Fouille de flots de données

Daniel Aloise <daniel.aloise@polymtl.ca>

5 janvier 2022

Algorithmes de fouille de flots de données

- Les données ne sont pas nécessairement stockées pour toujours... ou même du tout !
- Dans certaines applications, il peut être avantageux de calculer des statistiques *on the fly*, au fur et à mesure que les données arrivent pour que nous puissions les jeter en suite.
- Dans un algorithme de fouille d'un flot de données, nous n'avons qu'une seule chance de voir chaque donnée.
- Alors, nous devons décider quoi faire avec chaque donnée à ce moment-là.

Caractéristiques des flots de données

- Entrée continue et rapide des données.
  - ⋆ Big Data : dimension vélocité
- Mémoire limitée pour stocker les données (moins que linéaire dans la taille d'entrée).
- Temps limité pour traiter chaque donnée.
- Accès séquentiel (pas d'accès aléatoire).
- Une seule chance (ou peut-être très peu de chances) de voir chaque donnée du flot.

Exemple

- Calcul de la moyenne d'un flot de nombres :

Exemple

- Calcul de la moyenne d'un flot de nombres :
  - Deux variables : *sum* qui accumule la somme de nombres à date, et n la quantité de nombres qu'on a vu jusqu'à présent.
  - Pour chaque nouveau nombre X_i, on ajoute cela à *sum* et on incrémente n.
  - On renvoie $\mu = sum/n$ à chaque fois que la moyenne est demandée.

Exemple

- Calcul de la variance d'un flot de nombres :

$$\sigma^2 = \frac{\sum_{i=1}^{n}(X_i - \mu)^2}{n}$$

Le problème est que nous ne stockons pas les X_i du flot...

Exemple

- Calcul de la variance d'un flot de nombres :

$$\sigma^2 = \frac{\sum_{i=1}^{n}(X_i - \mu)^2}{n}$$

Le problème est que nous ne stockons pas les X_i du flot...

- Tout n'est pas perdu, il existe une autre formule pour la variance :

$$\sigma^2 = \left(\frac{1}{n}\sum_{i=1}^{n} X_i^2\right) - \mu^2$$

- Problème résolu !

Algorithmes de fouille de flots de données

- De nombreuses quantités ne peuvent pas être calculées exactement selon le modèle de flots :
  - ex. calcul de la médiane
- Mais même si nous ne pouvons pas calculer exactement quelque chose, nous pouvons souvent trouver une assez bonne estimation.
- La qualité de l'estimation dépend de la quantité de mémoire que nous avons.

Fouille de flots de données

- On maintient un *sketch* (c.-à-d. une représentation) du flot de données pour répondre aux requêtes.
- On verra dans le cours trois exemples d'algorithmes pour la fouille de flots de données :
  1. échantillonnage de flots
  2. filtrage de flots
  3. comptage de bits

Échantillonnage de flots

- Puisque nous ne pouvons pas stocker tout un flot de données, une option consiste à stocker un échantillon de ses données
- Deux approches différentes :
  1. Conserver un échantillon aléatoire de taille fixe sur un flot potentiellement infini
  2. Échantillonner une proportion fixe de données dans le flot (disons 1 sur 10)
- À chaque unité de temps, t, nous gardons un échantillon aléatoire de données P.

Échantillon de taille fixe

- Supposons que nous devons maintenir un échantillon P de taille exactement p :
  - ex. contrainte imposée par la taille de la RAM

Échantillonnage de réservoir

- Stockez les premières p données du flot dans P.
- Supposez que nous avons vu $n-1$ données, et maintenant la n-ème donnée arrive ($n > p$) :
  - Avec probabilité p/n, gardez la n-ème donnée, sinon la rejetez
  - Si nous avons choisi la n-ème donnée, elle remplace au hasard l'une des données de P

Échantillon de taille fixe

- *Claim* : Cet algorithme maintient un échantillon P avec la propriété que chacune des n données vues jusqu'ici a la même probabilité p/n d'être dans l'échantillon.

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Échantillon de taille fixe

- *Claim* : Cet algorithme maintient un échantillon P avec la propriété que chacune des n données vues jusqu'ici a la même probabilité p/n d'être dans l'échantillon.

- Preuve par induction : Supposons qu'après n données, l'échantillon contienne chaque donnée vue jusqu'ici avec probabilité p/n.
  - Nous devons montrer qu'après avoir vu la donnée $n+1$, l'échantillon maintient la propriété.
  - c.-à-d., l'échantillon contient chaque donnée vue jusqu'ici avec probabilité $p/(n+1)$.

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Échantillon de taille fixe

- Case de base : Après avoir vu $n = p$ données l'échantillon P a la propriété désirée.

