TD 5

Exercice 1. 2-SAT

L'algorithme utilisé pour k-SAT ($k \geq 3$) fonctionne également pour le problème 2-SAT. En réalité, il suffit de considérer la boucle intérieure WALK, comme nous allons le voir. On considère, comme dans le cours, la marche aléatoire donnée par la distance entre l'affectation courante et une affectation satisfaisante fixée. *Attention, on considère la vraie marche aléatoire, pas la marche aléatoire avec une infinité d'états.*

On note, pour $d \geq 0$, Z_d la variable aléatoire qui représente le nombre d'étapes nécessaires pour atteindre l'origine depuis l'état d, et E_d l'espérance de Z_d.

1. Donner le graphe associé à la marche aléatoire.
2. Exprimer E_d en fonction de E_{d-1} et E_{d+1}, en donnant les cas de base.
3. Montrer que $E_{d+1} = E_d + 2(n-d) - 1$ pour tout $d < n$.
4. En déduire que $E_d = d(2n-d)$ pour tout $d \leq n$.
5. Conclure en donnant et analysant un algorithme Monte Carlo pour 2-SAT, de complexité polynomiale et dont la probabilité de succès est $\geq 1/2$.
6. En déduire un algorithme Monte Carlo de complexité polynomiale pour 2-SAT, avec cette fois-ci une probabilité de succès $\geq 1 - 2^{-n}$ où n est le nombre de variables de la formule en entrée.

Exercice 2. *Connectivité dans les graphes*

Une application très directe des marches aléatoires est d'étudier des problèmes sur les graphes. On s'intéresse ici à la s-t-connectivité : étant donné un graphe non orienté $G = (V, E)$ et deux sommets s et t de G, la question est de savoir s'il existe un chemin entre s et t.

1. Le problème peut se résoudre grâce à un parcours en profondeur. Quelle est la complexité de l'algorithme obtenu ?

L'inconvénient du parcours en profondeur est de nécessiter une mémoire $\Omega(n)$ où n est le nombre de sommets du graphe (puisqu'on doit conserver la liste des sommets visités). On souhaite montrer qu'on peut résoudre le problème avec une mémoire $O(\log n)$. Pour cela, on effectue une marche aléatoire sur le graphe : à chaque étape, on passe d'un sommet à l'un de ses voisins de manière équiprobable, en commençant initialement en s.

2. Pour deux sommets u et v du graphe, on note h_{uv} l'espérance du nombre d'étapes que met une marche aléatoire pour aller de u et v. *On admet que* $h_{uu} = 2m/\deg(u)$ *où* $\deg(u)$ *est le nombre de voisins de* u *et* m *le nombre d'arêtes du graphe* [1].
   - **i.** Donner une condition nécessaire et suffisante pour avoir $h_{uv} < +\infty$.
   - **ii.** Montrer que $h_{uu} = \sum_{v \in N(u)} (1 + h_{vu})/\deg(u)$ où $N(u)$ est l'ensemble des voisins du sommets u.
   - **iii.** En déduire que pour toute arête $uv \in E$, $h_{uv} < 2m$.
3. On fixe un sommet u et on note c_u l'espérance du nombre d'étapes nécessaire à une marche aléatoire partant de u pour visiter tous les sommets du graphe.
   - **i.** Donner une condition nécessaire et suffisante pour que $c_u < +\infty$.
   - **ii.** Montrer que si $c_u < +\infty$, il existe un chemin ayant au plus $2(n-1)$ arêtes visitant tous les sommets du graphe. *Remarque : le chemin peut visiter plusieurs fois un même sommet.*
   - **iii.** En déduire que si $c_u < +\infty$, $c_u < 4m(n-1)$.
   - **iv.** Montrer que s'il existe un chemin entre s et t, une marche aléatoire partant de s passe par t au cours des $4n^3$ premières étapes avec probabilité $\geq 1/2$.
4. En déduire un algorithme Monte Carlo pour décider s'il existe un chemin entre s et t ayant un probabilité de succès $\geq 1 - 2^{-n}$, et montrer que l'algorithme n'utilise qu'une mémoire logarithmique.

1. Remarque : dans la définition de h_{uu}, on impose à la marche aléatoire d'effectuer au moins un pas.