INF554: Utilisation de l'aléatoire en algorithmique **Automne 2013**

Cours 4 — 14 octobre

Enseignant : Frédéric Magniez *Rédacteur : USTINOV Ivan*

4.1 Yao's minimax principle

Définition 4.1. *Fonction* $Cost(A, X) \geq 0$ *|* A *est un algorithme* $\in C$ *et* X *est entrée* $\in I$

Théorème 4.2. $\underset{\mathcal{A} \text{ - distribution sur } C}{MinMaxCost}(\mathcal{A}, X) = \underset{\mathcal{D} \text{ - distribution sur } I}{MaxMinCost}(A, \mathcal{D})$

Exemple

- I : Tableaux de taille n
- C : Algorithmes de tri par comparaisons (au plus n^2 comparaisons)
- $Cost(A, X)$ = nombre de comparaisons

Preuve: Notons : $\mathcal{A}$ - distribution sur C, $\mathcal{D}$ - distribution sur I

$$\underset{\mathcal{A}}{MinMaxCost}(\mathcal{A}, X) \tag{4.1}$$

$$\underset{\mathcal{A}}{Min}\underset{\mathcal{D}}{MaxCost}(\mathcal{A}, \mathcal{D}) \tag{4.2}$$

$$\underset{\mathcal{D}}{Max}\underset{\mathcal{A}}{MinCost}(\mathcal{A}, \mathcal{D}) \tag{4.3}$$

$$\underset{\mathcal{D}}{MaxMinCost}(A, \mathcal{D}) \tag{4.4}$$

Lemme 4.3. *Théorème du minimax de von Neumann pour* $x \in \mathbb{R}_+^n$, $y \in \mathbb{R}_+^m$, $a_ij \in \mathbb{R}_+$, $i \in (n)$, $j \in (m)$

$$\underset{x}{Max}\underset{y}{Min} \sum x_i y_j a_{ij} = \underset{y}{Min}\underset{x}{Max} \sum x_i y_j a_{ij}$$

Alors, (4.2) = (4.3).

(4.2) est plus generale que (4.1), donc (4.2) $\geq$ (4.1)

Fixons $\mathcal{A}$ et $\mathcal{D}$ telles que $v = Cost(\mathcal{A}, \mathcal{D})$ maximal, donc $\exists$x tel que $Const(\mathcal{A}, x) \geq v$

donc $\underset{x}{MaxCost}(\mathcal{A}, x) \geq \underset{\mathcal{D}}{MaxCost}(\mathcal{A}, \mathcal{D})$, et a pour but du x, $\mathcal{A}$ donc (4.1) $\geq$ (4.2), Alors (4.1) = (4.2)

intuitivement : (4.4) $\geq$ (4.3)

$\forall \mathcal{A}$ $\exists$A tq $Cost(\mathcal{A}, \mathcal{D}) \geq Cost(A, \mathcal{D})$, donc, (4.3) $\geq$ (4.4)

Alors, (4.3) = (4.4), à la fin on obtient (4.1) = (4.4) □

4.2 Algorithmes de streaming

4.2.1 Modèle et motivation

Définition 4.4 (Masive data). *L'entrée x (Data stream) est trop grosse pour être écrite en mémoire RAM (Random Access Memory). Sa taille est* $o(n)$ *(sous-linéaire), idéalement* $O(\log(n))$.

- *Nécessite un accès séquentiel à l'entrée : on lit x par morceaux (un bit, un entier, une lettre, ... un élément de taille fixée petite).11*
- *Une seule passe ou plusieurs possibles selon les cas (mais toujours un nombre constant de passes).*
- *A la fin des passes, l'algorithme doit calculer ou approcher une fonction.*

Exemple: Analyse de l'ADN, du graphe du WEB ... les données sont trop grandes pour être stoqués en mémoire RAM. Nombre constant de passes.

