clement.rouvroy@ens.psl.eu

Arbres équilibres et (pas) géométrie algorithmique

Consignes Le candidat respectera le langage imposé (OCaml) et ne devra pas utiliser de librairie hors-programme. Il est invité à faire des schémas clairs et précis de ses algorithmes lors des appels au correcteur, c'est à la fois un gain de temps pour les deux et une manière de montrer qu'il a compris ce qu'il manipule. Les preuves écrites doivent être formelles, sauf si la consigne précise que ce n'est pas nécessaire, la rigueur sera évaluée. L'examinateur ne débugera pas votre code, en revanche en cas de doute ("ai-je le droit à telle ou telle librairie ?", "je suis sortie de la VM, comment y revenir ?", "ai-je le droit à une indication pour cette question ?") n'hésitez pas à poser votre question. Elle ne vous dévalorisera pas, si la réponse peut vous retirer des points, l'examinateur vous demandera avant de vous donner la réponse si vous l'acceptez (si vous n'avez pas de chance, le jour de l'oral il vous retirera les points rien que pour avoir posé la question, par exemple si vous demandez "comment trier une liste en $O(n\log(n))$?" ou un autre résultat classique du programme, ça sera sûrement retenu contre vous). Enfin, si l'examinateur n'est pas à votre portée quand vous avez besoin d'une aide / d'une question oral à donner, gardez le bras levé et lisez la suite, ne restez jamais passif.

Introduction du sujet

Tiré de X/ENS Info A MPI 2024 et un TP MP⋆ du lycée Masséna.

(Info A) Ce sujet traite de l'implémentation d'ensembles finis par des arbres équilibrés dits AVL, et de leur application à un problème de géométrie algorithmique classique, la localisation d'un point dans le plan.

Ce TP se focalise sur la progrmamation d'AVL après une courte partie théorique (dénombrement).

Dans tout ce sujet, on note n le nombre de noeuds de l'arbres et h la $h$auteur de l'arbre.

I Question de cours

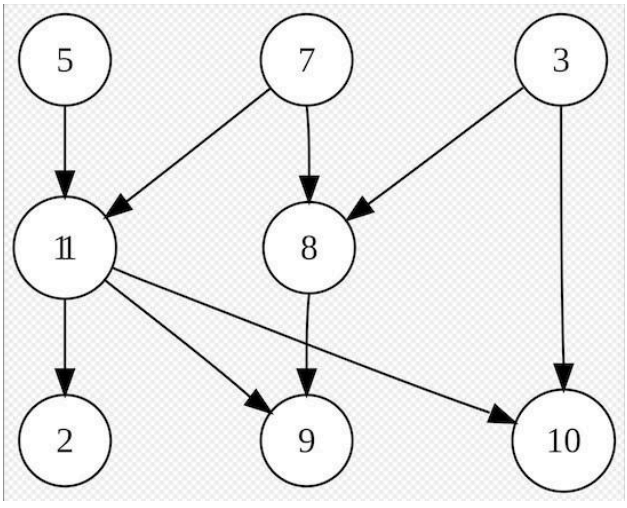


Figure 1: $\mathcal{G}_1$

Question 1 à l'écrit

1. Donner un parcours en profondeur du graphe $\mathcal{G}_1$ en respectant l'ordre naturel (on commence le parcours par le sommet 1, puis si on a plusieurs choix possibles on prend toujours l'entier le plus petit).
2. Donner un parcours en largeur du graphe $\mathcal{G}_1$ en respectant l'ordre naturel.

VRAI OU FAUX

a. Pour tout chemin de a à b, il existe un chemin simple de a à b.

b. Pour tout chemin de a à b, il existe un chemin élémentaire de a à b.

c. L'énonce "donner les composantes connexes de $\mathcal{G}$" permet de savoir si $\mathcal{G}$ est orienté ou non-orienté.

d. La complexité en temps du parcours en profondeur est $O(|V|)$.

e. Un cycle non-orienté peut-être constitué de deux sommets.

f. La liste d'adjacence optimise le temps de réponse à la question "i a pour voisin j".

g. La matrice d'adjacence optimise la mémoire.

h. Vous avez répondu aléatoirement à au moins une des questions.

II Préliminaires

On s'intéresse aux sous-ensembles d'un ensemble $\mathcal{U}$, $\mathcal{U}$ est munie d'un ordre total, c'est-à-dire que

$$\forall (x,y) \in \mathcal{U}^2, x < y \text{ ou } x = y \text{ ou } x > y$$

Question 2 à l'écrit

Combien y a-t-il d'arbres binaires de recherche contenant exactement les trois éléments x, y et z, avec $x < y < z$? Les dessiner.

Définition 2.1 AVL

Pour équilibrer nos arbres binaires de recherche, on va imposer une condition supplémentaire: pour tout noeud (l, x, r) on a

$$|h(l) - h(r)| \leq 1$$

Autrement dit, la hauteur du fils gauche a un écart d'au plus 1 avec la hauteur du fils droit. On appelle ce type d'arbre des arbres AVL, du nom de leur auteurs Adelson-Velsky(1922 - 2014) et Landis (1921 - 1997).

Question 3 à l'écrit

Combien y a-t-il d'arbres binaires de recherche AVL contenant exactement les trois éléments x, y et z, avec $x < y < z$? Les dessiner.

On supposera dans la suite que la borne des arbres rouges-noires ($h \leq 2\log(n) + 1$) reste valide.