  Chaque donnée parmi $n = p$ est dans l'échantillon avec une probabilité $p/p = 1$

- Hypothèse inductive : Après avoir vu $n = p$ données l'échantillon P contient chaque donnée avec probabilité p/n
  - Maintenant, il arrive la donnée $n + 1$
  - Pour une donnée déjà dans P, la probabilité que l'algorithme la conserve dans P est :

$$\left(1 - \frac{p}{n+1}\right) + \left(\frac{p}{n+1}\right)\left(\frac{p-1}{p}\right) = \frac{n}{n+1}$$

Échantillon de taille fixe

- Conclusion :
  - À l'instant n, les données sont dans P avec prob. p/n (hypothèse inductive).
  - À l'instant $n+1$, la donnée demeure dans P avec prob. $n/(n+1)$.
  - Donc, prob. que la donnée soit dans P à l'instant $n+1$ est $\underbrace{\frac{p}{n}}_{\text{proba qu'elle soit dans échantillon}} \cdot \underbrace{\frac{n}{n+1}}_{\text{proba pr qu'elle reste dans échantillon}} = \frac{p}{n+1}$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Échantillonnage d'une proportion fixe

taille

- Scénario : flot de requêtes d'un moteur de recherche.
- Flot de tuples : $(user, query, time)$ quand requete a été faite
- On a de l'espace pour stocker $1/10$ du flot de requêtes.
- Solution simple :
  - Générez un entier aléatoire entre $[0..9]$ pour chaque requête
  - Stockez la requête si l'entier est 0, sinon rejeter

Échantillonnage d'une proportion fixe

- Question Google : quelle fraction des requêtes de l'utilisateur typique sont répétées au cours d'une semaine ?
- Supposons que chaque usager fait x requêtes une (1) fois et d requêtes deux (2) fois.

  x requestes uniques
  d requetes doublees
- Réponse correcte : $\frac{d}{(x+d)}$

EXAM

Problème avec approche simple

- L'échantillon va contenir $\frac{x}{10}$ des requêtes uniques.
- Seules $\frac{d}{100}$ des requêtes doublées seront aussi doublées dans l'échantillon :

$$\frac{d}{100} = \frac{1}{10} \cdot \frac{1}{10} \cdot d$$

- Des d requêtes originalement doublées $\frac{18d}{100}$ apparaîtront exactement une fois :

proba selectionner 1ere

proba ne pas selectionner 1ere

proba selectionner 1ere mais pas la 2e

EXAMEN: ÉCOUTER à 9min
1. Montrer pk l'algo ne permet pas de repondre a la question
2. Cmt echantilloner pr que l'analyse(réponse) soit correcte

$$\frac{18d}{100} = \left[\left(\frac{1}{10} \cdot \frac{9}{10}\right) + \left(\frac{9}{10} \cdot \frac{1}{10}\right)\right] \cdot d$$

Réponse diapo plus bas

- Alors la réponse (incorrecte) à la question de départ à partir de l'échantillon sera :

$$\frac{d/100}{x/10 + d/100 + 18d/100} = \frac{d}{10x + 19d} \ll \frac{d}{x + d}$$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Solution : échantillonner des requêtes par usager

Au lieu de prendre 1/10 des requetes, on prend 1/10 des usagers

- Prenez $\frac{1}{10}$ des usagers ainsi que toutes leurs requêtes pour l'échantillon. Toutes les requetes qui seront doublées pour ces usagers, elles seront doublées dans un échantillon
- Utilisez une fonction hash qui hache le id de l'usager de façon uniforme en 10 *buckets*.
  - rappel : (user, query, time)
- Flot de tuples avec des clés :
  - La clé est un sous-ensemble des composants de chaque tuple
  - ex. tuple est (usager, recherche, heure) et la clé est l'usager
  - Le choix de la clé dépend de l'application

Solution généralisée

- Pour obtenir un échantillon de taille $\frac{a}{b}$ du flot.
  - Hachez chaque clé uniformément en b *buckets*
  - Retenez le tuple si la valeur de hachage de sa clé est au plus a

    Si ca dépasse a, on ne le garde pas

Filtrage de flots de données

- Considérons que chaque élément du flot de données est un tuple.
- Input : liste de clés S
- But : Filtrer les tuples du flot selon S
- Applications :
  - Filtrage de *spam* si nous connaissons une liste des « bonnes »adresses mail. Si un mail provient de l'une d'elles, ce n'est pas un *spam*.
  - *Web crawler* : doit filtrer les URL déjà visitées.

Filtrage de flots de données

- On veut filtrer le flot et conserver uniquement les éléments intéressants :
  - Certains usagers, certaines catégories, certains attributs, etc.

- Solution triviale : table de hachage (*hash table*) qui stocke toutes les clés de S.

Filtrage de flots de données

- On veut filtrer le flot et conserver uniquement les éléments intéressants :
  - Certains usagers, certaines catégories, certains attributs, etc.
- Solution triviale : table de hachage (*hash table*) qui stocke toutes les clés de S.
- Que faire si nous n'avons pas assez de mémoire pour stocker toutes les clés de S dans une table de hachage ?

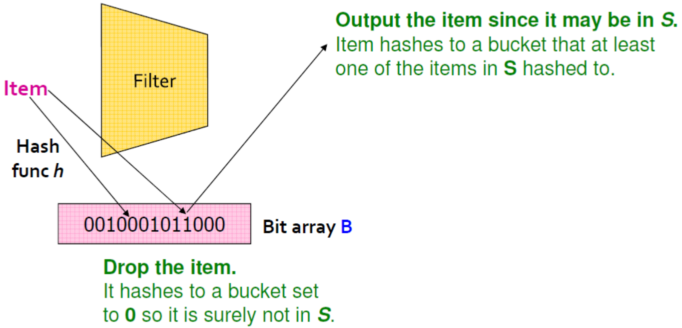
Première solution

Soit un ensemble de clés S que nous voulons filtrer :

1. Créez un tableau de bits B de n bits, initialement tous à 0
2. Choisissez une fonction de hachage h avec plage $[0, n)$
3. Hachez chaque $s \in S$ à l'un des n buckets, et mettez ce bit à 1, c'est-à-dire, $B[h(s)] = 1$
4. Hachez chaque élément a du flot et gardez seulement ceux dont les résultats de $B[h(a)] = 1$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Filtres de Bloom - exemple pratique