Exemple (Missing number):

Stream: suite de n entiers distincts de $[[1; n+1]]$

Sortie: trouver l'entier manquant

Contrainte: mémoire en $O(\log(n))$ bits, 1 passe

ALGORITHME

$s \leftarrow 0$

Tant que Stream non vide

    $x \leftarrow$ lire Stream

    $s \leftarrow s + x$

retourner $\frac{(n+1)(n+2)}{2} - s$

Définition 4.5 (Paramètres d'un algorithme de streaming).

paramètre	*valeur idéale*
nombre de passes	1 *ou* $O(1)$ *passes*
mémoire RAM	$O(\log^{O(1)}(n))$ *bits*
temps par morceaux	$O(\log^{O(1)}(n))$ *opérations*
temps de post-processing (pour donner la réponse après passage du stream)	$O(\log^{O(1)}(n))$ *opérations*

Lemme 1. *Tout langage régulier peut être reconnu en une passe par un algorithme de streaming de mémoire constante.*

4.2.2 Moments de Fréquence

Définition 4.6.

Stream: $a_1, a_2, ..., a_n \in [[1, m]]$. *$n$ est inconnu et m est connu*

Fréquences: $f_j = |\{i \in [[1; n]], a_i = j\}|, j \in [[1; m]]$

Moments d'ordre k: $F_k = \sum_{j=1}^{m} (f_j)^k$

- $F_0 = |\{j \in [[1; m]], f_j \neq 0\}|$ - *nombre d'éléments distincts*
- $F_1 = n$ - *nombre d'éléments*
- F_2 = *"repeat rate" ou "surprise index". F_2 grand $\Rightarrow$ f_j anormalement grand (possibilité d'attaque).*
- $F_\infty = \max_j f_j$
- $F_k = o(n^{1-2/k})(\log n + \log m)$

Définition 4.7. *μ est un $(\xi, \delta) - estimateur$ d'une valeur v si $\mathbb{P}[|\mu - v| > \delta|] \leq \xi$*

Remarque : En pratique, si on a un $(\xi, \delta) - estimateur$ avec $\delta < 1$ et $\xi < 1$ constantes, alors il est possible de construire un estimateur pour (ξ, δ) quelconque. Complexité typique : $O(\frac{1}{\delta} \log(\frac{1}{\xi}))$

Algorithme pour estimer F_1 :

Déterministe: espace en $O(\log(n))$ bits

Probabiliste: espace en $O(\log(\log(n)))$ bits

ALGORITHME

$a \leftarrow 0$

Tant que Stream non vide

    lire Stream

    $a \leftarrow a + 1$ avec probabilité $1/2^a$

retourner $2^a - 1$

Théorème 4.8. *Soit X_i la valeur de a après i éléments ($X_0 = 0, X_1 = 1$, ...)*

$$\mathbb{E}(2^{X_i}) = i + 1 \tag{4.5}$$

Preuve: Soit $\mathbb{P}_{i,j} = \mathbb{P}(X_i = j)$

$$\mathbb{E}(2^{X_i}) = \sum_j \mathbb{P}_{i,j} 2^j$$

$$\begin{aligned} \mathbb{P}_{i,j} &= \mathbb{P}(X_i = j | X_{i-1} = j)\mathbb{P}_{i-1,j} + \mathbb{P}(X_i = j | X_{i-1} = j-1)\mathbb{P}_{i-1,j-1} \\ &= (1 - \frac{1}{2^j})\mathbb{P}_{i-1,j} + \frac{1}{2^{j-1}}\mathbb{P}_{i-1,j-1} \end{aligned}$$

$$\begin{aligned}
\mathbb{E}(2^{X_i}) &= \sum_j (2^j - 1)\mathbb{P}_{i-1,j} + 2^{1-j}\mathbb{P}_{i-1,j-1} \\
&= \sum_j \mathbb{P}_{i-1,j}2^j - \sum_j \mathbb{P}_{i-1,j} + 2\sum_j \mathbb{P}_{i-1,j-1} \\
&= \mathbb{E}(2^{X_{i-1}}) - 1 + 2 \\
\mathbb{E}(2^{X_i}) &= \mathbb{E}(2^{X_{i-1}}) + 1
\end{aligned}$$

□

Inégalité de Markov

Soit X variable aléatoire positive, et soit $\mu = \mathbb{E}(X)$. $\forall a > 0, \mathbb{P}(X \geq a\mu) \leq 1/a$.
Application à notre problème : $\mathbb{P}(v \geq 2n) \leq 1/2$

Inégalité de Tchebychev

Soit X variable aléatoire telle que

$$\begin{aligned}
\mathbb{E}(X) &= \mu \\
\mathrm{Var}(X) &= \mathbb{E}(X^2) - (\mathbb{E}(X))^2 = \sigma^2
\end{aligned}$$

