Introduction à l'informatique
Algorithmes et structures linéaires

Portail René Descartes

Luminy

Université d'Aix-Marseille

Bref rappel

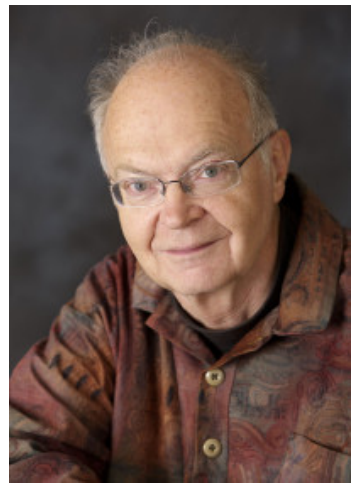
al-Khuwârizmî

Un algorithme est la description non ambigüe d'une séquence finie d'instructions permettant de résoudre un problème

Propriétés :

- Finitude : termine en temps fini
- Définition précise
- Entrées : données
- Sorties : résultat
- Rendement : utilise des opérations basiques

D. Knuth



Plan du cours

1. Abstraction et structures de données

2. Briques de base des algorithmes

3. Premiers algorithmes sur les tableaux

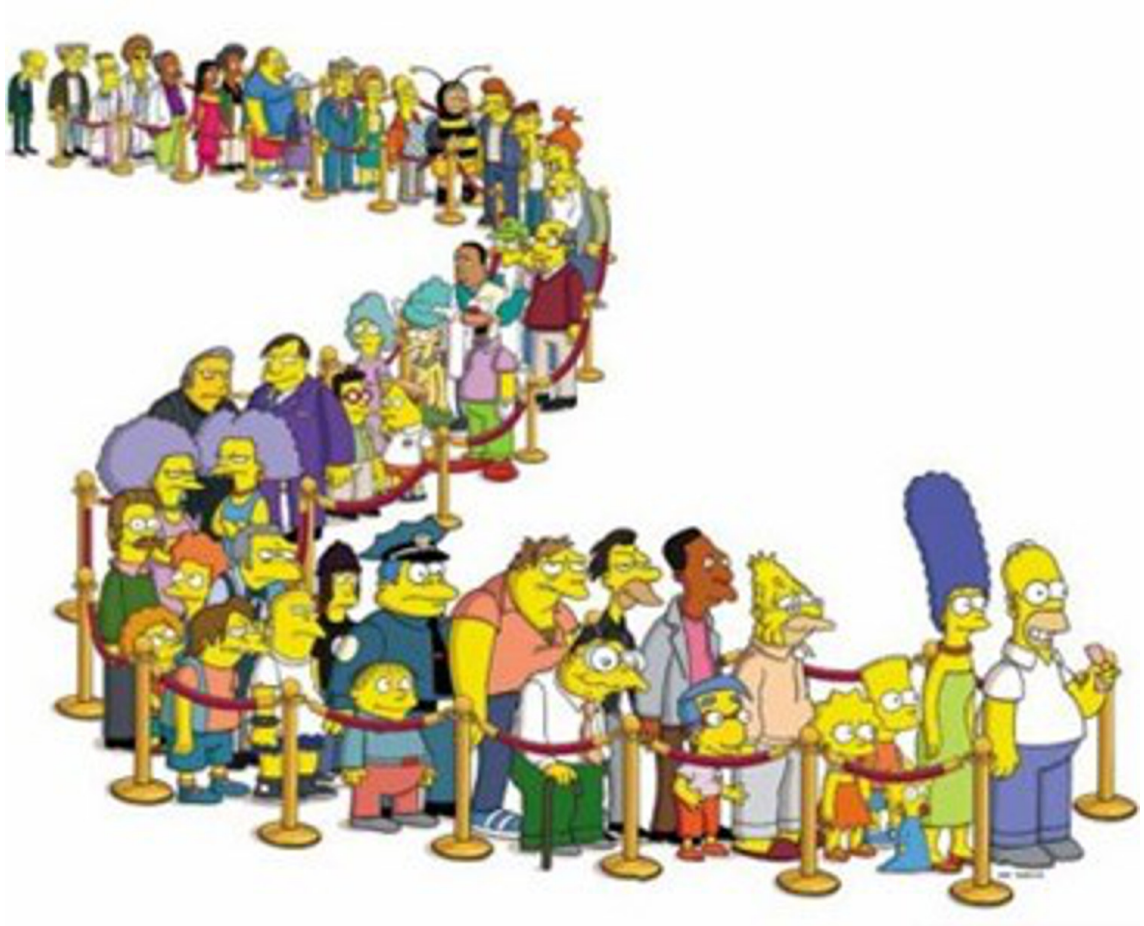
Plan du cours

Que voyez-vous ?

Que voyez-vous ?

