INF561: Algorithmes de streaming II **Hiver 2013**

Cours 6 — 13 février

Enseignant : Frédéric Magniez *Rédacteur : Jorge GONZÁLEZ SUITT*

6.1 Exemples

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

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

6.1.2 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 $\leq 1/n$

6.1.3 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 $\mathrm{O}(n)$.

6.1.4 Exercice : Element distinctness

Stream: $a_1, a_2, \ldots, a_n$, n entiers de $[[1; n+1]]$ tels qu'il existe une unique paire $i < j$ pour laquelle $a_i = a_j = v$. (Un élement apparaît deux fois)

Sortie: trouver v.

On cherche un algorithme avec mémoire locale en $O(\log n)$ et un nombre minimal de passes.

6.2 Moments et fréquences : retour

6.2.1 Estimer F_1

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

Sortie: approximation de n

ALGORITHME

$c \leftarrow 0$

Tant que Stream non vide

    lire Stream

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

retourner $2^c - 1$

Lemme:

$\mathbb{E}(v) = n$

v requiert $O(\log(\log(n)))$ bits

6.2.2 Outils mathématiques et astuces

Inégalité de Markov

Soit X variable aléatoire positive, et soit $\mu = \mathbb{E}(X)$. $\forall a > 0, \mathbb{P}(X \geq a\mu) \leq 1/a$.

A: pplication 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

$$\forall a > 0, \mathbb{P}(|X - \mu| > a\sigma) < 1/a^2 \tag{6.1}$$

D: ans 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

$$\mathbb{P}(|w-\mu| \geq \epsilon\mu) \leq e^{-l/24} \tag{6.2}$$

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

Théorème 6.1. *Il existe un (ϵ, δ) estimateur de F_1 en*
- *1 passe*
- *mémoire* $\mathrm{O}(\frac{\log(1/\delta)}{\epsilon^2}\log(\log(n)))$

6.2.3 Redéfinitions

Définition 6.2.

Stream: $a_1, a_2, ..., a_n \in [[1, m]]$. *$n$ et m sont connus désormais.*

Fréquences: $f_i = |\{j \in [[1; n]], a_j = i\}|, i \in [[1; m]]$ *nombre d'occurrences de i.*

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

- F_0 =*nombre d'éléments distincts*
- $F_1 = n$
- $F_\infty = \max_j f_j$

6.2.4 Estimer F_0

Définition 6.3 (2-universal family).

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

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

$$\forall a, b \in [[0; p-1]], h_{a,b} : \begin{array}{ccc} [[1; m]] & \to & [[1; p]] \\ x & \mapsto & a \cdot x + b \mod p \end{array} \tag{6.3}$$

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

Preuve: $\left\{ \begin{array}{c} \forall x \neq y \in [[1; m]] \\ \forall u, v \in [[1; M]] \end{array} \right.$ $\exists! a, b \in [[0; p-1]]$ tq $\left\{ \begin{array}{ll} a \cdot x + b = u & \mod p \\ a \cdot y + b = v & \mod p \end{array} \right.$ (système d'équations linéaires non dégénéré, car $x \neq y$)

Par conséquent $\mathop{\mathbb{P}}_{a,b} \left(\left\{ \begin{array}{c} h_{a,b}(x)=u \\ h_{a,b}(y)=v \end{array} \right. \right) = 1/p^2$ □

Algorithme MinHash

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

$a, b \in_{\mathfrak{R}} [[0; p-1]]$

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 6.5.

$$\mathbb{P}(F_0/6 \leq p/min \leq 6F_0) \geq 2/3 \tag{6.4}$$

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 $Y_k = \begin{cases} 1 \text{ si } h(a_k) > \frac{6p}{F_0} \\ 0 \text{ sinon} \end{cases}$ 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

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

□

6.2.5 Autres résultats

1 passe:
- F_2 en $O(\log(n) + \log(m))$ bits
- F_∞ en $O(m \log(\log(n)))$ bits
- F_k en $O(m^{1-1/k}(\log(n) + \log(m)))$

