TD Info1

23 septembre 2004

1 Approfondissement du cours

1.1 Calcul du pgcd

Voici un programme:

```
f(a,b) =
 if b=0
  then a
  else if a<b
        then f(b,a)
        else f(a-b,b)
```

a. Que fait ce programme?
b. Justifiez sa terminaison.
c. Quel est l'intéret de ce programme comparé à l'algorithme d'Euclide classique?

1.2 Calcul de x^n

a. Pourquoi, dans la méthode binaire, doit-on enlever le premier SX?
b. Calculez les successeurs de 19,21,28,22 et 23 dans l'arbre de Knuth.

1.3 Recherche d'un élément dans un tableau

a. Écrire un programme qui dit si un élément x est présent dans un tableau(vecteur) T de taille n.
b. Si le tableau T est trié (par ordre croissant), y-a-t-il un moyen de faire un programme plus efficace?

2 Exercices

Exercice 1 Soit $(F_n)_{n\in\mathbb{N}}$ la suite de Fibonacci :

$$F_0 = 0, \quad F_1 = 1, \quad F_{n+2} = F_{n+1} + F_n \text{ pour tout } n \geq 0$$

a. Montrer que

$$F_n = \frac{1}{\sqrt{5}}(\phi^n - \overline{\phi}^n)$$

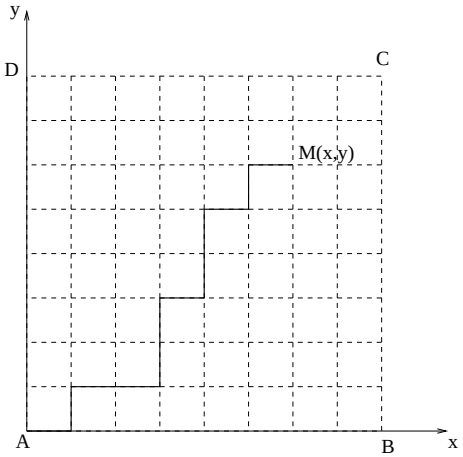
où ϕ et $\overline{\phi}$ sont les racines de l'équation $1 + z - z^2 = 0$.

b. Trouver une relation simple entre ϕ^n et les nombres de Fibonacci.

c. Montrer que $\{a\phi + b \mid a,b \in \mathbb{Z}\}$ est stable par produit. On notera $\mathbb{Z}[\phi]$ cet ensemble. En déduire le coût (en additions et multiplications entières)
   - d'une multiplication dans $\mathbb{Z}[\phi]$;
   - d'une élévation au carré dans $\mathbb{Z}[\phi]$.

d. Donner un algorithme « efficace » du calcul des nombres de Fibonacci et évaluer son coût.

Exercice 2 Soit un carré $ABCD$ de côté n découpé en n^2 carrés égaux. On relie A à un sommet quelconque $M(x,y)$ du quadrillage, en suivant les mailles avec la règle : « l'abscisse et l'ordonnée ne décroissent jamais ».



a. Évaluer le nombre de chemins reliant A à $M(x,y)$, noté $[A \to M]$. En particulier, préciser $[A \to C]$.

b. On affecte chaque sommet $M(x,y)$ du coefficient $[A \to M]$. Évaluer la somme de ces coefficients
   (i) sur une parallèle à BD coupant AB
   (ii) à l'intérieur du triangle ABD, sur son contour
   (iii) sur une parallèle à l'un des côtés
   (iv) à l'intérieur du carré $ABCD$, sur son contour.

c. Étant donnés les points $M(x,y)$, $M'(x+1,y)$ et $M''(x,y+1)$, calculer le nombre de chemins reliant A à C passant par le segment MM', et celui de chemins reliant A à C passant par le segment MM''. En déduire

$$C_{2n}^n = \sum_{y=0}^{n} C_{x+y}^x . C_{2n-x-y-1}^{n-y} = \sum_{x=0}^{n} C_{x+y}^x . C_{2n-x-y-1}^{n-x}$$

d. Calculer la somme, pour tous les chemins allant de A à C, des aires limitées par AB, BC et le chemin considéré.

