Programme et Algorithme III
Travaux Dirigés 1

CYU & ZUST

09 novembre, 2021

- **Exercice 1**: *Grand-O*

  Pour chacun des fonctions $T_i(n)$ suivant, déterminer sa complexité asymptotique dans la notation Grand-O. Exemple : $T_0(n) = 3n \in O(n)$.

  ^: puissance
  2^n == 2*2*2*….*2

  1. $T_1(n) = 6n^3 + 10n^2 + 5n + 2$ ∊ O(n^3)
  2. $T_2(n) = 3log_2 n + 4$ ∊ O(log n)
  3. $T_3(n) = 2^n + 6n^2 + 7n$ ∊ O(2^n) ou ∊O(a^n)
  4. $T_4(n) = 7k + 2$ ∊ O(1)
  5. $T_5(n) = 4log_2 n + n$ ∊ O(n)
  6. $T_6(n) = 2log_{10} k + kn^2$ ∊ O(n^2)

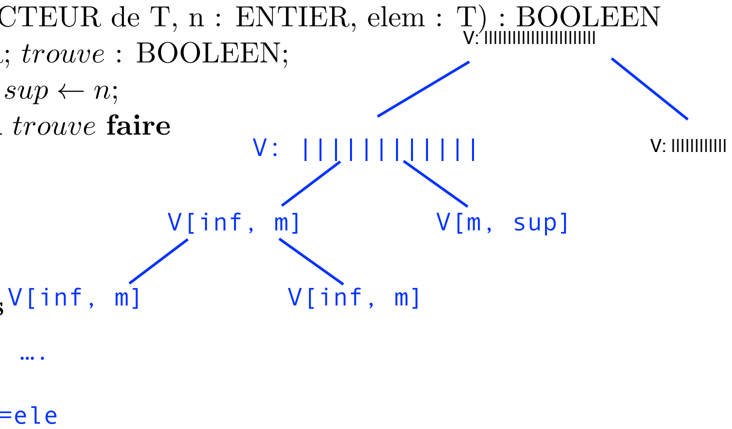
- **Exercice 2:** *Complexité de recherche*

  L'algorithme de recherche dichotomique est donné dans **Algorithme 1**.

Algorithme 1 Recherche dichotomique

Entrée: Un vecteur trié de type T , un élément de type T à chercher
Sortie: Si l'élément cherché existe dans le vecteur
  FONCTION dico(V : VECTEUR de T, n : ENTIER, elem : T) : BOOLEEN
  VAR inf, sup, m : ENTIER; *trouve* : BOOLEEN;
  $trouve \leftarrow FAUX$; $inf \leftarrow 1$; $sup \leftarrow n$;
  tant que $inf \leq sup$ **et non** $trouve$ **faire**
    $m \leftarrow (inf + sup)/div2$;
    si $V[m] = elem$ **alors**
      $trouve \leftarrow VRAI$;
    sinon
      si $V[m] < elem$ **alors**
        $inf \leftarrow m + 1$;
      sinon $sup \leftarrow m - 1$;
      fin si
    fin si
  fin tant que
  Retourner trouve

n: T(n)
2n: T(2n)=T(n)+const



O(log n)

Calculer la complexité temporelle de l'algorithme de recherche dichotomique en fonction du nombre de comparaisons dans le pire et dans le meilleur des cas. Comparer avec l'algorithme de recherche séquentiel. O(n)

- **Exercice 3:** *Addition de matrices*

Considérer les deux matrices quadratiques A et B de taille n:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{21} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{21} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix}, \tag{1}$$

L'addition de ces deux matrices donne la matrice C quadratique de taille n:

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{21} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_{nn} \end{pmatrix}, \tag{2}$$

avec

$$c_{ij} = a_{ij} + b_{ij}, \forall i, \forall j \tag{3}$$

type t_matrix = array[1..n, 1..n] of real;
var A, B, C : t_matrix;

1. Définir le type des matrices quadratiques et déclarer les variables A, B, et C.
2. Écrire un algorithme qui effectue l'addition des deux matrices A et B et stocke les résultats en C.
3. Déterminer la fonction de temps maximale ("worst case") $T(n)$ pour des matrices de taille n.
4. Déterminer la complexité Grand-O pour des matrices de taille n.

- **Exercice 4:** *Tour de Hanoï*

Implémenter l'algorithme de solution au problème de *Tour de Hanoï* en langage C, avec n désignant le nombre de disques (Pseudo-code à transparent P32).

```
pour i de 1 à n:
   pour j de 1 à n:
      C[i,j] := A[i,j]+B[i,j]
   fin pour
fin pour
```

Dans la boucle intérieur:
Une affectation j:=j+1,
une comparaison j<=n,
et une affectation C[i,j]=A[i,j]+B[i,j]
le tout est répété n*n fois. ====3n^2
Dans la boucle extérieur:
Une affectation i:=i+1,
une comparaison i<=n
une affectation de j:=1
le tout est répété n fois. =====3n
uniquement une fois: i:=1. ===== 1

T(n)=3n^2+3n +1 ∈ O(n^2)