ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Sections d'Informatique et de Systèmes de Communication

Série d'exercices 3 5 Octobre 2007

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 ← 0
2: for i = 1, ..., n do
3:     for j = 1, ..., i do
4:         a ← a + 1
5: return a
```

Call : $f_2(n)$

```
1: a ← 0
2: for i = 1, ..., n do
3:     for j = 1, ..., 2 · n do
4:         a ← a + 1
5:     for k = 1, ..., ⌊√n⌋ do
6:         a ← a + 1
7: return a
```

Call : $f_3(n)$

```
1: a ← 0
2: for i = 1, ..., n do
3:     for j = 1, ..., i do
4:         for k = j + 1, ..., i + j do
5:             a ← a + 1
6: return a
```

Call : $f_4(n)$

```
1: a ← 0
2: for i = 1, ..., n do
3:     a ← a + ln(n)
4: for i = 1, ..., n do
5:     for j = i, ..., n do
6:         a ← 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. Algorithme

Supposons que nous disposons d'une machine avec une mémoire infinie qui peut effectuer les opérations suivantes :

- Multiplier deux nombres stockés dans la mémoire
- Stocker le résultat en mémoire

Pour un n fixé, nous aimerions programmer cette machine (i.e. développer un algorithme) qui calcule la valeur de a^n (si l'input a est donné) en effectuant aussi peu de multiplications que possible.

a) Supposons que n est une puissance de 2. Trouver un algorithme qui calcule a^n en utilisant aussi peu de multiplications que possible. Exprimer le nombre de multiplications nécessaires en fonction de n.

b) Montrer que pour n'importe quel n il existe un algorithme qui calcule a^n en $O(\log_2(n))$ multiplications.

Indication : Utiliser le principe de Horner vu en cours.

3. L'Algorithme de Karatsuba

a) Soient $f(x) = 1+2x+x^2+2x^3$ et $g(x) = 2+2x+x^2+x^3$ deux polynômes de degré 3 sur $\mathbb{R}$. En utilisant l'algorithme naïf pour multiplier $f(x)$ et $g(x)$, de combien de multiplications d'éléments de $\mathbb{R}$ a-t-on besoin ?

b) Considérons maintenant l'algorithme de Karatsuba pour $f(x)$ et $g(x)$ dans la question précédente. Que valent f_0, f_1, g_0, g_1 ? Que valent u et v ?

c) Utiliser l'algorithme de Karatsuba pour effectuer la multiplication, et compter le nombre de multiplications sur $\mathbb{R}$ qui sont nécessaires.

4. Elever une matrice au carré

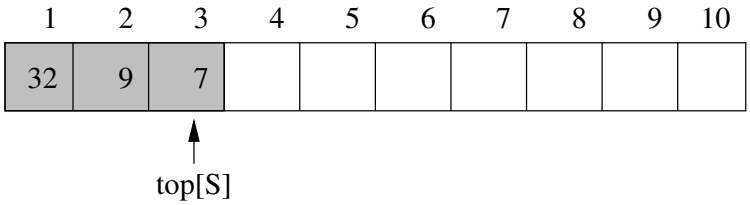
Soit A la matrice 2×2 sur $\mathbb{R}$: $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$

a) Trouver un algorithme qui calcule A^2 en utilisant 5 multiplications d'éléments de $\mathbb{R}$ (et autant d'additions d'éléments de $\mathbb{R}$ que nécessaire).

b) Peut-on généraliser cet algorithme récursivement à toute matrice $n \times n$ pour obtenir un algorithme $O(n^{\log_2(5)})$ (comme nous l'avons fait dans le cours pour l'algorithme de Strassen) ? Pourquoi ?

5. 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 $top[S]$) ?

$\texttt{Push}(S, 7)$, $\texttt{Push}(S, \texttt{Pop}(S) + \texttt{Pop}(S))$, $\texttt{Pop}(S)$, $\texttt{Push}(S, 11)$, $\texttt{Push}(S, 12)$

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 ?