Leçon 1 : Tri rapide (quick sort)

AYIKPA KACOUTCHY JEAN : Enseignant - Chercheur

Table des matières

1- Concept du tri rapide

I

1.1- Généralité du tri rapide

Les algorithmes récursifs utilisent l'approche *« diviser pour régner »*.

Le paradigme *« diviser pour régner »* donne lieu à 3 étapes à chaque niveau de récursivité :

1. *Diviser* le problème initial en un certain nombre de sous problèmes.

2. *Régner* sur les sous problèmes en les résolvant récursivement. Par ailleurs si la taille d'un sous problème est assez réduite, on peut le résoudre directement.

3. *Combiner* les solutions aux sous problèmes en une solution complète pour le problème initial.

Le tri rapide suit très fidèlement la règle « diviser pour régner ».

1.2- Description du tri rapide

Les trois (3) étapes du processus *« diviser pour régner »* employées dans le tri rapide pour trier un sous tableau typique *Tab[p..r]* avec $p < r$, sont les suivants :

Diviser : le tableau *Tab[p..r]* est partitionné (réarrangé) en *deux sous tableaux non vides Tab[p..q] et Tab[q+1..r]* tels que :

chaque élément de *Tab[p..q]* soit inférieur ou égal à chaque élément de *Tab[q+1..r]*. L'indice q est calculé pendant la procédure de partitionnement.

Régner : les deux sous tableaux *Tab[p..q] et Tab[q+1..r]* sont triés par des appels récursifs à la procédure principale du tri rapide.

Combiner : comme les sous tableaux sont triés sur place, aucun travail n'est nécessaire pour les recombiner. Le tableau *Tab[p..r]* tout entier est maintenant trie.

La procédure du tri rapide est le suivant :

Procédure TRI_RAPIDE (Tab, p, r)

var

q : entier

Début

Si p < r alors

q ← Partition(Tab, p, r)

TRI_RAPIDE (Tab, p, q)

TRI_RAPIDE (Tab, q+1, r)

Finsi

Fin

1.3- Partition du tableau

Principe :

Pour faire la partition d'un Tableau Tab en deux sous-tableaux L1 et L2 :

on choisit *une valeur quelconque dans du tableau Tab (la dernière par exemple) que l'on dénomme pivot,* puis on construit

- le sous-tableau L1 comme comprenant tous les éléments de L dont la valeur est inférieure ou égale au pivot,
- et l'on construit la sous-tableau L2 comme constituée de tous les éléments dont la valeur est supérieure au pivot.

Tab = [4, 23, 3, 42, 2, 14, 45, 18, 38, 16]

- Prenons comme pivot la dernière valeur *pivot = 16*

Nous obtenons par exemple :

L1 = [4, 14, 3, 2] *// tous les valeurs du tableau inférieures ou égales au pivot*

L2 = [23, 45, 18, 38, 42] *// tous les valeurs du tableau supérieures au pivot*

A cette étape voici *l'arrangement de Tab* :

Tab = L1 + pivot + L2 => [4, 14, 3, 2, 16, 23, 45, 18, 38, 42]

- En appliquant la même démarche au deux sous-listes : *L1 (pivot=2)* et *L2 (pivot=42)*
- Cas de L1 prenons comme pivot la dernière valeur *pivot = 2*

Nous aurons deux sous-tableaux à savoir *L11(pour les valeurs inférieures ou égales au pivot) et L12(pour les valeurs supérieures au pivot)*

L11=[ ] liste vide *// tous les valeurs du tableau inférieures ou égales au pivot*

L12=[3, 4, 14] *// tous les valeurs du tableau supérieures au pivot*

L1=L11 + pivot + L12 => [2,3, 4,14]

- Cas de L2 prenons comme pivot la dernière valeur *pivot = 42*

Nous aurons deux sous-tableaux à savoir *L21(pour les valeurs inférieures ou égales au pivot) et L22(pour les valeurs supérieures au pivot)*

L21=[23, 38, 18] *// tous les valeurs du tableau inférieures ou égales au pivot*

L22=[45] *// tous les valeurs du tableau supérieures au pivot*

L2=L21 + pivot + L22 => [23, 38, 18, 42, 45]

A cette étape voici le nouvel *arrangement de Tab* :

Tab = [(2,3, 4, 14), 16, (23, 38, 18, 42, 45)]

On répétera cette action jusqu'au rangement complet de Tab

etc...

Ainsi de proche en proche en subdivisant le problème en deux sous-problèmes, à chaque étape nous obtenons un pivot bien placé.