Que voyez-vous ?

Que voyez-vous ?



Que voyez-vous ?

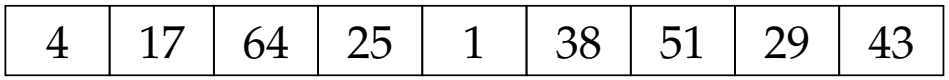
Que voyez-vous ?

Au final...

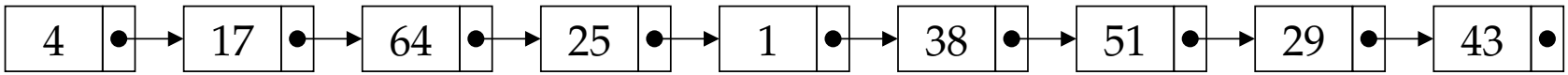
- Informations :
  - natures distinctes
  - relations entre elles spécifiques
  - structuration / organisation

Au final...

- Informations :
  - natures distinctes
  - relations entre elles spécifiques
  - structuration / organisation
- Structures de données linéaires :
  - Tableaux



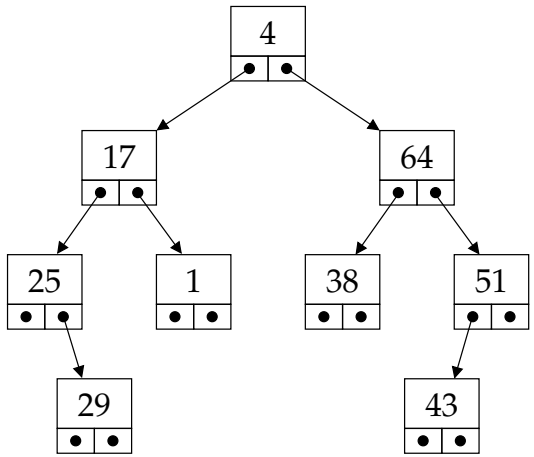
  - Listes chaînées

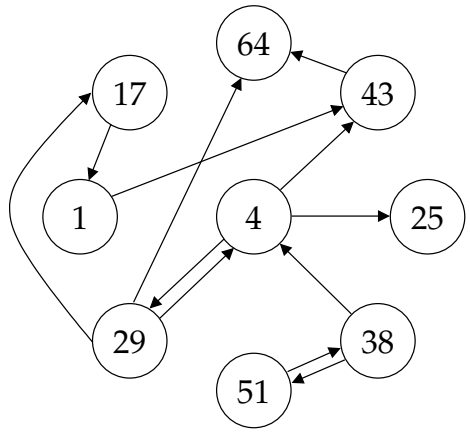


    - ◇ Piles (assiettes, pioche...)
    - ◇ Files (attente, pool d'impression...)
    - ◇ Tables de hachage (accès rapide)

Au final...

- Informations :
  - natures distinctes
  - relations entre elles spécifiques
  - structuration / organisation
- Structures de données non-linéaires :
  - Arbres (fichiers, décisions...)
  - Graphes (réseaux)





Plan du cours

Types et entrées-sorties

Types de base

Nom	Mot-clé	Exemples
Nombres booléens	booléen	0, 1
Nombres entiers	entier	$-131, -7, 0, 4, 1732$
Nombres réels	réel	$-2754{,}37, -13{,}0, 157\text{E}-13, 3{,}1415927$
Caractères	car	'a', ' ;', 'Z', '4'
Chaînes de caractères	chaîne	"hier", "toto", "amphithéâtre"

Appels d'entrées-sorties

Nom	Mot-clé	Exemples
Écriture	écrire	écrire("Hey !")
Lecture	lire	lire(variable)

écrire("Quel est votre nom ?") ;
lire(nom) ;
écrire("Bonjour ", nom, " !") ;

Procédures et variables

- Procédure : suite d'instructions nommée, que l'on peut appeler pour l'utiliser
- Variable : espace de stockage (emplacement mémoire) pour conserver des informations d'un type donné
- Exemple : Comment demander à une personne son nom, son année de naissance, et lui répondre l'anniversaire qu'elle a fêté ou va fêter cette année ?