Source : http ://www.mmds.org/

- Si $a \in S$ alors $B[h(a)] = 1$ par déf.
- Si $a \notin S$ alors $B[h(a)] = 0$ ou 1

Cela crée des faux positifs, mais pas des faux négatives

Exemple

- $|S| = 1$ milliard d'adresses email.
- $|B| = 1$ Go $= 8$ milliards de bits.
- Si l'adresse e-mail est en S, alors il hache à un bucket dont le bit vaut 1 (pas de faux négatifs)
- Environ $1/8$ des bits sont mis à 1, donc environ $1/8$ des adresses qui ne sont pas en S sont erronément filtrées (faux positifs) 1/8 de faux positifs, car la fonction de hash elle hash les valeurs de maniere uniforme, la meme proba de tomber partout. 1/8 de chance pour tomber sur la valeur 1 et 7/8 à 0
- Alors, le probabilité de faux positifs est $\frac{S}{B}$ = nombre approx. de buckets dont le bit vaut 1 S: nb d'objets à filter B: mémoire à filtrer

Analyse de faux positifs

- Mais nous pouvons faire une analyse plus approfondie.

Analyse de faux positifs

- Mais nous pouvons faire une analyse plus approfondie.
- Si nous lançons $|S|$ fléchettes dans $|B|$ cibles également probables, quelle est la probabilité qu'une cible soit atteinte ?
- Dans notre cas :

cibles	buckets
fléchette	code hash $h(s)$ des différentes clés $s \in S$

Analyse de faux positifs

- Les faux positifs arrivent lorsqu'un bucket (cible) est associé au code hash d'une clé quelconque de S.
- Quelle est la probabilité qu'une cible reçoive au moins une fléchette ?
- Quelle est la probabilité qu'un bit de B soit 1 ?
- Probabilité qu'un bucket soit marqué par un élément de S : $1/|B|$ 1/B est la proba de le selectionner
- Probabilité qu'aucun des éléments de S marquent un bucket en particulier $(1 - 1/|B|)^{|S|}$

proba de chances de ne pas le selectionner = 1 - (1/B)

Analyse de faux positifs

- Les faux positifs arrivent lorsqu'un bucket (cible) est associé au code hash d'une clé quelconque de S.
- Quelle est la probabilité qu'une cible reçoive au moins une fléchette ?
- Quelle est la probabilité qu'un bit de B soit 1 ?
- Probabilité qu'un bucket soit marqué par un élément de S : $1/|B|$
- Probabilité qu'aucun des éléments de S marquent un bucket en particulier $(1 - 1/|B|)^{|S|}$
- On peut réécrire cela par $(1 - 1/|B|)^{\left(|B| \cdot \frac{|S|}{|B|}\right)} \approx e^{-|S|/|B|}$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Analyse de faux positifs

- Les faux positifs arrivent lorsqu'un bucket (cible) est associé au code hash d'une clé quelconque de S.
- Quelle est la probabilité qu'une cible reçoive au moins une fléchette ?
- Quelle est la probabilité qu'un bit de B soit 1 ?
- Probabilité qu'un bucket soit marqué par un élément de S : $1/|B|$
- Probabilité qu'aucun des éléments de S marquent un bucket en particulier $(1 - 1/|B|)^{|S|}$
- On peut réécrire cela par $(1 - 1/|B|)^{\left(|B| \cdot \frac{|S|}{|B|}\right)} \approx e^{-|S|/|B|}$
- Donc, la probabilité qu'un bucket soit marqué vaut $1 - e^{-|S|/|B|}$

Analyse de faux positifs

- Reprenons notre exemple : ($|S| = 10^9$ fléchettes , $|B| = 8 \cdot 10^9$ cibles)
- Notre estimation précédente de faux positifs était $1/8 = 0.125$
- La nouvelle est de $1 - e^{-1/8} = 0.1175$

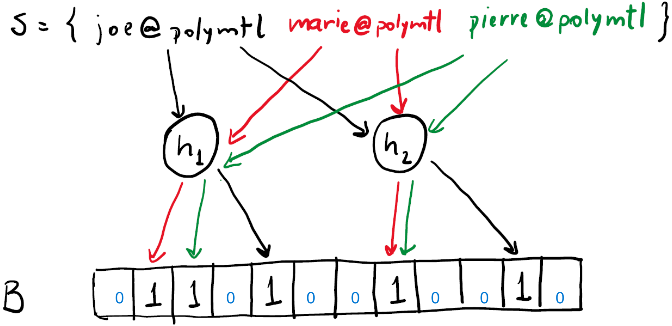
  = 11,75%

Filtre de Bloom

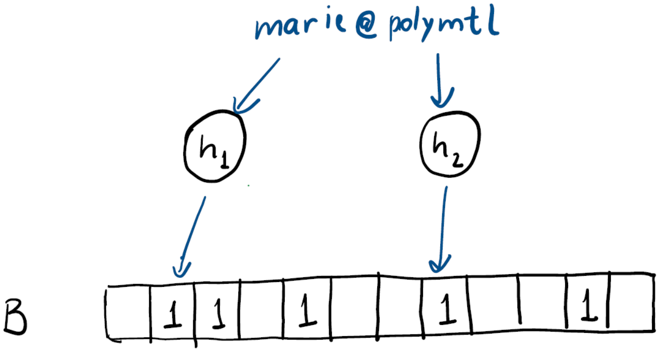
- Considérez : $|S| = m$, $|B| = n$
- Utilisez k fonctions hash $h_1, \ldots, h_k$
- Initialisation :
  - faite $B \leftarrow 0$
  - hachez chaque élément $s \in S$ en utilisant chaque fonction de hachage h_i, c.-à-d. faite $B[h_i(s)] = 1$ pour $i = 1, \ldots, k$
- Run-time :
  - Quand un élément ayant clé x arrive :
    - Si $B[h_i(x)] = 1$ pour tout $i = 1, \ldots, k$, alors déclarez $x \in S$
    - Sinon, jeter x

