ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Sections d'Informatique et de Systèmes de Communication

Série d'exercices 3 11 Octobre 2010

1. Running time Considérons les fonctions $f_i(n)$ dont les implémentations sont données ci-dessous en pseudo-code:

Call: $f_1(n)$
1: $a \leftarrow 0$
2: **for** $i = 1, \ldots, n$ **do**
3: **for** $j = 1, \ldots, i$ **do**
4: $a \leftarrow a + 1$
5: **return** a

Call: $f_2(n)$
1: $a \leftarrow 0$
2: **for** $i = 1, \ldots, n$ **do**
3: **for** $j = 1, \ldots, 2 \cdot n$ **do**
4: $a \leftarrow a + 1$
5: **for** $k = 1, \ldots, \lfloor \sqrt{n} \rfloor$ **do**
6: $a \leftarrow a + 1$
7: **return** a

Call: $f_3(n)$
1: $a \leftarrow 0$
2: **for** $i = 1, \ldots, n$ **do**
3: **for** $j = 1, \ldots, i$ **do**
4: **for** $k = j + 1, \ldots, i + j$ **do**
5: $a \leftarrow a + 1$
6: **return** a

Call: $f_4(n)$
1: $a \leftarrow 0$
2: **for** $i = 1, \ldots, n$ **do**
3: $a \leftarrow a + \ln(n)$
4: **for** $i = 1, \ldots, n$ **do**
5: **for** $j = i, \ldots, n$ **do**
6: $a \leftarrow a + n$
7: **return** a

a) Trouver des formules fermées pour $f_1(n)$, $f_2(n)$, $f_3(n)$ et $f_4(n)$ (C'est-à-dire sans $\sum$, $\prod$ et sans récursion).

b) Exprimer chacune de ces fonctions avec la notation θ, sous la forme $\theta(n^r \cdot \ln^s(n))$.

2. Relations de récurrence

a) Soit $T : \mathbb{N} \to \mathbb{R}$ une fonction monotone croissante (c'est-à-dire que $T(n) \leq T(n+1)$) telle que pour tout $n \in \mathbb{N}$ on a

$$T(10n) = 3 \cdot T(n) + 10^3 \cdot \sqrt{n}$$

Montrer que $T(n) = O(\sqrt{n})$.

b) * Soit $T : \mathbb{N} \to \mathbb{R}$ une fonction monotone croissante telle que $T(1) = 1$, et pour tout $n \in \mathbb{N}$ on a

$$T(2n) = n \cdot T(n)$$

Nous voulons montrer que $T(n) = O(n^{\frac{\log_2(n)+1}{2}})$.

- (i) Soit n une puissance de 2, disons $n = 2^k$. Trouver une formule pour $T(n)$ en fonction de k.
- (ii) En déduire que pour tout $n \in \mathbb{N}$ on a

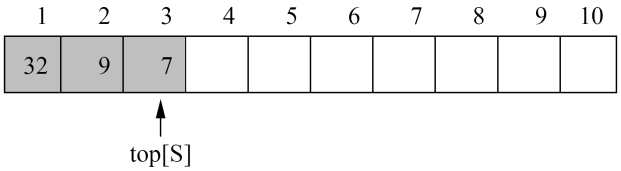
$$T(n) \leq 2^{\frac{\log_2 n \cdot (\log_2(n)+1)}{2}}$$

(utiliser le fait que $T(n) \leq T(2^{k+1})$, où $k = \lfloor \log_2 n \rfloor$ ainsi que la partie (i)).

- (iii) En déduire que $T(n) = O(n^{\frac{\log_2(n)+1}{2}})$ (il s'agit simplement d'utiliser des manipulations algébriques pour arriver au résultat).

3.Stacks

a) Supposons que nous avons un stack implémenté par un array dans l'état initial suivant:



Quel est l'état du stack après les instructions suivantes (c'est-à-dire quelles sont les valeurs dans l'array, et que vaut l'attribut $top[S]$)?

`Push`$(S, 7)$
`Push`$(S,$ `Pop`$(S) +$ `Pop`$(S))$
`Pop`(S)
`Push`$(S, 11)$
`Push`$(S, 12)$

b) Supposons que nous voulons utiliser un stack pour résoudre le probléme de vérification de parenthèses pour l'input suivant:

$$[(1 + 2) * 3 - ((4/5) + 6)) * 7]$$

Quel est l'output attendu? En utilisant l'algorithme donné dans le cours avec cet input, quelles sont les opérations effectuées sur le stack, et dans quel ordre? Quel est l'output?

4. Permutations avec stacks et files d'attente

Considérons une machine avec les caractéristiques suivantes:

- Elle dispose d'un (seul) stack S.
- Elle peut lire un caractère de l'input et le déposer sur le stack—c'est l'opération `Push`.
- Elle peut enlever un caractère du stack et l'ajouter à l'output—`Pop`.
- Il n'y a pas d'autres opérations ni de stock supplémentaire.

Par exemple, la suite suivante d'instructions

`Push`, `Push`, `Push`, `Pop`, `Push`, `Pop`, `Pop`, `Push`, `Pop`, `Pop`

crée, si l'input $i = (1, 2, 3, 4, 5)$ est donné, l'output $(3, 4, 2, 5, 1)$.

a) Existe-t-il une suite d'instructions qui sort, pour le même input i, l'output $(1, 2, 3, 5, 4)$? Et l'output $(2, 3, 5, 4, 1)$? Ou encore $(3, 1, 2, 5, 4)$?

b) Montrer qu'il est possible d'obtenir la permutation $(p_1, p_2, \ldots, p_n)$ de $(1, 2, \ldots, n)$ s'il n'existe pas d'indices $i < j < k$ tel que $p_j < p_k < p_i$.

c) Si nous remplaçons le stack par une file d'attente, quelles permutations pouvons nous alors obtenir?