Théorie de la complexité et calcul probabiliste

4.1 Calcul probabiliste

- Le lancer de pièces peut-il aider ?
- Supposons que nous essayions de résoudre un problème de décision et que nous ayons un algorithme qui, lorsqu'on lui donne une entrée $x \in$, produit soit « vrai », soit « probablement faux ». En supposant que chaque fois qu'il produit « vrai », c'est correct, alors que chaque fois qu'il produit « probablement faux », la probabilité que cela soit correct est d'au moins 1/2
- pouvons-nous utiliser cet algorithme pour décider ?
- la réponse à cette question à toutes fins pratiques est « oui ».
- avant de formaliser ce concept d'algorithme probabiliste (ou aléatoire), nous considérons un exemple simple.

- Soit Z [x_1,. . . , x_n] désignent l'ensemble des polynômes en n variables à coefficients entiers. Étant donné deux de ces polynômes f, g ∈ Z [x_1,. . . , x_n], pouvons-nous décider efficacement s'ils sont identiques?
- Nous devons faire attention à la façon dont les polynômes sont présentés afin de savoir comment mesurer la taille d'entrée. Par exemple le polynôme suivant

$$f(x_1, \ldots, x_{2n}) = (x_1 + x_2)(x_3 + x_4) \cdots (x_{2n-1} + x_{2n}),$$

pourrait clairement être encodé en utilisant l'alphabet = {∗, 0, 1, x, (,), +, -} avec la taille d'entrée O (n log n). Cependant, si nous développons les parenthèses, ce même polynôme semblerait alors avoir une taille d'entrée O ($n2^n$ log n).

- Le degré d'un polynôme est simplement le nombre maximum de variables, comptées en fonction de leurs multiplicités, apparaissant en un seul terme lorsque le polynôme est exprimé sous sa forme développée. Ainsi, l'exemple ci-dessus a le degré n tandis que

$$g(x_1, x_2, x_3) = x_1^2 x_3 + x_2^2 x_3^2 + x_3^3,$$

a le degré 4.

- Décider si deux polynômes f et g sont identiques équivaut clairement à décider si f - g est identique à zéro, nous considérons donc ce problème à la place.

POLY NON-ZÉRO
Entrée: un polynôme entier $f \in Z[x_1, \ldots, x_n]$.
Question: f n'est-il pas identiquement nul?

- Considérez l'algorithme probabiliste suivant pour ce problème. Nous écrivons un $a \in_R A$ pour signifier que a est choisi uniformément au hasard dans l'ensemble A, tandis que $a_1, \ldots, a_n \in_R A$ dénote le fait que $a_1, \ldots, a_n$ sont choisis indépendamment et uniformément au hasard parmi A.

Algorithm *Probabilistic algorithm for NON-ZERO POLY.*

Input: an integer polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree k.
Algorithm:
choose $a_1, \ldots, a_n \in_R \{1, 2, \ldots, 2kn\}$
if $f(a_1, \ldots, a_n) \neq 0$
    then output true
    else output false.

- cet algorithme devrait très bien fonctionner. Si f n'est pas identiquement nul, alors nous ne sortirons false que si nous choisissons accidentellement une racine de f, ce qui semble plutôt improbable.
- Nous pouvons toujours répéter cette procédure, et si jamais elle produit « vrai », alors nous savons que f n’est pas identiquement nul (puisque nous avons trouvé un point auquel il est non nul).
- Cependant, si après avoir répété cela cent fois avec des choix aléatoires indépendants pour a_1,. . . ,a_n si nous obtenons toujours la réponse «faux», nous pouvons alors être presque certains que cela est correct.
- La différence importante est que nous n'essayons pas tous les certificats possibles. Au lieu de cela, cet algorithme choisit simplement un certificat possible au hasard et vérifie s'il est bon.
- La raison intuitive pour laquelle cela fonctionne est que si le polynôme d'entrée n'est pas identique à zéro, alors il y a beaucoup de bons certificats et la probabilité qu'un certificat choisi au hasard soit bon sera élevée.
- D'un autre côté, si le polynôme d'entrée est identique à zéro, alors il n'y a pas de bons certificats et donc l'algorithme répondra toujours correctement «faux».

