

Algorithmique et structures de données : examen de première session

Ensimag 1A

Année scolaire 2011–2012

Consignes générales :

- Durée : 3 heures.
- Calculatrices, portables et tous instruments électroniques interdits. Livres interdits.
Autres documents autorisés.
- Le barème est donné à titre indicatif.
- Les deux exercices sont indépendants et peuvent être traités dans le désordre.
- La syntaxe Ada ne sera pas un critère déterminant de l'évaluation des copies. En d'autres termes, les correcteurs n'enlèveront pas de point pour une syntaxe Ada inexacte mais compréhensible (pensez à bien commenter votre code!).
- **Merci d'indiquer votre numéro de groupe de TD et de rendre votre copie dans le tas correspondant à votre groupe.**

Exercice 1 : Problème de la galerie d'art (7 pts)

L'effervescence est à son comble dans la capitale : le musée du Louvre va ouvrir une nouvelle salle, contenant des œuvres d'art magnifiques. Afin de se prémunir des convoitises des voleurs, le musée a décidé de placer des caméras dotées de la dernière technologie dans la salle, afin de surveiller ses moindres recoins. Chaque caméra sera placée dans un coin de la salle, et peut voir toute partie de la salle qui n'est pas cachée par un mur (voir Figure 1). Néanmoins, ces caméras ont un coût unitaire exorbitant, et le budget du musée est limité en ces temps de crise. Le Louvre fait appel à vos talents d'algorithme afin de répondre aux deux questions suivantes :

1. combien de caméras faut-il pour couvrir toute la salle ?

2. Où placer ces caméras ?

Serez-vous capable d'aider le Louvre à protéger ses œuvres d'art ?

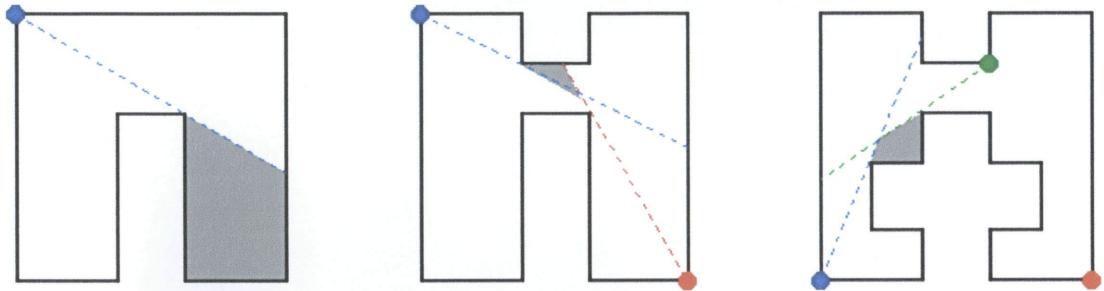


FIGURE 1 – Exemples de galeries surveillées par des caméras, avec un nombre insuffisant de caméras : certaines parties (grisées) sont cachées par des murs. La galerie est représentée en vue du dessus par un polygone, et les caméras (représentées par des points) sont placées sur des sommets du polygone, c'est-à-dire dans des coins de la salle.

Formalisation du problème

Le problème de la galerie d'art se formalise de la forme suivante.

Définition 1 (Polygone, sommet d'un polygone) *Un polygone est une figure géométrique plane formée d'une suite cyclique de segments consécutifs et délimitant une portion du plan. Un point intersection de deux segments consécutifs est appelé sommet du polygone.*

Exemples : triangle, carré, losange, parallélogramme, pentagone, hexagone, ...

Définition 2 (Surveillance d'un polygone) *Soit P un polygone simple (c'est-à-dire sans auto-intersection et sans trou). Un ensemble S de sommets du polygone surveille P si pour tout point p à l'intérieur de P , il existe s dans S tel que le segment $[sp]$ est entièrement inclus dans P .*

La figure 1 montre trois contre-exemples, avec respectivement 1, 2 et 3 sommets dans S . Dans chaque cas, il existe des points p (les points de la zone grise) pour lesquels il n'existe pas de sommet s de S tel que $[sp]$ est entièrement inclus dans le polygone.

Question 1 (0,5 pt) Pour chacun des trois polygones de la figure 2, donner le nombre de caméras nécessaires et suffisants pour sa surveillance. Justifier (vous pouvez par exemple dessiner une solution).

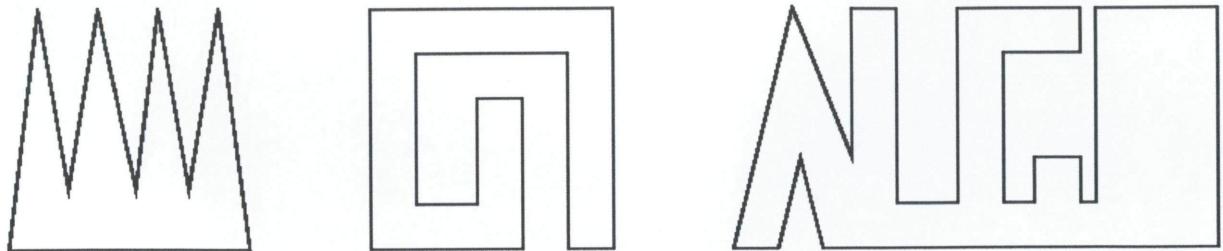


FIGURE 2 – Quel est le nombre minimal de caméras suffisantes pour surveiller chacune de ces trois galeries ?