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

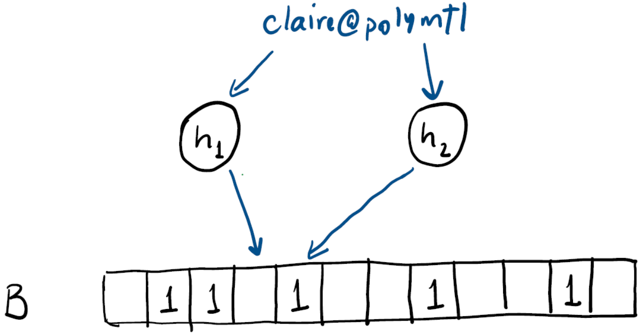
Exemple


S = { joe@polymtl marie@polymtl pierre@polymtl }
h1
h2
B
0 1 1 0 1 0 0 1 0 0 1 0

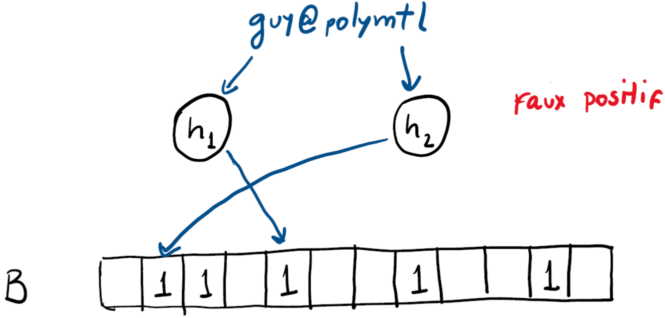
Exemple



Exemple



Exemple



POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Analyse de faux positifs

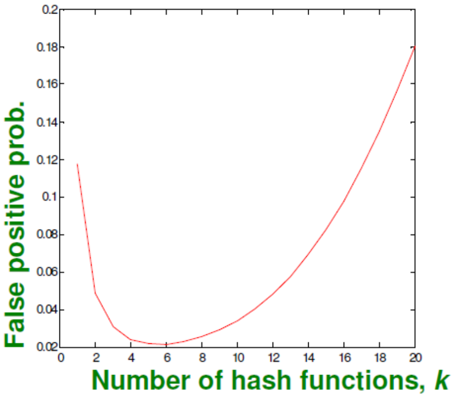
- Nous avons maintenant k fonctions de hachage.
- Quelle fraction du vecteur binaire B est mise à 1 ?
  - On lance $k \cdot m$ fléchettes sur n cibles
  - Donc la fraction de bits 1 est $(1 - e^{-km/n})$
- Nous filtrons l'élément x si $B[h_i(x)] = 1$ pour tout $i = 1, \ldots, k$.
- Les fonctions de hachage sont indépendantes.
- Alors, la probabilité d'un faux positif est $(1 - e^{-km/n})^k$.

Analyse de faux positifs

m: nb objets   n: nb bits en memoire

- $m = 1$ millard, $n = 8$ millards
  - $k = 1 : (1 - e^{-1/8}) = 0.1175$
  - $k = 2 : (1 - e^{-1/4})^2 = 0.0493$
- Qu'est-ce qui se passe si nous continuons à augmenter k ?
- Valeur optimale de $k = n/m \ln(2)$ nous donner valeur reelle, pas un entier
- Dans notre cas : $k = 8 \ln(2) = 5.54 \approx 6$
- Avec $k = 6 : 0.0235$

À NOTER FEUILLE NOTES valeur optimale



Source : http ://www.mmds.org/

Filtre de Bloom

- Les filtres Bloom garantissent l'absence de faux négatifs et utilisent une mémoire limitée :
  - Idéal pour le prétraitement avant des contrôles plus coûteux
- Approprié pour implémentation sur hardware :
  - Les calculs de fonction de hachage peuvent être parallélisés
- Est-il préférable d'avoir 1 gros B ou k petits B ?
  - C'est pareil : $(1 - e^{-km/n})^k$ contre $(1 - e^{-m/(n/k)})^k$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Comptage de bits

- Un modèle utile de traitement de flots est celui dont les requêtes sont faites sur une fenêtre de longueur N (les N plus récentes données).
- Cas intéressants :
  - N est si grand que les données ne peuvent pas être stockées en mémoire ou sur disque.
  - Il y a tellement de flots que leurs fenêtres ne peuvent pas être toutes stockées.

