

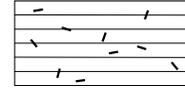
Chapitre 10 Algorithmes probabilistes

- Jouent à pile ou face
- Se comportent différemment lorsque exécutés deux fois sur un même exemplaire
- Défient parfois l'intuition

1

Première surprise: le hasard peut être utile!

- Créer le genre humain (???)
- Estimer π
- Cryptographie
- Vérifier la primalité
- Accélérer une recherche
- Réduire l'effet de mauvais exemplaires



2

Deuxième surprise: le hasard peut être précis

Exemple: pile = succès face = échec

Prob[succès] = $\frac{1}{2}$

Prob[succès après 2 essais] = $\frac{3}{4}$

...

Prob[succès après n essais] = $1 - (\frac{1}{2})^n$

$n=1000 \Rightarrow$ échec moins probable qu'une erreur interne de l'ordinateur après une seconde de calcul!

3

3 types d'algos probabilistes

- Numérique: retourne une solution approximative à un problème numérique (ex: la simulation). Plus de temps \Rightarrow plus grande précision.
- Monte Carlo: retourne toujours réponse (ex: oui ou non) mais peut se tromper. Plus de temps \Rightarrow plus grande probabilité que la réponse soit bonne. En général, on **ne peut pas** déterminer efficacement si réponse obtenue bonne.

4

- Las Vegas: ne retourne jamais une réponse inexacte, mais parfois ne trouve pas de réponse du tout! Plus de temps \Rightarrow plus grande probabilité de succès sur **chaque** exemplaire. **ne s'agit pas** seulement d'être malchanceux sur quelques exemplaires catastrophiques. **||**

5

Quand Colomb a découvert l'Amérique?

- Numérique: Au 15^{ème} siècle, entre 1485 et 1495, entre 1493 et 1494, ...
- Monte Carlo: 1492, 1501, 1492, 1492, 1487, 555, 1487, ... , ...
- Las Vegas: 1492, 1492, sais pas, 1492, sais pas, 1492, ... , ...

6

10.3 Temps espéré \neq temps moyen

- $\text{Moyen}(n) = \frac{\sum_{|w|=n} \text{temps sur exemplaire } w}{\#\{w : |w| = n\}}$

- Espéré: défini sur chaque exemplaire w

$$\text{Espoir}(w) = \sum_{\text{choix de l'algo}} \text{temps}(w \text{ avec } \text{choix}) \times \Pr[\text{choix}]$$

- $\text{Espéré}(n) = \text{Max}_{|w|=n} \{\text{Espoir}(w)\}$

7

10.4 Génération pseudo-aléatoire

Ex: $p = \overbrace{\hspace{10em}}^{\leftarrow 100 \text{ chiffres } \rightarrow}$ premier $\equiv 3 \pmod{4}$

$q = \overbrace{\hspace{10em}}$ premier $\equiv 3 \pmod{4}$

$0 < s < pq$ et $\text{pgcd}(s, pq) = 1$. Alors:

$s \leftarrow (s \times s) \pmod{pq}$
 imprimer parité(s)

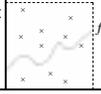
fournit une suite de bits presque toujours indistinguable d'une suite de bits aléatoire.

8

10.5 Algos numériques

On veut estimer

$$\int_0^1 f(x) dx$$

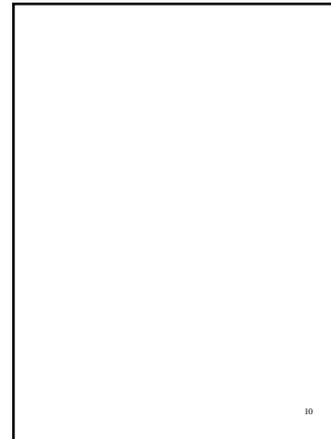


```

c ← 0
pour i ← 1 jusqu'à n faire
  x ← uniforme(0, 1)
  y ← uniforme(0, 1)
  si y ≤ f(x) alors c ← c + 1
retourner c/n
    
```

100 fois plus de points \Rightarrow erreur espérée 10 fois plus petite, c.à.d. un chiffre de précision de plus.

9



10

Pour la même précision, un algo déterministe utilisera d points, pour un $d \ll n$.

Mais en s dimensions, pour avoir la même précision avec l'algorithme probabiliste, il faut toujours n points. Par contre, l'algo déterministe aura probablement besoin de d points.