Procédures et variables

- Procédure : suite d'instructions nommée, que l'on peut appeler pour l'utiliser
- Variable : espace de stockage (emplacement mémoire) pour conserver des informations d'un type donné
- Exemple : définition

```
Procédure bonjour()
    nom : chaîne ;
    annee, date : entier ;
Début
    écrire("En quelle année sommes-nous ?") ;
    lire(date) ;
    écrire("Quel est votre nom ?") ;
    lire(nom) ;
    écrire("Quelle est votre année de naissance ?") ;
    lire(annee) ;
    écrire("Bonjour ", nom, " !") ;
    écrire("C'est l'année de vos ", date − annee, " ans.") ;
Fin
```

Procédures et variables

- Procédure : suite d'instructions nommée, que l'on peut appeler pour l'utiliser
- Variable : espace de stockage (emplacement mémoire) pour conserver des informations d'un type donné
- Exemple : exécution de `bonjour();`

```
En quelle année sommes-nous? 2018
Quel est votre nom? Camille
Quelle est votre année de naissance? 1967
Bonjour Camille !
C'est l'année de vos 51 ans.
```

Paramètres donnée

- Paramètre donnée : information pas toujours demandée directement à l'utilisateur, déjà connue (issue d'une saisie ou d'un calcul préliminaire)

- Exemple : définition

```
Procédure bonjour(d date : entier)
    nom : chaîne ;
    annee : entier ;
Début
    écrire("Quel est votre nom ?") ;
    lire(nom) ;
    écrire("Quelle est votre année de naissance ?") ;
    lire(annee) ;
    écrire("Bonjour ", nom, " !") ;
    écrire("C'est l'année de vos ", date − annee, " ans.") ;
Fin
```

Paramètres donnée

- Paramètre donnée : information pas toujours demandée directement à l'utilisateur, déjà connue (issue d'une saisie ou d'un calcul préliminaire)

- Exemple : exécution de `bonjour(2018);`

```
Quel est votre nom? Camille
Quelle est votre année de naissance? 1967
Bonjour Camille !
C'est l'année de vos 51 ans.
```

Paramètres donnée

- Paramètre donnée : information pas toujours demandée directement à l'utilisateur, déjà connue (issue d'une saisie ou d'un calcul préliminaire)

- Autre exemple : définition

Procédure distance(**d** x_1, x_2, y_1, y_2 : **réel**)
    dist : **réel** ;
Début
    écrire("La distance entre les points (", x_1, ",", y_1, ") et (", x_2, ",", y_2, ") est : ") ;
    dist := $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$;
    écrire(dist, ".") ;
Fin

- Autre exemple : exécution de `distance(1,3,2,4);`

```
La distance entre les points (1,2) et (3,4) est: 2.828427.
```

Affectations

```
1.  Procédure affect1()
2.      a, b, c : entier ;
3.  Début
4.      a := 7 ;
5.      b := 3 ;
6.      c := b − a ;
7.      a := a − c ;
8.  Fin
```

	a	b	c
2.	indéfini	indéfini	indéfini
4.	7	indéfini	indéfini
5.	7	3	indéfini
6.	7	3	−4
7.	11	3	−4

```
1.  Procédure affect2()
2.      a, b, c : entier ;
3.  Début
4.      a := 9 ;
5.      c := a + b ;
6.      b := c ;
7.      c := 2 ;
8.  Fin
```

	a	b	c
2.	indéfini	indéfini	indéfini
4.	9	indéfini	indéfini
5.	9	indéfini	?
6.	9	?	?
7.	9	?	2

Affectations

```
1.  Procédure affect1()
2.      a, b, c : entier ;
3.  Début
4.      a := 7 ;
5.      b := 3 ;
6.      c := b − a ;
7.      a := a − c ;
8.  Fin
```

	a	b	c
2.	indéfini	indéfini	indéfini
4.	7	indéfini	indéfini
5.	7	3	indéfini
6.	7	3	−4
7.	11	3	−4

```
1.  Procédure affect2()
2.      a, b, c : entier ;
3.  Début
4.      a := 9 ;
5.      c := a + b ;
6.      b := c ;
7.      c := 2 ;
8.  Fin
```

	a	b	c
2.	indéfini	indéfini	indéfini
4.	9	indéfini	indéfini
5.	9	indéfini	?
6.	9	?	?
7.	9	?	2

Conditions

- Condition : expression booléenne
  $\rightsquigarrow$ permet l'exécution d'une séquence d'instructions dans des cas précis

- Exemple : définition

Procédure parite(**d** n : **entier**)
Début
    Si ($n \equiv 0 \mod 2$) **alors**
        écrire(n, " est un nombre pair.") ;
    Sinon
        écrire(n, " est un nombre impair.") ;
    Finsi
Fin

- Exemple : exécution de `parite(86);`

```
86 est un nombre pair.
```

- Exemple : exécution de `parite(1377);`

```
1377 est un nombre impair.
```

Itérations

- Itération : répétition d'une séquence d'instructions

- Exemple : définition

```
Procédure enfants(d date : entier)
    nom : chaîne ;
    annee, age, i, nb : entier ;
Début
    écrire("Combien d'enfants avez-vous ?") ;   lire(nb) ;
    i := 1 ;
    Tant que i ⩽ nb faire
        écrire("Nom d'un de vos enfants ?") ;   lire(nom) ;
        écrire("Son année de naissance ?") ;   lire(annee) ;
        age := date − annee ;
        écrire(nom, " a eu ou va avoir ", age, "ans cette année.") ;
        i := i + 1 ;
    Finfaire
Fin
```