Alors

$$\forall a > 0, \mathbb{P}(|X - \mu| > a\sigma) < 1/a^2 \tag{4.6}$$

Dans notre problème, si l'on veut $\mathbb{P}(|v - n| \geq \frac{n}{2}) \leq \frac{1}{4}$ donc $\sigma = \frac{n}{4} \Rightarrow \sigma^2 = \frac{n^2}{16}$, ce qui n'est pas très mauvais. Nous cherchons donc les petites variances

Lemme:
$\mathrm{Var}(v) = \frac{n(n+1)}{2} \leq \frac{(\mathbb{E}(v))^2}{2}$

Ce n'est pas très bon... Il faudrait diminuer la variance, et pour ce faire on utilisera l'astuce suivante.

Mean trick

ALGORITHME

$c_t \leftarrow 0$, $t = 1, 2, \ldots, k$, k paramètre.

tant que Stream non vide

lire élément suivant

$\forall t$, avec probabilité $\frac{1}{2^{c_t}}$: $c_t \leftarrow c_t + 1$

retourner la moyenne w des $v_t = 2^{c_t} - 1$

Remarque:

$\mathbb{E}(w) = \mathbb{E}(v_t) = n$

$\mathrm{Var}(w) = \frac{1}{k^2} k \,\mathrm{Var}(v_t) = \frac{1}{k}\,\mathrm{Var}(v_t) \leq \frac{1}{2k}(\mathbb{E}(v_t))^2 \leq \frac{n^2}{2k}$

Remarque:

Soit $\epsilon > 0$. Si $a = \epsilon\sqrt{2k}$, alors

$\mathbb{P}(|w-n| \geq \epsilon n) = \mathbb{P}(|w-n| \geq \sqrt{\frac{n^2}{2k}}a) \leq \mathbb{P}(|w-n| \geq \sigma a)$, cette dernière inégalité, parce qu'on autorise davantage d'évènements puisque σ est plus petit que l'autre quantité.

$\leq \frac{1}{a^2} = \frac{1}{2\epsilon^2 k}$ d'après Markov

$\leq 1/4$ si $k = \frac{2}{\epsilon^2}$

On a obtenu ainsi que si $k = \frac{2}{\epsilon^2}$, $\mathbb{P}(|w-n| \geq \epsilon n) \leq 1/4$.
Maintenant on voudrait $\mathbb{P}(|w-n| \geq \epsilon n) \leq \delta$, quel que soit $\delta > 0$.

Median trick

$v_1, v_2, ..., v_l$ des $(\epsilon, 1/4)$ estimateurs indépendants (i.e. $\forall i, \mathbb{P}(|v_i - \mu| \geq \epsilon\mu) \leq 1/4$). $\mathbb{E}(v_i) = \mu$. Soit w leur médiane. Alors, w satisfait

$$\mathbb{P}(|w-\mu| \geq \epsilon\mu) \leq e^{-l/24} \tag{4.7}$$

Pour nous, si $l \sim \log(1/\delta)$ alors $\mathbb{P}(|w-\mu| \geq \epsilon\mu) \leq \delta$.

Théorème 4.9. *Il existe un (ϵ, δ) estimateur de F_1 en*
- *1 passe*
- *mémoire* $\mathrm{O}(\frac{\log(1/\delta)}{\epsilon^2}\log(\log(n)))$

4.2.3 Estimer F_0

Définition 4.10 (2-universal family).

$\exists H \subseteq \{h : [[1;m]] \to [[1;M]]\}$ *tel que* $\left\{ \begin{array}{c} \forall x \neq y \in [[1;m]] \\ \forall u, v \in [[1;M]] \end{array} \right.$, $\mathbb{P}_h\left(\left\{ \begin{array}{l} h(x)=u \\ h(y)=v \end{array} \right. \right) = 1/M^2$

Si les valeurs sont uniformément réparties, $(\min_i(a_i)) = m/F_0$

Conséquences: $\forall x \in [[1;m]], \forall u \in [[1;M]], \mathbb{P}(h(x) = u) = 1/M$

interprétation: Si h uniformément choisi dans H
- $\forall x \in [[1;m]], h(x)$ uniformément réparti sur $[[1;M]]$
- $\forall x \neq y \in [[1;m]]$, $h(x)$ et $h(y)$ indépendants