\Rightarrow quand s est plus grand que 2 ou 3, l'algo probabiliste l'emporte.

11

10.6 Algos Monte Carlo

- peut se tromper, mais
- trouve une solution correcte avec bonne probabilité, **quel que soit** l'exemplaire à traiter.

Aucun avertissement en cas d'erreur

p -correct: retourne une solution correcte avec probabilité p ou plus, $0 < p < 1$.

peut être **biaisé**, i.e. posséder la propriété de ne jamais se tromper sur certaines de ses réponses.

12

Exemple de Monte Carlo: multiplication de matrices

Pour vérifier si

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 4 & 3 \end{bmatrix} \times \begin{bmatrix} 1 & 4 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 13 & 11 & 22 \\ 31 & 32 & 58 \\ 23 & 31 & 50 \end{bmatrix}$$

on génère un vecteur X d'éléments 0 et 1, et on vérifie si

$$[] \times [] \times X = [] \times X.$$

Exemple: $X = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$

13

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 4 & 3 \end{bmatrix} \begin{bmatrix} 1 & 4 & 5 \\ 3 & 2 & 4 \\ 2 & 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} 13 & 11 & 22 \\ 31 & 32 & 58 \\ 23 & 31 & 50 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 13 & 11 & 22 \\ 31 & 32 & 58 \\ 23 & 31 & 50 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 5 & 4 & 3 \end{bmatrix} \begin{bmatrix} 6 \\ 7 \\ 5 \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} 35 \\ 89 \\ 73 \end{bmatrix}$$

L'algorithme retourne **vrai**.

14

fonction *Freivalds*(A, B, C, n)
vecteur $X[1..n]$
pour $i \leftarrow 1$ **à** n **faire** $X[i] \leftarrow 0$ **ou** 1
si $A \times (B \times X) = C \times X$
alors retourner vrai
sinon retourner faux

Remarque:
Freivalds(A, B, C, n) faux \Rightarrow
 $A \times B \neq C$ hors de tout doute
mais...

Freivalds(A, B, C, n) vrai \Rightarrow ?
 $\text{Prob}[A \times B = C] \geq \frac{1}{2}$?

NON!

15

La **bonne** façon:

$A \times B = C \Rightarrow$
 $\text{Prob}\{\text{Freivalds}(A, B, C, n) \text{ vrai}\} = 1$

$A \times B \neq C \Rightarrow$
 $\text{Prob}\{\text{Freivalds}(A, B, C, n) \text{ faux}\} \geq \frac{1}{2}$
bonne réponse

Pourquoi cette dernière inégalité?

Donc $\text{Prob}\{\text{bonne réponse}\} \geq \frac{1}{2}$
quel que soit l'exemplaire.

Temps dans $O(n^2)$ mais pas très impressionnant!

16

Amplification de l'avantage:
fonction *amplif*(A, B, C, n, k)
pour $i \leftarrow 1$ **à** k **faire**
si *Freivalds*(A, B, C, n) faux
alors retourner faux
retourner vrai

$A \times B = C \Rightarrow$
 $\text{Prob}\{\text{amplif}(A, B, C, n, k) \text{ vrai}\} = 1$

$A \times B \neq C \Rightarrow$
 $\text{Prob}\{\text{amplif}(A, B, C, n, k) \text{ faux}\} \geq 1 - 2^{-k}$

Donc: $\text{Prob}\{\text{erreur}\} \leq 2^{-k}$ quel que soit l'exemplaire!

N.B.: $2^{-300} \approx 10^{-100}$.

17

$A \times B \neq C \Rightarrow$
 $\text{Prob}\{\text{Freivalds}(A, B, C, n) \text{ faux}\} \geq \frac{1}{2}$?

$A \times B \neq C \Rightarrow$
 \exists colonne i de $(AB - C)$ non nulle.

Pour chaque choix de vecteur X hormis l'entrée i , on aura:

- $(AB - C)(\ll X \text{ sans } i \gg) = 0 \Rightarrow$
 $\text{Prob}\{(AB - C) X \neq 0\} = \text{Prob}\{X_i \neq 1\} = \frac{1}{2}$
- $(AB - C)(\ll X \text{ sans } i \gg) \neq 0 \Rightarrow$
 $\text{Prob}\{(AB - C) X \neq 0\} \geq \text{Prob}\{X_i \neq 0\} = \frac{1}{2}$

18

Test probabiliste de primalité: essai 1

fonction *premier1*(n)
 $d \leftarrow$ uniforme($2.. \lfloor \sqrt{n} \rfloor$)
si ($n \equiv 0 \text{ mod } d$) **retourner faux**
sinon retourner vrai

faux-biaisé: bien!

mais...

retourne presque toujours vrai, premier ou pas, donc peu utile!