Théorème 1 (Václav Chvátal (1973)) Soit n le nombre de sommets d'un polygone P simple du plan. $\lfloor \frac{n}{3} \rfloor$ sommets sont suffisants pour surveiller P (rappel : $\lfloor x \rfloor$ désigne la partie entière inférieure de x).

Question 2 (0,5 pt) Donner un exemple, avec $n = 6$, pour lequel $\frac{n}{3} = 2$ sommets sont nécessaires. Indice : vous pouvez vous inspirer d'un des polygones donné dans les figures précédentes.

Une démonstration constructive du théorème de Chvátal, dûe à Steve Fisk (1978) est la suivante (voir Figure 3) :

1. ajouter des arêtes à l'intérieur de P afin de le découper en triangles ;
2. attribuer une couleur parmi {rouge, vert, bleu} à chaque sommet de P , de manière à ce que pour tout triangle les sommets soient de couleurs différentes ;
3. déterminer parmi les trois couleurs celle qui est la moins attribuée aux sommets de P , et placer une caméra en chacun de ces sommets.

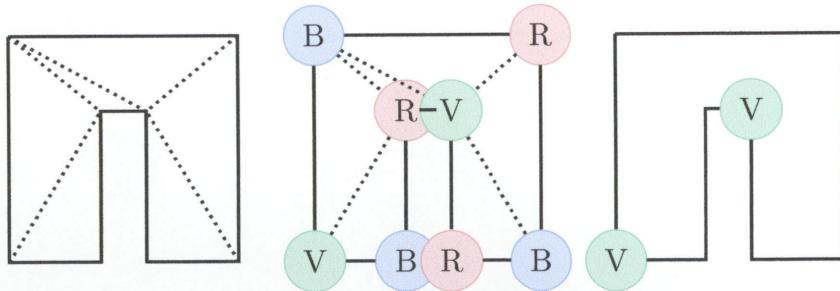


FIGURE 3 – Etapes de la démonstration constructive de Fisk. Ici, deux caméras sont suffisantes : il suffit de les placer sur les sommets verts.

Question 3 (1 pt) Quels triangles surveille chaque caméra ? Expliquer pourquoi cette démonstration justifie la borne $\lfloor \frac{n}{3} \rfloor$ donnée par Chvátal.

Algorithme “Diviser pour régner”

David Avis et Godfried Toussaint ont proposé en 1981 un algorithme “Diviser pour régner” qui permet de trouver où placer les caméras, c'est-à-dire quels sont les sommets de la couleur la moins attribuée. Cet algorithme prend en entrée le polygone P à la fin de l'étape 1. de Fisk, c'est-à-dire **avec les arêtes intérieures le découplant en triangles** (comme sur la Figure 3 à gauche).

L'**étape de récursion** consiste à diviser le polygone P en deux sous-polygones P_1 et P_2 . Pour cela, on cherche une arête intérieure $[s_1s_2]$ reliant deux sommets de P , et on prend pour P_1 le sous-polygone situé d'un côté de l'arête, et pour P_2 le sous-polygone situé de l'autre côté.

L'**étape de fusion** des résultats consiste à modifier les couleurs des sommets de P_2 afin que la couleur de s_1 dans P_2 soit la même que dans P_1 , et de même pour s_2 . Par exemple, si s_1 est rouge pour P_1 et vert pour P_2 et si s_2 est vert pour P_1 et bleu pour P_2 , alors dans P_2 : tous les sommets verts deviennent rouges, tous les sommets bleus deviennent verts, et tous les sommets rouges deviennent bleus.

Question 4 (0,5 pt) Soient $n+1, n_1+1$ et n_2+1 le nombre de sommets de P, P_1 et P_2 respectivement (noter $n+1$ le nombre de sommets de P plutôt que n , et de même pour P_1 et P_2 , permet de simplifier les calculs). Soit $C(n)$ la complexité en pire cas de l'algorithme. Donner $C(n)$ en fonction de $C(n_1)$ et $C(n_2)$. On admettra ici que la recherche du segment $[s_1s_2]$ se fait en temps n , mais vous devez évaluer la complexité en pire cas de l'étape de fusion des résultats.

Question 5 (1 pt) Pour quelles valeurs de n_1 et n_2 $C(n)$ est-il maximum ? Justifier et donner un grand O pour $C(n)$ dans ce cas, en fonction de n , et quand n tend vers l'infini.

Question 6 (1 pt) De la même manière, pour quelles valeurs de n_1 et n_2 $C(n)$ est-il minimum ? Justifier et donner un grand Θ pour $C(n)$ dans ce cas, en fonction de n , et quand n tend vers l'infini.

