Conception d'algorithmes et applications (LI325)

Cours 9: Introduction aux algorithmes probabilistes

Ana Bušić

http://www.di.ens.fr/~busic/LI325.html

ana.busic@inria.fr

Motivation

Tri rapide (aussi appelé "tri de Hoare", QuickSort) :

QuickSort

Entrées : Un tableau T de n clés.
Sorties : Le tableau T trié dans l'odre croissant.

Si $n \leq 1$, **Retourner** T

Choisir un élément x de T (pivot).

En comparant chaque autre élément de T à x, former les sous-tableaux $T_{\leq}$ (éléments $\leq x$) et $T_{>}$ (éléments $> x$).

Trier $T_{\leq}$ et $T_{>}$ en utilisant QuickSort.

Retourner $T_{\leq}$, x, $T_{>}$.

Propriétés :

- Opère uniquement sur la séquence à trier (tri sur place).
- Pire cas : complexité quadratique.
  Ex : si pivot est toujours le premier élément et le tableau est trié.

L'algorithme RandQuickSort

Une “solution” :

RandQuickSort

Entrées : Un tableau T de n clés.
Sorties : Le tableau T trié dans l'odre croissant.

Si $n \leq 1$, **Retourner** T

Choisir un élément x de T (pivot) **au hasard**.

En comparant chaque autre élément de T à x, former les sous-tableaux $T_{\leq}$ (éléments $\leq x$) et $T_{>}$ (éléments $> x$).

Trier $T_{\leq}$ et $T_{>}$ en utilisant RandQuickSort.

Retourner $T_{\leq}$, x, $T_{>}$.

Rappels de probabilités

(Cas des espaces finis ou dénombrables.)

- Un espace de probabilités est un couple (Ω, P) où Ω est un ensemble et $P : \Omega \to \mathbb{R}^+$ une fonction telle que :

$$\sum_{\omega \in \Omega} P(\omega) = 1.$$

  P est appelée une fonction de probabilité.

- Interpretation : Ω est l'ensemble des résultats élémentaires d'une experience non déterministe, chaque élément $\omega \in \Omega$ représente un résultat "élémentaire" possible qui a une "chance" (probabilité) $P(\omega)$ de se produire.

- Exemple d'une experience : jeter un dé (non-pipé)
  $\Omega = \{1, 2, 3, 4, 5, 6\}$.
  $P(1) = P(2) = \ldots = P(6) = 1/6$.

Événements

- Un événement est une partie A de Ω. Sa probabilité est :

$$P(A) = \sum_{\omega \in A} P(\omega).$$

- Interpretation : un événement est une réunion de tous les résultats élémentaires correspondant à une réponse positive à une question portant sur l'expérience et sa probabilité est la somme des probabilités des événements élémentaires qui le composent.

- Exemple (jet d'un dé).
  L'événement "le résultat est un nombre pair" : $A = \{2, 4, 6\}$.
  $P(A) = 1/2$.

Variable aléatoire, espérance, variance

- Une variable aléatoire $X : \Omega \to V$ est une fonction à valeurs dans un ensemble V (fini ou dénombrable).

  Nous n'allons considérer ici que les v.a. à valeurs réelles ($V \subset \mathbb{R}$).
- Notation : $P(X = a) = P(\{\omega \in \Omega : X(\omega) = a\})$.
- L'espérance d'une v.a. :

$$\begin{aligned}\mathbb{E}(X) &= \sum_{\omega \in \Omega} X(\omega) P(\omega) \\ &= \sum_{a \in V} a P(X = a).\end{aligned}$$

  Note : Toute v.a. réelle n'a pas forcement une espérance (si l'espace Ω est infini, la série peut diverger).
- Exemple (jet d'un dé). Variable aléatoire $X(\omega) = \omega$.
  $\mathbb{E}(X) = 1 * 1/6 + 2 * 1/6 + \ldots + 6 * 1/6 = 3.5$

Variable aléatoire, espérance, variance

- Une variable aléatoire $X : \Omega \to V$ est une fonction à valeurs dans un ensemble V (fini ou dénombrable).

