Classes de complexité probabilistes

Machine de Turing probabiliste

- **Définition:** Une machine de Turing probabiliste correspond à une machine de Turing qui peut à un moment donné tirer à pile ou face sur un ruban $i > 1$; à ce moment là, elle écrit O (resp. 1) avec probabilité $1/2$ sur la case en face de la tête de lecture du ruban i.
- $Pr(M\ accepte\ x) = \dfrac{\text{nombre de branches acceptantes}}{\text{nombre total de branches}}$.
- $Pr(erreur\ sur\ x)$ est donnée par
  1. $Pr(M\ n'accepte\ pas\ x)$ pour $x \in L$
  2. $Pr(M\ accepte\ x)$ pour $x \notin L$.

Classe Probabilistic P

- **Définition:** Un langage L est dans PP s'il existe une machine de Turing probabiliste qui fonctionne en temps polynômial telle que
  1. $x \in L$, $Pr(M\ accepte\ x) > 1/2$
  2. $x \notin L$, $Pr(M\ accepte\ x) \leq 1/2$

Classe Probabilistic P et classes stand

- **Théorème:** $NP \subset PP \subset PSPACE$.

  Démonstration $NP \subset PP$**: Soit** M **la machine ND qui accepte** $L \in NP$**. Soit** M' **la machine probabiliste obtenue à partir de la machine non-déterministe qui sur l'entrée** w**, tire à pile ou face, si elle tire pile accepte, si elle tire face alors simule** M **sur** w**. On a** $Pr(M' \; accepte \; x) > 1/2$ **pour** $x \in L$**,** $Pr(M' \; accepte \; x) = 1/2$ **si** $x \in L$**.**

  Démonstration $PP \subset PSPACE$**: Soit** M **la machine probabiliste qui accepte** $L \in PP$**. Il suffit de considérer la machine** M' **qui simule** M **sur tous les tirages possibles des choix aléatoires, compte le nombre de branches acceptantes de la machine probabiliste, et qui accepte ssi ce nombre est au moins la moitié du nombre de branches totales.**

Classe Bounded Probabilistic P

- **Définition:** Un langage L est dans BPP s'il existe une machine de Turing probabiliste qui fonctionne en temps polynômial telle que pour tout x, $Pr(erreur) < 1/4$.
- Autrement dit,
  1. $x \in L$, $Pr(erreur) < 1/4$
  2. $x \notin L$, $Pr(erreur) < 1/4$

Classe Zero Error Probabilistic P

- **Définition:** Un langage L est dans RP s'il existe une machine de Turing probabiliste qui fonctionne en temps polynômial telle que
  1. $x \in L$, $Pr(erreur) < 1/2$
  2. $x \notin L$, $Pr(erreur) = 0$
- **Définition:** Un langage L est dans coRP s'il existe une machine de Turing probabiliste qui fonctionne en temps polynômial telle que
  1. $x \in L$, $Pr(erreur) = 0$
  2. $x \notin L$, $Pr(erreur) < 1/2$
- **Définition:** $ZPP = RP \cap coRP$.

Amplification de l'erreur

- **Théorème:** Dans les définitions de RP, coRP, ZPP on peut remplacer $1/2$ par n'importe quelle constante $0 < \epsilon \leq 1/2$.

  Par exemple pour RP: **Théorème:** un langage L est dans RP ssi il existe une machine de Turing probabiliste qui fonctionne en temps polynômial telle que

  1. $x \in L$, $Pr(erreur) < \epsilon$
  2. $x \notin L$, $Pr(erreur) = 0$

  Démonstration. Soit L dans RP reconnu par M. Soit k un entier. La machine $M*$ qui sur une entrée w simule k fois M et accepte ssi au moins un calcul de M accepte. $M*$ vérifie $Pr(M* \ refuse \ x) \leq 1/2^k$ pour $x \in L$, et $Pr(M* \ accepte \ x) = 0$ pour $x \notin L$. Choisir k tel que $1/2^k < \epsilon$.

Amplification de l'erreur

- **Théorème:** Dans la définition de BPP on peut remplacer $1/4$ par n'importe quelle constante $0 < \epsilon < 1/2$.

  Démonstration. Soit L dans BPP reconnu par M. Soit k un entier. On considère la machine $M*$ qui sur une entrée w de longueur n simule de façon indépendante k fois M et décide à la majorité des réponses de chacune des simulations.

  Soit $\delta = 1 - \epsilon$. La probabilité que exactement j simulations donnent une bonne réponse est $C_k^j \delta^j \epsilon^{k-j}$. La probabilité que $M*$ se trompe est la probabilité qu'au plus $k/2$ simulations donnent une réponse correcte, soit $\sum_{j=0}^{k/2} C_k^j \delta^j \epsilon^{k-j}$. Puisque $\delta > \epsilon$, $\delta^j \epsilon^{k-j} \leq \delta^{k/2} \epsilon^{k/2}$, et donc $M*$ se trompe avec une probabilité $\leq (\sum_{j=0}^{k/2} C_k^j)(\epsilon\delta)^{k/2} \leq 2^k (\epsilon\delta)^{k/2} = (4\epsilon\delta)^{k/2}$. Il suffit de choisir k assez grand pour que $(4\epsilon\delta)^{k/2} < 1/2 - \epsilon$.