Le point principal de l'algorithme est la fonction Partition qui réarrange le sous tableau *Tab[p..r]*. En plus, elle émet l'indice de partitionnement q.

fonction Partition(A, p ,r : entier) : entier

var

i , j , piv , temp : entier

début

piv ← Tab[r];

i ← p-1;

j ← r;

repeter

repeter

i ← i+1

jusquà Tab[i] >= piv *// recherche l'indice de la valeur supérieur e ou égale au pivot*

repeter

j ← j-1

jusquà Tab[j] <= piv *// recherche l'indice de la valeur inférieure ou égale au pivot*

*//****** Permuter les valeurs des indices de i et j *******

temp ← Tab[i];

Tab[i] ← Tab[j];

Tab[j] ← temp

jusqu à j <= i;

*//****************** Refaire une autre permutation après la sortie de la boucle avec les indice i,j et r*

Tab[j] ← Tab[i];

Tab[i] ← Tab[r];

Tab[r] ← temp;

renvoyer i

FinPartition

Application 1 :

II

Exercice

[solution n°1 p.11]

Quelle règle utilise le tri rapide ?

- Diviser avec précision
- Diviser pour régner
- Diviser pour vaincre

Exercice

[solution n°2 p.11]

- Soit G et D deux entiers avec G < D, T est un tableau de taille allant de G à D et k un nombre compris entre G et D. Utilisant le principe du tri rapide donner :

Le tableau principal : ______

Le sous-tableau T1 : ______

Le sous-tableau T2 : ______

- Compléter l'instruction avec les paramètres ci-dessus la procédure tri_rapide

Procédure tri_rapide(______ , ______ , ______ : entier)

var

k : entier

Début

Si ______ alors

______ ← Partition(______ , ______ , ______)

______ (______ , ______ , ______)

______ (______ , ______ , ______)

Finsi

Fin

2- Version stochastique du tri rapide

III

2.1- Concept

Un algorithme est dit stochastique si son comportement est déterminé non seulement par son entrée mais aussi par les valeurs produites par un générateur de nombres aléatoires.

On modifie la fonction *partition* pour qu'elle est un comportement stochastique en utilisant une fonction hasard(a,b) qui renvoie de manière équiprobable un entier de l'intervalle [p..r].

2.2- Algorithme stochastique du tri rapide

- *Fonction partition Stochastique*

Fonction partitionStochastique (Tab, p , r : Entier) : Entier

var s : entier

Début

s <- hasard (p , r)

permuterTab (Tab , p , r)

Renvoyer *partition (Tab , p , r)*

Fin

NB : Pour la fonction partition voir la section 1.3- Partition du tableau

- *La procédure du tri rapide stochastique est la suivant :*

Procédure TRI_RAPIDE (Tab, p, r)

var

q : entier

Début

Si p < r alors

q ← *partitionStochastique* (Tab, p, r)

TRI_RAPIDE (Tab, p, q)

TRI_RAPIDE (Tab, q+1, r)

Finsi

Fin

Application 2 :

IV

Exercice

[*solution* n°3 p.11]

Un algorithme est dit stochastique si

○ son comportement est déterminé par une entrée et aussi par les valeurs produites par un générateur de nombres aléatoires

○ son comportement est déterminé une seule valeur en entrée

○ son comportement est déterminé par plusieurs valeurs en entrée

Exercice

[*solution* n°4 p.12]

Soit G et D deux entiers avec G < D, T est un tableau de taille allant de G à D donner la fonction partition Stochastique

Fonction partitionStochastique (______ , ______ , ______ : Entier) : Entier

var

s : entier

Début

______ <- ______ (______ , ______)

permuterT(______ , ______ , ______)

______ partition (______ , ______ , ______)

Fin

Solutions des exercices

> Solution n°1

Exercice p. 7

Quelle règle utilise le tri rapide ?

- ○ Diviser avec précision
- ◉ Diviser pour régner
- ○ Diviser pour vaincre

> Solution n°2

Exercice p. 7

- Soit G et D deux entiers avec G < D, T est un tableau de taille allant de G à D et k un nombre compris entre G et D. Utilisant le principe du tri rapide donner :

Le tableau principal : `T[G..D]`

Le sous-tableau T1 : `T[G..k]`

Le sous-tableau T2 : `T[k+1..D]`

- Compléter l'instruction avec les paramètres ci-dessus la procédure tri_rapide

Procédure tri_rapide(`T`, `G`, `D` : entier)

var

k : entier

Début

Si `G<D` alors

`k` ← Partition(`T`, `G`, `D`)

`tri_rapide`(`T`, `G`, `k`)

`tri_rapide`(`T`, `k+1`, `D`)

Finsi

Fin

> **Solution** n°3

Exercice p. 10

Un algorithme est dit stochastique si

⦿

son comportement est déterminé par une entrée et aussi par les valeurs produites par un générateur de nombres aléatoires

O son comportement est déterminé une seule valeur en entrée

O son comportement est déterminé par plusieurs valeurs en entrée

> **Solution** n°4

Exercice p. 10

Soit G et D deux entiers avec G < D, T est un tableau de taille allant de G à D donner la fonction partition Stochastique

Fonction partitionStochastique (`T`, `G` , `D` : Entier) : Entier

var

s : entier

Début

`s` <- `hasard`(`G` , `D`)

permuterT(`T` , `G`, `D`)

`Renvoyer` partition (`T` , `G` , `D`)

Fin