Itérations

- Itération : répétition d'une séquence d'instructions

- Exemple : exécution de `enfants(2018);`

```
Combien avez-vous d'enfants? 3
Nom d'un de vos enfants? Alice
Son année de naissance? 1998
Alice a eu ou va avoir 20 ans cette année.
Nom d'un de vos enfants? Bruno
Son année de naissance? 1995
Bruno a eu ou va avoir 23 ans cette année.
Nom d'un de vos enfants? Coralie
Son année de naissance? 2007
Coralie a eu ou va avoir 11 ans cette année.
```

Itérations

- Itération : répétition d'une séquence d'instructions
- Exercice : comment calculer la somme des n premiers entiers appartenant à $\mathbb{N}^*$?

Itérations

- Itération : répétition d'une séquence d'instructions

- Exercice : comment calculer la somme des n premiers entiers appartenant à $\mathbb{N}^*$?

- Une solution "naïve" :

```
Procédure somme(d n : entier)
    i, s : entier ;
Début
    s := 0 ;
    i := 1 ;
    Tant que i ⩽ n faire
        s := s + i ;
        i := i + 1 ;
    Finfaire
    écrire("s = ", s) ;
Fin
```

Itérations

- Itération : répétition d'une séquence d'instructions

- Exercice : comment calculer la somme des n premiers entiers appartenant à $\mathbb{N}^*$?

- Une solution "naïve" :

Procédure somme(**d** n : **entier**)
    i, s : **entier** ;
Début
    $s := 0$;
    $i := 1$;
    Tant que $i \leqslant n$ **faire**
        $s := s + i$;
        $i := i + 1$;
    Finfaire
    écrire("s = ", s) ;
Fin

- La solution optimale :

Procédure somme(**d** n : **entier**)
Début
    écrire("s = ", $\frac{n \times (n+1)}{2}$) ;
Fin

Pourquoi ?

Fonctions

- Fonction : “procédure” particulière qui renvoie une valeur d’un type déterminé

- Exemple : définition et appel

```
Fonction min(d a, b : entier) : entier
Début
    Si (a < b) alors
        retour a ;
    Finsi
    retour b ;
Fin

Procédure minimum()
    x, y : entier ;
Début
    écrire("Que vaut x ?") ; lire(x) ;
    écrire("Que vaut y ?") ; lire(y) ;
    écrire("Le minimum entre x et y est ", min(x,y), ".") ;
Fin
```

Paramètres résultat

- Paramètre.s résultat : quand on veut rendre visible le changement de valeur d'un ou plusieurs paramètre.s donnée
- Exemple : définition et appel

```
Procédure min_max(d a, b : entier, r min, max : entier)
Début
    Si (a < b) alors
        min = a ;
        max = b ;
    Sinon
        min = b ;
        max = a ;
    Finsi ;
Fin
```

```
Procédure appel()
    k, ℓ, x, y : entier ;
Début
    min_max(x, y, k, ℓ) ;
    écrire("min(x,y) = ", k, ".") ;
    écrire("max(x,y) = ", ℓ, ".") ;
Fin
```

Paramètres résultat

- Paramètre.s résultat : quand on veut rendre visible le changement de valeur d'un ou plusieurs paramètre.s donnée
- Un autre exemple : définition et appel

Procédure permut(**dr** a, b : **entier**)
Début
    $c := a$;
    $a := b$;
    $b := c$;
Fin

Procédure permutation()
    x, y : **entier** ;
Début
    $x := 5$;
    $y := 3$;
    permut(x, y) ;
    écrire("$x =$ ", x, ".") ;
    écrire("$y =$ ", y, ".") ;
Fin

- Exécution de `permutation();`

```
x = 3.
y = 5.
```

Plan du cours

Tableaux

D♢	7♢	D♠	7♠	V♠	7♡	R♣

Tableaux

- Un tableau t de taille n est un vecteur de dimension n composé de n éléments de même type, en général représenté comme suit :

$$t = (t_1, t_2, \ldots, t_{n-1}, t_n)$$

$t[1 \ldots n]$ =

$t[1]$	$t[2]$	$\ldots$	$t[n-1]$	$t[n]$

- Un tableau de nombres entiers :

4	17	64	25	1	38	51	29	43

- Un tableau de caractères :

'a'	'l'	'g'	'o'	'r'	'i'	't'	'h'	'm'	'e'

Un tableau t de taille n contenant des éléments de E est une application $t : \{1, \ldots, n\} \to E$.