- La différence importante est que nous n'essayons pas tous les certificats possibles. Au lieu de cela, cet algorithme choisit simplement un certificat possible au hasard et vérifie s'il est bon.
- La raison intuitive pour laquelle cela fonctionne est que si le polynôme d'entrée n'est pas identique à zéro, alors il y a beaucoup de bons certificats et la probabilité qu'un certificat choisi au hasard soit bon sera élevée.
- D'un autre côté, si le polynôme d'entrée est identique à zéro, alors il n'y a pas de bons certificats et donc l'algorithme répondra toujours correctement «faux».

- Théorème 4.1: Supposons f ∈ Z [x_1,. . . , x_n] a un degré au plus k et n'est pas identiquement nul. Si a_1,. . . , a_n sont choisis indépendamment et uniformément au hasard parmi {1,. . . , N} alors

$$\Pr[f(a_1, \ldots, a_n) = 0] \leq \frac{k}{N}.$$

- Ce théorème implique que l'algorithme POLY NON-ZERO, si l'entrée

f ∈ Z [x_1,. . . , x_n] n'est pas identiquement nul alors avec probabilité au moins 1/2, il affichera «vrai», tandis que s'il est identique à zéro, il affichera toujours «faux».

Input: a polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$ of degree k.
Algorithm:
for $i = 1$ to 100
    choose $a_1, \ldots, a_n \in_R \{1, \ldots, 2kn\}$
    if $f(a_1, \ldots, a_n) \neq 0$ then output true
next i
output false.

- S’il produit un jour « vrai », il est certainement correct, tandis que s’il produit « faux », sa probabilité d’erreur est d’au plus $1/2^{100}$. Une telle procédure, connue sous le nom algorithme probabiliste.
- Notez qu'il est également efficace (en supposant que nous avons une source d'aléatoire et que l'évaluation du polynôme en a_1, ..., a_n peut être réalisée en temps polynomial). Une telle procédure est connue sous le nom algorithme probabiliste à temps polynomial.
- Un problème évident avec un tel algorithme probabiliste est qu'il nécessite la prise en considération de l’aspect aléatoire.
- Dans les chapitres précédents, nous avons examiné les ressources informatiques du temps et de l'espace. Lors de l’évaluation de l’efficacité d’un algorithme probabiliste, nous devons également prendre en compte la quantité d’aléatoire qu’il requiert.
- Nous le mesurons par le nombre de bits aléatoires utilisés lors de son calcul.

- Exemple Choix d'un entier a $\in_R \{0,. . . , n\}$ en utilisant des bits aléatoires.
- Supposons que l'on nous donne une séquence infinie de bits aléatoires indépendants. Pour choisir un entier aléatoire a $\in R \{0,. . . , n\}$,où $(2^{k-1} \leq n < 2^k)$,on utilise la procédure suivante.

lire k bits aléatoires $b_1,. . . , b_k$ de notre séquence.
Si $a = b_1 \cdot \cdot \cdot b_k \in \{0,. . . , n\}$ (où a est encodé en binaire)
alors sortie a
sinon répéter.

Sur une seule itération, la probabilité qu'une sortie soit produite est

$$\Pr[a \in \{0, \ldots, n\}] = \frac{n+1}{2^k} > \frac{1}{2}.$$

- Ainsi, le nombre attendu d'itérations avant qu'une sortie ne se produise est inférieur à deux et, avec une probabilité d'au moins $1 - 1/2^{100}$, une sortie se produit en une centaine d'itérations.
- De plus, lorsqu'une sortie se produit, elle est choisie uniformément au hasard parmi $\{0,\ldots,n\}$. Puisque si $m \in \{0,\ldots,n\}$ et on note a_j la valeur de a choisie à la $j^{ème}$ itération de cette procédure alors

$$\begin{aligned}\Pr[\text{Output is } m] &= \sum_{j=1}^{\infty} \Pr[a_j = m \text{ and } a_1, \ldots, a_{j-1} \geq n+1] \\ &= \frac{1}{2^k} \sum_{j=0}^{\infty} \left(1 - \frac{n+1}{2^k}\right)^j \\ &= \frac{1}{n+1}.\end{aligned}$$

