Algorithmique avancée

Série d'exercices n° 1 – Algorithmes randomisés

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 SAT est exponentiellement faible.
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 $m \geq cn$.

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 ⋃ {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) ?
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
$$p_{k^\star} \leq e^{-O(n^\varepsilon)} \text{ pour n suffisament grand.}$$
4. En déduire
$$\mathbb{P}[\text{MonGlouton renvoie une clique de taille } \leq (1-\varepsilon)\log_2 n] \to 0 \text{ quand n est grand, i.e. } n \to +\infty\,.$$

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.

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 $O(n \log n)$.

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 ≤ n − 1 ; i++) do
    tab[i] = i;
end
for (i = 0 ; i ≤ n − 2 ; i++) do
    j = Uniforme (0, n − i)          /⋆ Un entier aléatoire j tel que 0 ≤ j ≤ n − i ⋆/;
    échanger(tab[i], tab[i + j]);
end
```

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