TD Info1

30 septembre 2004

1 Problème de l'arrêt des machines de Turing

1.1 Ordre lexicologique

Soit Σ un alphabet fini, on définit la relation $\preceq$ sur les mots de Σ^*:

$$u \preceq v \Leftrightarrow \begin{array}{c} |u| < |v| \\ \text{ou} \\ |u| = |v| \text{ et } u \leq_{lex} v \end{array}$$

Montrer que $\preceq$ est un ordre total pour lequel il existe un plus petit élément. En déduire une bijection $\mathbb{N} \to \Sigma^*$.

1.2 Codage des machines de Turing

Soit $\mathcal{M} = (Q,\Gamma,\Sigma,\delta,0,Q_f)$ une machine de Turing. On suppose que $Q = [1 \dots n]$. On considère $\Gamma' = \Gamma \cup \{L,R,|\}$.

a. Donner un codage d'une règle $(p,x) \to (q,y,D)$ $(p,q) \in Q$, $(x,y) \in \Gamma, D = L/R$ dans Γ'^*.

b. En déduire un codage de la machine de Turing.

c. Montrer alors que l'ensemble des machines de Turing est dénombrable.

1.3 Problème de décision

Définition 1 Soit E un ensemble, $F \subseteq E$ et $c : E \to \Sigma^*$ un codage. Le problème est décidable s'il existe une machine de Turing M qui:

- $\forall x \in F$, M s'arrête si on lui donne en entrée $c(x)$ et rend $True$.
- $\forall x \in E \setminus F$, M s'arrête si on lui donne en entrée $c(x)$ et rend $False$.

Montrer qu'un problème est décidable si et seulement si il existe une machine de Turing qui reconnaît $c(F)$ et une machine de Turing qui reconnaît $c(E \setminus F)$.

1.4 Problème de l'arrêt des machines de Turing

Soit $E = \{(M,w)\}$ l'ensemble des couples (machine de Turing/mot). On considère que les machines et les mots sont ordonnées avec l'ordre lexicologique. Soit $F = \{(M_i,w_i) \mid M_i \text{ ne reconnaît pas } w_i\}$. Montrer qu'il n'existe pas de machine de Turing reconnaissant F. En déduire que le problème de l'arrêt des machines de Turing est indécidable.

2 Excercices

Exercice 1

a. Construire une machine de Turing qui reconnaît les mots de $\Sigma = \{a,b\}$ dont la longueur est une puissance de 2.

b. Construire une machine de Turing qui reconnaît les mots de $\Sigma = \{a,b\}$ qui comportent autant de a que de b.

Exercice 2

On considère la machine de Turing $\mathcal{M} = (Q,\Gamma,\Sigma,\delta,q_0,Q_f)$ définie par:

$$\begin{array}{lcl} Q & = & \{q_0,q_1,q_2,q_3\} \\ \Gamma & = & \{g,d,X,S,\#\} \\ \Sigma & = & \{g,d,S\} \\ Q_f & = & \{q_3\} \end{array}$$

et δ définie par le tableau suivant:

	g	d	X	S	$\#$
q_0	(q_0,g,R)	(q_1,X,L)	(q_0,X,R)	(q_0,S,R)	$(q_2,\#,L)$
q_1	(q_0,X,R)	(q_1,d,L)	(q_1,X,L)	$-$	$-$
q_2	$-$	$-$	(q_2,X,L)	(q_3,S,R)	$-$
q_3	$-$	$-$	$-$	$-$	$-$

a. Indiquer si les mots suivants sont reconnus par $\mathcal{M}$:
   - Sggddg
   - Sgdggdd
   - ggdd

b. Quel est le langage reconnu par cette machine de Turing?

TD Info1

7 octobre 2003

1 Fonctions primitives récursives

Le but de l'exercice est d'associer à toute suite finie d'entiers $S = (a_0,\ldots,a_n)$ un code numérique $\hat{s}$ de façon à ce que les opérations usuelles sur les listes soient primitives récursives.