4.2 Machines de Turing probabilistes et RP

- Un problème est traitable si et seulement si leur solution est donnée par un algorithme probabiliste à temps polynomial.
- Afin de donner une définition formelle d'un algorithme de temps polynomial probabiliste, nous introduisons un nouveau type de machine de Turing.
- Une machine de Turing probabiliste ou PTM est un DTM avec une bande supplémentaire, appelée bande de lancer de pièces, qui contient une séquence infinie de bits aléatoires indépendants uniformément répartis
- Cette bande a une tête en lecture seule appelée tête de tirage de pièces.
- La machine effectue des calculs de la même manière qu'un DTM, sauf que la tête de tirage de pièces peut lire un peu de la bande de tirage de pièces en une seule étape.

- La fonction de transition dépend maintenant non seulement de l'état actuel et du symbole dans le carré courant de la bande ordinaire, mais également du bit aléatoire dans le carré actuellement balayé par la tête de tirage de pièces.
- La fonction de transition dit à la machine quatre choses : le nouvel état ; le nouveau symbole à écrire dans le carré courant de la bande ordinaire ; le mouvement à gauche ou à droite de la tête de lecture-écriture et le mouvement à gauche ou à droite de la tête de tirage de pièces
- Notez que puisque le ruban de tirage de pièces est infini dans une seule direction, la tête de tirage de pièces n'est pas autorisée à sortir de l'extrémité du ruban

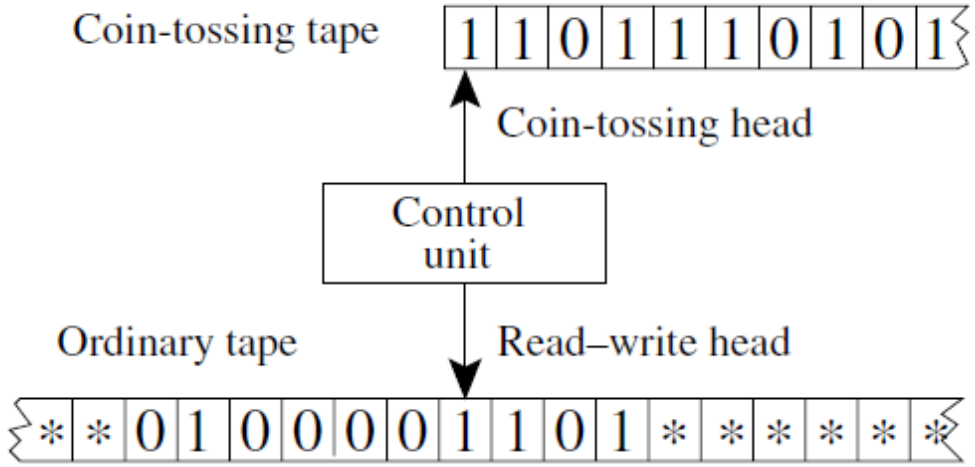

Figure. 1 Une machine de Turing probabiliste**.**

- Puisque le calcul d'un PTM, M, sur une entrée x ∈ dépend non seulement de x mais aussi des bits aléatoires utilisés lors de son calcul, son temps d'exécution est une variable aléatoire : t_M (x).
- On dit qu'un PTM s'arrête s'il s'arrête après un nombre fini de pas sur chaque entrée x ∈ $\sum_0^*$, quels que soient les bits aléatoires utilisés dans son calcul.
- La complexité temporelle d'un PTM à l'arrêt, M, est TM : N → N définie par

$$T_M(n) = \max \{t \mid \text{there exists } x \in \Sigma_0^n \text{ such that } \Pr[t_M(x) = t] > 0\}.$$

- PTM, M, a un temps d'exécution polynomial s'il existe un polynôme p (n) tel que TM (n) ≤ p (n), pour tout n ∈ N. Donc, par définition, tout PTM avec un temps d'exécution polynomial s'arrête.