Applications

- On observe la vente de certains items sur Amazon :
  - Un flot par item
  - Un bit par transaction
  - 0 : transaction ne contient pas l'item ; 1 : contient l'item
- On étudie la fréquence de certains hashtags sur Twitter :
  - Un flot par hashtag
  - Un bit par tweet
  - 0 : tweet ne contient pas le hashtag ; 1 : contient le hashtag

Fenêtre glissante

passé à gauche

futur à droite (nouvelle valeur ajoutée à droite)

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

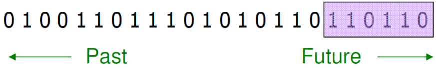
q w e r t y u i o p a s d f g h j k l z x c v b n m

← Past Future →

Source : http ://www.mmds.org

Comptage de bits

- Étant donné un flot de 0 et 1s.
- Soyez prêt à répondre aux questions telles que :
  - Combien y a-t-il de 1s dans les derniers k bits ? où $k \leq N$
  - Combien de fois avons-nous vendues X dans les k dernières ventes ?
- Solution évidente (impossible pour $k \approx N$) :
  - Stocker les k bits les plus récents - pas possible
  - Quand un nouveau bit arrive, ignorez le $k+1er$ bit



Source : http ://www.mmds.org

Comptage de bits

- Vous ne pouvez pas obtenir une réponse exacte sans stocker toute la fenêtre...

Comptage de bits

- Vous ne pouvez pas obtenir une réponse exacte sans stocker toute la fenêtre...
- Une approximation nous satisfait

Un essai : solution simple

- Q : Combien y a-t-il de 1 dans les derniers k bits ?
- Une solution simple qui ne résout pas vraiment notre problème : hypothèse d'uniformité.
- Maintenir deux compteurs :

  S : nombre de 1s depuis le début du flot
  Z : nombre de 0s depuis le début du flot

- Combien y a-t-il de 1s dans les derniers k bits ? $k \cdot \frac{S}{S+Z}$

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Un essai : solution simple

- Q : Combien y a-t-il de 1 dans les derniers k bits ?
- Une solution simple qui ne résout pas vraiment notre problème : hypothèse d'uniformité.
- Maintenir deux compteurs :

$$\begin{aligned} S &: \text{ nombre de 1s depuis le début du flot} \\ Z &: \text{ nombre de 0s depuis le début du flot} \end{aligned}$$

- Combien y a-t-il de 1s dans les derniers k bits ? $k \cdot \frac{S}{S+Z}$
- Comment faire si le flot n'est pas uniforme ?

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Un essai : solution simple

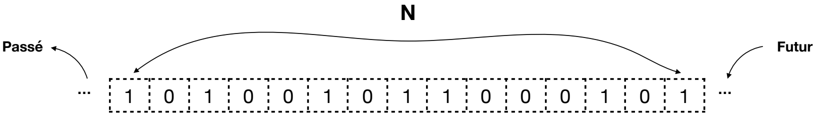
- Q : Combien y a-t-il de 1 dans les derniers k bits ?
- Une solution simple qui ne résout pas vraiment notre problème : hypothèse d'uniformité.
- Maintenir deux compteurs :

  S : nombre de 1s depuis le début du flot
  Z : nombre de 0s depuis le début du flot
- Combien y a-t-il de 1s dans les derniers k bits ? $k \cdot \frac{S}{S+Z}$
- Comment faire si le flot n'est pas uniforme ?
- Et si la distribution change avec le temps ?

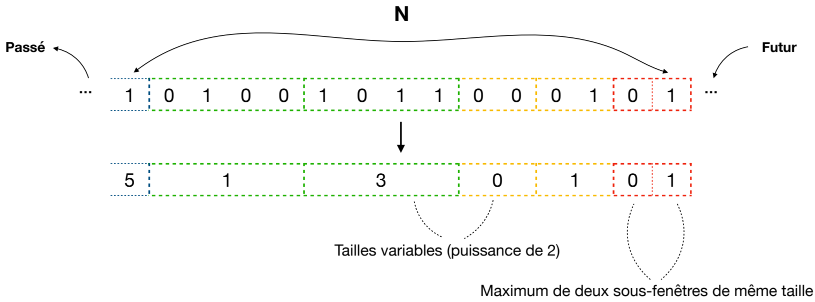
Comptage de bits

- On peut définir plusieurs sous-fenêtres glissantes sur le flot.
- Tailles variables : plus grandes dans le passé, plus petites dans le présent (puissances de 2).
- Les sous-fenêtres sont fusionnées au long du temps :
  - Maximum de deux sous-fenêtres de même taille.
  - Les sous-fenêtres ne peuvent pas se chevaucher.
- On garde pour chaque sous-fenêtre le nombre de 1 ainsi que sa taille.

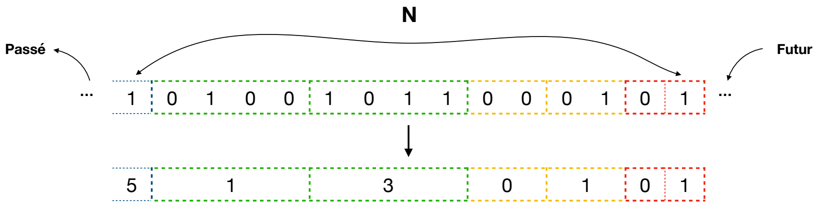
Exemple



Exemple


N
Passé
Futur
... 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 ...
5 1 3 0 1 0 1
Tailles variables (puissance de 2)
Maximum de deux sous-fenêtres de même taille

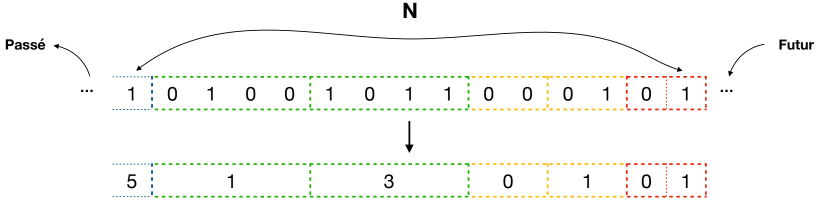
Exemple



Q: Combien y a-t-il de 1s dans les derniers $K \leq N$ bits?

