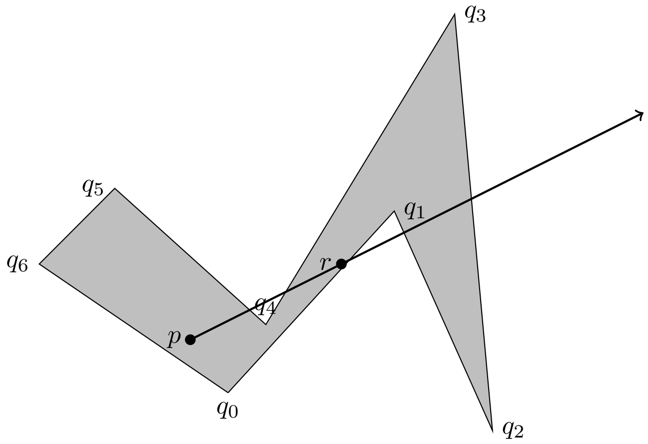


FIGURE 1 – Le point p est dans le polygone Q car $[p, r)$ intersecte 5 fois le bord de Q.

Nb : dans la suite, pour simplifier, on supposera toujours que le rayon $[p, r)$ ne traverse pas un sommet de Q, ni se s'aligne avec un côté de Q. C'est une hypothèse réaliste si r est bien choisi ou si r est tiré au hasard.

Nb : on vous redonne dans la page suivante quelques fonctions de base de géométrie algorithmique, que vous pouvez utiliser dans vos algorithmes.

1. Adaptez la fonction INTERSECTION-SEGMENTS du cours (rappelée plus loin) pour en faire une fonction INTERSECTION-RAYON(p_1, p_2, p_3, p_4) qui retourne vrai si et seulement le rayon $[p_1, p_2)$ intersecte le segment fermé $[p_3, p_4]$.
2. Ecrire maintenant l'algorithme qui teste si un point p est à l'intérieur de Q. On attend ici un algorithme simple de complexité linéaire en le nombre de sommets de Q. Son prototype sera **Fonction** ESTINTÉRIEUR ?*(**E** p : Point, **E** Q : Polygone) : booléen*

   NB : On écrira $Q.n$ pour avoir le nombre de sommets de Q, et $Q[i]$ pour accéder au i-ème sommet, avec l'indice i pris modulo $Q.n$. N'oubliez pas de choisir un r.
3. Si maintenant Q est un polygone convexe, combien de fois le rayon $[p, r)$ peut-il intersecter le bord de Q ?
4. En fait, on peut être beaucoup plus efficace dans le cas où Q est convexe (le polygone est supposé stocké dans l'ordre trigonométrique). Imaginez que l'on regarde la ligne polygonale allant de q_i à q_j sur le bord du polygone, $0 < i < j < n$. Si q_i est d'un côté de la droite (p, r) et q_j de l'autre côté, alors on peut trouver le côté potentiel d'intersection avec la droite (pr) par dichotomie (côté situé entre les indices i et j). Ecrivez cet algorithme (efficace) CHERCHE-CÔTÉ, qui retourne l'entier k tel que la droite (pr) intersecte le côté $[q_k q_{k+1}[$.

   Fonction CHERCHE-CÔTÉ*(**E** p,r : Point, **E** Q : Polygone, **E** i,j : entier) : entier*
5. Quelle est la complexité en pire cas de CHERCHE-CÔTÉ en fonction de i et j ? Puis en fonction en n ?
6. Est-ce que le côté retourné par CHERCHE-CÔTÉ a toujours une intersection non vide avec $[p, r)$?
7. Si on choisit r comme étant un point milieu sur un côté de Q, quelles sont les possibilités d'intersection entre $[p, r)$ et Q ? Faites un dessin pour (les) illustrer. Quel est l'intérêt d'un tel choix ?
8. On définit r comme étant le milieu entre q_{n-1} et q_0. En déduire l'algorithme rapide ESTINTÉRIEURCONVEXE ? qui teste si un point p est intérieur à un convexe Q. Quelle est sa complexité en pire cas en fonction de n ?

// Retourne un nombre > 0 si et seulement si r est à gauche du rayon $[p, q)$, un nombre < 0 ssi r est à droite de $[p, r)$, et 0 si p, q, r sont alignés.

Fonction Orientation(**E** p, q, r : *Point*) : *réel ;*
début
  Retourner $(q.x - p.x) * (r.y - p.y) - (q.y - p.y) * (r.x - p.x)$;
fin

// Sachant que r est sur la droite (pq), détermine si r appartient au segment $[pq]$.

Fonction Sur-Segment(**E** p, q, r : *Point*) : *booléen ;*
début
  Retourner $\min(p.x, q.x) \leq r.x \leq \max(p.x, q.x)$ *et* $\min(p.y, q.y) \leq r.y \leq \max(p.y, q.y)$;
fin

// Détermine si les segments $[p_1, p_2]$ et $[p_3, p_4]$ s'intersectent.

Fonction Intersection-Segments(**E** p_1, p_2, p_3, p_4 : *Point*) : *booléen;*
début
  $d_1 \leftarrow$ Orientation(p_3, p_4, p_1);
  $d_2 \leftarrow$ Orientation(p_3, p_4, p_2);
  $d_3 \leftarrow$ Orientation(p_1, p_2, p_3);
  $d_4 \leftarrow$ Orientation(p_1, p_2, p_4);
  si $((d_1 < 0$ *et* $d_2 > 0)$ *ou* $(d_1 > 0$ *et* $d_2 < 0))$ *et* $((d_3 < 0$ *et* $d_4 > 0)$ *ou* $(d_3 > 0$ *et* $d_4 < 0))$ **alors Retourner** *Vrai* ;
  sinon si $d_1 = 0$ *et* Sur-Segment(p_3, p_4, p_1) **alors**
    Retourner *Vrai ;*
  sinon si $d_2 = 0$ *et* Sur-Segment(p_3, p_4, p_2) **alors**
    Retourner *Vrai ;*
  sinon si $d_3 = 0$ *et* Sur-Segment(p_1, p_2, p_3) **alors**
    Retourner *Vrai ;*
  sinon si $d_4 = 0$ *et* Sur-Segment(p_1, p_2, p_4) **alors**
    Retourner *Vrai ;*
  sinon Retourner *Faux* ;
fin