IENAC S03

Algorithmique, λ-calcul et typage

Durée : 2h
Tous documents autorisés

Ce sujet comporte 2 pages.

1 Typage et λ-calcul

1. [2pt] Les deux λ-termes `a` et `b` sont-ils équivalents ? Justifiez votre réponse.

```
let a = fun x y -> x ((fun c d -> c) y x)
let b = fun z -> (fun y -> y) z
```

2. [2pt] Donner les types générés par OCaml pour les termes précédents (en justifiant mais sans nécessairement développer l'arbre de typage).
3. [1pt] Donner les règles de typage pour le type suivant :

```
type 'a option = None | Some of 'a
```

2 Algorithmique

2.1 Algorithme de Dijkstra

L'algorithme de Dijkstra permet de calculer tous les plus courts chemins depuis un sommet s d'un graphe $\mathcal{G}$ vers tous les autres sommets. L'algorithme fait évoluer un ensemble $\mathcal{E}$ de sommets pour lesquels le plus court chemin depuis s est connu ; $\mathcal{E}$ est initialisé avec $\mathcal{G} \setminus \{s\}$. On note $p(i, j)$ la longueur de l'arc entre le sommet i et le sommet j (∞ s'il n'existe pas d'arc entre i et j), $d[i]$ la longueur du plus court chemin *connu* de s à i.

1 : PourTous les sommets i de $\mathcal{G}$
2 : $\quad d[i] := p(s, i)$;
3 : $\mathcal{E} := \mathcal{G} \setminus \{s\}$;
4 : TantQue $\mathcal{E}$ n'est pas vide
5 : $\quad k :=$ le sommet de $\mathcal{E}$ de $d[k]$ minimal ;
6 : $\quad \mathcal{E} := \mathcal{E} \setminus \{k\}$;
7 : $\quad$ PourTous les sommets i de $\mathcal{E}$
8 : $\quad\quad d[i] := min(d[i], d[k] + p(k, i))$;

On note n le nombre de sommets du graphe et m le maximum de n et du nombre d'arcs du graphe.

1. [3pt] En détaillant les étapes, donner la complexité d'une implémentation « naïve » de l'algorithme.
2. [2pt] Comment peut-on améliorer l'algorithme en utilisant un maximier ?

2.2 Recherche dans un nuage de points

On souhaite résoudre le problème suivant :

Soit un ensemble de point du plan, déterminer ceux qui sont dans une région donnée.

Les algorithmes seront implémentés au choix en C ou en OCaml.

1. On commence par construire un arbre binaire dont chaque nœud sera étiqueté par un point. On note $\mathcal{P}$ l'ensemble des points. Cet arbre est construit suivant le principe suivant :
   - choisir un point $p \in \mathcal{P}$, le retirer de $\mathcal{P}$ et construire un nœud étiqueté par p ;
   - on divise le plan en deux par la droite *horizontale* passant par p ; les points situés dans le demi-plan inférieur seront insérés (récursivement) dans le fils gauche et les autres dans le fils droit ;
   - à l'étape suivante, on choisit un nouveau point et on divise les autres à l'aide d'une droite *verticale* : les points à gauche seront insérés dans dans le fils gauche et les points à droite dans le fils droit ;
   - on alterne division horizontale et verticale à chaque étape et on s'arrête quand $\mathcal{P}$ est vide ;

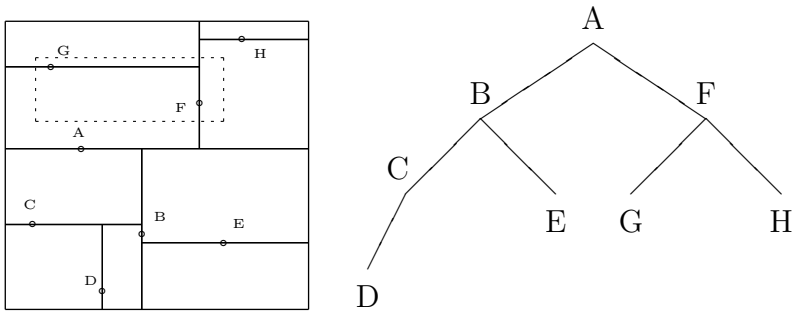

FIG. 1 – Un nuage de points et son arbre binaire.

La figure 1 représente un tel arbre. On choisit le point A à la racine de l'arbre puis on insère d'abord les points situés plus bas. On choisit alors le point B pour diviser verticalement le demi-plan inférieur, et on insère d'abord les points à gauche (C et D), etc. Les points seront ainsi insérés dans l'ordre $A \ldots H$.

   (a) [1pt] Définir un type pour représenter ces arbres binaires.

   (b) [3pt] Écrire la fonction de construction de l'arbre qui prend en paramètre l'ensemble des points et renvoie un arbre les contenant.

   (c) [1pt] Écrire l'équation de récurrence sur la complexité de cette fonction quand l'ensemble des points est divisé en deux parties de tailles égales à chaque étape.

2. On veut maintenant retrouver les points qui sont situés dans un rectangle donné du plan (spécifié par ses points inférieur droit et supérieur gauche). Il suffit alors de parcourir l'arbre et à chaque nœud, on ajoute au résultat le point du nœud s'il est contenu dans le rectangle puis on continue soit dans le fils gauche, soit dans le fils droit, soit les deux, suivant que le rectangle est au dessous (resp. à gauche), au dessus (resp. à droite) ou coupe la droite horizontale (resp. verticale) associée au nœud.

   (a) [4pt] Écrire une fonction qui prend en paramètre un rectangle et renvoie la liste des points de $\mathcal{P}$ contenus dans ce rectangle.

   (b) [1pt] La complexité de cette recherche dépend fortement du rectangle considéré (taille, position). Évaluer cette complexité (en justifiant) dans quelques cas différents (i.e. on ne se limite pas au « pire cas »). On supposera dans cette question que l'arbre est bien équilibré.