- Nous considérerons parfois des PTM qui peuvent ne pas s'arrêter. Pour un tel PTM, M, la complexité temporelle n'est pas définie, cependant, on peut toujours définir son temps de fonctionnement attendu comme étant ETM : N → N tel que

$$ET_M(n) = \max\left\{t \mid \text{there exists } x \in \Sigma_0^n \text{ such that } \mathrm{E}[t_M(x)] = t\right\}.$$

- Un PTM, M, a un temps d'exécution polynomial attendu si et seulement s'il existe un polynôme p (n) tel que $ET_M(n) \leq p(n)$, pour tout $n \in N$.

4.3 La classe de complexité des langages décidable en temps polynomial aléatoire ou RP

- Un langage L appartient à RP si et seulement s'il existe un PTM, M, avec un temps d'exécution polynomial tel que sur toute entrée $x \in \sum_0^*$:
- (i) si $x \in L$ alors Pr [M accepte x] $\geq 1/2$;
- (ii) si $x \notin L$ alors Pr [M accepte x] = 0.
- Il est facile de voir que l'algorithme probabiliste pour POLY NON-ZERO pourrait être implémenté sur un PTM.
- Si l'entrée est le polynôme zéro, alors l'algorithme rejette toujours et donc la condition (ii) ci-dessus est satisfaite
- Si l'entrée est un polynôme non nul, alors avec une probabilité d'au moins 1/2, l'algorithme accepte (voir le théorème) Par conséquent, la condition (i) ci-dessus est également satisfaite.

- Théorème 4.2 : Le test de primalité de Miller – Rabin est un algorithme probabiliste à temps polynomial. Compte tenu de l'entrée n
- (i) si n est premier, alors l'algorithme produit toujours « premier » ;
- (ii) si n est composé alors Pr [l'algorithme produit « composite »] $\geq 1/2$
- D'où COMPOSITE $\in$ RP ou de manière équivalente PRIMIER $\in$ co-RP.

- La seule question qui reste est de savoir si l'algorithme s'exécute en temps polynomial.
- Nous introduisons une version restreinte de ce langage pour laquelle cela est certainement vrai.
- La taille d'entrée dans ce cas est clairement O (n^2) et le degré du polynôme est O (n).
- L'algorithme POLY NON-ZERO nous oblige donc à calculer le déterminant d'une matrice d'entiers n × n. Ceci peut être réalisé en temps polynomial (puisque l'évaluation d'un déterminant peut être réalisée en temps polynomial)
- donc DET de NON-ZERO POLY appartient à RP.

- Une autre façon de définir RP est qu'il consiste en langages L avec la propriété que si $x \in L$ alors la probabilité qu'une chaîne de longueur polynomiale aléatoire soit un certificat succinct est d'au moins 1/2; tandis que si $x \notin L$ alors aucun certificat de ce type n'existe.
- Par conséquent, nous avons le résultat suivant: $P \subseteq RP \subseteq NP$.
- Si un langage appartient à RP, nous pouvons réduire notre probabilité de rejeter par erreur une entrée correcte en répétant le calcul.
- Le résultat suivant montre qu'en répétant le calcul polynomialement plusieurs fois, nous pouvons réduire considérablement la probabilité d'une erreur.

- Proposition 4.1 Si L ∈ RP et p (n) ≥ 1 est un polynôme alors il existe une PTM, M à temps polynomial, tel que sur l'entrée x ∈ $\sum_0^*$;
- (i) si x ∈ L alors Pr [M accepte x] ≥ $1-2^{-p(n)}$
- (ii) si x ∉ L alors Pr [M accepte x] = 0.
- Nous avons prouvé au chapitre 3 que PRIMIER ∈ NP ∩ co-NP.
- Nous montrerons au TD que COMPOSÉ ∈ RP et donc PRIMIER ∈ co-RP
- Bien que l'on sache maintenant que PRIME ∈ P, ce résultat n'est pas d'intérêt théorique ou historique.
- Les algorithmes probabilistes restent de loin le moyen le plus pratique de tester la primalité.