Terminaison, correction et complexité

- Terminaison : pour prouver qu'une itération se termine, il suffit de trouver une quantité (un variant) qui :
  1. est un entier positif avant le début de l'itération
  2. est un entier positif à la fin de chaque étape de l'itération
  3. décroit strictement à chaque étape de l'itération

Terminaison, correction et complexité

- Terminaison : pour prouver qu'une itération se termine, il suffit de trouver une quantité (un variant) qui :
  1. est un entier positif avant le début de l'itération
  2. est un entier positif à la fin de chaque étape de l'itération
  3. décroit strictement à chaque étape de l'itération

- Correction : pour prouver qu'une itération produit un résultat, il suffit de trouver une propriété P (un invariant) telle que :
  1. (Entrée) P est vraie à la 1ère étape de l'itération
  2. (Récurrence) pour tout entier i, si P est vraie à la $i$ème étape, alors P est vraie à la $i+1$ème étape
  3. (Sortie) après la dernière étape, P et la condition d'itération devenue fausse doivent permettre de prouver que le résultat est celui attendu

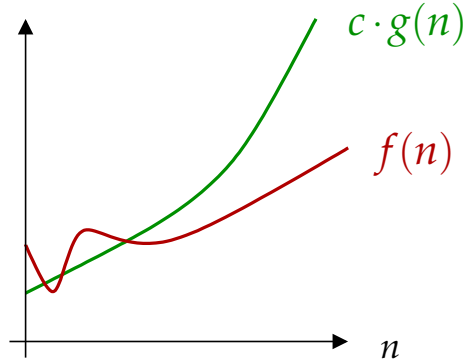
Terminaison, correction et complexité

- Terminaison : pour prouver qu'une itération se termine, il suffit de trouver une quantité (un variant) qui :
  1. est un entier positif avant le début de l'itération
  2. est un entier positif à la fin de chaque étape de l'itération
  3. décroit strictement à chaque étape de l'itération

- Correction : pour prouver qu'une itération produit un résultat, il suffit de trouver une propriété P (un invariant) telle que :
  1. (Entrée) P est vraie à la 1ère étape de l'itération
  2. (Récurrence) pour tout entier i, si P est vraie à la $i$ème étape, alors P est vraie à la $i+1$ème étape
  3. (Sortie) après la dernière étape, P et la condition d'itération devenue fausse doivent permettre de prouver que le résultat est celui attendu

- Complexité : $f(n)$ est en $O(g(n))$ s'il existe un entier n_0 et une constante $c > 0$ tels que

$$\forall n \geqslant n_0, f(n) \leqslant c \cdot g(n)$$



Parcours d'un tableau

- Objectif : réaliser un traitement sur l'ensemble des éléments d'un tableau
  ⟶ parcours séquentiel

Procédure parcours(**d** tab[1 . . . n] : **tab_t**)
    i : **entier** ;
Début
    $i := 1$;
    Tant que $i \leqslant n$ **faire**
        traiter(tab[i]) ; # p.ex. **écrire**
        $i := i + 1$;
    Fin faire
Fin

D♢	7♢	D♠	7♠	V♠	7♡	R♣

```
parcours(tab[1... 7]);
```

```
D♢
7♢
D♠
7♠
V♠
7♡
R♣
```

Parcours d'un tableau

- Objectif : réaliser un traitement sur l'ensemble des éléments d'un tableau
  $\longrightarrow$ parcours séquentiel

Procédure parcours(**d** tab$[1 \ldots n]$: **tab_t**)
    i : **entier** ;
Début
    $i := 1$;
    Tant que $i \leqslant n$ **faire**
        traiter(tab$[i]$) ; # p.ex. **écrire**
        $i := i + 1$;
    Fin faire
Fin

D♢	7♢	D♠	7♠	V♠	7♡	R♣

```
parcours(tab[1... 7]);
```

```
D♢
7♢
D♠
7♠
V♠
7♡
R♣
```

- Terminaison ? ok
- Correction ? ok
- Complexité ? $3n + 1$ opérations $\rightsquigarrow O(n)$

Recherche dans un tableau non trié

- Objectif : répondre vrai ou faux selon qu'un élément donné appartient ou non à un tableau
  $\longrightarrow$ recherche séquentielle

Recherche dans un tableau non trié

- Objectif : répondre vrai ou faux selon qu'un élément donné appartient ou non à un tableau
  $\longrightarrow$ recherche séquentielle

Fonction r(**d** tab$[1 \ldots n]$: **tab_t**, **d** e : **t**) : **booléen**
    i : **entier** ;
Début
    $i := 1$;
    Tant que $(i < n)$ **et** $(t[i] \neq e)$ **faire**
        $i := i + 1$;
    Fin faire
    retour tab$[i] = e$;
Fin

Procédure rech(**d** tab$[1 \ldots n]$: **tab_t**, **d** elem : **t**)
Début
    écrire(rech(tab$[1 \ldots n]$, elem)) ;
Fin

4	17	64	25	1	38	51

```
recherche(tab[1... 7], 32);
```

```
faux
```

```
recherche(tab[1... 7], 64);
```

```
vrai
```

Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?

Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

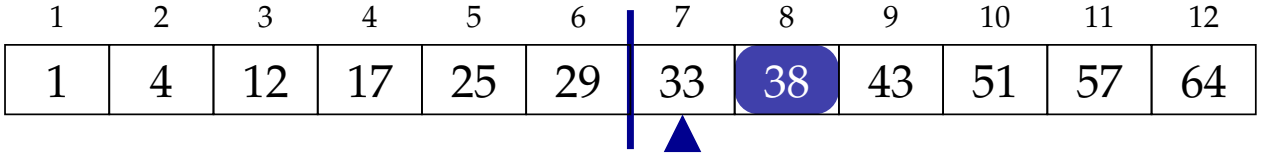
- Recherche séquentielle ? $\rightsquigarrow$ complexité en $O(n)$, peut mieux faire...

Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

- Recherche séquentielle ? $\rightsquigarrow$ complexité en $O(n)$, peut mieux faire...



Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

- Recherche séquentielle ? $\rightsquigarrow$ complexité en $O(n)$, peut mieux faire...

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

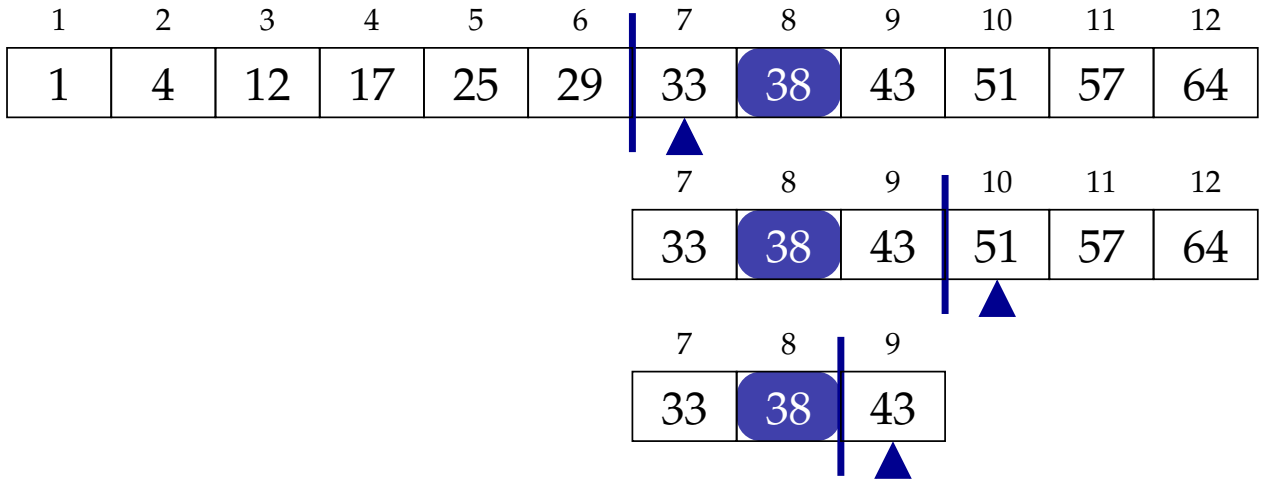
7	8	9	10	11	12
33	38	43	51	57	64

Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

- Recherche séquentielle ? $\rightsquigarrow$ complexité en $O(n)$, peut mieux faire...

