Les tableaux

D.E ZEGOUR

École Supérieure d'Informatique

ESI

Les tableaux

Définition

Ensemble d’éléments de même nature (type)

Le type peut être quelconque: entier, réel, booléen, caractères ou un type construit

Dans ce dernier cas, on dira qu'on a un vecteur de structures ou d'objets composés.

Accès à un élément du vecteur : opération d'indexation.
Quelque soit i dans [1, n] : T(i) délivre la valeur de l’élément d'indice i dans le vecteur T.

Vecteur ordonné :
- Ensemble muni d'une relation d'ordre.
- Quelque soit i dans [1, n-1] : T(i) <= T(i+1)

Les tableaux

Tableau statique et tableau dynamique

Tableau statique	Tableau dynamique
L'allocation de l'espace se fait tout à fait au début d'un traitement.	L'allocation de l'espace se fait pendant l'exécution
En termes techniques, on dit que l'espace est connu à la compilation.	La libération de l'espace est alors possible pendant l'exécution

Les tableaux

Machine abstraite

ELEMENT (T [i, j, ...])
Accès à l'élément T[i, j, ...]du vecteur T.

AFF_ELEMENT (T [I, J, ...], Val)
Affecter à l'élément T[i, j, ...] la valeur Val.

ALLOC_TAB (T)
Allocation d'un tableau de taille spécifiée par la définition de T. L'adresse est rendue dans la variable T.

LIBER_TAB (T)
Libération de l'espace mémoire pointé par T.

Les tableaux

Algorithmes

4 catégories

Parcours :accès par valeur, accès par position

Mise à jour : insertion, suppression

Algorithmes sur plusieurs tableaux : fusion, interclassement, éclatement,...

Tri sur les tableaux

Les tableaux

Algorithme : Parcours : Recherche séquentielle

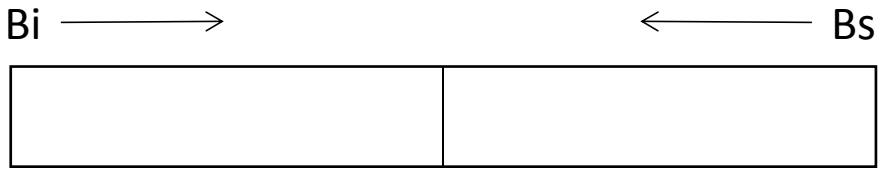
Tableau non ordonné

```
I := 1
Trouv := FAUX
TANTQUE I <= M et NON Trouv
    Si Element( V[I]) = D
        Trouv := VRAI
    SINON
        I := I + 1
    FSI
FINTANTQUE
```

Tableau ordonné

```
I := 1
Trouv, Arret := FAUX
TANTQUE I <= M et NON Trouv et NON Arret
    Si Element(V[I]) = D
        Trouv := VRAI
    SINON
        SI Element(V[I]) > D
            Arret := VRAI
        SINON
            I := I + 1
        FSI
    FSI
FINTANTQUE
```

Les tableaux



Algorithme : Parcours : Recherche dichotomique

```
Bi := 1
Bs := M
Trouv := FAUX
TANTQUE Bi <= Bs et NON Trouv
        Milieu := (Bi + Bs ) /2
        Si Element(V[Milieu]) = V
             Trouv := VRAI
        SINON
              SI V < Element(V[Milieu])
                   Bs := Milieu - 1
               SINON
                    Bi := Milieu + 1
                FSI
         FSI
FINTANTQUE
```

Nombre maximal de comparaison = Log2(N)

Les tableaux

Algorithme : Mise à jour

Insertion par valeur :Insérer un élément donné après ou avant une valeur donnée du vecteur.

Insertion par position : Insérer un élément donné à la position p donnée du vecteur.

Insertion par valeur dans un tableau ordonné

Suppression (physique) par valeur dans un tableau (ordonné ou pas)

Suppression logique par valeur dans un dans un tableau (ordonné ou pas)

Les tableaux

Algorithme : Insertion par valeur dans un tableau ordonné T[1..M] contenant N éléments

```
Si  N < M
   // Recherche de la position
   I:=1
   Trouv := Faux
   Tq I <=N et Non Trouv
      Si ( Element(T[i]) >= V
         Trouv := Vrai
      Sinon
          I := I +1
      Fsi
   Ftq
   // Insertion
    Si non trouv
       Aff_element(T[i], V)
   Sinon
       Pour j:= N+1, i, -1
          Aff_element(T[j], Element(T[j-1] ) )
       Fpour
       Aff_element(T[i], V)
   Fsi
   N:= N+1
Sinon
  Ecrire('Saturation')
Fsi
```

