Rapport du projet d'algo2 : Décomposition en polygones monotones

Aymeric COLLANGE, Romain DULERY

10 avril 2015

Syntaxe d'appel du programme : ./main data.in polygonOutput.svg

1 Présentation du programme

L'algorithme principal parse le fichier de données, rentre les points dans un tableau, trie ce dernier et crée l'arbre binaire de recherche, avec les segments pour clefs, afin de réaliser le traitement du polygone, tout en fournissant le rendu svg.

Pour le tri, nous avons implanté un tri fusion, triant le tableau de points selon leurs abscisses afin de permettre un parcours du polygone de gauche à droite.

Explication du fonctionnement de certaines fonctions :

1. Noeuds_Voisins
   Pour trouver le noeud de l'arbre de clef juste inférieure à celle de la cible : on procède en deux étapes :
   (a) Si la cible possède un fils gauche, on renvoit le noeud le plus à droite du sous-arbre engendré.
   (b) Sinon, on renvoit le premier père gauche que l'on trouve en remontant vers la racine, en cas d'existence de ce dernier.

   On procède de manière analogue pour trouver le noeud de clef juste supérieure.

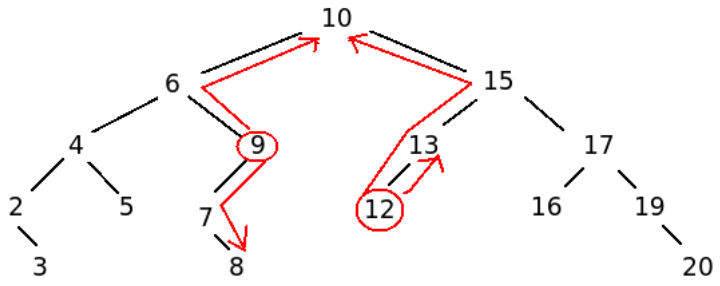

FIGURE 1 – Illustration Noeuds_Voisins

Sur la figure 1, on peut voir ce procédé, on s'intéresse aux noeuds voisins de 9 et de 12 :

Pour 9, on utilise (a) pour le petit voisin, et (b) pour le grand.
Pour 12, on utilise (b) pour le petit voisin, et (b) également pour le grand.

2. Compte_Position
   Pour trouver le nombre de noeuds de clefs inférieures à celle de la cible, on procède en deux étapes, une initialisation et une boucle.
   (a) Déjà on initialise le compteur au compte du fils gauche de la cible, s'il existe.
   (b) Ensuite on remonte vers la racine, et à chaque itération, si on tombe sur un père gauche, on ajoute au compteur le compte du sous-arbre gauche de ce père, en cas d'existence de ce dernier, plus un pour prendre en compte le père. Pour cela il suffit d'ajouter le compte du père auquel on retranche le compte du noeud courant.

   On procède de manière analogue pour trouver le nombre de noeuds de clefs supérieures à celle de la cible.

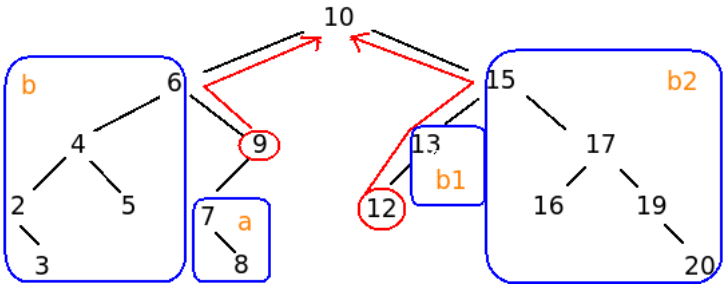

FIGURE 2 – Illustration Compte_Position

Sur la figure 2, on peut voir ce procédé, on s'intéresse aux noeuds de clefs inférieures à 9, et aux noeuds de clefs supérieures à 12.
Pour 9, on commence par l'étape a, puis à l'étape b en remontant vers la racine on arrive sur 6 qui est un père gauche, on augmente donc le compteur. Ensuite 10 est un père droit, et c'est la racine donc on s'arrête là.
Pour 12, il n'y a pas de fils droit donc on ne procède pas à l'étape a, et l'étape b se fait en deux temps, b1 et b2, jusqu'à arriver à 10, la racine, qui est un père gauche donc non pris en compte.

2 Problèmes rencontrés et améliorations

Nous n'avons pas réussi à terminer ce projet, il nous manque la gestion de l'arbre dans le traitement du polygone, notamment à cause d'un problème de compréhension à propos de l'utilisation de l'arbre dans l'algorithme évoqué à la fin du sujet. De plus, la relation d'ordre sur les segments est à revoir, celle que nous avons choisie, consistant à comparer les ordonnées des milieux, possède de nombreuses lacunes.

Au sujet des difficultés rencontrées pendant le codage, le plus difficile fut la gestion de la généricité, afin de pouvoir utiliser les segments comme clefs. Nous avons perçu l'idée dans un premier temps, en considérant une analogie avec l'introduction des tableaux de taille indéterminée à l'aide de pointeurs. L'idée est la même dans le sens où on accède à un cas particulier de quelque chose de général. Le plus difficile fut la syntaxe, il était peu intuitif de déclarer un package au sein d'une procédure.

Améliorations possibles :

- Utilisation d'un arbre auto-équilibré (type AVL ou rouge-noir)
- Factorisation de certaines portions de code, par exemple dans la procédure supprimer de struct.adb, où Suppr1G et Suppr1D peuvent être fusionnées

- Amélioration de la fonction de comparaison entre deux segments
- Adaptation de la taille du cadre svg à la taille du polygone
- Optimisation du coût en mémoire