Algorithme

Chapitre 1

A lire Introduction to algorithms
lien du cours http ://sylvain.berbiqui.org

1.1 Introduction

on parle de fonction recurcive pour le recurcif et de terme pour le Λ calcul.

1.1.1 mesure de la complexité

La mesure est universelle
L'algorithme est un schema theorique.
2 notion de compleité :

- notion de temps : ce compte en nombre de passage
- notion d'espace memoire Hors connaissance du cours

taille combinatoire d'un entier X est X.
Le cout d'un algorithme est le nombre d'opération fondamentales exécutées par A sur I (Input).

$$Moy_A(n) = \sum_{I \in D_n} P_r[I].cout_A(I)$$

Chercher la preuve de :

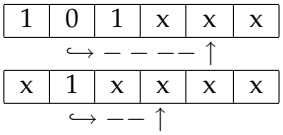
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

exercice p. 19

Au pire cas, l'algorithme fait 3 comparaisons.
en moyenne, il fait $\frac{8}{3}$ comparaisons.
complexité d'un pb : non defini pour le moment.

exercice p. 20

complexité au pire du cas n = 2 : 2
complexité au pire du cas n = 3 : 3
complexité au pire du cas n = 4 : 3
complexité au pire du cas n = 5 : 5



exo p23

on prend cos et sin

exercice p. 25

reponse 3
reponse 2

exercice p. 26

1)
$Max(f,g) \leq (f+g)(n)$
$\frac{1}{2}f(n) + \frac{1}{2}g(n) \leq f(n) \leq (f+g)(n)$
2)
$C_n^k \geq |a_k|n^k + \ldots + |a_0| \geq p(n) \geq \frac{1}{\sum_{i=1}^{k}|a_i|}n^k$

exercice p. 27

Supposons que f(n) $\in \theta$ (g(n)) est-il vrai que $2^{f(n)} \in \theta(2^{g(n)})$
$\rightarrow$ NON
f(n) = n^2 }
g(n) = $2n^2$ } $\Rightarrow 2^{n^2} \leq c2^{2n^2} = 2^{n^2} * c2^{n^2}$

exercice p. 29

$n! \ldots logn$

1.2 TRI

1.2.1 par insertion

exercice p. 56

p = 0 OK
$(p \rightarrow p+1) \Rightarrow 2p \rightarrow 2p+1$
$\underline{a_1 b_1, \ldots, a_p b_p} H_1 R, a_{p+1} b_{p+1}$
on sait que a_p est plus petit que a_{p+1}

exercice p. 57

cout = $2(\frac{P(P+1)}{2}) + \frac{P(P+1)}{2} = \frac{1}{8}(3n^2+6n) - 2 \sim \frac{3}{8}n^2$
$\neq \frac{2P(P+1)}{2} - 1 = \frac{n^2+n}{2} - 1 \sim \frac{n^2}{2}$

1.2.2 tri par fusion

```
procedure tri_fusion (tab [1..n])
    if taille (tab) = 1
        return tab
    else
        tab_1 = tri_fusion(tab[1..n/2])
        tab_2 = tri_fusion(tab[n/2..n])
        return fusion(tab_1, tab_2)
fin procedure
```

fusion (tab1, tab2) fait un trie des listes passées en parametre.

complexité : nlog(n)

1.2.3 tri à bulle

exercice p.75

```
algorithme procedure trie_a_bulle
    parametre globaux liste l
debut
    pour j=n decroissant jusqu'à 0 et bool faire
        bool = 0
        pour i=0 jusqu'a i¡j faire
            si liste[i] ¿ liste[i + 1]
                bool = 1
                swap(liste[i], liste[i + 1])
        fin pour
    fin pour
fin algo
```

complexité au pire en n^2
complexité $\overline{x}$:

1.2.4 tri par tas

exercice p 103

1 21 5 34 32 67 6 0
1 0 5 34 32 67 6 21
0 1 5 34 32 67 6 21
0 1 5 34 32 21 6 76
0 1 5 6 32 21 34 67
0 1 5 6 21 32 34 67
Il y a 14 comparaison

exercice p 104

renvoi la derniere valeur du tableau

exercice p 105

```
Algorithme procedure couleur
    i = 1
    couleur = color(rtab.[i])
    j = i
    i = i + 1
    tant que i != n faire
        si color(i) == couleur
            i++
        sinon
            swap(tab.[j], tab[i])
            i = i + 1
            j = j + 1
        fin si
    fin tant que
fin procedure
```