  Nous n'allons considérer ici que les v.a. à valeurs réelles ($V \subset \mathbb{R}$).
- Notation : $P(X = a) = P(\{\omega \in \Omega : X(\omega) = a\})$.
- L'espérance d'une v.a. :

$$\mathbb{E}(X) = \sum_{\omega \in \Omega} X(\omega)P(\omega) = \sum_{a \in V} aP(X = a).$$

  Note : Toute v.a. réelle n'a pas forcement une espérance (si l'espace Ω est infini, la série peut diverger).
- Exemple (jet d'un dé). Variable aléatoire $X(\omega) = \omega$. $\mathbb{E}(X) = 1 * 1/6 + 2 * 1/6 + \ldots + 6 * 1/6 = 3.5$

  Supposons que : $P(1) = 1/12$, $P(2) = P(3) = P(4) = P(5) = 1/6$, $P(6) = 1/4$. Alors $\mathbb{E}(X) = 47/12 = 3.917$

- Propriété de linéarité de l'espérance :
  Soient X et Y deux v.a. sur le même espace de probabilités et $a, b \in \mathbb{R}$. Alors la variable aléatoire $Z = aX + bY$ vérifie :

$$\mathbb{E}(Z) = a\mathbb{E}(X) + b\mathbb{E}(Y).$$

- La variance d'une v.a. réelle est :

$$Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2).$$

  (un indicateur de dispersion)

  Note : Une v.a. qui a une espérance peut ne pas avoir de variance.

- Propriétés :
  $Var(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2$.
  $Var(aX + b) = a^2 Var(X)$.

Indépendance

- Intuitivement : deux événements (ou deux v.a.) sont indépendants si l'information obtenue sur l'un ne modifie pas la connaissance qu'on a de l'autre.
- Def. Deux événements A et B sont indépendants si :

$$P(A \cap B) = P(A)P(B).$$

Indépendance

- Intuitivement : deux événements (ou deux v.a.) sont indépendants si l'information obtenue sur l'un ne modifie pas la connaissance qu'on a de l'autre.
- Def. Deux événements A et B sont indépendants si :

$$P(A \cap B) = P(A)P(B).$$

- Exemple (jet d'un dé non-pipé).
  Les événements $A = \{2, 4, 6\}$ et $B = \{1, 2, 3\}$ ne sont pas indépendants :
  $P(A) = 1/2, P(B) = 1/2$, mais $P(A \cap B) = 1/6$ et $P(A)P(B) = 1/4$.

Indépendance

- Intuitivement : deux événements (ou deux v.a.) sont indépendants si l'information obtenue sur l'un ne modifie pas la connaissance qu'on a de l'autre.
- Def. Deux événements A et B sont indépendants si :

$$P(A \cap B) = P(A)P(B).$$

- Exemple (jet d'un dé non-pipé).
  Les événements $A = \{2, 4, 6\}$ et $B = \{1, 2, 3\}$ ne sont pas indépendants :
  $P(A) = 1/2, P(B) = 1/2$, mais $P(A \cap B) = 1/6$ et $P(A)P(B) = 1/4$.

  Les événements $A = \{2, 4, 6\}$ et $B = \{1, 6\}$ sont indépendants :
  $P(A) = 1/2, P(B) = 1/3$, $P(A \cap B) = P(A)P(B) = 1/6$.

Probabilités conditionnelles

Soient A et B deux événements avec $P(B) > 0$. La probabilité de A sachant B est définie par

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Probabilités conditionnelles

Soient A et B deux événements avec $P(B) > 0$. La probabilité de A sachant B est définie par

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Propriétés :

- Un événement A est indépendant de B ssi $P(A|B) = P(A)$.

Probabilités conditionnelles

Soient A et B deux événements avec $P(B) > 0$. La probabilité de A sachant B est définie par

$$P(A|B) = \frac{P(A \cap B)}{P(B)}.$$

Propriétés :

- Un événement A est indépendant de B ssi $P(A|B) = P(A)$.
- Pour tout B avec $P(B) > 0$, on peut définir une nouvelle fonction $P(\cdot|B)$ par $P(\omega|B) = P(\{\omega\}|B)$. Cette fonction est aussi une fonction de probabilité.

RandQuickSort

RandQuickSort

Entrées : Un tableau T de n clés.
Sorties : Le tableau T trié dans l'odre croissant.