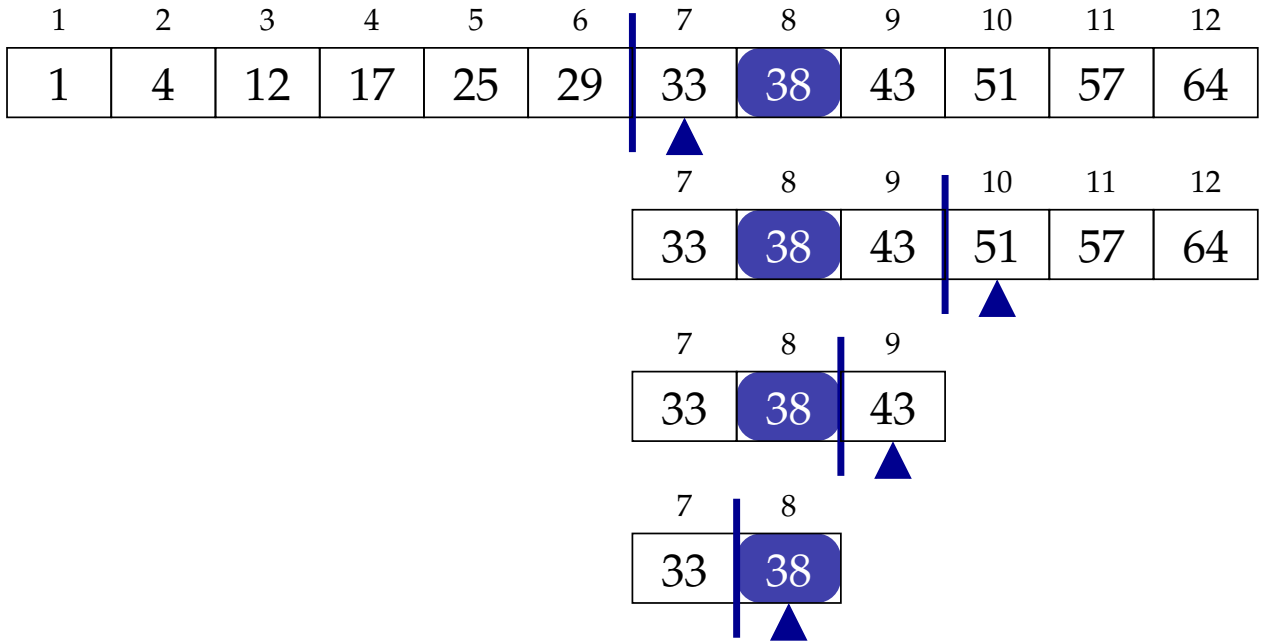


Recherche dans un tableau trié

- Question : comment faire dans un tableau trié, *i.e.* dont les éléments sont ordonnés ?
- Exemple : recherche de 38 dans le tableau suivant

1	2	3	4	5	6	7	8	9	10	11	12
1	4	12	17	25	29	33	38	43	51	57	64

- Recherche séquentielle ? $\rightsquigarrow$ complexité en $O(n)$, peut mieux faire...



Recherche dans un tableau trié

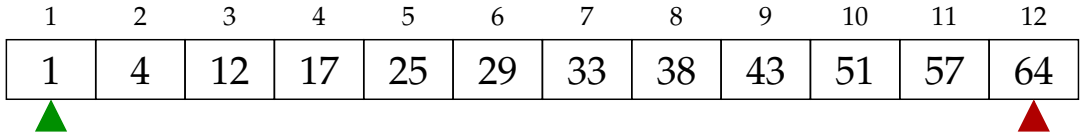
```
Fonction rechdich(d tab[1...n] : tab_t, d e : t) : booléen
    m, inf, sup : entier ;
    trouvé : booléen ;
Début
    trouvé := faux ;
    inf := 1 ;
    sup := n ;
    Tant que (inf ⩽ sup) et (trouvé = faux) faire
        m := ⌈(inf + sup)/2⌉ ;
        Si (t[m] = e) alors
            trouvé := vrai ;
        Sinon si (t[m] < e) alors
            inf := m + 1 ;
        Sinon
            sup := m − 1 ;
        Fin si
    Fin faire
    retour trouvé ;
Fin
```

Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent

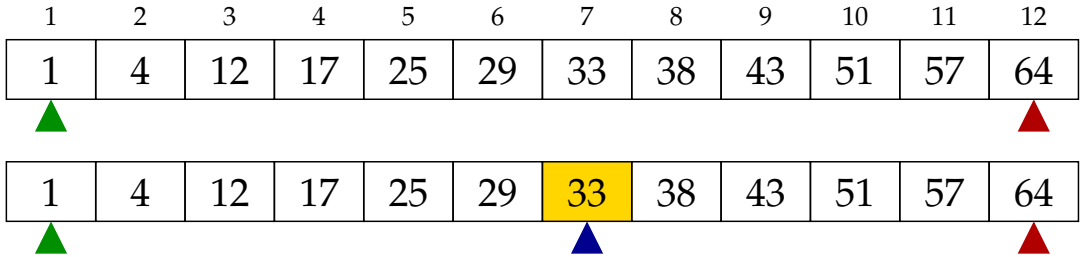
Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



