INF561: Algorithmes de streaming	**Hiver 2012**

Cours 5 — 1 février

Enseignant : Frédéric Magniez — *Rédacteur : Aurélien LAMBERT*

5.1 Motivations et modèle

Définition 5.1 (Masive data). *L'entrée x (Data stream) est trop grosse pour être écrite en mémoire RAM (Random Access Memory). Sa taille est* $\mathrm{o}(n)$ *(sous-linéaire), idéalement* $\mathrm{O}(\log(n))$.

- *Nécessite un accès séquentiel à l'entrée : on lit x par morceaux (un bit, un entier, une lettre, ... un élément de taille fixée petite).*
- *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: Développé par AT&T (American Telephone & Telegraph) pour analyser le traffic sur ses routeurs. Analyse des adresses IP passant par ses routeurs. Les données générées en 24 heures sont bien trop grandes pour être stoquée : une seule passe possible.

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 $\mathrm{O}(\log(n))$ 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 5.2 (Paramètres d'un algorithme de streaming).

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

Exemple (Identity destiny):

Donnée: $X, Y \in \{0,1\}^n$

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

Sortie: décider si $x = y$

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

Remarque: déterministe en une passe $\Rightarrow$ memoire en O(n) au moins

---

Algorithme

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

calcul des fingerprint de x et y ($fp(x) \leftarrow \sum_i x_i a^i \mod p$, $fp(y) \leftarrow \sum_j y_j a^j \mod p$)

retourner $fp(x) = fp(y)$

Exemple:

Stream: n entiers de $[[1; n]]$

Sortie: décider s'ils sont tous distincts

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

Algorithme

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

$h \leftarrow 0, h_r \leftarrow 0$

tant que Stream non vide

  $hr \leftarrow 1 + a \cdot h_r \mod p$

  $i \leftarrow$ lire Stream

  $h \leftarrow h + a^{i-1} \mod p$

retourner $h = h_r$

Exemple:

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

Sortie: décider s'ils sont tous distincts

Algorithme

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

$h \leftarrow 0, h_r \leftarrow 1, s \leftarrow 0$

tant que Stream non vide

$hr \leftarrow 1 + a \cdot h_r \mod p$

$i \leftarrow$ lire Stream

$s \leftarrow s + i$

$h \leftarrow h + a^{i-1} \mod p$

$h_r \leftarrow h_r - a^{\frac{(n+1)(n+2)}{2} - s - 1} \mod p$

retourner $h = h_r$

Preuve:
- Si les entiers sont tous distincts, alors $\frac{(n+1)(n+2)}{2} - s$ désigne l'entier manquant, donc $h = h_r$ et l'algorithme retourne vrai.
- Si les entiers ne sont pas tous distincts, $h_r(X)$ est un polynome à n coeffiscients, donc est différents de $h(X)$. De plus, les facteurs de $h(X)$ sont inférieurs à n, donc strictement inférieurs à p. Donc l'algorithme retourne vrai avec une probabilité inférieure à $1/n$.

5.2 Moments et fréquences

Définition 5.3.

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

- $F_0 = |\{j \in [[1; m]], f_j \neq 0\}|$
- $F_1 = n$
- $F_2 =$ *"repeat rate" ou "surprise index". F_2 grand $\Rightarrow f_j$ anormalement grand (possibilité d'attaque).*
- $F_\infty = \max_j f_j$

5.2.1 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 5.4. *Soit X_i la valeur de a après i éléments ($X_0 = 0, X_1 = 1, ...$)*

$$\mathbb{E}(2^{X_i}) = i + 1 \tag{5.1}$$

$$\mathrm{Var}(2^{X_i}) = \frac{i(i+1)}{2} \leq \frac{1}{2}\mathbb{E}(2^{X_i}) \tag{5.2}$$