4.4 Test de primalité

- En cryptographie, nous devons souvent choisir de grands nombres premiers aléatoires
- Cela peut être considéré comme deux problèmes distincts.
- Premièrement, choisir un grand entier au hasard et, deuxièmement, tester si l'entier choisi est premier ou non.
- Étant donné une source d'aléatoire, il est simple de choisir un entier aléatoire avec exactement k bits.
- Nous allons donc nous concentrer sur le deuxième problème.
- Nous n'avons vu qu'un algorithme déterministe de complexité exponentiel pour le test de primalité est inutile.
- Une avancée majeure dans les tests de primalité a été la découverte d'algorithmes probabilistes efficaces dans les années 1970

- L'un d'eux est l'algorithme de Miller-Rabin que nous présentons ci-dessous.
- Depuis 2002, nous disposons d’un algorithme de teste de primalité de temps polynomial déterministe dû à Agrawal. Cependant, cela a encore un temps d'exécution O ($\log^6 n$) et donc pour des raisons pratiques, l'algorithme de Miller – Rabin est plus utile.
- Rappelez-vous le problème complémentaire de PRIMIER.

COMPOSÉ

Entrée: un entier n.

Question: n est-il composé?

- On note l'ensemble des résidus non nuls mod n par $\mathbb{Z}_n^+ = \{a \in \mathbb{Z}_n \mid a \neq 0\}.$

- Lemme 4.1 : Soit $n \geq 3$ impair et $a \in \mathbb{Z}_n^+$. Écrivez $n - 1 = 2^k m$, avec m impair. Si l'une des deux conditions suivantes est remplie, n est composé:
- (i) $a^n - 1 \neq 1 \bmod n$; (on l'appelle le témoin de Fermat F_n)
- (ii) $a^n - 1 = 1 \bmod n$, $a^m = 1 \bmod n$ et aucune des valeurs de la séquence sont congruents à $-1 \bmod n$. (on l'appelle le témoin de Miller)

- Un entier composé n est un nombre de Carmichael si les seuls témoins de Fermat pour n sont ceux $a \in \mathbb{Z}_n^+$ qui ne sont pas premiers avec n. Le plus petit exemple d'un tel nombre est $561 = 3 \cdot 11 \cdot 17$.

- Le résultat suivant nous dit que si nous pouvions ignorer les nombres de Carmichael, alors le test de primalité serait extrêmement simple
- Proposition 4.2 Si n est composé mais pas un nombre de Carmichael alors $| F_n | > n / 2$.
- Théorème 4.3 Si C(x) désigne le nombre de nombres de Carmichael inférieur ou égal à x alors $C(x) > x^{2/7}$, en particulier il existe une infinité de nombres de Carmichael.
- Malgré ce fait, certaines implémentations de crypto systèmes qui nécessitent des nombres premiers aléatoires utilisent en fait l'algorithme du teste de primalité de Fermat.
- La justification en est que, puisqu'il y a beaucoup plus de nombres premiers que de nombres de Carmichael, donc le choix d'un nombre de Carmichael par erreur est négligeable.

4.5 Class de langage probabiliste sans erreur à temps polynomial, ZPP

- Un langage L est décidable en temps polynomial probabiliste sans erreur (ZPP) ssi il existe une PTM, M, avec un temps d'exécution polynomial attendu tel que pour toute entrée $x \in \sum_0^*$:
- (i) si $x \in L$ alors Pr [M accepte x] = 1;
- (ii) si $x \notin L$ alors Pr [M accepte x] = 0.
- ZPP=RP ∩ co-RP.
- Il est très difficile de trouver des exemples de langages dans ZPP \ P. En effet, cette classe pourrait bien se révéler vide, mais il reste à la prouver.

4.6 Class de langage probabiliste d'erreur bornée à temps polynomial, BPP

- Le langage L appartient à BPP s'il existe une PTM, M, avec un temps d'exécution polynomial tel que sur toute entrée $x \in \sum_0^*$:
- (i) si $x \in L$ alors Pr [M accepte x] $\geq 3/4$;
- (ii) si $x \notin L$ alors Pr [M accepte x] $\leq 1/4$.
- **Proposition 4.3** $P \subseteq ZPP = RP \cap co\text{-}RP \subseteq RP \subseteq RP \cup co\text{-}RP \subseteq BPP$.