1 Si $n \leq 1$, **Retourner** T
2 Choisir un élément x de T (pivot) **selon la loi uniforme** (chaque élément de T ayant la même probabilité d'être sélectionné).
3 En comparant chaque autre élément de T à x, former les sous-tableaux $T_{\leq}$ (éléments $\leq x$) et $T_{>}$ (éléments $> x$).
4 Trier $T_{\leq}$ et $T_{>}$ en utilisant RandQuickSort.
5 **Retourner** $T_{<}$, x, $T_{>}$.

Complexité de RandQuickSort

- Est aussi une variable aléatoire : dépend des choix aléatoires faits, pas nécessairement identiques d'une execution à l'autre.
- Dans le cas le pire, toujours quadratique, mais peu probable que RandQuickSort sélectionne à chaque étape un pivot minimal ou maximal).

Complexité de RandQuickSort

- Est aussi une variable aléatoire : dépend des choix aléatoires faits, pas nécessairement identiques d'une execution à l'autre.
- Dans le cas le pire, toujours quadratique, mais peu probable que RandQuickSort sélectionne à chaque étape un pivot minimal ou maximal).
- Mais *quel que soit le tableau initial* T, le nombre moyen de comparaisons effectuées par RandQuickSort est au plus de $2nH_n$, où $H_n = \sum_{k=1}^{n} 1/k = \ln(n) + O(1)$.

Complexité de RandQuickSort

- Est aussi une variable aléatoire : dépend des choix aléatoires faits, pas nécessairement identiques d'une execution à l'autre.
- Dans le cas le pire, toujours quadratique, mais peu probable que RandQuickSort sélectionne à chaque étape un pivot minimal ou maximal).
- Mais *quel que soit le tableau initial* T, le nombre moyen de comparaisons effectuées par RandQuickSort est au plus de $2nH_n$, où $H_n = \sum_{k=1}^{n} 1/k = \ln(n) + O(1)$.
- Il est aussi possible, mais sensiblement plus difficile de démontrer qu'il est très peu probable (en un sens quantifiable) que la complexité soit très éloignée de son espérance.

Théorème. Quel que soit le tableau initial T de n éléments, le nombre moyen de comparaisons effectuées par RandQuickSort est au plus de $2nH_n$, où $H_n = \sum_{k=1}^{n} 1/k = \ln(n) + O(1)$.

Preuve.

- Notons les éléments de T après le tri par : $s_1 \leq s_2 \ldots s_n$.
- Soit $X_{i,j}$ la variable aléatoire qui vaut 1 si s_i et s_j sont comparés au cours de l'exécution de RandQuickSort, et 0 sinon.
  Notons par $p_{i,j} = P(X_{i,j} = 1)$.
- Le nombre de comparaisons effectuées par RandQuickSort : $X = \sum_{1 \leq i < j \leq n} X_{i,j}$.
- $\mathbb{E}(X_{i,j}) = 0 * (1 - p_{i,j}) + p_{i,j} = p_{i,j}$.
- Par linéarité de l'espérance :

$$\mathbb{E}(X) = \sum_{1 \leq i < j \leq n} \mathbb{E} X_{i,j} = \sum_{1 \leq i < j \leq n} p_{i,j}.$$

$\square$

Pour évaluer les $p_{i,j}$:

- s_i et s_j seront comparés ssi l'un d'entre eux est pris comme pivot avant qu'aucun des éléments $s_{i+1}, \ldots, s_{j-1}$ le soit (sinon, s_i et s_j se retrouvent dans les 2 sous-tableaux différents).

  Donc $p_{i,j} = \frac{2}{j-i+1}$.
- Nous avons :

$$\mathbb{E}(X) = \sum_{1 \leq i < j \leq n} p_{i,j} = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1} = 2 \sum_{i=1}^{n} \sum_{k=2}^{n-i+1} \frac{1}{k}$$
$$< 2 \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{1}{k} = 2n \sum_{k=1}^{n} \frac{1}{k} = 2nH_n.$$

Coupe minimale

- Soit $G = (V, E)$ un graphe connexe, non orienté, à arêtes multiples mais sans boucles.
- Une coupe de G est l'ensemble $\mathcal{C}$ d'arêtes de G qui ont une extrémité dans chacun des deux ensembles X et Y. Ces deux ensembles forment une partition de V.
- Une coupe minimale est une coupe de taille (nombre d'arêtes) minimale.
- Problème : Pour un graphe G donné, trouver une coupe minimale.