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

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

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

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

□

5.2.2 Amélioration de l'estimateur

On voudrait un (ϵ, δ) estimateur (ie un estimateur V de S tel que $\mathbb{P}(|V - S| > \epsilon S) < \delta$). Par exemple, $\epsilon = \delta = 1/100$

Théorème 5.5 (Inegalite 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

$$\forall a > 0, \mathbb{P}(|X - \mu| > a\sigma) < 1/a^2 \tag{5.3}$$

Remarque: Ici, $\sigma = \frac{\mathbb{E}(X)}{\sqrt{2}}$ donc $\forall a > 0, \mathbb{P}(|X - \mu| \geq \frac{a}{\sqrt{2}}\mu) \leq \frac{1}{a^2}$ donc $a/\sqrt{2} = 1/100 \Rightarrow \mathbb{P} \leq 100^2$ ... inutile

Théorème 5.6 (moyenne). *On calcule k copies indépendantes du même estimateur. $Y_1, Y_2, ..., Y_k$ indépedants et $Y = \sum_i Y_i/k$ le nouvel estimateur*

$$\mathbb{E}(Y) = \mathbb{E}(Y_i) = \mu \tag{5.4}$$

$$\mathrm{Var}(Y) = \frac{\mu^2}{2k} \tag{5.5}$$

Remarque: Finalement $k = \frac{1}{2\delta\epsilon^2}$:
- Pour $\delta = \epsilon = 1/100$, k est bien trop grand.
- Pour $k = 2/\epsilon^2$, $\delta = 1/4$

Preuve:

$$\begin{aligned}\mathrm{Var}(Y) &= \sum_i \mathrm{Var}(Y_i/k) \\ &= \frac{1}{k^2}\sum_i \mathrm{Var}(Y_i) \\ &= \frac{1}{k^2} k \frac{\mu^2}{2}\end{aligned}$$

□

Théorème 5.7 (médiane). *$Z_1, Z_2, ..., Z_l$ des $(\epsilon, 1/4)$ estimateurs indépendants (ie $\forall i, \mathbb{P}(|Z_i \mu| > \epsilon\mu) < 1/4$) et Z leur médiane*

$$\mathbb{P}(|Z - \mu| > \epsilon\mu) < e^{-l/24} \tag{5.6}$$

Pour avoir un (ϵ, δ) estimateur, $l \sim \log(1/\delta)$

Théorème 5.8. *Il existe un (ϵ, δ) estimateur de F_1 en*
- *1 passe*
- *mémoire* $\mathrm{O}(\frac{\log(1/\delta)}{\epsilon^2}\log(\log(n)))$

5.2.3 Estimer F_0

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

Définition 5.9 (2-universal family).
$\exists H \subseteq \{h : [[1;m]] \to [[1;M]]\}$ *tel que* $\left\{\begin{array}{c}\forall x \neq y \in [[1;m]] \\ \forall u, v \in [[1;M]]\end{array}\right., \mathbb{P}_h\left(\left\{\begin{array}{c}h(x)=u \\ h(y)=v\end{array}\right.\right) = 1/M^2$

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

$$\forall a, b \in [[0; p-1]], h_{a,b} : \begin{array}{rcl} [[1;m]] & \to & [[1;p]] \\ x & \mapsto & a \cdot x + b \mod p \end{array} \tag{5.7}$$

$\{h_{a,b}, a, b \in [[0; p-1]]\}$ 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 $\left\{ \begin{array}{l} a \cdot x + b = u \mod p \\ a \cdot y + b = v \mod p \end{array} \right.$ (système d'équations linéaires non dégénéré, car $x \neq y$)

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

Algorithme MinHash

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

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

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 5.11.

$$\mathbb{P}(F_0/6 \leq p/min \leq 6F_0) \geq 2/3 \tag{5.8}$$

Améliorable avec les même techniques que F_1

Preuve:

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

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

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

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

□

5.2.4 Autres résultats

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

5.2.5 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 5.12.
L'algorithme renvoie T tel que $\begin{cases} \forall i \in T, f_i > c_i > f_i - n/k \\ \forall j, f_j > n/k \Rightarrow j \in T \end{cases}$