Théorème 4.11 (Construction). *Soit $m \leq p < 2m$ premier. $M = p$*

$$\forall a, b \in [[0;p-1]], h_{a,b} : \begin{array}{ccc} [[1;m]] & \to & [[1;p]] \\ x & \mapsto & a \cdot x + b \mod p \end{array} \tag{4.8}$$

$\{h_{a,b}, a, b \in [[0;p-1]]\}$ *est une 2-universal family*

Preuve: $\left\{ \begin{array}{r} \forall x \neq y \in [[1;m]] \\ \forall u, v \in [[1;M]] \end{array} \right.$ $\exists! a, b \in [[0; p-1]]$ tq $\left\{ \begin{array}{l} a \cdot x + b = u \mod p \\ a \cdot y + b = v \mod p \end{array} \right.$ (système d'équations linéaires non dégénéré, car $x \neq y$)
Par conséquent $\mathop{\mathbb{P}}_{a,b} \left(\left\{ \begin{array}{l} h_{a,b}(x) = u \\ h_{a,b}(y) = v \end{array} \right. \right) = 1/p^2$ □

ALGORITHME MINHASH

$m \leq p < 2m$ premier, $min \leftarrow p$

$a, b \in [[0; p-1]]$

Tant que Stream non vide

$x \leftarrow$ lire Stream

$min \leftarrow \min\{a \cdot x + b \mod p; min\}$

retourner p/min

---

analyse:
- 1 passe
- mémoire en $O(\log(m))$ bit
- temps par élément en $O(1)$ opérations arithmétiques
- temps post-processing en $O(1)$ opérations arithmétiques

Théorème 4.12.

$$\mathbb{P}(F_0/6 \leq p/min \leq 6F_0) \geq 2/3 \tag{4.9}$$

Améliorable avec les même techniques que F_1

Preuve:

$$\begin{aligned} \mathbb{P}(p/min > 6F_0) &= \mathbb{P}(\exists k, h(a_k) < \frac{p}{6F_0}) \\ &\leq \sum_k \mathbb{P}(h(a_k) < \frac{p}{6F_0}) \\ &\leq F_0 \max_k \mathbb{P}(h(a_k) < \frac{p}{6F_0}) \\ &\leq F_0 \frac{p}{6F_0} \frac{1}{p} \\ &\leq 1/6 \end{aligned}$$

$$\mathbb{P}(p/min < F_0/6) = \mathbb{P}(\forall k, h(a_k) > \frac{6p}{F_0})$$

Soit $Y_k = \left\{ \begin{array}{l} 1 \text{ si } h(a_k) > \frac{6p}{F_0} \\ 0 \text{ sinon} \end{array} \right.$ et $Y = \sum_k Y_k$

On a $\mathbb{E}(Y_k) = \frac{6}{F_0}$ et $\mathrm{Var}(Y_k) = \frac{6}{F_0}(1 - \frac{6}{F_0})$
Donc $\mathbb{E}(Y) = 6$ et $\mathrm{Var}(Y) = 6(1 - \frac{6}{F_0}) < 6$

Finalement

$$\begin{aligned}\mathbb{P}(p/min < F_0/6) &= \mathbb{P}(Y = 0)\\ &\leq \mathbb{P}(|Y - \mathbb{E}(Y)| \geq 6)\\ &\leq 1/6\end{aligned}$$

□

4.3 Exemples

4.3.1 Identity testing

Donnée: $x, y \in \{0,1\}^n$, n connu

Stream: $(x_i, i, "x")$ ou $(y_j, j, "y")$ dans un ordre quelconque

Sortie: décider si $x = y$ avec erreur unilatérale 1/n.

Contrainte: mémoire en O(log(n)), 1 passe

Remarque: déterministe en une passe ⇒ mémoire en O(n) au moins

Idée : Polynomial identity testing (1.2.3).
Evaluer $P = \sum_{i=1}^{n} x_i x^{n-i}$ et $Q = \sum_{i=1}^{n} y_i x^{n-i}$. Soit p premier, $n^2 < p < 2n^2$, $a \in [[0; p-1]]$.