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Exemple



Q: Combien y a-t-il de 1s dans les derniers $K \leq N$ bits?

$\hat{y}$ = nombre estimé de bits

y = nombre réels de bits

$$\text{Erreur} = \frac{\hat{y} - y}{y}$$

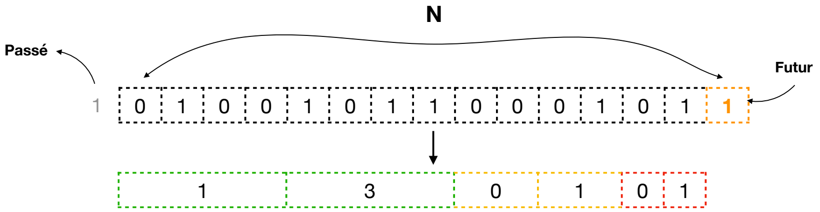
Exemple pour K = 9:

$$\hat{y} = 1 + 0 + 1 + 0 + (3/4)*3 = 4.25$$

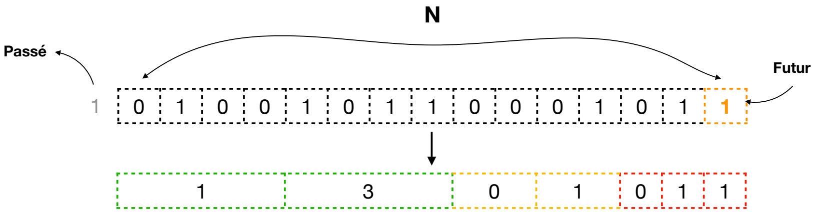
$$y = 4$$

$$\text{Erreur} = \frac{4.25 - 4}{4} = 0.0625$$

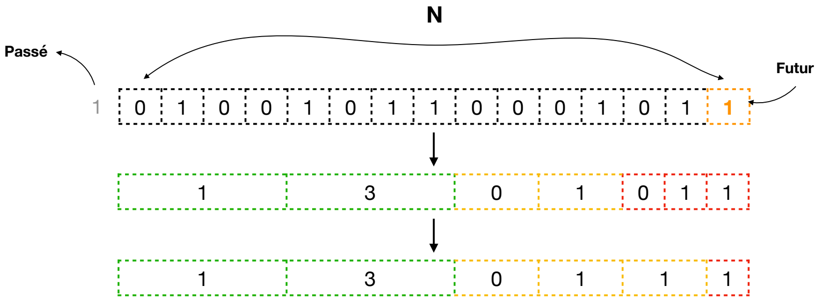
Exemple



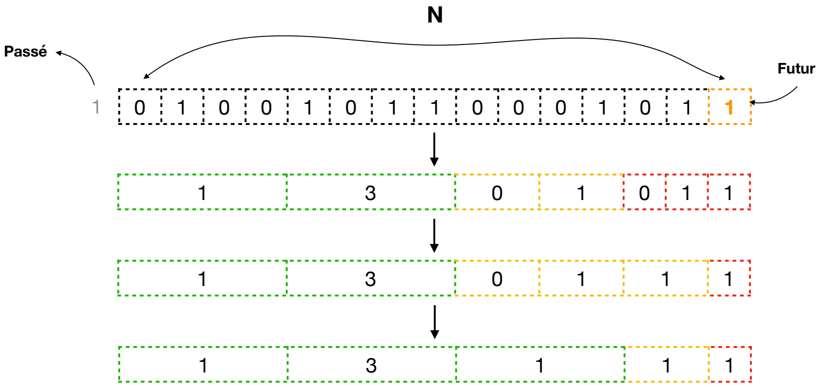
Exemple



Exemple

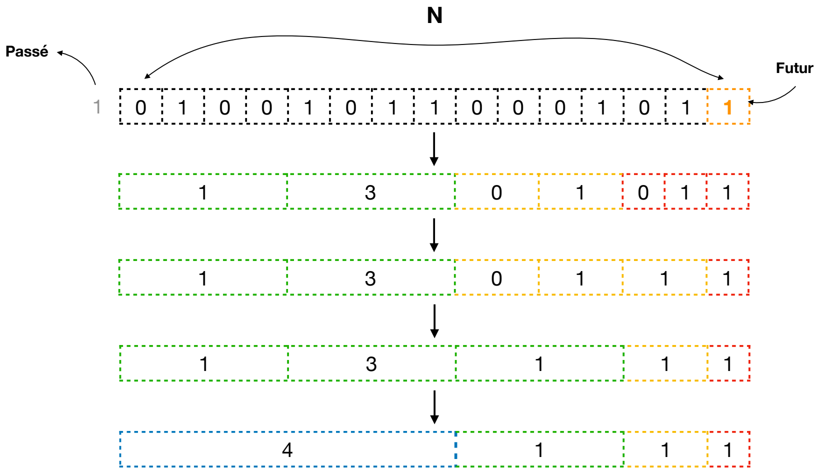


Exemple

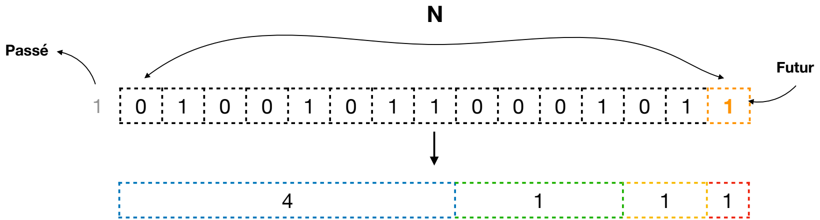


POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Exemple



Exemple



Exemple pour K = 9:

$$\hat{y} = 1 + 1 + 1 + (2/8)*4 = 4$$

$$y = 5$$

$$\text{Erreur} = \frac{4-5}{5} = -0.2$$

- 20%, on a sous-estimé le nb de 1

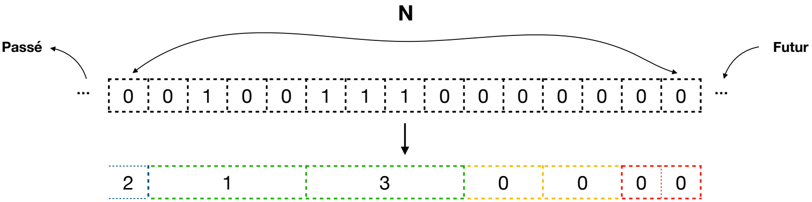
Est-ce qu'il y a une borne et cmb elle est?
L'erreur nest pas bornee et on peut avoir une valeur qui va a l'infini

Exercice

- Quel est le pire cas pour l'estimation calculée par cette méthode ?

Exercice

- Quel est le pire cas pour l'estimation calculée par cette méthode ?



Exemple pour K = 7:

$\hat{y} = 0 + 0 + (1/4)*3 = 0.75$

$y = 0$

erreur illimitée

Méthode DGIM

La méthode DGIM garantit 3 propriétés intéressantes :

1. Stockage de l'ordre de $O(\log^2 N)$ bits pour une fenêtre de taille N.
2. Temps de l'ordre de $O(\log N)$ pour traiter un nouveau bit.
3. Erreur relative inférieure ou égale à 50%.

Les N derniers bits sont divisés en *buckets* (sous-fenêtres). Chaque *bucket* contient :

- Le *timestamp* le plus récent modulo N (marque le début du *bucket*, la fin est implicite).
- Le nombre de 1 dans le *bucket* (appelé taille du *bucket*).

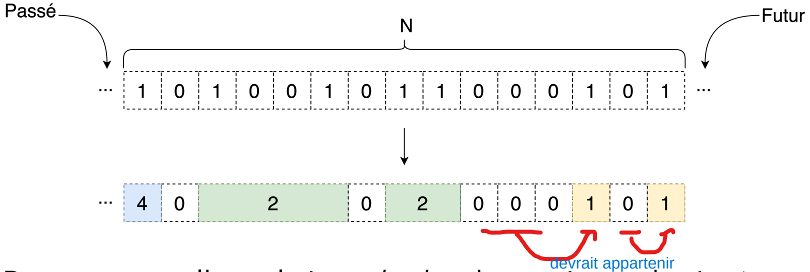
Méthode DGIM

Il y a 6 contraintes pour DGIM :

1. Le bit à droite (le plus récent) d'un *bucket* doit être un 1.
2. Tous les 1 doivent être dans un *bucket*.
3. Chaque bit est au plus dans un *bucket*.
4. Il y a au plus deux *buckets* de la même taille.
5. Toutes les tailles sont des puissances de 2.
6. La taille des *buckets* augmente vers la gauche (vers le passé)

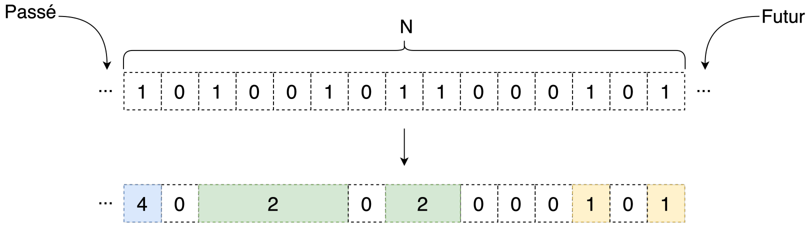
À chaque fois qu'on a plus que deux sous-fenêtres de même taille, on les fusionne.

Exemple



Remarquez que l'on colorie un *bucket* du premier au dernier 1. Cependant, puisque la fin est implicite, un *bucket* se termine quand le suivant commence. Cette représentation est utilisée dans le livre Mining of Massive Datasets.

Exemple



Exemple pour $K = 11$:

$\hat{y} = 1 + 1 + 2 + \frac{1}{2} * 2 = 5$

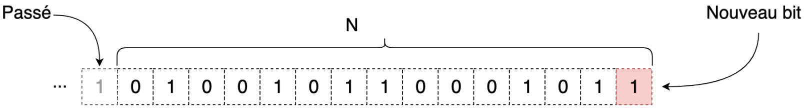
$y = 5$

Erreur $= \frac{5-5}{5} = 0$ donc il n'y a pas d'erreur

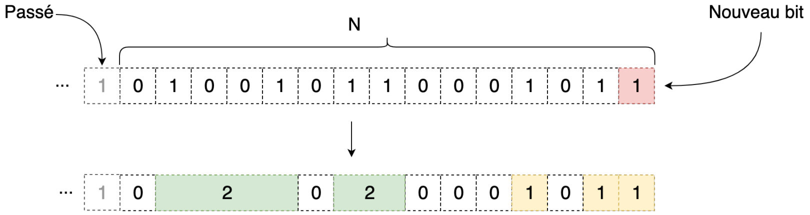
VOIR NOTES CAHIER
et enregistrement 1h13