5.3 Problèmes de graphe

5.3.1 Modèle

Définition 5.13. *$G = (V, E)$ graphe non dirigé*

$|V| = n$ connu et $|E| = m$ inconnu
Potentiellement, $E \sim n^2$

Stream: *E dans un ordre arbitraire*

Exemple (Connectivité):

Solution: se résoud en simulant UnionFind (on retient au plus $n-1$ arrêtes). Le graphe obtenu est connexe si et seulement si G est connexe.

complexité: $\mathrm{O}(n \log(n))$ bit

Exemple (Bipartition):

$$\begin{aligned} G \text{ biparti} \quad &\Leftrightarrow \quad \exists(A,B), \left\{ \begin{array}{ccc} V &=& A \sqcup B \\ E &\subseteq& A \times B \end{array} \right. \\ &\Leftrightarrow \quad G \text{ n'a pas de cycle impair} \end{aligned}$$

Algorithme

$F \leftarrow \emptyset$

Tant que Stream non vide

  $e \leftarrow$ lire Stream

  Si $F \cup \{e\}$ n'a pas de cycle, $F \leftarrow F \cup \{e\}$

  Sinon, si le cycle est impair, rejeter

Accepter

Analyse:

- Mémoire en $\mathrm{O}(n \log(n))$ bits
- Si G biparti, G n'a pas de cycle impair, donc $F \cup \{e\} \subseteq E$ non plus.
- Si G n'est pas biparti, G a au moins un cycle impair. Supposons que l'algorithme accepte. Alors quand arrive la dernière arrête du cycle, toutes les autres arrêtes du cycle sont soit présentes, soit formaient un cycle pair, donc sont contournable par un chemin de longueur impaire. Cette dernière arrête forme donc un cycle de longueur impaire, ce qui est absurde.

5.3.2 Maximum cardinality matching

Définition 5.14. *Trouver $M \subseteq E$ tel que* $\left\{ \begin{array}{l} \forall e, f \in M, e \cap f = \emptyset \\ M \textit{ est de cardinal maximum} \end{array} \right.$

Dans la suite, *Opt* est un tel matching (inconnu). En particulier, $|Opt| < n/2$

Exemple (Algorithme glouton):

ALGORITHME
$M \leftarrow \emptyset$
Tant que Stream non vide
  $e \leftarrow$ lire Stream
  Si $M \cup \{e\}$ est 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 5.15. *M matching maximal $\Rightarrow |Opt|/2 \leq |M| \leq |Opt|$*

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, $|Opt| = \sum\limits_{u \in V(M)} charge(u) \leq |V(M)| = 2|M|$

□

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 $\begin{cases} |M| \geq (1-\epsilon)|Opt| \\ \text{mémoire en } O(n \log^{O(1)}(n) g(\epsilon)) \end{cases}$

5.3.3 Maximum Weight Matching

Définition 5.16. *Chaque arrête à un poids $w(e) > 0$*

Stream: *$(e, w(e))$ dans un ordre quelconque*

Problème: *Trouver $M \subseteq E$ tel que* $\begin{cases} \forall e, f \in M, e \cap f = \emptyset \\ w(M) = \underset{e \in M}{w}(e) \text{ maximal} \end{cases}$

Remarque:
- *Si $\forall e \in E, w(e) = 1$, il s'agit du cas précédent.*
- *Si les arrêtes sont données dans un ordre croissant de poids, l'algorithme glouton peut-être particulièrement inefficace.*

Théorème 5.17. *Il existe un algorithme probabiliste qui calcule M*
- *en 1 passe*
- *avec mémoire en $O(n \log^2(n))$*

– *tel que* $w(Opt)/4.91 \leq \mathbb{E}(w(M)) \leq w(Opt)$

Remarque:

– On en déduit une version déterministe avec mémoire en $O(n \log^3(n))$.
– en 2005, un algorithme proche du glouton à 5.828 près.
– en 2008, une version améliorée à 5.585 près.

