École polytechnique
Promotion 2001

Majeure informatique

**Cours de conception et analyse d'algorithmes**
PC 6 – algorithmes probabilistes

### 1. Un algorithme probabiliste pour 2-SAT

Une formule 2-SAT consiste en $p$ clauses $u_i = y_i \vee z_i$ sur $n$ variables. On considère l'algorithme probabiliste suivant pour chercher une assignation satisaisant la formule ($m$ étant un paramètre) :

- partir d'une affectation quelconque de valeurs aux variables $x = (x_1, \ldots, x_n)$.
- pour $c$ de 1 à $m$ faire
  - s'il existe une clause non satisfaite,
    - changer la valeur d'une des deux variables choisie au hasard
  - sinon renvoyer "satisfiable"
- renvoyer "non satisfiable"

**i.** L'algorithme est-il de type Monte Carlo ou Las Vegas ? Préciser. Quelle est la probabilité d'erreur si la formule est non satisfiable ?

On suppose qu'il existe une affectation $u$ satisfaisant toutes les clauses.

**ii.** Soit $E(i)$ le maximum, sur toutes les assignations $x$ telles que la distance de Hammming $d(x, u)$ soit inférieure ou égale à $i$, du nombre moyen d'itérations de l'algorithme en partant de $x$. Montrer que :

$$\left\{ \begin{array}{lcl} E(0) & = & 0 \\ E(n) & = & 1 + E(n-1) \\ E(i) & \leq & 1 + (E(i-1) + E(i+1))/2 \text{ pour } 1 \leq i \leq n-1. \end{array} \right.$$

**iii.** En déduire successivement que $E(i) \leq 2(n - i + 1) - 1 + E(i - 1)$ pour tout $i \in \{1 \ldots, n-1\}$, puis une borne supérieure sur $E(n)$.

**iv.** Par l'inégalité de Markov, en déduire que si $m > 2n^2$, alors la probabilité d'obtenir l'affectation correcte est au moins égale à $1/2$.

**v.** Que penser d'une généralisation à 3-SAT ?

### 2. Coupure minimale

Soit $G$ un graphe connexe non orienté, dont les arêtes peuvent être multiples, avec $n$ sommets. Une *coupure* dans $G$ est un ensemble d'arêtes dont la suppression rend $G$ non connexe. Une coupure minimale est une coupure de cardinalité minimale. Voici un algorithme probabiliste pour la détermination d'une coupure minimale.

*Contracter* $\{u, v\}$ signifie : fusionner ces deux sommets en un seul sommet $w$ ; les arêtes incidentes à $w$ sont les arêtes qui étaient précédemment incidentes à $u$ ou à $v$ ; puis, supprimer les boucles (induites par les arêtes $\{u, v\}$).

L'algorithme contracte une suite d'arêtes choisies au hasard, jusqu'à ce que le graphe n'ait plus que deux sommets ; il renvoie alors comme valeur de coupure le nombre d'arêtes entre ces deux sommets.

**i.** Montrer que l'opération de contraction ne diminue pas la valeur de la coupe minimale.

**ii.** Soit $k$ la taille de la coupe minimale. Montrer que $G$ a au moins $kn/2$ arêtes.

**iii.** Quelle est la probabilité qu'après $\ell$ étapes, aucune des arêtes d'une coupe minimale $C$ de taille $k$ n'ait été contractée ? En déduire une borne inférieure sur la probabilité que l'algorithme renvoie la valeur d'une coupe minimale.

**iv.** Quelle est la complexité de l'algorithme ? Construire un algorithme polynomial renvoyant la bonne valeur avec probabilité $1 - O(1/n^2)$.

**3. Vérification de calculs matriciels**

L'algorithme le plus efficace pour la multiplication de deux matrices carrées de côté $n$ actuellement connu est de complexité $O(n^{2.376})$. Hélas, il est extrêmement complexe, et toute implémentation est sujette à erreurs.

On souhaite tester le programme en vérifiant sur un certain nombre de couples $(A, B)$ si le résultat $C$ fourni est bien le bon.