Nombre maximal de décalages = N

Les tableaux

V1 : TABLEAU[1..N1] DE ENTIER
V2 : TABLEAU[1..N2] DE ENTIER
V3 : TABLEAU[1..N1+N2] DE ENTIER

Algorithme : Interclassement de deux tableaux ordonnés

```
I, J:= 1
K := 0
TANTQUE I <= N1 et J <= N2
    K := K + 1
    SI Element(V1[I]) < Element(V2[J])
        Aff_element(V3[K], Element(V1[I]))
        I := I + 1
    SINON
        Aff_element(V3[K], Element( V2[J]))
        J := J + 1
    FSI
FINTANQUE
```

```
TANTQUE I <= N1
    K := K + 1
    Aff_element(V3[K], Element(V1[I]) )
    I := I + 1
FINTANQUE

TANTQUE J <= N2
    K := K + 1
    Aff_element(V3[K], Element( V2[J]))
    J := J + 1
FINTANQUE
```

Nombre maximal d'
d'affectations = N1 + N2

Les tableaux

Algorithme : Tri par bulles

```
POUR I :=1, M-1
    POUR J := M, I+1, -1
        SI Element(V[J-1]) > Element(V[J])
            Temp := Element(V[J-1])
            Element(V[J-1] ):= Element(V[J])
            Element(V[J]) := Temp
        FSI
    FINPOUR
FINPOUR
```

Nombre maximal de permutations = N (N-1)/2

Les tableaux

Algorithme : Tri par interclassement

```
Pas := 1
Tq Pas <= N
   Pour I=1, N-Pas, 2Pas
      Interclasser (I, Pas, I + Pas, Min(Pas,N-(i+Pas) + 1 )
   Fpour
   Pas := 2 pas
Ftq
```

V1 V2 V3 V4 V5 V6 V7 V8 V9

[V1] [V2] [V3] [V4] [V5] [V6] [V7] [V8] [V9]

[V1 V2] [V3 V4] [V5 V6] [V7 V8] [V9]

[V1 V2 V3 V4] [V5 V6 V7 V8] [V9]

[V1 V2 V3 V4 V5 V6 V7 V8] [V9]

[V1 V2 V3 V4 V5 V6 V7 V8 V9]

Nombre maximal
d'affectations = N Log2(N)

Les tableaux

Représentation en mémoire : Tableau à n dimensions

Une déclaration typique : $A(a_1:b_1; a_2:b_2; .....a_n:b_n)$

Ordre de rangement des sous tableaux :

$A(a_1, *, *, ..., *)$, $A(a_1+1, *, *, ..., *)$, $A(a_1+2, *, *, ..., *)$,
....., $A(i, *, *, ..., *)$, ... $A(b_1, *, *, ..., *)$

A l'intérieur de chaque sous tableau A(i, *, *, ..., *), l'ordre suivant des sous sous-tableaux est considéré :

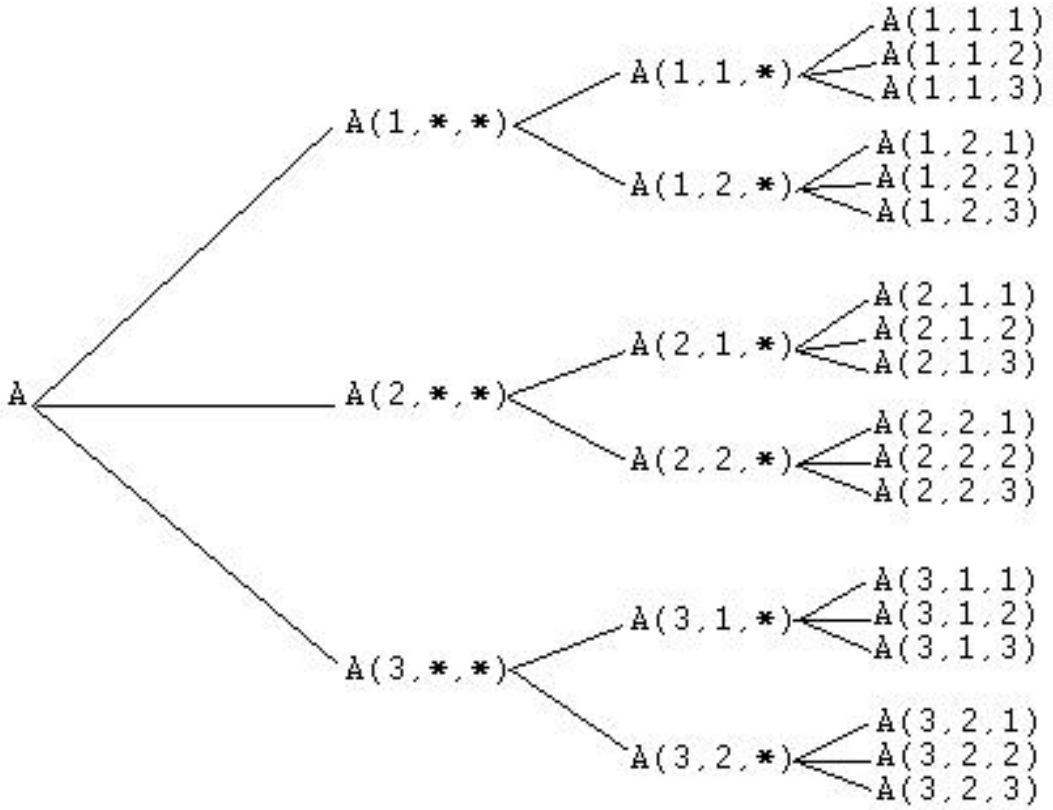
$A(i, a2, *, *, ..., *)$, $A(i, a2+1, *, *, ..., *)$, $A(i, a2+2, *, *, ..., *)$,
....., $A(i, b2, *, *, ..., *)$

Etc.