Lemme (Rappel): Si $x \neq y$, $\mathbb{P}_{a \in \mathbb{Z}_p}[P(a) = Q(a) \mod p] \leq \frac{n}{p} \leq \frac{1}{n}$.
Si $x = y$ alors $\forall a \in \mathbb{Z}_p P(a) = Q(a)$.

ALGORITHME

Prendre p premier, $n^2 < p < 2n^2$, $a \in_{\mathfrak{R}} [[0; p-1]]$

$h_x, h_y \leftarrow 0$

tant que: Stream non vide

lire l'élement suivant

si: $u = (1, i, "x")$

alors: $h_x \leftarrow h_x + a^{n-1} \mod p$ ($\log n$ multiplications par exponenciation rapide.

si: $v = (1, i, "y")$

alors: $h_y \leftarrow h_y + a^{n-1} \mod p$

accepter si $h_x = h_y$

rejeter sinon

Analyse: $h_x = P(a) \mod p$
$h_y = Q(a) \mod p$

4.3.2 Permutation

Stream: n entiers de $[[1;n]]$, $a_1, a_2, \ldots, a_n$

Sortie: décider s'ils sont tous distincts en 1 passe et mémoire en $\mathrm{O}\log n$

Remarque: tous distincts $\Leftrightarrow$ permutation de $[[1;n]]$ (mais stockage d'une permutation en $\mathrm{O}(n)$)

On transforme a_i en $(a_i, 1, "x"), (i, 1, "y")$ et on applique l'algorithme précédent. On a ainsi $P = \sum_{i=1}^{n} x_i x^{n-i}$ et $Q = \sum_{i=1}^{n} y_i x^{n-a_i} = \sum_{j=1}^{n} c_j x^{n-j}$ où $c_j = \sharp\{i : a_i = j\}$, $0 \leq c_j \leq n$.

Comme $p \geq n^2$, $Q = P \mod p \Leftrightarrow Q = P \Leftrightarrow c_j = 1 \forall j \Leftrightarrow a_i$ sont tous distincts.

En conséquence, si tous les a_i sont distincts, l'algorithme accepte avec probabilité 1. Sinon, l'algorithme accepte avec probabilité $\leq 1/n$

4.3.3 Permutation 2

Stream: $a_1, a_2, \ldots, a_n$, n entiers de $[[1;n+1]]$

Sortie: décider s'ils sont tous distincts

Ce problème est une combinaison entre *missing element* et le précédent.

Algorithme

$n^2 < p < 2n^2$ premier, $a \in_{\mathfrak{R}} [[0;p-1]]$

$h_x, h_y \leftarrow 0$

$S \leftarrow \frac{(n+1)(n+2)}{2} = 1 + 2 + \cdots + (n+1)$

$i \leftarrow 0$

tant que Stream non vide

  $h_y \leftarrow h_y + a^i \mod p$

  lire Stream, a_i

  $h_x \leftarrow h_x + a^{n+1-a_i} \mod p$

  $s \leftarrow s - a_i$

  $i \leftarrow i + 1$

$h_y \leftarrow h_y + a^{n+1} \mod p$, ainsi $h_y = (1 + x + \cdots + x^n)(a) \mod p$

$h_x \leftarrow h_x + a^{n+1-s} \mod p$, ainsi $h_x = \sum_{i=1}^{n} \left(x^{n+1-a_i} + x^b\right)(a) \mod p$, où $b = \frac{(n+1)(n+2)}{2} - \sum_{i=1}^{n} a_i = \sum_{i=1}^{n} i - \sum_{i=1}^{n} a_i$.

accepter si $h_x = h_y$.

Preuve:

- Si les a_i sont tous distincts, alors b désigne l'entier manquant, donc $P = Q$ et l'algorithme retourne accepte avec probabilité 1.
- Si les a_i ne sont pas tous distincts, et $1 \leq b \leq n+1$, alors deux valeurs dans $\{1, 2, \ldots, n+1\}$ ne sont pas prises et il y a un monôme x^j dans P avec coefficient 0, où $0 \leq j \leq n$. Donc $P \neq Q$ et alors l'algorithme accepte avec probabilité $\leq 1/n$.

Une approche déterministe utiliserait une mémoire en $\mathrm{O}(n)$.