Algorithmique avancée

Série d'exercices n° 1 – Algorithmes randomisés
(Correction)

Exercice 1 : k-SAT

Soit $x_1, x_2, \cdots, x_n$ n variables booléennes. On désigne par *littéral* x_i ou sa négation $\bar{x}_i$. Une clause de 3-SAT est de la forme :

$$r \vee s \vee t$$

où r, s et t appartiennent à l'ensemble des littéraux $\{x_1, \cdots, x_n, \bar{x}_1 \cdots, \bar{x}_n\}$. Une formule 3-SAT est constituée d'un ensemble de clauses. On dit qu'elle est SAT ssi il existe des affectations des x_i qui vont satisfaire toutes les clauses.
On construit une formule aléatoire 3-SAT (notée par la suite $F_{n,m}$) avec $2n$ littéraux et m clauses choisies uniformément au hasard parmi les $8\binom{n}{3}$ clauses possibles.

1. Montrer que si $m = 6n$ alors la probabilité que $F_{n,m}$ soit soit SAT est exponentiellement faible.

▷ *Soit X la v.a. qui compte le nombre d'affectations des x_i telles que $F_{n,m}$ soit SAT. Pour l'affectation numéro j (parmi les 2^n affectations), on définit*

$$X_j = \begin{cases} 0, & \textit{si l'affectation ne satisfait pas } F_{n,m} \\ 1, & \textit{sinon.} \end{cases}$$

De même, on définit X_j^k

$$X_j^k = \begin{cases} 0, & \textit{si l'affectaction numéro } j \textit{ ne satisfait pas la } k-\textit{ième clause de } F_{n,m} \\ 1, & \textit{sinon.} \end{cases}$$

On a

$$X = \sum_j X_j = \sum_j \left(\prod_k X_j^k \right)$$

*et $\mathbb{E}[X] = \sum_j \mathbb{E}[X_j]$ (par linéarité de l'espérance) puis $\mathbb{E}[X] = \sum_j \left(\prod_k \mathbb{E}[X_j^k]\right)$ car $\mathbb{E}[AB] = \mathbb{E}[A]\mathbb{E}[B]$ si A et B sont indépendants. X_j^a et X_j^b sont indépendants car les deux clauses a et b sont indépendantes.
Maintenant on a $\mathbb{E}[X_j^k] = \mathbb{P}[X_j^k] = \frac{7}{8}$. Après calcul, on trouve*

$$\mathbb{E}[X] = \sum_{j=1}^{2^n} \left(\prod_{k=1}^{6n} \frac{7}{8} \right) = \left(2\frac{7^6}{8^6} \right)^n .$$

Comme $27^6/8^6 \sim 0.898 < 1$, l'espérance de X est exponentiellement faible. On utilise l'inégalité de Markov pour conclure :

$$\mathbb{P}[X \geq 1] \leq \mathbb{E}[X] \textit{(exponentiellement petit quand n est grand).}$$

2. Généraliser le résultat pour $k > 3$ en montrant qu'il existe une constante c telle qu'une formule k-SAT avec n variables et m clauses est SAT avec une probabilité exponentiellement faible pour $m \geq cn$.

▷ *Il suffit de substituer $\frac{7}{8}$ par $\frac{2^k-1}{2^k}$. Pour $m = cn$, l'espérance de X est cette fois-ci*

$$\left(2 \left(\frac{2^k - 1}{2^k} \right)^c \right)^n .$$

On choisit une constante c telle que $\left(\frac{2^k-1}{2^k}\right)^c < \frac{1}{2}$.

Exercice 2 : Clique maximale

Une clique d'un graphe est un ensemble de sommets tel que le sous-graphe induit est complet. On considère l'algorithme suivant qui prend en entrée un graphe avec n sommets :

Algorithme 1 : MonGlouton

Data : Graphe avec n sommets
Result : Une clique maximale
On initialise un ensemble K à $K = \{\}$;
for $i := 1$ *to* n **do**
  if *sommet* i *est adjacent à tous les sommets dans* K **then**
    $K = K \bigcup \{i\}$;
  end
end
return K;

1. Montrer que cet algorithme renvoie bien une clique maximale (au sens de l'inclusion). Quelle est sa complexité (en fonction de n le nombre de sommets) ?