19

Théorème de Fermat

p premier et $1 \leq a \leq p-1$
 $\Rightarrow a^{p-1} \equiv 1 \text{ mod } p$

Contraposée:
Soient n entier et $1 \leq a \leq n-1$;
 $\neg(a^{n-1} \equiv 1 \text{ mod } n) \Rightarrow n$ composé.

fonction *premier2*(n)
 $a \leftarrow$ uniforme($2.. n-1$)
si ($a^{n-1} \equiv 1 \text{ mod } n$) **retourner vrai**
sinon retourner faux

Toujours faux-biaisé: bien!
Toujours pas p-correct car trop de faux témoins de primalité.

20

Test de Miller-Rabin

fonction *premier3*(n impair > 4)
 $a \leftarrow$ uniforme ($2 \dots n-1$)
retourner vrai ssi $a \in B(n)$
 où
 $B(n) = \{ 2 \leq a \leq n : n-1 = 2^s t \text{ et } a^t \bmod n = 1 \}$
 ou
 $\exists i, 0 \leq i < s, a^{2^i t} \bmod n = n-1 \}$

Théorème 10.6.2:

- n premier $\Rightarrow \Pr[\text{vrai}] = 1$
- n composé $\Rightarrow \Pr[\text{vrai}] < 1/4$

\Rightarrow faux-biaisé et $3/4$ -correct!! 21

Amplifier l'avantage

sans biais: vitesse tortue

Considérons d'abord:

fonction *ridicule*(x)

si pile alors retourner vrai
 sinon retourner faux

résoud n'importe quoi...tout en étant $1/2$ -correct...pourtant on ne gagne rien à répéter cet algo.

\Rightarrow observation uno: sans biais, impossible d'amplifier à moins d'être p -correct pour un $p > 1/2$.

Et si $p > 1/2$?

22

Amplifier l'avantage

sans biais: vitesse tortue

Algo p -correct, $p > 1/2$, pour un problème de décision: que faire?

Répéter et prendre la majorité!

Ex: $p = 3/4$, voyons 3 répétitions

$\Pr[\text{err, err, err}] = 1/4 \times 1/4 \times 1/4$

$\Pr[\text{OK, err, err}] = 3/4 \times 1/4 \times 1/4$

$\Pr[\text{err, OK, err}] = 1/4 \times 3/4 \times 1/4$

$\Pr[\text{err, err, OK}] = 1/4 \times 1/4 \times 3/4$

$\Pr[\text{majorité d'erreurs}] = \frac{10}{64} = 16\%$

\Rightarrow avantage passé de 75% à 84% 23

Amplifier l'avantage

sans biais: vitesse tortue

Général: algo $(1/2 + \epsilon)$ -correct

Prendre majorité de k répétitions

$$\Pr[i \text{ OK}] \geq \binom{k}{i} (1/2 + \epsilon)^i (1/2 - \epsilon)^{k-i}$$

$$\Pr[\text{majorité d'erreurs}] \leq \sum_{i=0}^{k/2} \binom{k}{i} (1/2 + \epsilon)^i (1/2 - \epsilon)^{k-i}$$

Statistiques: 95% requiert

$$k > 2.706 \left(\frac{1}{4\epsilon^2} - 1 \right)$$

$k=269$ pour passer de 55% à 95%

Mais 99.5% requiert seulement

$$k > 6.636 \left(\frac{1}{4\epsilon^2} - 1 \right)$$

24

Amplifier l'avantage

avec biais: vitesse lapin

Algo p -correct, disons faux-biaisé.

Que faire?

Déjà vu! Répéter et répondre vrai à moins d'un seul faux.

Ex: $p = 3/4$, voyons 3 répétitions.

Seule possibilité d'erreur est de répondre vrai sur un exemplaire

faux, et pour un tel exemplaire:

$\Pr[\text{err, err, err}] = 1/4 \times 1/4 \times 1/4$

$\Pr[\text{erreur après 3}] = \frac{1}{64} = 2\%$

\Rightarrow avantage passé de 75% à 98% 25

Amplifier l'avantage

avec biais: vitesse lapin

Algo p -correct, faux-biaisé.

