Chapitre 2

Algorithmes probabilistes

Hugo Salou MPI*

Dernière mise à jour le 31 octobre 2022

1 Introduction

Définition: Un algorithme *déterministe* est un algorithme tel que pour chaque entrée I de l'algorithme, l'exécution de l'algorithme produit toujours exactement la même suite d'états.

Remarque:
Un algorithme déterministe produit donc toujours les même sorties sur les mêmes entrées.

Définition: Un algorithme *probabiliste* est un algorithme opérant sur un ensemble $\mathscr{E}$, tel que la suite d'états obtenus par exécution de l'algorithme sur une entrée $e \in \mathscr{E}$ est une variable aléatoire.

Remarque:
Avec cette définition, un algorithme déterministe est un algorithme probabiliste.

Définition (Algorithme de type Las Vegas)**:** Étant donné un problème P, un algorithme probabiliste répondant au problème P est dit de type Las Vegas dès lors que, s'il se termine, c'est en donnant une réponse correcte.

Définition (Algorithme de type Monte-Carlo)**:** Étant donné un problème P, un algorithme probabiliste répondant au problème P est dit de type Monte-Carlo dès lors que son temps d'exécution dépend uniquement de son entrée. L'algorithme peut cependant répondre de manière erronée au problème P avec une « certaine » probabilité.

Remarque:
Dans le cas d'un problème de décision (la réponse de l'algorithme est oui ou non), un algorithme de type Monte-Carlo est dit
- « à erreur unilatérale » s'il existe une des réponses (oui ou non) r telle que, si l'algorithme répond r, alors il a raison (r est la réponse au problème) ;
- « à erreur bilatérale » si pour chaque réponse l'algorithme se trompe avec une probabilité non nulle.

2 Algorithme de Monte-Carlo

On considère le problème : « étant donné trois matrices A, B, C de $\mathscr{M}_n(\mathbb{Z}/2\mathbb{Z})$, a-t-on $A \cdot B = C$? »

Un algorithme trivial serait de calculer $A \cdot B$ et on vérifie, point à point, que $A \cdot B = C$. La complexité cet algorithme est en $\Theta(n^3)$ à cause du produit matriciel.

Un algorithme de Monte-Carlo serait le suivant.

0. à faire

Algorithme 1 Algorithme de Monte-Carlo répondant au problème

Entrée A, B, C trois matrices et $k \in \mathbb{N}$

pour $j \in [\![1, k]\!]$ **faire**
  $r \leftarrow \mathcal{U}\big((\mathbb{Z}/2\mathbb{Z})^n\big)$ ▷ n
  $r_1 \leftarrow B \cdot r$ ▷ n^2
  $r_2 \leftarrow A \cdot r_1$ ▷ n^2
  $r_3 \leftarrow C \cdot r$ ▷ n^2
  si $r_3 \neq r_2$ **alors**
    retourner Non
retourner Oui

Dans le pire cas, la complexité est en $k \times n^2$. On cherche la probabilité d'erreur de cet algorithme. Pour cela, on utilise le lemme suivant.

Lemme: Si $D \neq 0$, et $r \sim \mathcal{U}\big((\mathbb{Z}/2\mathbb{Z})^n\big)$, alors $P(D \cdot r = 0) \leqslant \frac{1}{2}$.

■

D'où, l'algorithme ci-dessous est tel que sa probabilité d'échec est de $\frac{1}{2^k}$. Or, l'algorithme a une complexité de $\mathcal{O}(k\,n^2)$.

3 Algorithme de type Las-Vegas

On étudie le tri rapide. On considère les fonctions "Partitionner," puis "Tri Rapide."

Algorithme 2 Fonction "Partitionner" utilisée dans le tri rapide

Entrée T le tableau à trier, g, d et p trois entiers (bornes du tableau)
Sortie un entier J et le sous-tableau $T[g..d]$ est modifié en $\bar{T}$ de sorte que $\bar{T}$[1] $[J] = \underline{T}[p]$, et $\forall i \in [\![g, J-1]\!]$, $\bar{T}[i] \leqslant \bar{T}[J]$, et $\forall i \in [\![J+1, d]\!]$, $\bar{T}[i] \geqslant \bar{T}[J]$, et $\forall i \in [\![0, n-1]\!] \setminus [\![g, d]\!]$, $\bar{T}[i] = \underline{T}[i]$, et $\bar{T}$ est une permutation de $\underline{T}$.

Échanger(T, g, d)
$J \leftarrow g$
$I \leftarrow g$
tant que $I < d$ **faire**
  si $T[I] > T[d]$ **alors** ▷ *Cas* "$T[I] >$ pivot"
    $I \leftarrow I + 1$
  sinon ▷ *Cas* "$T[I] \leqslant$ pivot"
    Échanger(T, I, J)
    $J \leftarrow J + 1$
    $I \leftarrow I + 1$
Échanger(T, J, d)
retourner J

Remarque:
On admet que $\bar{T}$ est une permutation de $\underline{T}$. On admet également que, $\forall i \in [\![0, n-1]\!] \setminus [\![g, d]\!]$, $\bar{T}[i] = \underline{T}[i]$.

