Option Informatique

Emeric Tourniaire

20 septembre 2016

Table des matières

Chapitre 1

Introduction

1.1 Programme

1. Arbres 2. Graphes 3. Automates

· Machine de Babbag -> permet de faire des calculs de manière automatisée : premiers programmes.
· Ada Lovelace.
· Alan Turing : calculabilité. Qu'est-ce qu'on peut calculer ?
- Problème de l'arrêt. - Modèle de la machine de Turing. - // Church - Travail sur le décryptage d'Enigma.

1.2 Complexité

· Karp 1972
· Nombre d'étapes faites par un ordinateur.
· Les problèmes polynomiaux restent polynomiaux.
· Certains problèmes sont équivalents.
- P : résolubles polyniomalament. - NP - $P = NP$? Conjecture actuelle = non (très important) · Notation de Landau $O()$ ou $\Theta()$

1.3 Domaines

· Algorithmes exacts
- Démonstration · Algorithmes heuristiques
- pas très sérieux

· Optimisation linéaire
- Simplexe (exponentiel) / Ellipses (polynomial) - La méthode du simplexe est employée (elle est polynomiale sauf pour des points isolés)

· Cryptographie
- Décrypter = récupérer le contenu sans la clé - Chiffrer = chiffre un message avec la clé - Déchiffrer = lit le message avec la clé · TLS / RSA...

· Calcul numérique et info appliquée
- Données continues (analyse) - Données discrètes (ADN)
· Robotique

· Combinatoire : nombre fini de solutions et on doit trouver la meilleure.
- Jeux, optimisation

Chapitre 2

Structures de données

2.1 Tableau

Plage de mémoire continue dans laquelle on met des éléments. Accès instantané : tous les éléments prennent la même place.

```
[| 3 ; 5; 2 |]
t.(2);;
t.(2) <- -4;;
```

| Opération | Coût | | :————— : | :——— : | | taille | O(1) | | accès | O(1) | | modif | O(1) | | ajouter / supprimer | O(n) | | tri | O(nlog n) |

2.2 Listes

Pointeur vers une adresse mémoire qui contient une valeur et un pointeur vers l'adresse suivante.

| Opération | Coût | | :——————————————— : | :– : | | accéder à la tête | O(1) | | accéder à la queue (tout sauf le premier élément) | O(1) | | ajouter un élément en kème position | O(k) | | accéder à un élément en kème position | O(k) |

Remarque : La récursivité c'est bien, mais c'est mieux quand on sait s'en servir...

Chapitre 3

Arbres

3.1 Définitions générales

Généralités On parle d'arbres binaires étiquetés. L'ensemble de ces arbres est noté $A(N, F)$, où N est l'ensemble des valeurs possibles pour les nœuds et F est l'ensemble des valeurs possibles pour les feuilles.

Définition L'ensemble $A(N, F)$ est défini par :
· $F \subset A(N, F)$
· Si $a, b \in A(N, F), n \in \mathbb{N}$, alors $(n, a, b) \in A(N, F)$

```
type ('n, 'f) arbre_binaire =
  | Feuille of 'f
  | Noeud of 'n * ('n, 'f) arbre_binaire * ('n, 'f) arbre_binaire;;
```

Définition Le squelette d'un arbre, c'est l'arbre sans les étiquettes.

```
type squelette_binaire =
  | Rien
  | Jointure of squelette_binaire * squelette_binaire;;
```

3.2 Propriétés combinatoires

Définition La taille d'un arbre (ou d'un squelette) est le nombre de ses noeuds internes.

```
let rec taille_arbre sq = match sq with
  | Rien -> 0
  | Jointure(g, d) -> 1 + taille_arbre g + taille_arbre d;;
```

Définition La hauteur d'un arbre est définie par induction. La hauteur d'une feuille est nulle. La hauteur d'un arbre est $1+max(hauteur_{filsgauche}, hauteur_{filsdroit})$.

```
let rec hauteur_arbre sq = match sq with
  | Rien -> 0
  | Jointure(g, d) -> 1 + max (hauteur_arbre g) (hauteur_arbre d);;
```

Théorème Un arbre de taille n possède n+1 feuilles.

Démonstration

Par induction - Le résultat est vrai pour tout arbre de taille 0. - Soit $n \in \mathbb{N}$. Supposons le théorème vrai pour tout arbre de taille $\leqslant n$. - Soit $A = (n, g, d)$ un arbre de taille $n + 1$. - Le nombre de feuilles de A vaut :

- $\#f(A) = \#f(g) + \#f(d)$
- Le nombre de noeuds de A vaut :
- $\#n(A) = 1 + \#n(g) + \#n(d)$ - Donc $\#n(g) \leqslant n$ et $\#n(d) \leqslant n$
- Par hypothèse d'induction,
- $\#f(g) = \#n(g) + 1$ - $\#f(d) = \#n(d) + 1$
- Donc $\#f(a) = \#n(d) + \#n(g) + 2$ - Donc $\#f(a) = \#n(a) + 1$

Autre démonstration : - Soit A un arbre, on note n le nombre de nœuds et f le nombre de feuilles. - n + f - 1 est le nombre d'enfants dans l'arbre. - Or le nombre d'enfants est aussi 2n. - Donc $n = f - 1$.

Théorème Soit A un arbre de hauteur h et de taille n.
Alors $1 + \lfloor lg(n) \rfloor \leqslant h \leqslant n$

Démonstration La hauteur est le plus long chemin de la racine à une feuille. Si l'arbre est de hauteur h, ce chemin comporte $h + 1$ points, donc h noeuds, donc $n \geqslant h$.