Simplification:

– On connait $w_{max} = \max_{e \in E} w(e)$. Sinon, on le calcul à la volée.
– On ne regarde pas les arrêtes de poids inférieur à $2\epsilon w_{max}/n$. En effet, notons Opt' l'ensemble Opt sans ces arrêtes

$$\begin{array}{rcl} w(Opt) & \leq & w(Opt') + \frac{n}{2}\frac{2\epsilon w_{max}}{n} \\ & \leq & w(Opt') + \epsilon w_{max} \\ & \leq & (1+\epsilon) w(Opt') \end{array}$$

D'où une erreur relative de ϵ.

Algorithme déterministe:

– $\phi > 0$ et $\gamma \geq 2$
– $E_i = \{e \in E, w(e) \in [\phi\gamma^i; \phi\gamma^{i+1}[\}$
– Pour chaque $\mathbb{E}_i$, calculer M_i le matching résultant de l'algorithme glouton
– Retourner M le matching obtenu en appliquant l'algorithme glouton sur les ensembles M_i arrivant dans l'ordre des i décroissants

Résultat:

– Si $\phi = 1$ et $\gamma = 2$, alors $w(Opt) \leq 8w(M)$.
– Si $\phi = \gamma^\delta$, avec $\delta \in_{\mathfrak{R}} [0; 1[$, alors $w(Opt) \leq 4.91\mathbb{E}(w(M))$. (Remarque : on peut supprimer l'aléa en essayant suffisament de valeurs de δ.)

$\forall e \in E_i$, on pose $w'(e) = \phi\gamma^i$.

Lemme 5.18. $w(Opt) < \gamma w'(Opt)$

Preuve: En effet, $w(e) < \gamma w'(e)$. □

Lemme 5.19.

$$w'(Opt) \leq \frac{2\gamma}{\gamma - 1} w'(M).$$

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 $w'(f) = \phi\gamma^i$. 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 $2\sum_{j \leq i} w'(g)\gamma^{j-i}$.

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

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

□

Le cas déterministe découle alors en posant $\phi = 1$ et $\gamma = 2$. Procédons maintenant au cas probabiliste.

Lemme 5.20. *Pour toute arête $e \in E$:*

$$\mathop{\mathbb{E}}_{\delta}(w'(e)) = w(e)\frac{(1 - 1/\gamma)}{\ln(\gamma)}.$$

Et donc en particulier

$$w(Opt) \leq \frac{\gamma \ln(\gamma)}{\gamma - 1} \times \mathop{\mathbb{E}}_{\delta}(w(M)).$$

Preuve: La deuxième partie du lemme découle directement de la première. Nous montrons donc maintenant la première égalité. Fixons une arête $e \in E$. Soit i entier et $\alpha \in [0; 1[$ tel que $w(e) = \gamma^{i+\alpha}$.

$$\left\{ \begin{array}{lcl} \gamma^{i+\delta} < \gamma^(i+\alpha) & \Rightarrow & w'(e) = \gamma^{i+\delta} \\ \gamma^{i+\delta} \geq \gamma^(i+\alpha) & \Rightarrow & w'(e) = \gamma^{i-1+\delta} \end{array} \right. \text{ Donc}$$

$$\begin{aligned}
\mathop{\mathbb{E}}_{\delta}(w'(e)) &= \int_0^{\alpha} \gamma^{i+\delta} d\delta + \int_{\alpha}^{1} \gamma^{i-1+\delta} d\delta \\
&= \tfrac{1}{\ln(\gamma)}(\gamma^i(\gamma^\alpha - 1) + \gamma^{i-1}(\gamma - \gamma^\alpha)) \\
&= \tfrac{w(e)}{\ln(\gamma)}(1 - 1/\gamma)
\end{aligned}$$

□