Coupe minimale

- Soit $G = (V, E)$ un graphe connexe, non orienté, à arêtes multiples mais sans boucles.
- Une coupe de G est l'ensemble $\mathcal{C}$ d'arêtes de G qui ont une extrêmité dans chacun des deux ensembles X et Y. Ces deux ensembles forment une partition de V.
- Une coupe minimale est une coupe de taille (nombre d'arêtes) minimale.
- Problème : Pour un graphe G donné, trouver une coupe minimale.
- Variantes déterministes :
  l'algorithme Ford et Fulkerson (à base de calculs de flots) ;
  l'algorithme Edmonds and Karp (complexité en $O(|V| \cdot |E|^2)$).

Coupe minimale

- Soit $G = (V, E)$ un graphe connexe, non orienté, à arêtes multiples mais sans boucles.
- Une coupe de G est l'ensemble $\mathcal{C}$ d'arêtes de G qui ont une extrêmité dans chacun des deux ensembles X et Y. Ces deux ensembles forment une partition de V.
- Une coupe minimale est une coupe de taille (nombre d'arêtes) minimale.
- Problème : Pour un graphe G donné, trouver une coupe minimale.
- Variantes déterministes :
  l'algorithme Ford et Fulkerson (à base de calculs de flots) ;
  l'algorithme Edmonds and Karp (complexité en $O(|V| \cdot |E|^2)$).
- Ici : un algorithme probabiliste très simple (mais pas sans défauts).

Contraction

- Soit $e \in E$ une arête de G.
- La contraction de e consiste à fusionner les deux sommets extrêmités de e et à retirer l'arête e et également toutes les boucles formées lors de la fusion.
- Le nouveau graphe est noté par $G_{/e}$.

Contraction

- Soit $e \in E$ une arête de G.
- La contraction de e consiste à fusionner les deux sommets extrêmités de e et à retirer l'arête e et également toutes les boucles formées lors de la fusion.
- Le nouveau graphe est noté par $G_{/e}$.
- Une autre vision de $G_{/e}$: le graphe dont les sommets forment une partition de l'ensemble des sommets de départ ; la contraction d'une arête correspond à remplacer 2 ensembles de la partition par leur union.

RandMinCut

RandMinCut

Entrées : Un graphe G à arêtes multiples sans boucles.
Sorties : Une coupe de G.

```
tant que |G| > 2 faire
    Choisir une arête selon la loi uniforme (chaque arête ayant la même probabilité d'être choisie);
    Remplacer G par G/e;
Retourner Toutes les arêtes de G.
```

RandMinCut

RandMinCut

Entrées : Un graphe G à arêtes multiples sans boucles.
Sorties : Une coupe de G.

tant que $|G| > 2$ **faire**
  Choisir une arête selon la loi uniforme (chaque arête ayant la même probabilité d'être choisie);
  Remplacer G par $G_{/e}$;

Retourner Toutes les arêtes de G.

- *RandMinCut*(G) retourne toujours une coupe de G.
- Cette coupe n'est pas toujours une coupe minimale.

RandMinCut

RandMinCut

Entrées : Un graphe G à arêtes multiples sans boucles.
Sorties : Une coupe de G.

```
tant que |G| > 2 faire
    Choisir une arête selon la loi uniforme (chaque arête ayant la même
    probabilité d'être choisie);
    Remplacer G par G/e;
Retourner Toutes les arêtes de G.
```

- $RandMinCut(G)$ retourne toujours une coupe de G.
- Cette coupe n'est pas toujours une coupe minimale.
- On va montrer que : Pour tout graphe à n sommets, la probabilité que RandMinCut retourne une coupe minimale de G est supérieure ou égale à $\frac{2}{n^2}$.

Lemme 1. Soit G un graphe à n sommets et $\mathcal{C}$ une coupe minimale de G. Si $\mathcal{C}$ est de cardinalité k, alors G a au moins $\frac{kn}{2}$ arêtes.

Lemme 1. Soit G un graphe à n sommets et $\mathcal{C}$ une coupe minimale de G. Si $\mathcal{C}$ est de cardinalité k, alors G a au moins $\frac{kn}{2}$ arêtes.