**i.** Quelle est la complexité du calcul de $Ax$ pour un vecteur $x$ donné, par une méthode naïve ?

**ii.** Soit $r$ un vecteur à coordonnées 0 ou 1 choisi au hasard. On suppose que $AB \neq C$. Montrer que la probabilité que $ABr = Cr$ est au plus 1/2.

**iii.** Comment et en quelle complexité peut-on être convaincu que $AB = C$ avec probabilité $1 - O(1/n)$ ?

**4. Stratégies de cache**

On s'intéresse aux *stratégies de cache* déterministes et probabilistes. Dans les ordinateurs, il existe au moins deux niveaux de mémoire physique : le cache, petit mais d'accès rapide ; et la RAM, beaucoup plus grande mais d'accès plus lent. Le problème, étant donné une suite de requêtes, est de choisir à tout instant quelles données garder dans le cache. Soit $k$ le nombre d'unités de mémoire du cache. À chaque fois qu'une requête sollicite une information non présente, il faut la charger dans le cache, et l'algorithme doit pour cela choisir une information à éliminer du cache.

On considère un algorithme $O$ optimal en ce sens qu'il a connaissance de toute la suite des requêtes futures et peut ainsi choisir sa stratégie d'élimination en conséquence. On va comparer les performances de $O$ à celles d'un algorithme déterministe quelconque $A$, puis à celle d'un algorithme probabiliste donné **Marker**.

**i.** On suppose qu'au démarrage $A$ et $O$ ont les mêmes $k$ objets $x_1, \dots x_k$ dans leurs caches. On construit une suite de requêtes de la façon suivante : la première requête est pour $x_{k+1}$ qui n'est dans aucun des deux caches ; à chaque étape, la requête porte sur l'unique élément, parmi $\{x_1, \dots, x_{k+1}\}$, qui n'est pas dans le cache courant de $A$. Montrer que lors des $k$ premières requêtes, puis des $k$ suivantes, $A$ connaît $k$ échecs et $O$ au plus un seul.

**ii.** On considère la stratégie probabiliste **Marker** suivante. À chaque emplacement du cache, on associe un bit initialement mis à 0. Pour chaque requête, on opère de la façon suivante :

- si l'objet est en cache, on met le bit correspondant à 1.
- si l'objet n'est pas en cache, on le charge, on rejette un objet à un emplacement choisi au hasard *parmi ceux dont le bit est à 0*, et on met le bit de cet emplacement à 1.

Une fois que les bits de tous les emplacements sont à 1, on les remet tous à 0 et on recommence. Une *étape* est la période entre deux remises à 0 des bits. Par abus, on dira qu'un objet est marqué s'il se trouve à un emplacement marqué.

**a.** Montrer que lors d'une étape de **Marker**, exactement $k$ objets distincts sont demandés.

**b.** On dit qu'un objet est *périmé* s'il était dans le cache au début de l'étape et n'est pas marqué, et *propre* s'il n'est ni marqué, ni périmé (il ne se trouve donc pas dans le cache de **Marker** dans ce cas). Soit $\ell$ le nombre d'objets propres requis lors de l'étape considérée, $d_I$ le nombre d'objets du cache de $O$ qui ne sont pas dans le cache de **Marker** en début d'étape, et $d_F$ l'analogue en fin d'étape.

Montrer que le nombre d'échecs de $O$ est au moins $\max(\ell - d_I, d_F)$, donc $(\ell - d_I + d_F)/2$. En déduire que le nombre d'échecs en $n$ étapes est au moins $(n\ell - k)/2$.

**c.** On suppose que lors d'une étape, on ait rencontré $j$ requêtes à des éléments propres, $i$ requêtes à des éléments impropres. Montrer que la probabilité qu'un élément impropre non encore requis soit présent dans le cache est $(k-i-j)/(k-i)$, et que la probabilité d'échec est donc $j/(k-i)$. En déduire que le pire des cas est : d'abord $\ell$ requêtes à des éléments propres, puis $k - \ell$ requêtes à des éléments impropres.

**d.** En déduire qu'en moyenne **Marker** rencontre au plus $2\sum_{j=1}^{k} 1/j = O(\log k)$ échecs quand $O$ en rencontre 1.