Lemme: "Partitionner" est correct.

■

1. La notation $\bar{T}$ représente le tableau T après l'algorithme, et la notation $\underline{T}$ représente le tableau T avant l'algorithme.

Algorithme 3 Tri rapide

Entrée T un tableau, g et d les bornes de ce tableau

si $d > g$ **alors**
  $p \leftarrow \textsc{ChoixPivot}(T, g, d)$
  $J \leftarrow \textsc{Partition}(T, g, d, p)$
  $\textsc{TriRapide}(T, g, J-1)$
  $\textsc{TriRapide}(T, J+1, d)$

La fonction “$\textsc{Tri}(T)$” est donc définie comme $\textsc{TriRapide}(T, 0, n-1)$ si T est un tableau de taille n.

Étudions rapidement l’influence du choix du pivot.

Cas 1 On définit “$\mathrm{ChoixPivot}(T, g, d) = g$.” Ainsi

À faire : Figure

Figure 1 – Arbre des appels récursifs de “TriRapide” avec le pivot à gauche

Ainsi, la complexité de cet algorithme, avec ce choix de pivot, est en $(n-1) + (n-2) + (n-3) + \cdots + 2 = \Theta(n^2)$.

Cas 2 On définit maintenant le choix du pivot comme l’indice de la médiane.

À faire : Figure

Figure 2 – Arbre des appels récursifs de “TriRapide” avec le pivot à la médiane

Rédigeons-le rigoureusement : soit $C_n = \max_{T \text{ tableau de taille } n} C(T)$. Posons $(u_p)_{p\in\mathbb{N}} = (C_{2^p})_{p\in\mathbb{N}}$. D’après l’algorithme de “TriRapide,” on a

$$\begin{aligned}
u_{p+1} &= 2^{p+1} - 1 + u_p + u_p \\
&= 2^{p+1} - 1 + 2u_p \\
&= (2^{p+1} - 1) + 2(2^p - 1) + 2^2 u_{p-1} \\
&= 2^{p+1} - 1 + 2^{p+1} - 2 + 2^2 u_{p-1} \\
&= 2^{p+1} - 1 + 2^{p+1} - 2 + 2^2(2^{p-1} - 1 + 2u_{p-2}) \\
&= 2^{p+1} - 1 + 2^{p+1} - 2 + 2^{p+1} - 2^2 + 2^3 u_{p-2}.
\end{aligned}$$

On a donc $u_0 = 1$ et $u_p = p \times 2^p - (2^p - 1)$. Or, la suite $(c_n)_{n\in\mathbb{N}}$ est croissante. Or,

$$\forall n \in \mathbb{N},\ 2^{\lfloor \log_2 n \rfloor} \leqslant n \leqslant 2^{\lfloor \log_2 n \rfloor + 1}$$

donc

$$u_{\lfloor \log_2 n \rfloor} \leqslant C_n \leqslant u_{\lfloor \log_2 n \rfloor + 1}.$$

D’où

$$c_n \leqslant (\lfloor \log_2 n \rfloor + 1) \times 2^{\lfloor \log_2 n \rfloor + 1} - 2^{\lfloor \log_2 n \rfloor - 1}.$$

Et donc, on en déduit que $c_n = \Theta\big(n \log_2(n)\big)$.

Remarque (Notations):
On fixe un tableau T de taille n. De plus, on suppose dans toute la suite, que $T \in \mathfrak{S}_n$.[2] On note alors $X_g^d[T]$ la variable aléatoire indiquant le nombres de comparaisons effectuées par l’algorithme $\textsc{TriRapide}(T, g, d)$, dès lors que $T(\llbracket g, d \rrbracket) \subseteq \llbracket g, d \rrbracket$.

On note de plus, $\mathrm{E}\big[X_g^d[T]\big]$ l’espérance de cette variable aléatoire.

2. T est une permutation de n éléments. Ici, $\mathfrak{S}_n$ représente l’ensemble des permutations de $\llbracket 1, n \rrbracket$.

Théorème: Le nombre moyen de comparaisons effectuées par l'algorithme de tri rapide pour une entrée T de taille n est équivalent à $2n \ln n$. Autrement dit,

$$\mathrm{E}\big[X_0^{n-1}[T]\big] \sim 2n \ln n.$$

■

Dans la preuve précédente, on a utilisé le lemme suivant.

Lemme: Soit $(g, d) \in \mathbb{N}^2$ et soit $T \in \mathfrak{S}_n$ une permutation telle que $T(\llbracket g, d \rrbracket) \subseteq \llbracket g, d \rrbracket$.

$$\mathrm{E}\left[X_g^d[T]\right] = \mathrm{E}\left[X_0^{d-g}[\mathrm{id}]\right].$$

■

Annexe A. Hors-programme