- **Remarque: la preuve montre que dans la définition de BPP on peut aussi remplacer $1/4$ par $(1/2)^{p(n)}$ pour tout polynôme p (n est la taille de l'entrée).**

- **Remarque: la même chose était vrai pour RP, coRP, et ZPP.**

Algorithme de Las Vegas

- **Définition:** Un algorithme de Las Vegas est un algorithme probabiliste qui donne toujours une réponse correcte en un temps qui en moyenne est polynômial (aucune hypothèse sur le temps au pire cas).
- Autrement dit: **Las Vegas**
  1. Zéro erreur
  2. Mais pas de temps garanti.

Algorithme de Monte Carlo

- **Définition:** Un algorithme de Monte Carlo est un algorithme probabiliste qui fonctionne toujours en temps polynômial, qui peut se tromper, mais dont l'erreur reste inférieure à $1/4$.
- Autrement dit: **Monte Carlo**
  1. Des erreurs sont possibles.
  2. Mais le temps est garanti.

Exemple d'algorithme de Las Vegas

- **Exemple:** Tri d'un ensemble S.
  Algorithme récursif: $Tri(S)$
  1. Choisir uniformément $y \in S$.
  2. Déterminer $S_1 = \{x \in S, x < y\}$ et $S_2 = \{x \in S, x \geq y\}$.
  3. Faire $Tri(S_1)$
  4. Faire $Tri(S_2)$
  5. Retourner le résultat de $Tri(S_1)$, suivi de y, suivi du résultat de $Tri(S_2)$.
- Cet algorithme est un algorithme de Las Vegas: il est toujours correct, de temps moyen $O(n \log n)$.
- **Pour cet algorithme, le temps au pire est $O(n^2)$.**

Exemple d'algorithme de Monte Carl

- **Exemple:** MinCut: trouver un ensemble minimal d'arètes d'un graphe telle que leur suppression crée plusieurs composantes connexes.
  Algorithme:
  1. Tant qu'il y a plus que deux sommets:
     (a) Choisir une arrête au hasard uniformément.
     (b) Fusionner les deux sommets.
  2. Retourner l'ensemble des arrêtes entre les deux sommets restants.
- Cet algorithme est un algorithme de Las Vegas: il fonctionne en temps $O(n)$, mais se trompe parfois.
- **Exercice: erreur $\leq 1 - 2/n^2$. En itérant $n^2/2$ fois cet algorithme, on peut garantir l'erreur $\leq 1/e$.**

BPP = Algorithmes de Monte Carlo

- **Théorème:** BPP correspond aux algorithmes de Monte Carlo.

  Démonstration. Par définition.

ZPP = Algorithmes de Las Vegas

- **Théorème:** ZPP correspond aux algorithmes de Las Vegas.

Démonstration. Sens $\Rightarrow$: soit $L \in ZPP$. Soient M et M' les machines qui attestent que L est respectivement dans RP et dans $co - RP$. Sur une entrée x, avec une probabilité $> 1/2$, M et M' donnent la même réponse. Lorsque cela ce produit, cette réponse est nécessairement correcte. Considérer la machine $M*$ qui simule de façon alternée M et M' sur son entrée, et recommence jusqu'à ce que les deux machines soient d'accord. $M*$ donne toujours une réponse correcte. Le temps n'est pas forcément polynômial, mais sa moyenne est plus faible que $(temps(M) + temps(M')(1 * 1/2 + 2 * 1/4 + 3 * 1/8 + \ldots)$ soit $0(temps(M) + temps(M'))$.

Démonstration. Sens $\Leftarrow$: Soit t le temps d'exécution de l'algorithme probabiliste qui reconnaît le langage L. Soit M la machine qui accepte un entrée x ssi l'algorithme randomizé accepte l'entrée x en un temps $\leq 3 * t$, rejette sinon. Pour $x \notin L$, M rejette toujours. Pour $x \in L$, par l'inégalité de Markov (lemme plus bas), avec une probabilité $\geq 2/3$ M a le temps de simuler complètement l'algorithme et donc ne se trompe pas. L est donc dans RP.

L est dans $coRP$ par l'argument symétrique.

Lemme [Inégalité de Markov]: X variable aléatoire ≥ 0, $Pr(X \geq cE(X)) \leq 1/c$ pour tout $c > 0$.

Démonstration du Lemme: Considère $f(x) = 1$ si $x \geq c * E[X]$, 0 sinon.

$Pr(X \geq cE(X)) = E[f(X)]$. Puisque $f(x) \leq x/(c * E[X])$ pour tout x,

$E[f(X)] \leq E[X/(c * E[X]) = 1/c$.

Classes probabilistes

- **Théorème:** PP clos par complément.

  Démonstration: exercice.

- **Théorème:** BPP et ZPP sont closes par complément.

  Démonstration. PP**: inverser les états d'acceptation et de rejet. ZPP: le complément de** RP **est clairement** $coRP$**, et réciproquement.**

- **Théorème:** $P \subset RP$, $P \subset coRP$.