1.1

Montrer que la fonction

$$\begin{array}{rccc} c_2 : & \mathbb{N}\times\mathbb{N} & \longrightarrow & \mathbb{N} \\ & (m,n) & \longmapsto & \frac{(m+n)(m+n+1)}{2} + m \end{array}$$

est primitive récursive et définit une bijection $\mathbb{N}\times\mathbb{N} \to \mathbb{N}$.

1.2

Trouver deux fonctions primitives récursives p_1^2 et p_2^2 de $\mathbb{N} \to \mathbb{N}$ telles que:

$$\begin{array}{rcll} p_1^2(c_2(n,m)) & = & n & \forall n,m \in \mathbb{N} \\ p_2^2(c_2(n,m)) & = & m & \forall n,m \in \mathbb{N} \\ c_2(p_1^2(l),p_2^2(l) & = & l & \forall l \in \mathbb{N} \end{array}$$

Ces deux fonctions sont appelées les *fonctions de projection associées à* c_2.

1.3

En déduire, pour chaque entier k, la fonction $C_k : \mathbb{N}^k \to \mathbb{N}$ bijective et les k fonctions de projection associées.

1.4

En déduire pour chaque série $s$un codage numérique $\hat{s}$ permetant de définir:

- une fonction primitive récursive longueur $: \mathbb{N} \to: \mathbb{N}$ qui donne la longeur de la liste s telle que $\hat{s} = n$.
- une fonction primitive récursive tete $: \mathbb{N} \to: \mathbb{N}$ qui donne le premier élément de la liste s telle que $\hat{s} = n$, et renvoie 0 si la liste est vide.
- une fonction primitive récursive queue $: \mathbb{N} \to: \mathbb{N}$ qui donne le code correspondant à la queue de la liste s telle que $\hat{s} = n$, et renvoie 0 si la liste est vide.

2 Complexité

Exercice 1 Pour deux entiers m et n on définit $Compte(m,n)$ ainsi:

```
s := 0
i := 1
if m > 1 then
   while i < n+1 do
     i := m * i
     for j := 1 to i do
       s := s + 1
     end
   end
Compte := s
```

a. Quelle est la complexité de ce programme (en nombre d'opérations entières) ?

b. Donnez un équivalent asymptotique simple de cette complexité.

Exercice 2 Soit $Ajout(n)$ la fonction défnie par le programme:

```
s := 0
for i := 1 to n do
  j := 1
  while j < i+1 do
    s := s + 1
    j := j + j
  end
end
Ajout := s
```

a. Quelle est la complexité de ce programme (en nombre d'additions) ?

b. Donnez un équivalent asymptotique simple de cette complexité.

TD Info1

14 octobre 2004

1. Un tri insertion optimal?

Dans le *tri par insertion* d'une liste, on trie successivement les premiers éléments de la liste, en insérant à la $i^{\text{ème}}$ étape le $i^{\text{ème}}$ élément à son rang parmi les $i-1$ éléments précédents (qui sont déjà triés entre eux).

1.1 Écrire un algorithme mettant en œuvre cette méthode.

1.2 Déterminer le nombre de comparaisons effectués par cet algorithme dans le cas le pire.

1.3 Exprimer le nombre de comparaisons effectués par cet algorithme en moyenne, pour une liste de n éléments distincts en fonction de la quantité $Inv(n)$ qui représente le nombre moyen d'inversions dans une permutation de n éléments.

1.4 Calculer $Inv(n)$ (on pourra utiliser un partitionnement judicieux de $\mathcal{S}_n$ en deux ensembles). En déduire le nombre moyen de comparaisons de l'algorithme.

1.5 Quel est le nombre d'affectations dans le cas le pire et en moyenne?

1.6 Optimiser l'algorithme précédent en réalisant une recherche plus efficace du rang d'insertion du $i^{\text{ème}}$ élément.

1.7 Déterminer le nombre de comparaisons effectués par cet algorithme dans le cas le pire.

1.8 Donner un équivalent asymptotique du nombre moyen de comparaisons. Cet algorithme est-il optimal?

2. Le problème du drapeau tricolore

On aligne n boules, réparties en un nombre quelconque de boules de couleur jaunes, vertes ou rouges. Ces boules sont diposées dans un ordre quelconque. La ligne de boules est représentée par un tableau v à une dimension de taille n, dont chaque élément $v(i)$ appartient à l'ensemble {vert, blanc, rouge}. Écrire un algorithme qui trie le tableau de telle façon que toutes les boules jaunes apparaissent au début, suivies des boules vertes puis des boules rouges. La contrainte est que le tri du tableau doit être réalisé en seul parcours.