Si l'arbre n'est pas complet, on le complète en A', également de hauteur h, complet.

On a alors $2^h - 1$ noeuds.
Donc on avait $n \leqslant 2^h - 1$
$lg(n) < h$
Donc $\lfloor lg(n) \rfloor \leqslant h - 1$

Démonstration bis, par induction · Si l'arbre contient 0 noeuds, le résultat est "vrai" (en prenant $lg(0) = -\infty$)
· Soit un arbre quelconque de hauteur h de taille n, avec g et d ses fils de hauteur hg hd, de taille ng nd.
· Par hypothèse, on a :

- $1 + \lfloor lg(ng) \rfloor \leqslant hg \leqslant ng$ - $1 + \lfloor lg(nd) \rfloor \leqslant hd \leqslant nd$

· Alors :

- $n = 1 + ng + nd \geqslant 1 + hd + hg \geqslant 1 + max(hd, hg) \geqslant h$
· On note $m = max(ng, nd)$
· On a :

- $h = 1 + max(hd, hg) \geqslant 2 + \lfloor lg(m) \rfloor \geqslant 1 + \lfloor lg(2m) \rfloor$
· De plus $\lfloor lg(2m) \rfloor = \lfloor lg(2m+1) \rfloor$ (changement de valeur sur les puissances de 2)
· Donc :

- $h \geqslant 1 + \lfloor lg(2m+1) \rfloor$ - $h \geqslant 1 + \lfloor lg(n) \rfloor$

Définition Un arbre est complet si toutes ses feuilles sont à hauteur h ou h - 1

3.3 Nombres de Catalan [HP]

Question : **Combien y a-t-il de squelettes de taille n distincts ?**
· n = 0 : 0
· n = 1 : 1
· n = 2 : 2
· n = 3 : 5

Définition On note C(n) le nombre de squelettes de taille n.

Propriété $C_0 = 1$
$\forall n \geqslant 1, C_n = \sum_{k=0}^{n-1} C(k)C(n-1-k)$

Démonstration On peut partitionner les squelettes de taille n en fonction de k le nombre de noeuds à gauche de la racine.
n | 0 | 1 | 2 | 3 | 4 ·— | - | - | - | - | –
C(n) | 1 | 1 | 2 | 5 | 14
Il y a alors n-1-k sommets à droite. Tout choix d'un arbre gauche et d'un arbre droit donne un arbre (total) distinct :
· $C_4 = C_0C_3 + C_1C_2 + C_2C_1 + C_3C_0$
· $C_4 = 5 + 2 + 2 + 5 = 14$

Ceci nous permet de calculer C(n).
Idée : **passer par une série génératrice**
Soit $C(z) = \sum_{n=0}^{+\infty} C_n z^n, z \in \mathbb{C}$
Si $C_n = o(\alpha^n)$, alors la série est bien définie pour $|z| < \dfrac{1}{\alpha}$
Alors :

$\cdot\ C(z) = C_0 + z \sum_{n=1}^{+\infty} C_n z^{n-1}$

$\cdot\ C(z) = 1 + z \sum_{n=1}^{+\infty} \sum_{k=0}^{n-1} C_k C_{n-1-k} z^{n-1}$

$\cdot\ C(z) = 1 + z \times C_z^2$

En effet :
$\cdot\ C(z)^2 = (\sum_{n=0}^{+\infty} C_n z^n) \times (\sum_{n=0}^{+\infty} C_n z^n)$

$\cdot\ C(z)^2 = \sum_{n=0}^{+\infty} \sum_{k=0}^{n} C_k z^k \times C_{n-k} z^{n-k}$

$\cdot\ C(z)^2 = \sum_{n=0}^{+\infty} \sum_{k=0}^{n} C_k C_{n-k} z^n$

$\cdot\ C(z)^2 = \sum_{n=1}^{+\infty} \sum_{k=0}^{n-1} C_k C_{n-1-k} z^{n-1}$

On a alors :
$\cdot\ C(z) = 1 + z \times C^2(z)$

Donc : $\cdot\ C(z) = \dfrac{1 - \sqrt{1-4z}}{2z}$

On peut ensuite redévelopper cette fonction pour retrouver :
$\cdot\ C_n = \dfrac{\binom{2n}{n}}{n+1}$

Ces nombres sont appelés nombres de Catalan.

Autre application : dénombrement des chaines bien parenthésées.
()((())()) : bien parenthésées
On peut construire une bijection entre les chaines bien parenthésées et les arbres.
Un fils gauche correspond à un encadrement, un fils droit à une concaténation.

3.4 Parcours d'un arbre

L'objectif de cette partie est d'analyser un arbre algorithmiquement.
Il existe 3 parcours "en profondeur d'abord", et 1 parcours en largeur.

```
   1
  / \
 a   6
    / \
   4   b
  / \
 c   d
```

Parcours | Idée | Exemple ·——- | ————————————————— | ——— ———-

Largeur | SOSA | 1, a, 6, 4, b, c, d Préfixe | $(racine, p(fils_gauche), p(fils_droit))$ | 1, a, 6, 4, c, d, b Infixe | $(p(f_g), r, p(f_d))$ | a, 1, c, 4, d, 6, b Suffixe | $(p(f_g), p(f_d), r)$ | a, c, d, 4, b, 6, 1

Astuce ! On fait le **contour de l'arbre** dans le sens trigo en regardant :
· **vers la droite** → Parcours préfixe
· **vers le haut** → parcours infixe
· **vers la gauche** → parcours suffixe

Le parcours infixe n'est pas injectif.
Le parcours suffixe est injectif (si on a la distinction feuille/noeud).