TD 4: Algorithmique fonctionnelle
Complexité

Nga Nguyen - Stefan Bornhofen - Peio Loubière

1 Puissance

Voici 2 versions différentes permettant de calculer la puissance x^n vues en TD 3. Comparer leur complexité :

```
1  Fonction puissance(x, n : Entier) : Entier
2  Début
3    Si n = 0 Alors
4      retourner 1
5    Sinon
6      retourner x*puissance(x,n-1)
7    FinSi
8  Fin
9
10 ================
11
12 Fonction puissance-dic(x, n : Entier) : Entier
13 Début
14   Si n = 0 Alors
15     retourner 1
16   SinonSi (n mod 2 = 0)
17         retourner puissance-dic(x*x,n/2)
18   Sinon
19         retourner x*puissance-dic(x,n-1)
20   FinSi
21 Fin
```

2 Jeu par élimination

Soit le jeu suivant consistant à tirer au sort un "vainqueur" parmi un groupe d'enfants, au moyen d'une pièce de monnaie :

- S'il n'y a qu'un enfant, il est évidemment vainqueur.
- S'il y en a plusieurs, on lance la pièce. Si c'est face qui sort, on élimine au hasard (par un jeu de type "chaises musicales" par exemple) un des enfants ; si c'est pile on en élimine deux (à moins qu'il n'en reste que deux, auquel cas on n'en élimine qu'un).
- Le gagnant est le dernier enfant non éliminé.

Question 1 : Si l'on considère le lancer d'une pièce comme opération fondamentale, trouver l'équation de récurrence correspondant à ce jeu dans le cas moyen.

Question 2 : Montrer par récurrence, à partir de cette équation, que l'algorithme précédent est en O(n), où n est le nombre initial d'enfants.

3 Mots décroissants

Soit A un alphabet fini, muni d'un ordre total $\leq$. On considère A^* l'ensemble des mots finis sur A.
Soit m un mot de longeur l. Pour tout $i, 1 \leq i \leq l$, on note m_i la ième lettre de m.
On dit qu'un mot m de longeur l est décroissant, s'il est vide ou si pour tout $i, 1 \leq i \leq l-1$, $m_{i+1} \leq m_i$. Si de plus il n'y a pas deux lettres égales dans le mot m, m est dit strictement décroissant.
Soit A^*_D le sous ensemble de A^* constitué des mots décroissants, et A^*_{SD} le sous-ensemble de A^*_D constitué des mots strictement décroissants.

Donner en pseudo-code un algorithme récursif pour chacune de ces fonctions puis déterminer sa complexité :

Question 3 : Decroissant : $A^* \rightarrow \{vrai, faux\}$
Cette fonction retourne vrai si et seulement si le mot considéré est décroissant.

Question 4 : Simplifie : $A^*_D \rightarrow A^*_{SD}$
Cette fonction retourne le mot de A^*_{SD} ayant le même ensemble de lettres que le mot initial, mais où les répétitions de lettres ont été éliminées.

Question 5 : Intersection : $A^*_{SD} \times A^*_{SD} \rightarrow A^*_{SD}$
Cette fonction retourne le mot de A^*_{SD} contenant les lettres qui sont dans les deux mots.