# TD 3 : QuickSelect et RandMinCut

**Exercice 1.** QuickSelect *particulier*

**1.** On s'intéresse à l'espérance $E_n$ du nombre de comparaisons effectuées par QuickSelect$(T, 1)$ où $T$ est un tableau de $n$ éléments.

**i.** Montrer que $E_n = (n-1) + \frac{1}{n}\sum_{m=1}^{n-1} E_m$.
**ii.** En déduire que $E_n \leq 2n - 2$.

**2.** De même, étudier l'espérance du nombre de comparaisons effectuées par QuickSelect$(T, 2)$.
**3.** Donner des algorithmes déterministes (simples !) pour ces deux cas, et comparer les complexités obtenues.

**Exercice 2.** RandMinCut *amélioré*

Le but de cet exercice est d'étudier une variante de l'algorithme RandMinCut qui a une bien meilleure complexité. L'idée est que les problèmes arrivent plutôt quand il reste peu de sommets : le risque de contracter deux sommets *par erreur* devient plus grand.

**1.** On fixe une coupe minimum d'un graphe $G$ à $n$ sommets, et on contracte aléatoirement des arêtes de $G$ jusqu'à ce que $G$ ait $n_0$ sommets. Quelle est la probabilité qu'une arête de la coupe minimale ait été contractée ?
**2.** Montrer que si $n_0 \geq 1 + n/\sqrt{2}$, la probabilité qu'au moins une arête de la coupe minimum ait été contractée est $\leq 1/2$.

On note Contraction$(G)$ l'algorithme qui contracte aléatoirement des arêtes de $G$ tant que $G$ a plus de $(1 + n/\sqrt{2})$ sommets. L'algorithme BetterMinCut est le suivant.

BetterMinCut$(G)$ :
1. Si $G$ a moins de 8 sommets, renvoyer une coupe minimale ;
2. $H \leftarrow$ Contraction$(G)$ ;
2. $C_1 \leftarrow$ BetterMinCut$(H)$ ;
3. $C_2 \leftarrow$ BetterMinCut$(H)$ ;
4. Renvoyer $\min(C_1, C_2)$.

On suppose disposer d'un algorithme calculant une coupe minimale exacte pour un graphe de moins de 8 sommets (par exemple par recherche exhaustive).

**3.** Montrer que la complexité de BetterMinCut est $O(n^2 \log n)$.
**4.** En notant $p_n$ la probabilité que BetterMinCut renvoie une coupe minimale quand le graphe en entrée a $n$ sommets, borner $p_n$ en fonction de $p_{n/\sqrt{2}+1}$.
**5.** *On admet que* $p_n = \Omega(1/\log n)$. Montrer qu'avec $O(\log^2 n)$ répétitions de l'algorithme, sa probabilité de succès est au moins $1 - 1/n^c$ pour une certaine constante $c$. Quelle est la complexité globale obtenue ?