III Implémentation

On se donne le type OCaml suivant:

```
type tree = E | N of int * tree * elt * tree
```

L'entier qui doit vous surprendre dans cette définition correspond à la hauteur de l'arbre AVL.

III.1 Vérification

Question 4 code

Ecrire une fonction `hauteur : tree -> int` donnant la hauteur d'un arbre AVL en $O(1)$.

Question 5 code

Ecrire une fonction `verification: tree -> int` qui vérifie que la hauteur de la racine est bien ce qu'elle doit être.

Question 6 code

Déduire de ces deux questions une fonction `verifie_avl: tree -> bool` qui dit si un arbre passé en argument est bien un AVL.

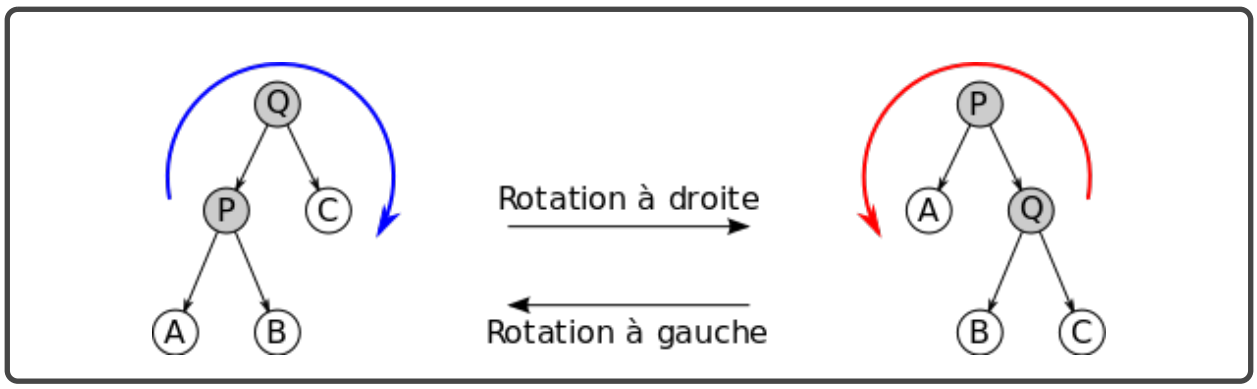
Votre fonction doit être $O(n)$ pour raporter des points.

La suite du sujet a pour but de construire des arbres qui sont acceptés par la fonction `verifie_avl`, Info A vous donne le code et vous demande des preuves dessus, ce sujet vous demande de trouver le code. Par conséquent, si vous avez du temps cet été vous gagnerez (beaucoup) à traiter Info A en autonomie, ça prolongera bien ce sujet et vous pourrez voir la différence théorie/pratique.

III.2 Rotation

Définition 3.1 rotation

On appelle rotation sur un arbre binaire de recherche une de ces deux transformations:



Question 7 code

Ecrire une fonction OCaml `rotation_droite: tree -> tree`, de même pour `rotation_gauche`. On supposera que les arbres ont la bonne forme (i.e. que les sous-arbres nécessaires existent).

Question 8 à l'écrit

Justifier (par un raisonnement sur la relation de comparaison) que l'opération rotation droite préserve une structure d'abre binaire de recherche si l'arbre initial en est un.

III.3 Equilibrage

On va ajouter les éléments 1 à 1 dans l'arbre AVL, donc avoir un algorithme pour un arbre non-AVL avec une différence de hauteur au plus 2 est suffisant car au pire un ajout augmente la hauteur de 1 donc on passe d'une différence au + 1 à une différence au + 2, on va donc équilibrer avec une différence de hauteur d'au + 2.

On se donne un arbre (r, g, d) ($r$acine, $g$auche et $d$roite). On suppose que g et d sont deux AVL corrects, que la hauteur de g et d différent d'au plus 2 et que le champ hauteur du noeud y n'est pas forcément le bon, même si l'abre total vérifie la propriété d'AVL. On se donne l'algorithme suivant:

- Si l'arbre total est un AVL on a terminé.
- Si l'arbre toal n'est pas un AVL
  - Si $h_g = h_d + 2$, $g = (x, \alpha, \beta)$
    - Si $h_\alpha \geq h_\beta$, l'arbre obtenu par raotation droite de l'arbre total est un AVL
    - Sinon on fait une rotation gauche de l'abre g puis une rotation droite de l'arbre total et on obtient aussi un AVL.
  - L'autre cas est symétrique.

Question 9 code

Implémenter l'algorithme récursif `equilibre: tree -> tree`.

Le lecteur curieux pourra trouver la preuve de cet algorithme en traintent les questions 6-13 du sujet Info A.

Question 10 code

Ecrire une fonction `insertion: elt -> tree -> tree` qui prend en entrée un élément, un AVL d'ensemble $\mathcal{U}$ et renvoie un AVL contenant $\mathcal{U} \cup \{\text{element}\}$. Evidemment c'est possible car le type `elt` est compatible pour la comparaison dans $\mathcal{U}$.

Question 11 code

Ecrire une fonction `suppression: elt -> tree -> tree`.

Question 12 à l'écrit

Créer un arbre AVL à 1000 noeuds insérés dans l'ordre croisant, quelle est sa hauteur ?

IV Bonus

Question classico-classique des ENS:

Question 13 à l'écrit

Combien y a-t-il d'arbres binaires de recherche à n noeuds ?