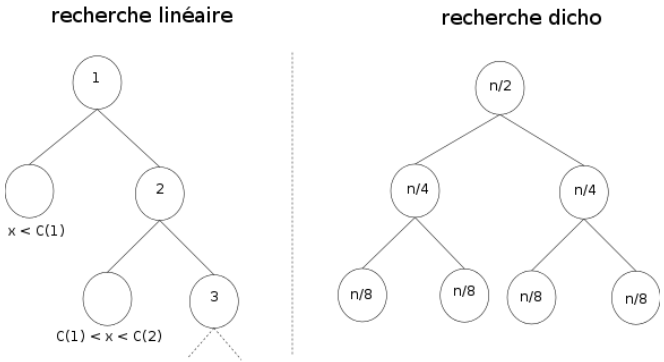
Probabilité :

$$\frac{(\frac{1-n_2}{n})n_2 + (\frac{1-n_1}{n})n_1}{2}$$

avec n_1 les boules rouge et n_2 les boules noires

1.2.5 Recherche dans une liste triée

exemple p.114



Chapitre 2

2.1 La programmation dynamique

exemple p.17

A_1A_2 : 10*100*5 = 5000
$(A_1A_2)A_3$: 10*5*50 = 2500
$\Rightarrow$ 7500
A_2A_3 : 100*5*50 = 25000
$A_1(A_2A_3)$: 10*100*50 = 50000
$\Rightarrow$ 75000

exemple

chercher le cout en trouvant le paranthesage maximal

A_1 A_2 A_3 A_4 A_5
10 100 5 50 40 10

	A_1	A_2	A_3	A_4	A_5
A_1	0	5000	$(A_1A_2)A_3$ 7500	$(A_1A_2)(A_3A_4)$ 17000	$(A_1A_2)(A_3A_4A_5)$ 17500
A_2	| |	0	25000	$A_2(A_3A_4)$ 30000	$A_2(A_3A_4A_5)$ 17000
A_3	| |	| |	0	10000	$(A_3A_4)A_5$ 12000
A_4	| |	| |	| |	0	20000
A_5	| |	| |	| +	—	0

2.1.1 PLSC

Application de l'algo p. 53

X = 1 0 0 0 1 0 1 1 0 0 1
Y = 0 1 1 0 1 1 1 0 0 1 0

Y\ X	1	0	0	0	1	0	1	1	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	$\searrow$1	$\searrow$1	1	$\searrow$1	1	1	$\searrow$1	$\searrow$1	1
1	0	$\searrow$1	1	1	$\searrow$2	2	$\searrow$2	$\searrow$2	2	2	$\searrow$2
0	0	1	$\searrow$2	$\searrow$2	2	$\searrow$3	3	3	$\searrow$3	$\searrow$3	3
1	0	$\searrow$1	$\searrow$2	2	$\searrow$3	3	$\searrow$4	$\searrow$4	4	4	$\searrow$4
1	0	$\searrow$1	2	2	$\searrow$3	3	$\searrow$4	$\searrow$5	5	5	$\searrow$5
1	0	$\searrow$1	2	2	$\searrow$3	4	$\searrow$4	$\searrow$5	5	5	$\searrow$6
0	0	1	$\searrow$2	$\searrow$3	3	$\searrow$4	4	5	$\searrow$6	$\searrow$6	6
0	0	1	$\searrow$2	$\searrow$3	3	$\searrow$4	4	5	$\searrow$6	$\searrow$7	7
1	0	$\searrow$1	2	3	$\searrow$4	4	$\searrow$5	$\searrow$5	6	7	$\searrow$8
0	0	1	$\searrow$2	$\searrow$3	4	$\searrow$5	5	5	$\searrow$6	$\searrow$7	8

2.1.2 Algorithme glouton

exercice p.67

tant que x $\geq$ 5 faire
    ens = ens + {5}
    x = x - 5
fin tant que
tant que x $\geq$ 2 faire
    ens = ens + {2}
    x = x - 2
fin tant que
tant que x $\neq$ 0 faire
    ens = ens + {1}
    x = x - 1
fin tant que

2.1.3 codage de HUFFMAN

exemple p. 71

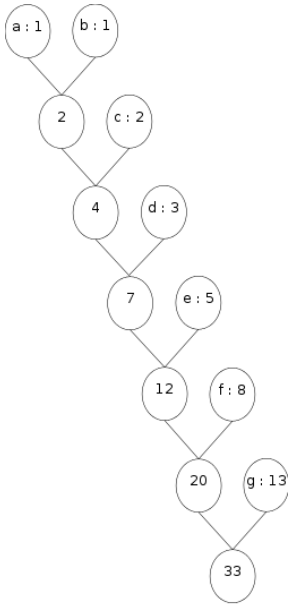
300 000 pour le code de longueur fixe, et 224 000 por le codage de longueur varaible.

exercice p. 78

exercice complementaire

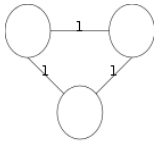
tableau des fréquences :

a	$\frac{1}{33}$
b	$\frac{1}{33}$
c	$\frac{2}{33}$
d	$\frac{3}{33}$
e	$\frac{5}{33}$
f	$\frac{8}{33}$
g	$\frac{13}{33}$



2.1.4 arbre de recouvrement minimal

exercice p. 85



–

–