3. Divers algorithmes de tri

On veut trier une suite de nombres (supposés différents) de la façon suivante : au départ, chaque nombre est mis dans une liste ne contenant que lui. Toutes ces listes sont mises dans une file d'attente (FIFO). L'algorithme consiste 'a sortir deux liste de la file, à les fusionner en une liste croissante et à la rentrer dans la file, jusqu'à ce que la file ne contiennent plus qu'une liste.

3.1 Écrire cet algorithme dans un "langage de programmation". Quelle est sa complexité?

3.2 Donner toutes les étapes du tri de la suite:

4,1,8,2,3,7,9,5,6

Si on remplace la file par une pile (FILO), à quel tri cet algorithme correspond-il?

3.3 On remplace les listes par des tas (cf cours) et la fusion de 2 listes par la fusion de tas. Donnez un algorithme qui effectue la fusion de deux tas. En déduire le coût de l'algorithme de tri. Quelle forme a l'arbre résultat quand on remplace une file par une pile?

TD 5 - Info1

21 octobre 2004

Ce TD est une initiation au *tri parallèle*. Nous étudierons les *réseaux de tri* qui permettent de trier une liste de taille n avec une complexité de l'ordre de $O(\ln^2 n)$.

1 Définitions

On dispose de *comparateurs* tous identiques qui permettent de comparer deux nombres x et y.

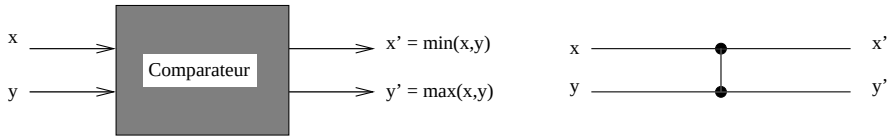


FIG. 1 – *Schématisation d'un comparateur*

Un réseau de comparaisons est un ensemble de comparateurs reliés par des câbles (sans cycle). Il a donc autant d'entrées que de sorties, et est représenté par n lignes parallèles. Un *réseau de tri* est un réseau de comparaisons tel que, pour toute séquence d'entrée (suite de n valeurs), la séquence de sortie est triée par ordre croissant.

La *profondeur* d'un câble est définie ainsi:
- Si c'est une entrée du réseau, la profondeur vaut 0.
- Si les entrées d'un comparateur sont des câbles de profondeur p_1 et p_2 alors la profondeur des **deux** câbles de sortie est $1 + max(p_1,p_2)$.

La profondeur d'un réseau est le maximum des profondeurs des câbles. Cette profondeur permet de donner une mesure du temps nécessaire au tri. Pourquoi?

2 Un exemple

Voici un exemple de réseau de taille 4.

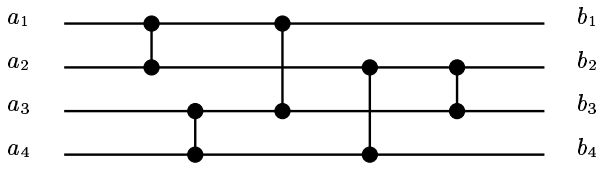


FIG. 2 – *Exemple de réseau de tri*

a. Simuler l'exécution de ce réseau avec l'entrée (9,3,1,5).
b. Si on rajoute un comparateur quelque part dans ce réseau, le nouveau réseau est-il toujours un réseau de tri?

c. Construire un réseau de tri de taille 4 qui reprend le principe du tri par insersion.
d. Montrer que tout réseau de tri de taille n a une profondeur au moins égale à $\log_2 n$.

A partir de maintenant on suppose que:
– la taille du réseau, n, est une puissance de 2.
– les valeurs d'entrée sont prises dans l'ensemble $\{0,1\}$.