Les tableaux

Représentation en mémoire : Tableau à n dimensions

A(1..3, 1..2, 1..3)



Les tableaux

Représentation en mémoire : Tableau à n dimensions

Adresse d'un élément $A(i_1, i_2 .., i_n)$ *?*

Posons $d_i = b_i - a_i + 1$

Adresse de $A(i_1, *, *, ...)$: $AD_1 = Base + (i_1-a_1)\ d_2d_3...d_n$

Adresse de $A(i_1, i_2, *, ...)$: $AD_2 = AD_1 + (i_2-a_2)d_3d_4...d_n$

Adresse de $A(i_1, i_2, ...i_n)$:

$AD_n = Base + (i_1-a_1)d_2d_3..d_n + (i_2-a_2)d_3d_4...d_n + ...+(i_{n-1}-a_{n-1})d_n + (i_n-a_n)$

Les tableaux

Représentation en mémoire : Tableau à n dimensions

Adresse de $A(i_1, i_2, ...i_n)$:

$AD_n = Base + (i_1-a_1)d_2d_3..d_n + (i_2-a_2)d_3d_4...d_n + ...+(i_{n-1}-a_{n-1})d_n + (i_n-a_n)$

Partie constante : $Base -(a_1 \prod_{i=2,n} d_i + a_2 \prod_{i=3,n} d_i + ... + a_{n-1}d_n + a_n)$

Partie variable : $i_1 \prod_{i=2,n} d_i + i_2 \prod_{i=3,n} d_i + ... + i_{n-1}d_n + i_n$

Si a1 = a2 = ... = an = 0

l'adresse de A(i1, i2, ...in) est

$i1d2d3..dn + i2d3d4...dn + ...+in-1dn + in = \sum_{j=1,n} (i_j . \prod_{i=j+1,n} d_i)$

Les tableaux

Implémentation (Statique C)

```
typedef % type d'un élément du tableau % Typeqq;
typedef Typeqq Typevect[10];

int Element (Typevect V , int I )
  {
    return ( V[I] );
  }

void Aff_element (Typevect  V,  int  I, int Val )
  {
    V[I] = Val;
  }
```

Les tableaux

Implémentation (Dynamique C)

```
/** Implémentation **\: TABLEAU DE ENTIERS**/

/** Tableaux dynamiques **/
typedef int Typeelem_W10i ;
typedef Typeelem_W10i *  Pointeur_W10i;
```

```
void Alloc_tab_W10i ( Pointeur_W10i *T )
   { *T = malloc(10 * sizeof(int)); }

 void Liber_tab_W10i ( Pointeur_W10i T)
   { free(T) ;}
```

Les tableaux

Implémentation (Dynamique C)

```
Typeelem_W10i Element_W10i ( Pointeur_W10i V  , int I1  )
   {
     return *(V + I1-1 );
   }

void Aff_element_W10i ( Pointeur_W10i V,  int I1 ,
Typeelem_W10i Val )
   {
     *(V + I1-1 ) = Val ;
   }
```

Les tableaux

Implémentation (Statique PASCAL)

```
CONST Max =100;  Taille arbitraire
TYPE
  Typeqq =  { type d'un élément du tableau } ;
  Typevect = ARRAY[1..Max] OF Typeqq;
```

```
 FUNCTION Element ( VAR V:Typevect; I: INTEGER ) : Typeqq;
   BEGIN
     Element :=  V[I];
   END;

PROCEDURE Aff_element ( VAR V :Typevect; I:INTEGER; Val :
Typeqq );
   BEGIN
     V[I] := Val;
   END;
```

Les tableaux

Implémentation (Dynamique PASCAL)

```
{ Implémentation : TABLEAU DE ENTIERS}

{ Tableaux dynamiques }
Type
   Typeelem_W10I = INTEGER;
   Pointeur_W10I= ^Typetab_W10I;
   Typetab_W10I = ARRAY[1..10] OF Typeelem_W10I;
```

```
PROCEDURE Alloc_tab_W10I ( var T : Pointeur_W10I );
   BEGIN
     NEW(T)
   END;

 PROCEDURE Liber_tab_W10I ( T : Pointeur_W10I );
   BEGIN
     DISPOSE(T)
   END;
```

Les tableaux

Implémentation (Dynamique PASCAL)

```
 FUNCTION Element_W10I ( V:Pointeur_W10I; I1  : INTEGER ) :
Typeelem_W10I;
   BEGIN
     Element_W10I :=  V^[I1 ];
   END;

  PROCEDURE Aff_element_W10I ( V :Pointeur_W10I; I1  :
INTEGER; Val : Typeelem_W10I );
   BEGIN
     V^[I1 ] := Val;
   END;
```