S4 2019-2020

Nathan TONNELLE

Algorithmique

INFO0401

Table des matières

[La note du TP : 2](#_Toc27495588)

# La note du TP :

A faire dans un document

Tous les algos qu'on fait pendant l'année sont programmé et dans un dossier (algo TD, TP)

Divisée en 3 parties :

Document 40% note TP

Partie exécutive : 30% note TP

Oral : 30% note TP

Question sur un algo tirer au hasard (montrer son exécution)

# Tp1 : tous les algos du td1

Ecrire l'algo : (langue algo)

Donner son utilisation

Quelques propriétés

Donner sa complexité (estimé)

Code

Exemple d'exécution

Horaire

Piles :

* Implémentation d'une pile
* Différents algos
  + - Remplir une pile
    - Vider une pile
    - Les fonctions de bases pile :
      * Est\_Vide(p)
      * Sommet(p)
      * Eunparler(p)
      * Dépiler(p)
    - Egalite de 2 piles
    - Tuer une pile
    - Recherche d'une pile occurrente de x depuis une pile
    - Fonction Ackermann
* Liste chainée
  + - Construction d'une liste chainée
    - Vider une liste chainée
    - Insertion d'un élément dans une liste chainée (au début, milieu, fin)
    - Soustraire un élément dans une liste (3 config)
    - Concaténation de 2 listes chainée
    - Les opérations logiques sur liste chainée (et ou intersection, union)
    - Ajouter en fin de liste (l)
    - Addition de 2 polys en utilisant les listes chainées
    - Trier une liste chainée
    - Fonction existante : est-ce qu'une liste est contenue dans une liste

Extrait (C1, C2) : Boolean

* Les 2 premières valeurs de C1, C2 sont égales
* Les2 dernières valeurs de C1, C2 sont égales
* Des autres sont égales 2 à 2

Tris :

* Tris à bulles
* Tris rapides
* Tri topologique
* Tri couleur
* Tri par parties

Les arbres

* Les différents parcours
* Construction d'un arbre
* Recherche d'un élément dans un arbre binaire
* La hauteur d'un arbre binaire
* St-h équilibré
* Affichage d'un arbre
* Calcul des nœuds internes et externes d'un arbre
* Calcul des nombres de feuilles
* Egalité entre 2 arbres binaires
* L'arbre d'Hoffmann
* Un arbre 2 3

# Les tris

Organiser une collection d'objets selon un ordre déterminé. Les objets à trier sont des éléments d'un ensemble muni d'un ordre total. Souvent ces objets sont des entiers. On utilise le plus souvent un tableau non dimensionnel pour réaliser ce tri.

## Le tri à bulle

### Principe

#### Ev

13 14 15 21 6 7 3 2 5 1

Comparer 2 valeurs adjacentes et les échanger jusqu'à remonter la plus grande valeur.

A la 1ere itération : on remonte la plus grande valeur

A la 2eme itération : on remonte la 2eme plus grande valeur

A la 3eme itération : on remonte la 3eme plus grande valeur

… jusqu'à la dernière valeur contenue dans la 1ere case du tableau qui constitue la primitive de base égale à elle même

1ere T 13 14 15 6 7 3 2 5 1 **21**

2eme T 13 14 6 7 3 2 5 1 **15 21**

3eme T 13 6 7 3 2 5 1 **14 15 21**

4eme T 6 7 3 2 5 1 **13 14 15 21**

5eme T 6 3 2 5 1 **7 13 14 15 21**

6eme T 3 2 5 1 **6 7 13 14 15 21**

7eme T 2 3 1 **5 6 7 13 14 15 21**

8eme T 2 1 **3 5 6 7 13 14 15 21**

9eme T 1 **2 3 5 6 7 13 14 15 21**

10eme T **1 2 3 5 6 7 13 14 15 21**

### Algorithme :

Données :

T tableau d'entiers

I, inf, sup, taille : entier

Début :

Lire (sup)

Lire (t)

Taille <- sup

Tant que taille > 1 faire

Pour i allant de 1 à taille-1 faire

Si T[i] > T[i+1] alors

Echanger (t[i],T[i+1])

finSi

FinPour

Taille <- taille-1

FinTantQue

Afficher (T)

Fin

## Tri couleur

Hollandais : **000**0000000

T 000**0**00**0**000**00**000**0**0**0** initial

Plusieurs approches pour réaliser ce tri couleur

T **000** 000 **00**00000**0**0 0000

Classé classé non classé classé

i j j+1 m

### Algo

Données :

T : tableau

I, j, k : entier

N : entier

Début :

Lire(T)

I<-0

J<-0

K<-n

Tant Que (k=/= j) faire

Si t[j+1]=bleue alors

Echange(t[j+1],t[i+1])

I++

J++

Sinon si (t[j+1]=rouge) alors

Echange( t[j+1],t[k])

k—

sinon

j++

finSi

FintTantQue

Fin

T : A B C A C B C B i=0, j=0, k=taille

T : A B C A C B C B i=0, j=1, k=8

T : B A C A C B C B i=0, j=2, k=8

…

## Tri rapide

Ex : 1 3 15 5 14 6 21 2 13 7

Sur une partie du tableau à traiter avant la séparation en 2 parties

2 4 5 3 8 1 21 16 9

Progression de i tant que t[i]<pivot

## Algo

I, j, k, droite, gauche, pivot, taille, sup, inf : entier

Début :

Droite<-sup;

Gauche <- inf

Pivot<-T[droite]

J<-droite-1

I<-inf

Tant Que i =/= j faire

Tant Que (T[i]<pivot) & (i=/= j)faire

I<-i+1

Fin tant que

Tant que (T[j]>pivot)&(i=/=j) faire

j<-j+1

fin tant que

echanger (T[i], T[j])

fin tant que

si T[i]>pivot alors

echanger (T[i], pivot)

finSi

si i> gauche +1

T1 rapide (T,inf,i)

Sinon

T1 rapide (T,i+1,sup)

Fin Si

Fin

1 3 15 5 14 6 21 2 17 7

1 3 2 5 14 6 21 15 13 7

1 3 2 5 6 14 21 15 13 7

1 3 2 5 6 7 21 15 13 14