3 Trieuse bitonique

Une *séquence bitonique* est une séquence "doublement monotone": croissante puis décroissante, ou inversement. Les séquences bitoniques binaires sont de la forme $0^i1^j0^k$ ou $1^i0^j1^k$.

Un *séparateur* de taille n est un réseau de comparaisons qui compare l'entée k à l'entrée $k + n/2$.

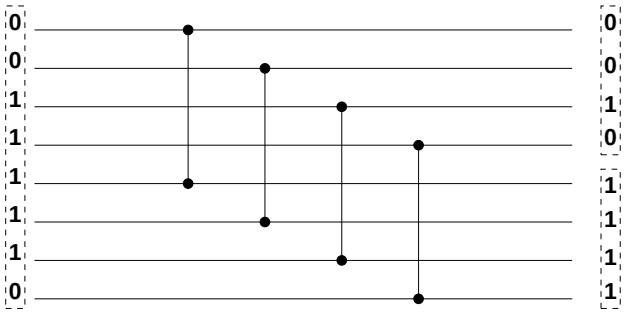


FIG. 3 – *Séparateur de taille 8*

a. Montrer que si la séquence d'entrée est bitonique, alors la sortie du séparateur est formée de deux séquences de taille $n/2$, l'une étant bitonique, l'autre étant pure (que des 0 ou que des 1), tous les nombres de la séquence du bas étant supérieurs à ceux de la séquence du haut. Indication: faire une étude par cas en supposant que la séquence d'entrée est de la forme $0^i1^j0^k$.
b. Construire un *trieur bitonique* en utilisant des séparateurs.
c. Quelle est la profondeur de ce trieur? Et le nombre de comparateurs utilisés?

4 Réseau de tri

a. Construire un *fusionneur* qui, étant donné deux séquences triées de taille $n/2$, sort une séquence triée de taille n.
b. Quelle est la profondeur du fusionneur? Et le nombre de comparateurs utilisés?
c. En déduire la contruction d'un réseau de tri pour des séquences binaires quelconques. Quelle est la profondeur de ce réseau de tri? Et le nombre de comparateurs utilisés?

5 Principe du zero-un

Pour le moment, on n'a considéré que des entrées binaires. Le but de cette partie est de montrer qu'un réseau de tri binaire peut trier n'importe quels nombres.

Soit f monotone croissante. Montrer que si un réseau de comparaisons transforme une séquence $\langle a_1,a_2,\ldots,a_n\rangle$ en une séquence $\langle b_1,b_2,\ldots,b_n\rangle$, alors ce réseau transforme la séquence $\langle f(a_1),f(a_2),\ldots,f(a_n)\rangle$ en la séquence $\langle f(b_1),f(b_2),\ldots,f(b_n)\rangle$.

En déduire que si un réseau de tri est capable de trier toutes les séquences binaires, alors il est aussi capable de trier toute séquence de nombres.

TD6 - Info1

28 octobre 2004

1. Le problème des 8 dames

Un échiquier est composé de 8×8 cases. Une dame peut prendre une pièce qui se trouve sur la même ligne/colonne/diagonale qu'elle.

1.1 Essayer de placer "à la main" 8 dames sans qu'aucune ne soit en prise avec une autre.

1.2 Ecrire un algorithme qui permet de résoudre ce problème "brutalement". Quelle est sa complexité au pire ? Et si on a de la chance ?

1.3 Discuter des moyens permettant d'améliorer un peu l'algorithme précédent.

2. Gestion de la mémoire

On souhaite enregistrer n fichiers $F_1, \ldots, F_n$ sur une mémoire de taille L. Chaque fichier F_i occupe un espace mémoire a_i. On suppose que $\sum_{i=1}^{n} a_i > L$; on ne peut donc pas enregistrer tous les fichiers, mais seulement un sous-ensemble $Q \subset F$. Le *quotient d'utilisation* est défini par:

$$\frac{\sum_{F_i \in Q} a_i}{L}$$

2.1 On cherche a enregistrer un maximum de fichier.

a. Proposer une stratégie pour résoudre ce problème. De quel type d'algorithme s'agit-il ? Justifiez la correction de cette stratégie.

b. Ecrire un algorithme qui met en oeuvre cette stratégie, en supposant que les fichiers F_i sont ordonnés "convenablement". Quelle est sa complexité au pire ?

c. A quel point le quotient d'utilisation peut-il être petit ?