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

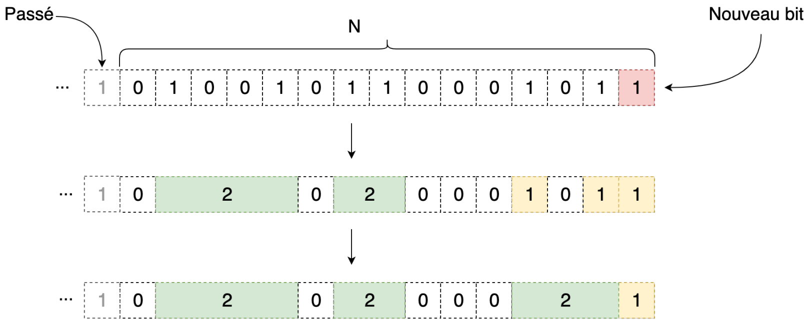
Exemple



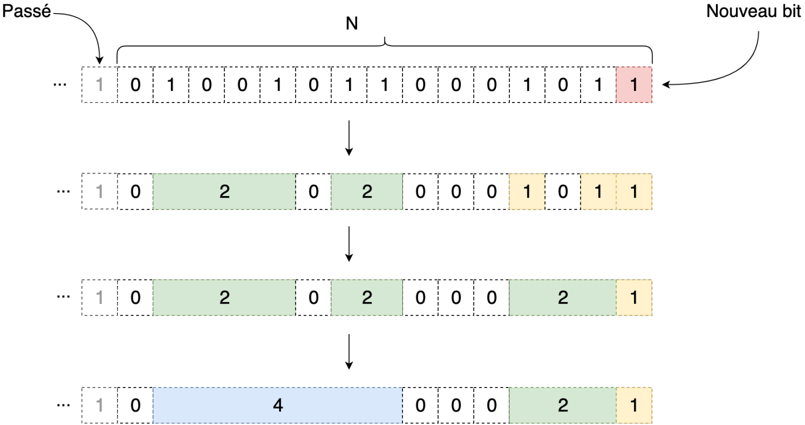
Exemple



Exemple


Passé
N
Nouveau bit
... 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1
... 1 0 2 0 2 0 0 0 1 0 1 1
... 1 0 2 0 2 0 0 0 2 1

Exemple


Passé
N
Nouveau bit
... 1 0 1 0 0 1 0 1 1 0 0 0 1 0 1 1
... 1 0 2 0 2 0 0 0 1 0 1 1
... 1 0 2 0 2 0 0 0 2 1
... 1 0 4 0 0 0 2 1

Méthode DGIM

- On ne connaît pas la distribution des 1 dans le *bucket* partielle.
- Si une requête inclut partiellement un *bucket*, l'estimation DGIM = somme 1 *bucket* entiers + moitié de 1 du *bucket* partielle.
- DGIM garantit un seuil maximum (*upper bound*) d'erreur sur l'estimation de 50% !

Méthode DGIM

Preuve :

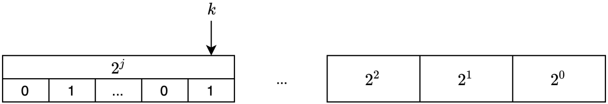
- Supposons que l'estimation nécessite partiellement un *bucket* b de taille 2^j.
- Notons la valeur réelle y et l'estimation $\hat{y}$.
- Il y a deux cas : l'estimation est supérieure ou inférieure à la valeur réelle.

Méthode DGIM

On surestime la valeur reelle

1) Estimation supérieure à la valeur réelle

- Dans le pire des cas, seul le bit le plus à droite du *bucket* b est considéré et il n'y a qu'un *bucket* de chaque taille plus petite que celle de b



- La valeur réelle $y = 2^0 + 2^1 + \cdots + 2^{j-1} + 1 = 2^j$
- L'estimation est $\hat{y} = 2^0 + 2^1 + \cdots + 2^{j-1} + \frac{1}{2}2^j = 2^j + 2^{j-1} - 1$
- L'erreur relative est $\frac{\hat{y}-y}{y} = \frac{2^{j-1}-1}{2^j} \leq 50\%$

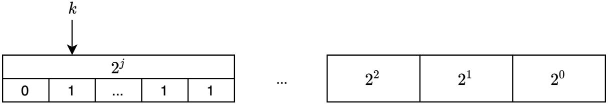
POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE

Méthode DGIM

on sous-estime la valeur reelle

2) Estimation inférieure à la valeur réelle

- Dans le pire cas, tous les 1 de b sont dans la partie du *bucket* considéré



- L'estimation du *bucket* partiel est $\frac{1}{2} \times 2^j = 2^{j-1}$, alors que la valeur réelle du *bucket* est 2^j
- Puisqu'il y a au moins un *bucket* de chaque taille, la valeur réelle est supérieure à $2^0 + 2^1 + \cdots + 2^{j-1} + 2^j = 2^{j+1} - 1$
- L'erreur relative est donc inférieure à 50%

POLYTECHNIQUE MONTRÉAL
LE GÉNIE EN PREMIÈRE CLASSE