Examen de

λ-calcul, typage et algorithmique

Durée : 2 h

Tous documents autorisés

Ce sujet comporte 2 pages.

1 λ-calcul typé

1. [1pt] Typer le terme suivant en développant l'arbre de typage :

   ```
   fun f g x -> f (g x)
   ```

2. [2pt] Est-ce que le terme suivant est typable ?

   ```
   (fun id -> id id) (fun x -> x)
   ```

   Comment peut-on le transformer pour le rendre typable ?

3. [3pt] Donner les règles de typage nécessaires pour le type suivant :

   ```
   type 'a t =
       X
     | Y of 'a * 'a t
     | C of unit
   ```

2 Problème du sac à dos

Soient n objets de poids $w_i \in \mathbb{N}$ et de valeur $p_i \in \mathbb{N}$ et un *sac* de capacité W. On cherche à choisir des objets parmi ces n

- dont la somme des poids est inférieure à W ;
- dont la somme des valeurs est maximale.

Formellement, soit un vecteur de x_i, des variables booléennes avec $x_i = 1$ ssi l'objet i est sélectionné. On chercher à maximiser

$$z(X) \quad = \quad \sum_i x_i p_i$$

en respectant la contrainte

$$\sum_i x_i w_i \quad \leq \quad W$$

2.1 Algorithme glouton

Un algorithme glouton pour résoudre ce problème consiste à choisir en priorité les objets les plus efficaces, c'est-à-dire ceux de ratio maximal valeur/poids.

- [3pt] Écrire cet algorithme en pseudo-code. Évaluer sa complexité.
- [2pt] Réécrire l'algorithme en langage C ou OCaml.
- [2pt] Cet algorithme peut-être arbitrairement mauvais : pour tout $\epsilon\mathbb{R}^+$, il existe un problème (i.e. des w_i, des p_i et W) dont l'optimum O et la solution calculée par le glouton de valeur G sont tels que $\frac{G}{O} \leq \epsilon$. Fabriquer un tel problème pour $n = 2$. On pourra prendre $W = \lceil \frac{2}{\epsilon} \rceil$.

2.2 Programmation dynamique

Le problème du sac à dos possède la propriété de sous-structure optimale, c'est-à-dire que l'on peut construire la solution optimale du problème à i variables à partir du problème à $i-1$ variables. Cette propriété permet d'utiliser une méthode de résolution par programmation dynamique.

Soit $R(k, c)$ le problème réduit aux k premiers objets et de contenance c (donc $R(n, W)$ est le problème total). La solution optimale de $R(k, c)$ est soit :

- la solution optimales du problème à $k-1$ variables avec la même contenance c ($R(k-1, c)$), à laquelle on ajoute $x_i = 0$;
- la solution optimale du problème à $k-1$ variables avec la contenance $c - w_i$ ($R(k-1, c-w_i)$), auxquelles on ajoute $x_i = 1$.

On note $T(k, c)$ la valeur de la solution optimale de $R(k, c)$. On a $T(0, c) = 0$ pour tout c.

- [1pt] En utilisant cette méthode, calculer la solution optimale pour l'instance à 4 objets de poids 8, 4, 3, et 2, tous de valeur 1, pour une contenance de 9.
- [1pt] Écrire la formule de récurrence permettant de calculer la fonction T.
- [3pt] En utilisant une technique de programmation dynamique, écrire le pseudo-code du calcul de T.
- [2pt] Estimer la complexité temporelle et spatiale de l'algorithme obtenu.