6.2.6 Eléments les plus fréquents

ALGORITHME

k paramètre

$T \leftarrow \emptyset$

Tant que Stream non vide

$i \leftarrow$ lire Stream

Si $i \in T, c_i \leftarrow c_{i+1}$

Sinon si $|T| < k-1, T \leftarrow T \cup \{i\}$

Sinon $\forall j \in T, c_j \leftarrow c_j - 1$

$\forall j \in T$, si $c_j = 0, T \leftarrow T \setminus \{j\}$

retourner T

Théorème 6.6.
L'algorithme renvoie T tel que $\begin{cases} \forall i \in T, f_i > c_i > f_i - n/k \\ \forall j, f_j > n/k \Rightarrow j \in T \end{cases}$
De plus, l'espace en mémoire est en $\mathrm{O}(k \log n)$

Corollaire: Avec deux passes il es possible de trouver exactement les éléments i tels que $f_i > n/k$, avec leurs respectives fréquences et espace de mémoire en $\mathrm{O}(k \log n)$.

6.3 Problèmes de graphe

6.3.1 Modèle

Définition 6.7. *$G = (V, E)$ graphe non dirigé, $V = \{1, 2, \ldots, n\}$*
$|V| = n$ connu et $|E| = m$ inconnu
Potentiellement, $E \sim n^2$
Stream: *arêtes $e \in E$ dans un ordre arbitraire*

Problème 1:

Donnée: Graphe G comme Stream

Sortie: Matching M de G de cardinal maximum.

Problème 2:

Donnée: Graphe G comme Stream, chaque arête arrive avec poids entier $w_e \geq 1$.

Sortie: Matching M de G de poids maximum.

La taille de la sortie peut aller jusqu'à $n/2$. On cherche des algorithmes de streaming à 1 passe, avec taille de mémoire en une fonction polynomial en n fois $\log n$.

6.3.2 Maximum cardinality matching

ALGORITHME

$M \leftarrow \emptyset$

Tant que Stream non vide

$e \leftarrow$ lire Stream

Si $M \cup \{e\}$ est toujours un matching, $M \leftarrow M \cup \{e\}$

retourner M

Remarque: M est un matching maximal au sens de l'inclusion, ie $\forall e \in E, M \cup \{e\}$ n'est pas un matching.

Lemme 6.8. *Soit* OPT *un matching de taille maximale. Alors tout matching M maximal pour l'inclusion satisfait* $\Rightarrow |OPT|/2 \leq |M| \leq |OPT|$

Preuve (par chargement):
- Chaque arrête de OPT met une charge à une des extrémités dans $V(M)$ (les sommets issus des arrêtes de M).
- M maximal, donc $\forall e \in OPT$, l'une des extrémités de e au moins est dans $V(M)$.
- Chaque sommet est chargé au plus 1 fois (car OPT est un matching).
- Finalement, $|OPT| = \sum_{u \in V(M)} charge(u) \leq |V(M)| = 2|M|$

□

Remarque:
- On ne connait aucun moyen (déterministe ou probabiliste) de faire mieux en une passe.
- Pour tout ϵ fixé, on peut trouver en $|f(\epsilon)|$ passes un M tel que $\begin{cases} |M| \geq (1-\epsilon)|OPT| \\ \text{mémoire en } \mathrm{O}(n \log^{\mathrm{O}(1)}(n) g(\epsilon)) \end{cases}$

6.3.3 Maximum Weight Matching

Théorème 6.9. *Il existe un algorithme probabiliste qui calcule M*
- *en 1 passe*
- *avec mémoire en* $\mathrm{O}(n \log n)$
- *tel qu'il calcule un matching de poids au moins* $1/8$ *du poids maximum possible.*

ALGORITHME

Executer en parallèle l'algorithme glouton sur $E_i = \{e \in E, w(e) \in [2^i; 2^{i+1}[\}$, on obtient des matchings M_i tels que chaque M_i est maximal dans E_i (pour l'inclusion).