2.2 On souhaite maintenant sélectionner un ensemble Q qui maximise ce quotient d'utilisation.

a. Quelle serait la stratégie gloutonne pour résoudre ce nouveau problème ?

b. Cette stratégie donne-t-elle toujours le bon résultat ?

2.3 Supposons que l'on accorde à chaque fichier une certaine valeur v_i. Le but est de sélectionner un ensemble Q qui maximise

$$\sum_{F_i \in Q} v_i$$

a. De quel problème s'agit-il ?

b. Quelle méthode faut-il employer ?

c. En déduire une méthode pour résoudre le problème précédent.

3. Le spéculateur malhonnête

Un homme d'affaire veut jouer en bourse, et il s'est arrangé pour connaître à l'avance le cours des actions d'une société pour les n prochains jours. On suppose que:

- le cours d'une action varie d'un jour à l'autre, mais reste stable durant toute la journée: ces valeurs sont données par un tableau $C[1..n]$.
- le spéculateur peut accomplir chaque jour une seule opération (codée par u):
  a. il ne fait rien, $u = 0$
  b. il achète une action, $u = -1$
  c. il vend une action, $u = 1$
- Au départ, il ne possède aucune action. Au jour $n+1$, il est repéré et doit abandonner toutes les actions qu'il possède alors.

On veut déterminer quelles opérations permettent de dégager un bénéfice maximal.

3.1 Quel type d'algorithme permet de résoudre ce problème ?

3.2 Donner la formule $V(j, x)$ qui permet de calculer la valeur d'un stock de x actions le jour j.

3.3 En déduire un algorithme qui permet de calculer la stratégie optimale.

4. La triangulation des polyèdres convexes

Pour un polyèdre convexe, on appelle *triangulation* une partition formée par des triangles. On suppose qu'un polyèdre est décrit par une suite de n sommets adjacents $(s_1, \ldots, s_n)$ $(n \geq 3)$.

4.1 Pour un polyèdre donné ayant n sommets, combien existe-t-il de triangulations distinctes ?

4.2 On associe à chaque triangle un poids correspondant à la somme des longeurs de ses 3 côtés. Donner un algorithme efficace qui trouve une triangulation minimisant la somme des poids des triangles.

TD7 - Info1

4 novembre 2004

1. La triangulation des polyèdres convexes

Pour un polyèdre convexe, on appelle *triangulation* une partition formée par des triangles. On suppose qu'un polyèdre est décrit par une suite de n sommets adjacents $(s_1, \ldots, s_n)$ $(n \geq 3)$.

1.1 Pour un polyèdre donné ayant n sommets, combien existe-t-il de triangulations distinctes ?

1.2 On associe à chaque triangle un poids correspondant à la somme des longeurs de ses 3 côtés. Donner un algorithme efficace qui trouve une triangulation minimisant la somme des poids des triangles.

2. Arbre couvrant minimum

Soit un ensemble de n sommets. Un *arbre couvrant* est un ensemble de $n-1$ segments reliant tous les points.

2.1 Donner un algorithme efficace qui permet de donner un arbre couvrant *minimal* (qui minimise la somme des longueurs des $n-1$ segments).

2.2 De quel type d'algorithme s'agit-il ? Quelle est sa complexité ?

3. Appartenance d'un point à un polyèdre

Soit P un polyèdre donné par un ensemble de n sommets $s_1, \ldots, s_n$. On veut trouver un algorithme qui, étant donnés P et un point m, détermine si m est à l'intérieur de P.

3.1 On suppose que P est convexe. Donner un algortihme qui permet de résoudre ce problème en temps linéaire.

3.2 Si P est quelconque, on va utiliser une demi-droite Δ horizontale issue de m. Quel est le rapport entre le nombre de segments $[s_i, s_{i+1}]$ qui ont une intersection avec Δ et le fait que m soit à l'intérieur de P? En déduire un algorithme pour résoudre le problème dans le cas général. Quelle est sa complexité ?