Preuve.

- Soit $u \in V$ un sommet arbitraire. Alors $(\{u\}, V - \{u\})$ est une coupe de cardinalité au moins k, donc u est de degré au moins k.
- Vrai pour chaque sommet, il y a n sommets au total, et chaque arête relie 2 sommets.
- Donc, G a au moins $\frac{kn}{2}$ arêtes.

$\square$

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

$\Leftarrow$ Supposons que aucune arête de $\mathcal{C}$ n'a été contractée. Alors :

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

$\Leftarrow$ Supposons que aucune arête de $\mathcal{C}$ n'a été contractée. Alors :

- Toutes arêtes de $\mathcal{C}$ sont retournées.

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

$\Leftarrow$ Supposons que aucune arête de $\mathcal{C}$ n'a été contractée. Alors :

- Toutes arêtes de $\mathcal{C}$ sont retournées.

  Il n'y a que 2 raisons pour qu'une arête ne soit pas retournée : être contractée ou être transformée en boucle.

  Si une arête de $\mathcal{C}$ a été transformée en boucle, cela implique que ses 2 extrêmités ont été fusionées, ce qui n'est possible que si une autre arête de $\mathcal{C}$ a été contractée.

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

$\Leftarrow$ Supposons que aucune arête de $\mathcal{C}$ n'a été contractée. Alors :

- Toutes arêtes de $\mathcal{C}$ sont retournées.

  Il n'y a que 2 raisons pour qu'une arête ne soit pas retournée : être contractée ou être transformée en boucle.

  Si une arête de $\mathcal{C}$ a été transformée en boucle, cela implique que ses 2 extrêmités ont été fusionées, ce qui n'est possible que si une autre arête de $\mathcal{C}$ a été contractée.

- Aucune arête n'appartenant pas à $\mathcal{C}$ n'est retournée.

Lemme 2. Une coupe $\mathcal{C}$ sera retournée par RandMinCut si et seulement si aucune des arêtes de $\mathcal{C}$ n'est sélectionée pour être contractée.

Preuve.

$\Rightarrow$ est évident (à chaque étape l'arête contractée est retirée du graphe).

$\Leftarrow$ Supposons que aucune arête de $\mathcal{C}$ n'a été contractée. Alors :

- Toutes arêtes de $\mathcal{C}$ sont retournées.

  Il n'y a que 2 raisons pour qu'une arête ne soit pas retournée : être contractée ou être transformée en boucle.

  Si une arête de $\mathcal{C}$ a été transformée en boucle, cela implique que ses 2 extrêmités ont été fusionées, ce qui n'est possible que si une autre arête de $\mathcal{C}$ a été contractée.

- Aucune arête n'appartenant pas à $\mathcal{C}$ n'est retournée.

  Vrai car l'algorithme ne se termine que lorsqu'il ne reste que 2 sommets (correspondent aux 2 parties séparées par $\mathcal{C}$) : toute arête qui n'appartient pas à $\mathcal{C}$ joint 2 sommets de la même partie et a été éliminée soit par contraction soit comme boucle.

$\square$

Proposition.

Pour tout graphe à n sommets, la probabilité que RandMinCut retourne une coupe minimale de G est supérieure ou égale à $\frac{2}{n^2}$.

Proposition.
Pour tout graphe à n sommets, la probabilité que RandMinCut retourne une coupe minimale de G est supérieure ou égale à $\frac{2}{n^2}$.

Preuve. Soit $\mathcal{C}$ une coupe minimale de G et notons par k la cardinalité de $\mathcal{C}$.

Nous allons calculer la probabilité qu'aucune arête de $\mathcal{C}$ ne soit contractée au cours de $n-2$ étapes de l'algorithme.