  Démonstration: une MT est une machine probabiliste particulière.

- **Corollaire:** $P \subset ZPP$.

Classes probabilistes

- **Théorème:** $RP \subset BPP$, $coRP \subset BPP$, $BPP \subset PP$.

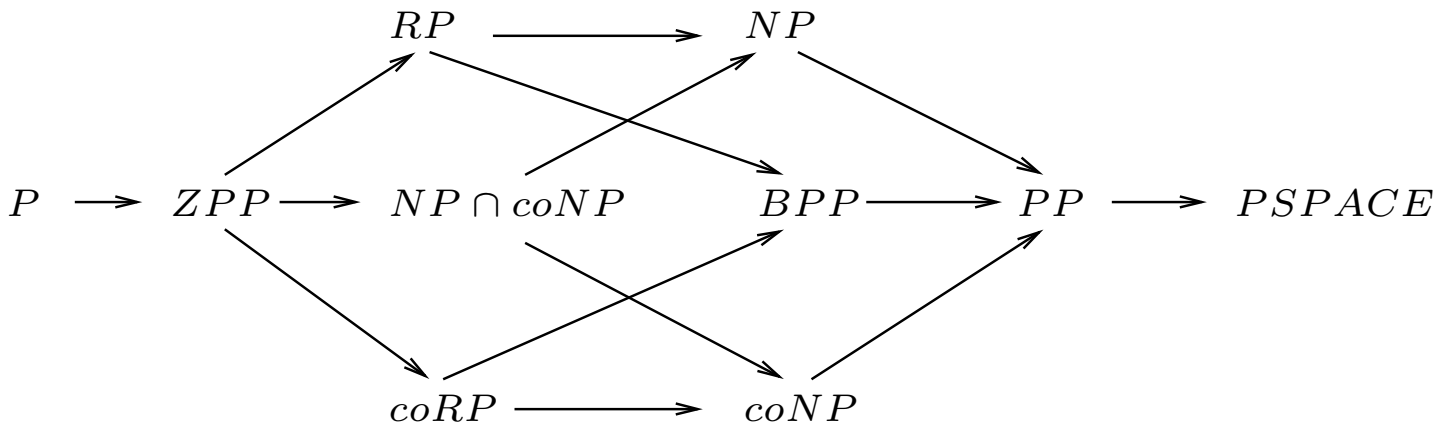
  Démonstration: évident.

- **Théorème:** $RP \subset NP$, $coRP \subset coNP$.

  Démonstration: considérer la MT probabiliste comme une machine non-déterministe.

- **Corollaire:** $ZPP \subset NP$. $ZPP \subset coNP$.

Résumé



Autres classes

- Classe RL: obtenue à partir de L, comme RP est obtenu à partir de P.
- Classe RNC^i: obtenue à partir de NC^i, , comme RP est obtenu à partir de P.
- Classe RNC: obtenue à partir de NC, , comme RP est obtenu à partir de P.

Exemple de problèmes: $UREACH$ es

- **Théorème:** Problème ***UREACH*:**
  - **Instance:** $G = (V, E)$ un graphe, $u, v \in V$ deux sommets
  - **Question:** Existe-t'il un chemin entre u et v?

  est dans RL.

Remarque: $UREACH \leq REACH$ **et donc le problème est dans** NL**.**
Démonstration. Algorithme:

1. **u:=s; i:=1**
2. **Tant que** $i < 2.n^3$
   (a) **Tirer une arrête** $(u, u') \in E$ **au hasard uniformément.**
   (b) $u := u'$**;** $i := i + 1$
   (c) **Si** $u = t$ **accepter**
3. **Sinon rejeter**

Cet algorithme fonctionne bien en espace $O(\log n)$**.**
On admettra (théorie des chaînes de Markov) que le temps moyen pour qu'une marche aléatoire explore tous les noeuds d'un graphe est $O(n^3)$**.**

Par l'inégalité de Markov quand l'algorithme rejette, il a plus d'une chance sur deux d'avoir visité tous les sommets, et donc de donner une bonne réponse.

Exemples de problèmes: $COUPLAG$

- **Théorème:** Problème ***COUPLAGEPARFAIT*** **:**
  - **Instance:** $G = (V, E)$ un graphe avec $2n$ sommets
  - **Question:** Il existe $S \subset E$, $|S| = n$, tq
    $e_1, e_2 \in S, e_1 \neq e_2 \Rightarrow e_1 \cap e_2 = \emptyset$

  est RNC^2.

  Démonstration. Admise. Utilise Lemme 1: $G \notin COUPLAGEPARFAIT$ **ssi** $det(matrice\ de\ tutte\ de\ G) = 0$**. Lemme 2: Determinant calculable dans** NC^2 **(algorithme de Csanky).**

Exemples de problèmes: $PRIMES$ e

- Problème ***PRIMES*:**
  - **Instance:** un entier p
  - **Question:** p est premier?

  est dans ZPP

  Démonstration. Admise
- **Remarque:** PRIMES a été prouvé récemment dans P.