Mieux! Même pas nécessaire que p soit plus grand que $1/2$.

Ex: $p = 0.01$

$$\Pr[\text{erreur après } k] = \left(\frac{99}{100} \right)^k$$

$\Rightarrow k=10$ passe de 1% à $\approx 10\%$

$k=30$ passe de 1% à $\approx 25\%$

$k=150$ passe de 1% à $\approx 75\%$

Ces valeurs de k seraient 10 fois moindres en partant de $p = 0.1$ 26

Algos de Las Vegas

- Las Vegas de type 1: utilise le hasard pour guider ses choix et arriver infailliblement à résoudre le problème demandé. Mauvais choix \Rightarrow plus long. Ex: tri, hashage.
- Las Vegas de type 2: utilise le hasard pour tenter de résoudre un problème, quitte à échouer. Mauvais choix \Rightarrow échec (avoué). Ex: 8 reines, factorisation.

27

Las Vegas de type 1

Rappel: sélection et médiane en temps $\Theta(n)$, en pire cas. L'idée était de partitionner sur la base d'une médiane approximée.

Revenons au choix du « pivot »: on peut démontrer que le choix trivial donne un algo ayant

- temps en pire cas dans $\Theta(n^2)$
- temps en moyenne dans $\Theta(n)$.

Un algorithme de Las Vegas de type 1 consiste à choisir le pivot au hasard...et alors?

28

Sélection et médiane par algo de Las Vegas

- temps espéré $\Theta(n)$, précisément égal à ce qu'était le temps en moyenne de l'algo à choix trivial
- en jouant de malchance, peut prendre autant de temps que le pire cas de l'algo à choix trivial
- peut même prendre ce pire temps sur un exemplaire qui aurait été facile pour l'algo à choix trivial!
- le grand avantage? Il n'existe plus de mauvais exemplaire!!!

29

Las Vegas de type 1

Particulièrement utile quand un algo déterministe existe, qui est

- bon en moyenne
- mauvais en pire cas

Alors un algo de Las Vegas pourra

- éliminer les exemplaires pire cas
- uniformiser les exemplaires
- maintenir un bon temps espéré

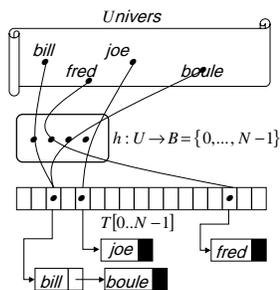
Autre exemple: quicksort

- $\Theta(n \log n)$ en moyenne
- quadratique en pire cas
- devient temps espéré $\Theta(n \log n)$

30

Las Vegas de type 1

Ex: adressage dispersé universel



31

$m = \#\{bill, fred, joe, boule, \dots\}$

$\beta = \frac{m}{N}$ facteur de chargement

Avec fonction de hashage fixée (ex: compilateur),

- β = longueur moyenne de chacune des N listes à la fin du hashage
- temps moyen requis pour placer les m identificateurs $\in O(m \times (1 + \beta))$
 \Rightarrow linéaire puisque $\beta \leq 1$
- temps pire cas $\in \Omega(m^2)$ (lorsqu'une seule liste à la fin)

32

Idee Las Vegas de type 1:
 Choisir $h : U \rightarrow B = \{0, \dots, N-1\}$
 une fonction aléatoire.
 Alors
 temps **espéré** $\in O(m \times (1 + \beta))$
 quelle que soit la séquence!

Problème: il y a $\# \{h : U \rightarrow B\} = N^{\#U}$
 fonctions h différentes. Le choix
 aléatoire d'une telle fonction
 exigerait de spécifier $(\#U) \times \lg N$
 bits. Il en coûterait donc $m \lg N$
 pour placer **chaque** identificateur!

Que faire?

33

Définition: l'ensemble de fonctions
 $H \subseteq \{h : U \rightarrow B\}$
 est une classe universelle2 si
 $(\forall x \neq y \in U)$

$$\Pr_{h \in H} [h(x) = h(y)] \leq \frac{1}{N}$$

\Rightarrow quels que soient les deux
 identificateurs x et y (distincts),
 la probabilité de collision est faible.