- Soit A_i l'événement "l'arête contractée à l'étape i n'est pas une arête de $\mathcal{C}$.
- Nous cherchons à minorer la probabilité de l'événement $A = \cap_{i=1}^{n-2} A_i$.
- Lemme 1. implique que G a au moins $kn/2$ arêtes, dont k dans $\mathcal{C}$. Donc,
$$P(A_1) \geq 1 - \frac{2k}{kn} = 1 - \frac{2}{n}.$$

- En supposant que A_1 c'est produit, il reste au moins $k(n-1)/2$ arêtes :

$$P(A_2|A_1) \geq 1 - \frac{2}{n-1}.$$

- En supposant que A_1 c'est produit, il reste au moins $k(n-1)/2$ arêtes :
$$P(A_2|A_1) \geq 1 - \frac{2}{n-1}.$$
- La probabilité que A_i se produise sachant que $A_1, \ldots, A_{i-1}$ se sont produits est la probabilité d'un choix d'une arête parmi au moins $k(n-i+1)/2$ (graphe à $n-i+1$ sommets, de taille de coupe minimale k) en évitant k :
$$P(A_i| \cap_{j=1}^{i-1} A_j) \geq 1 - \frac{2}{n-i+1}.$$

- En supposant que A_1 c'est produit, il reste au moins $k(n-1)/2$ arêtes :

$$P(A_2|A_1) \geq 1 - \frac{2}{n-1}.$$

- La probabilité que A_i se produise sachant que $A_1, \ldots, A_{i-1}$ se sont produits est la probabilité d'un choix d'une arête parmi au moins $k(n-i+1)/2$ (graphe à $n-i+1$ sommets, de taille de coupe minimale k) en évitant k :

$$P(A_i| \cap_{j=1}^{i-1} A_j) \geq 1 - \frac{2}{n-i+1}.$$

- Donc,

$$\begin{aligned} P(A) = P(\cap_{i=1}^{n-2} A_i) &= P(A_1) \prod_{i=2}^{n-2} P(A_i| \cap_{j=1}^{i-1} A_j) \\ &\geq \prod_{i=1}^{n-2} (1 - \frac{2}{n-i+1}) = \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} \\ &= \frac{2}{n(n-1)} \end{aligned}$$

Proposition. Pour tout graphe à n sommets et tout réel $0 < \epsilon < 1$, la probabilité que $n^2 \ln(1/\epsilon)/2$ répétitions indépendantes de RandMinCut donnent une coupe minimale est $\geq 1 - \epsilon$.

Proposition. Pour tout graphe à n sommets et tout réel $0 < \epsilon < 1$, la probabilité que $n^2 \ln(1/\epsilon)/2$ répétitions indépendantes de RandMinCut donnent une coupe minimale est $\geq 1 - \epsilon$.

Preuve.

- La probabilité qu'une exécution de l'algorithme ne donne pas une coupe minimale est $\leq 1 - 2/n^2$.

Proposition. Pour tout graphe à n sommets et tout réel $0 < \epsilon < 1$, la probabilité que $n^2 \ln(1/\epsilon)/2$ répétitions indépendantes de RandMinCut donnent une coupe minimale est $\geq 1 - \epsilon$.

Preuve.

- La probabilité qu'une exécution de l'algorithme ne donne pas une coupe minimale est $\leq 1 - 2/n^2$.
- La probabilité que k répétitions indépendantes échouent toutes est $\leq (1 - 2/n^2)^k$.

Proposition. Pour tout graphe à n sommets et tout réel $0 < \epsilon < 1$, la probabilité que $n^2 \ln(1/\epsilon)/2$ répétitions indépendantes de RandMinCut donnent une coupe minimale est $\geq 1 - \epsilon$.

Preuve.

- La probabilité qu'une exécution de l'algorithme ne donne pas une coupe minimale est $\leq 1 - 2/n^2$.
- La probabilité que k répétitions indépendantes échouent toutes est $\leq (1 - 2/n^2)^k$.
- Pour tout réel $x > 1$, on a $(1 - 1/x)^x < 1/e$. Pour $x = n^2/2$, cela donne : $(1 - 2/n^2)^{n^2/2} < 1/e$ et donc :

$$(1 - 2/n^2)^{n^2 \ln(1/\epsilon)/2} < e^{-\ln(1/\epsilon)} = \epsilon.$$

$\square$

Algorithmes Probabilistes

Algorithme probabiliste est un algorithme dans lequel on utilise l'instruction de la forme

"tirer un entier selon la loi uniforme entre 1 et n".

Algorithmes Probabilistes

Algorithme probabiliste est un algorithme dans lequel on utilise l'instruction de la forme

"tirer un entier selon la loi uniforme entre 1 et n".