Retourner M le matching obtenu en appliquant l'algorithme glouton sur les ensembles M_i arrivant dans l'ordre des i décroissants

Résultat:

Mémoire : $\log_2(w_{max})n\log(n)$, avec $w_{max} = \max_{e\in E} w(e)$.

Soit OPT un matching de poids maximal.

Réduction de la mémoire:

- Supposons connaître $w_{max} = \max_{e\in E} w(e)$. Sinon, on le calcul à la volée.
- Il suffit de garder les arrêtes de poids inférieur à $2\epsilon w_{max}/n$. En effet, notons OPT' l'ensemble OPT sans ces arrêtes. Alors

$$w(OPT) \leq w(OPT') + \frac{n}{2}\frac{2\epsilon w_{max}}{n} \leq w(OPT') + \epsilon w_{max} \leq (1+\epsilon)w(OPT').$$

  D'où une erreur relative de ϵ et une mémoire de $n\log n$ arêtes.

Pour chaque $\forall e \in E_i$, posons $w'(e) = 2^i$. Alors $w'(e) \leq w(e) < 2w'(e)$ et par conséquent :

Lemme 6.10. $w(OPT) < 2w'(OPT)$

Couplé au lemme suivant, le lemme précédent prouve que le matching M est de poids au moins 1/8 du poids de OPT.

Lemme 6.11.

$$w'(OPT) \leq 4w'(M).$$

Preuve: Considérons le schémas de charge suivant. Pour chaque arête $e \in OPT$, on cherche le plus grand i tel qu'il existe une arête $f \in M_i$ qui intersecte e. On charge alors un sommet de $e \cap f$ (il peut y en avoir deux si $e = f$) avec $w'(f) = 2^i$. Posons de plus j tel que $e \in OPT_j$. Alors la maximalité de M_j entraine qu'au moins un des sommets de e est recouvert par M_j, et donc que $i \geq j$, c'est-à-dire que $w'(f) \geq w'(e)$.

Par construction, chaque sommet est chargé au plus une fois et la charge total des sommets est donc au moins $w'(OPT)$.

Nous allons maintenant déplacer les charges des sommets sur les arêtes de M. Considérons un sommet u de charge M_j. Le sommet u est donc couvert par une arête $f = (u, v) \in M_j$. Cette arête est nécessairement unique car M_j est un couplage. Deux cas sont alors possibles :

1. Si f est aussi dans M, alors $f \in M_j \cap M$ on déplace la charge $w'(f)$ de u sur f.
2. Sinon, il existe une arête g responsable du fait que $f \notin M$. Nécessairement $g \in M_i \cap M$ avec $i > j$ et g intersecte f en v. (En effet, g ne peut intersecter f en u, car sinon la charge de u serait au moins $w'(g)$.) Encore une fois cette arête g est unique dans M_i. On déplace alors la charge $w'(f)$ de u sur g.

Au final, les charges sont uniquement sur les arêtes de M, mais une arête peut se retrouver charger plusieurs fois. Bornons la charge totale d'une arête $g \in M_i \cap M$ en fonction de i, c'est-à-dire en fonction de son poids $w'(g)$. Pour $j = i$ et $f = g$, le cas (1) ne peut se produire qu'au plus deux fois car f n'a que deux extrémités. Pour chaque valeur $j < i$, le cas (2) ne

peut se produire aussi qu'au plus deux fois. En effet, au plus deux arêtes $f \in M_j$ intersectent g. En conclusion, la charge totale d'une arête $g \in M_i \cap M$ et au plus $2\sum_{j\leq i} w'(g)2^{j-i}$.

On en déduit que la charge totale de toutes les arêtes de M est au plus

$$\begin{aligned}\sum_{i\geq 0}\sum_{g\in M\cap M_i} 2w'(g)\sum_{j\leq i} 2^{j-i} &\leq \sum_{g\in M} 2w'(g)\sum_{k\geq 0}\frac{1}{2^k}\\ &= 4w'(M).\end{aligned}$$

□