Etape de division

L'algorithme de division du polygone P d'Avis et Toussaint permet d'obtenir n_1 et n_2 supérieurs ou égaux à $\lfloor \frac{n}{4} \rfloor$. Il divise les sommets de P en 4 sous-ensembles de sommets successifs A, B, C et D , chacun de taille au moins $\lfloor \frac{n}{4} \rfloor$.

Question 7 (1 pt) Justifier qu'il existe au moins une arête intérieure de P qui relie soit A à C , soit B à D . Donner un algorithme simple en temps linéaire pour trouver une telle arête. On admettra que le nombre d'arêtes intérieures est $\Theta(n)$. Rappel : P est un polygone sans trou, et est découpé en triangles grâce aux arêtes intérieures.

Question 8 (0,5 pt) On suppose n multiple de 4, par souci de simplicité. Déduire de ce qui précède que $C(n) \leq C(\frac{n}{4}) + C(\frac{3n}{4}) + 2n$.

Question 9 (1 pt) Montrer par récurrence sur n que $C(n) \leq dn \log n$, avec d une constante positive bien choisie.

L'algorithme d'Avis et Toussaint est donc en $O(n \log n)$.

Exercice 2 : Structures de tas-arbres (Treaps) (13 pts)

Introduction

On se propose d'étudier ici une forme particulière d'arbres binaires de recherche : les treaps qui sont un croisement entre arbres binaires de recherche et tas. Les treaps permettent d'implémenter une structure de dictionnaires stockant un ensemble de clefs (et éventuellement des valeurs associées).

On définit un treap comme un arbre binaire tel que :

- À chaque nœud v est associée une clef c_v . Nous supposerons dans cet exercice que toutes les clefs utilisées sont toujours différentes.
- À chaque nœud v est associée une priorité p_v . Nous supposerons une fois encore qu'un treap ne contiendra jamais deux nœuds différents de priorités égales.
- Pour tout nœud v et ses deux enfants potentiels v_1, v_2 on a : $p_v > p_{v_1}$ et $p_v > p_{v_2}$.
- Pour tout nœud v et ses deux *sous-arbres* A_1, A_2 potentiels $\forall v_1 \in A_1, \forall v_2 \in A_2$ on a : $c_{v_1} < c_v < c_{v_2}$.

Lors de l'insertion d'un nœud dans un treap, une priorité aléatoire lui est associée. Nous supposerons ici disposer d'une fonction *Random* renvoyant une priorité aléatoire. Nous supposerons dans un souci de simplification qu'une même priorité n'est jamais obtenue deux fois. Enfin, nous supposerons que les clefs comme les priorités seront ici des entiers. La figure 4 illustre un exemple de treap contenant les clefs $\{1, 3, 7, 5, 2\}$.

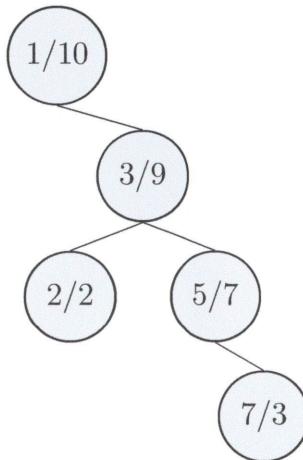


FIGURE 4 – Exemple de treap. Chaque nœud affiche clef/priorité.

Algorithmes

Question 10 (1 pt) Proposez une structure de données permettant de stocker un nœud d'un treap.

Question 11 (1 pt) Décrivez comment effectuer une recherche d'une clef. Quel est le coût au pire cas d'une telle recherche ?

Question 12 (0,5 pt) Dessinez les insertions successives dans un treap des clefs 8,4,6,10 associées aux priorités aléatoires respectives suivantes : 9,7,4,5.

Question 13 (0,5 pt) Ajoutez au treap obtenu lors de la question précédente les clefs 12 et 3 associées au priorités 6 et 11 (dessinez toutes les étapes). Quels problèmes rencontrerez-vous ? Comment résoudre ces problèmes ?

Question 14 (4 pts) Écrivez un algorithme d'insertion dans un treap. Les allocations mémoire doivent apparaître explicitement.

Question 15 (2 pts) Expliquez le fonctionnement d'un algorithme de suppression d'une clef. Une portion du code est-elle factorisable avec le code d'insertion ? Quel est le coût au pire cas d'une suppression ?

Opérations avancées

Question 16 (2 pts) Écrivez un algorithme de coût $O(h)$ permettant la séparation d'un treap en deux treaps tels que l'un des treap contienne tous les sommets inférieurs à une clef k donnée en paramètre et l'autre tous les sommets supérieurs à k . *Indication* : on considérera la possibilité d'ajouter au treap initial un nouveau sommet judicieusement choisi.

Question 17 (2 pts) Expliquez le fonctionnement d'un algorithme permettant la fusion de deux treaps en temps $O(h)$.