Deux usages principaux du aléatoire :

- RandQuickSort : pour assurer que sur toute instance la complexité moyenne est égale à la complexité moyenne (sous l'hypothèse uniforme) d'un algorithme déterministe correspondant. Technique de randomisation.

Algorithmes Probabilistes

Algorithme probabiliste est un algorithme dans lequel on utilise l'instruction de la forme

"tirer un entier selon la loi uniforme entre 1 et n".

Deux usages principaux du aléatoire :

- RandQuickSort : pour assurer que sur toute instance la complexité moyenne est égale à la complexité moyenne (sous l'hypothèse uniforme) d'un algorithme déterministe correspondant. Technique de randomisation.
- RandMinCut : algorithme plus simple que l'algorithme déterministe pour le même problème.

Algorithmes Probabilistes

Algorithme probabiliste est un algorithme dans lequel on utilise l'instruction de la forme

"tirer un entier selon la loi uniforme entre 1 et n".

Deux usages principaux du aléatoire :

- RandQuickSort : pour assurer que sur toute instance la complexité moyenne est égale à la complexité moyenne (sous l'hypothèse uniforme) d'un algorithme déterministe correspondant. Technique de randomisation.
- RandMinCut : algorithme plus simple que l'algorithme déterministe pour le même problème.

L'analyse faite est valable sur toutes les instances du problème (pas d'hypothèse probabiliste sur les instances).

Las Vegas vs. Monte Carlo

Deux types d'algorithmes probabilistes :

- Las Vegas : donne un résultat toujours correct.

  Exemple : RandQuickSort.

  La complexité est une variable aléatoire.

- Monte Carlo : donne un résultat qui peut être incorrect.

  Exemple : RandMinCut.

  Il est impératif de majorer la probabilité d'erreur.

Las Vegas vs. Monte Carlo

Deux types d'algorithmes probabilistes :

- Las Vegas : donne un résultat toujours correct.

  Exemple : RandQuickSort.

  La complexité est une variable aléatoire.

- Monte Carlo : donne un résultat qui peut être incorrect.

  Exemple : RandMinCut.

  Il est impératif de majorer la probabilité d'erreur.

  Un algorithme Monte Carlo d'erreur λ : si pour toute entrée la probabilité qu'il retourne un résultat faux est au plus λ.

Algorithmes Monte Carlo

Dans le cas des problèmes de décision (la réponse OUI ou NON) deux sous-classes :

- Algorithme Monte Carlo à erreur unilatérale : pour toute instance pour laquelle la réponse est OUI, l'algorithme répond OUI avec probabilité 1.
- Algorithme Monte Carlo à erreur bilatérale : pour au moins une instance positive et au moins une instance négative la probabilité qu'il réponde OUI est strictement comprise entre 0 et 1.

Algorithmes Monte Carlo

Dans le cas des problèmes de décision (la réponse OUI ou NON) deux sous-classes :

- Algorithme Monte Carlo à erreur unilatérale : pour toute instance pour laquelle la réponse est OUI, l'algorithme répond OUI avec probabilité 1.
- Algorithme Monte Carlo à erreur bilatérale : pour au moins une instance positive et au moins une instance négative la probabilité qu'il réponde OUI est strictement comprise entre 0 et 1.

RandMinCut (avec répetition) transformé en algorithme de décision (en ajoutant à l'entrée un entier k et en demandant de décider si toute coupe du graphe a un poids strictement supérieur à k) est un algorithme à l'erreur unilaterale.

Algorithmes Monte Carlo

Dans le cas des problèmes de décision (la réponse OUI ou NON) deux sous-classes :

- Algorithme Monte Carlo à erreur unilatérale : pour toute instance pour laquelle la réponse est OUI, l'algorithme répond OUI avec probabilité 1.
- Algorithme Monte Carlo à erreur bilatérale : pour au moins une instance positive et au moins une instance négative la probabilité qu'il réponde OUI est strictement comprise entre 0 et 1.

RandMinCut (avec répetition) transformé en algorithme de décision (en ajoutant à l'entrée un entier k et en demandant de décider si toute coupe du graphe a un poids strictement supérieur à k) est un algorithme à l'erreur unilaterale.

La probabilité d'erreur peut être rendue aussi petite que l'on veut en changeant le nombre de répetitions.