▷ *On montre par contradiction que l'algorithme renvoie bien une clique maximale. Supposons qu'un ensemble S forme une clique et qu'un sommet $j \notin S$ soit adjacent à tous les sommets de S. Lors de la boucle "for", le sommet j aurait du être rajouté à l'ensemble K "courant". Pour la complexité, si le graphe est donné comme une matrice, comme on teste les adjacences pour chaque colonne i les $O(n)$ lignes, la complexité est en $O(n^2)$.*

2. Soit $G(n, p = 1/2)$ le graphe aléatoire où chacune des $\binom{n}{2}$ arêtes est présente (indépendamment les unes des autres) avec la probabilité p (ici $1/2$). Soit alors p_k la probabilité que l'algorithme ci-dessus renvoie une clique de taille k. Comparer p_k à

$$\binom{n}{k}\left(1 - \frac{1}{2^k}\right)^{n-k}.$$

3. Soit ε une constante telle que $0 < \varepsilon < 1$. On définit $k^\star = (1 - \varepsilon)\log_2 n$. Montrer que

$$p_{k^\star} \leq e^{-O(n^\varepsilon)} \text{ pour n suffisament grand.}$$

▷ *Supposons que la taille de K est un entier k fixé, chaque sommet v pas encore dans K n'y rentre pas avec probabilité $1 - 2^{-k}$ (v n'est pas adjacent à tous les sommets de K). De même pour $v_1, v_2, \cdots, v_i$ i sommets distincts, aucun ne rentre dans K avec probabilité $(1 - 2^{-k})^i$. Il se peut que ces sommets forment une clique de taille i mais si l'algorithme commence par mettre d'autres sommets dans K alors ces $v_1, \cdots, v_k$ n'y seront jamais. D'où*

$$p_k \leq \binom{n}{k}(1 - 2^{-k})^{n-k}.$$

4. En déduire

$$\mathbb{P}[\text{MonGlouton renvoie une clique de taille } \leq (1-\varepsilon)\log_2 n] \to 0 \text{ quand n est grand, i.e. } n \to +\infty.$$

▷ *Pour $k = (1 - \varepsilon)\log_2 n$ on a*

$$(1-2^{-k})^{n-k} = (1-2^{-(1-\varepsilon)\log_2 n})^{n-(1-\varepsilon)\log_2 n} = (1-\frac{n^\varepsilon}{n})^n(1-\frac{n^\varepsilon}{n})^{-(1-\varepsilon)\log_2 n} \sim e^{-n^\varepsilon}(1-\frac{n^\varepsilon}{n})^{-(1-\varepsilon)\log_2 n} \leq 2e^{-n^\varepsilon}$$

Pour ce même k on a

$$\binom{n}{k} \leq \left(\frac{n\,e}{k}\right)^k \leq n^k e^k = \left(1 - \frac{n^\varepsilon}{n}\right)^{-(1-\varepsilon)\log_2 n} \ll e^{1/2n^\varepsilon}$$

(le $1/2$ est arbitraire mais ça marchera pour une constante quelconque). On multiplie et on obtient le résultat.

Exercice 3 : $2n$ bits aléatoires, n bit à 1 et n bit à 0

On veut générer *aléatoirement uniformément* une suite de $2n$ bits contenant exactement n bits à 1 et n bits à 0.

1. Un algorithme basique consisterait à tirer $2n$ bits jusqu'à ce qu'on obtienne satisfaction (n bits à 1 et n bits à 0). Montrer qu'en moyenne on effectuera alors $O(n^{3/2})$ tirages de bits. *Aide :* on peut utiliser la formule de Stirling $n! \sim \sqrt{2\pi n}(n/e)^n$ pour l'analyse.

▷ *La probabilité de tomber sur ce qu'on cherche est*

$$\frac{\binom{2n}{n}}{2^{2n}} = O(n^{-1/2}).$$

En moyenne, on bouclera donc sur $O(n^{1/2})$ où dans chaque boucle on lancera n tirages. D'où le résultat.

2. Un algorithme un peu moins basique consisterait à tirer au hasard les n positions des 1 dans le tableau à $2n$ éléments. En vous inspirant du cours, montrer qu'en moyenne ce nouveau algorithme s'exécute en $O(n \log n)$.

▷ *Pour collecter n coupons 2 à 2 différents, il faut en moyenne $O(n \log n)$ tirages. Ces coupons serviront à placer les* 1.

3. Que fait l'algorithme suivant (connu sous le nom de *Fisher-Yates Shuffle* ou *Knuth Shuffle*) ?

Algorithme 2 : KNUTH SHUFFLE

for $(i = 0\,;\, i \leq n-1\,;\, i++)$ **do**
  tab$[i] = i$;
end
for $(i = 0\,;\, i \leq n-2\,;\, i++)$ **do**
  $j =$ Uniforme $(0, n-i)$ /⋆ Un entier aléatoire j tel que $0 \leq j \leq n-i$ ⋆/;
  échanger(tab$[i]$, tab$[i+j]$);
end

▷ *Cet algorithme génère uniformément au hasard une permutation de n éléments après $O(n)$ itérations.*

4. En déduire un algorithme linéaire pour le problème courant (en supposant que la fonction Uniforme a un coût unitaire).

▷ *On place les* 1 *et les* 0 *arbitrairement puis on permute aléatoirement les $2n$ éléments.*