Résultat: Quelle que soit la
 séquence de m identificateurs, le
 temps **espéré** pour traiter cette
 séquence $\in O(m \times (1 + \beta))$.
 \Rightarrow linéaire puisque β constant!

34

Magie: de telles fonctions existent
 (et sont calculables efficacement).

Exemple: $U = \{0, \dots, m-1\}$
 $B = \{0, \dots, N-1\}$
 p premier $\geq m$

Alors
 $H = \{h_{ij} : U \rightarrow B \mid 1 \leq i, j < p\}$
 où
 $h_{ij}(x) = ((ix + j) \bmod p) \bmod N$
 est une classe universelle2.

\Rightarrow facile de tirer aléatoirement
 une fonction $h \in H$, et
 \Rightarrow facile de calculer $h(x)$.

35

Las Vegas de type 2

Rappel: un tel algo peut échouer,
 mais détecte alors son échec.

procédure $LV(x, \text{var } y, \text{var succès})$

Alors retour,
 succès *vrai* $\Rightarrow y$ est solution à x
 succès *faux* \Rightarrow pas de chance

$p(x)$ = probabilité de succès, et
 $(\forall \text{exemplaire } x)[p(x) > 0]$.

Mieux si possible:
 $(\exists \delta > 0)(\forall \text{exemplaire } x)[p(x) > \delta]$.

36

Répétition non bornée d'un algo
 de Las Vegas de type 2:

fonction obstiné(x)
répéter
 $LV(x, y, succès)$
jusqu'à succès
retourner y

Réponse: toujours correcte
 toujours obtenue...

...un de ces jours!

37

...mais quand au juste?

Soient
 p : probabilité de succès de LV
 s : temps espéré de LV en cas
 de succès
 e : temps espéré de LV en cas
 d'échec
 t : temps espéré de obstiné

Alors
 $t = ps + (1-p)(e+t)$
 d'où
 $t = s + \frac{(1-p)}{p}e$

Bref: $s \downarrow, e \downarrow, p \uparrow \Rightarrow t \downarrow$

38

Las Vegas de type 2

Ex: placement des huit reines

*Rappel: exploration du graphe des vecteurs k -prometteurs à l'aide d'un algorithme à retour arrière.
 $k=8 \Rightarrow 114$ explorés parmi 2057.*

Observation: les positions des reines qui résolvent le problème ont l'air plutôt arbitraire.

Approche Las Vegas « vorace »: parmi les positions qui restent, choisir au hasard, et abdiquer si un cul de sac est atteint.

39

Placement Las Vegas des 8 reines

Avantages:

• *conceptuellement plus simple*

• *plus rapide en principe*

$p: Pr[\text{succès}] = 0.1293$ (= # de

solutions/# total, ordinateur)

s : temps espéré en cas de succès

= coût de générer 9 vecteurs

e : temps espéré en cas d'échec

= 6.971 vecteurs (ordinateur)

t : temps espéré de l'algo

= $s + \frac{(1-p)}{p} e = 55.93$

(comparer aux 114 vecteurs engendrés par retour arrière!)

40

Placement Las Vegas des 8 reines

En fait: dans le cas de 8 reines, le coût de la génération des nombres aléatoires annule le gain en nombre de vecteurs générés.

Faire mieux?

Oui! En ajustant s , e et p .

Idée: générer les k premières reines aléatoirement, et les $8-k$ dernières par retour arrière.

$k=2$: trois fois plus rapide que retour arrière

$k=3$: seulement 2 fois plus rapide, même si moins de vecteurs

41

Placement Las Vegas de n reines

L'avantage de Las Vegas sur le retour arrière croît énormément lorsque n augmente.

Ex: $n=39$

• *pur retour arrière: 10^{10} vecteurs*

explorés avant première solution

• *pur Las Vegas: un million de fois*

plus rapide en implantation réelle!

• *hybride avec $k=29$: 2 millions de*

fois plus rapide en implantation,

et 20 millions moins de vecteurs.

Ex: $n=1000$

• *bonne idée de choisir $k=983$:-)*

42

Las Vegas de type 2

Ex: factorisation de grands entiers

- *problème important*
- *aucun algo efficace connu (en fait, la crypto actuelle repose sur sa difficulté!)*
- *ne semble pourtant pas NP-complet*
- *choix aléatoires mêlés à une stratégie judicieuse et à des estimés sophistiqués tirés de la théorie des nombres permettent parfois de réussir.*

43