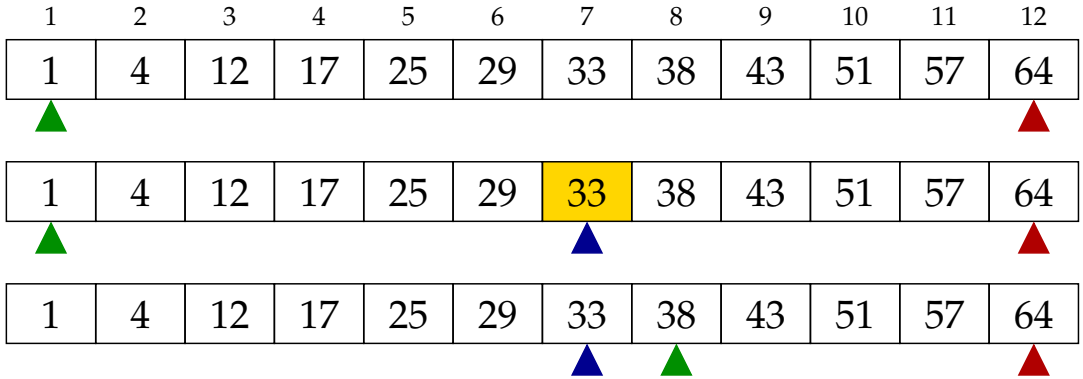
Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



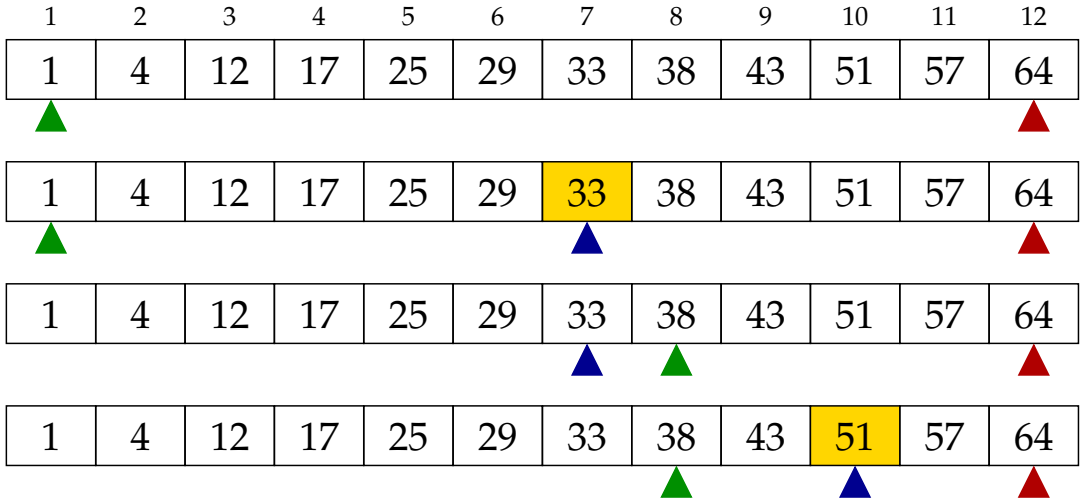
Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



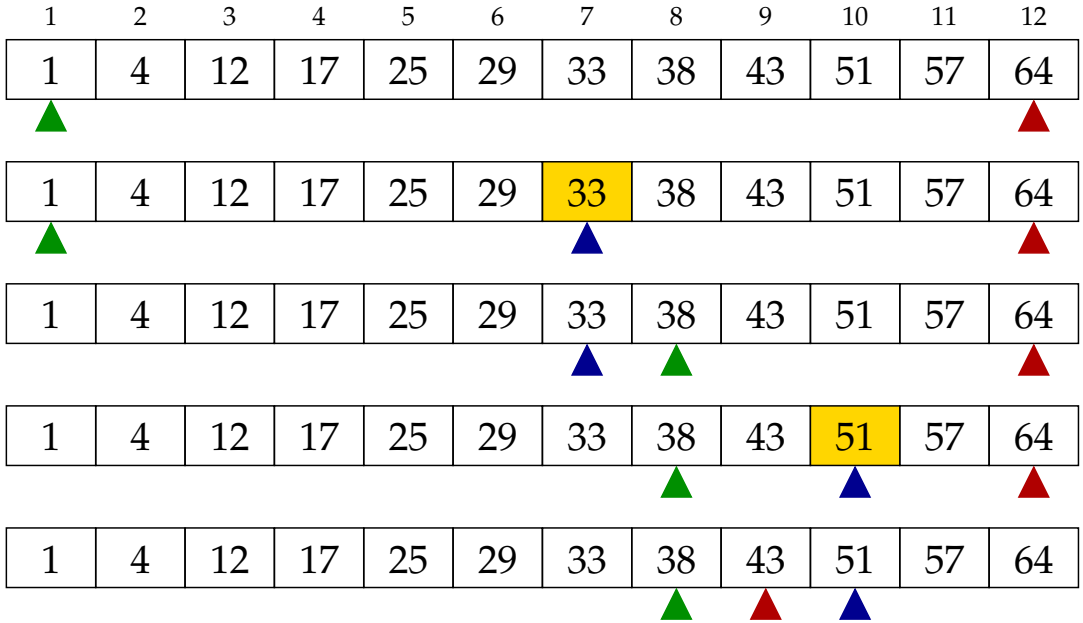
Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



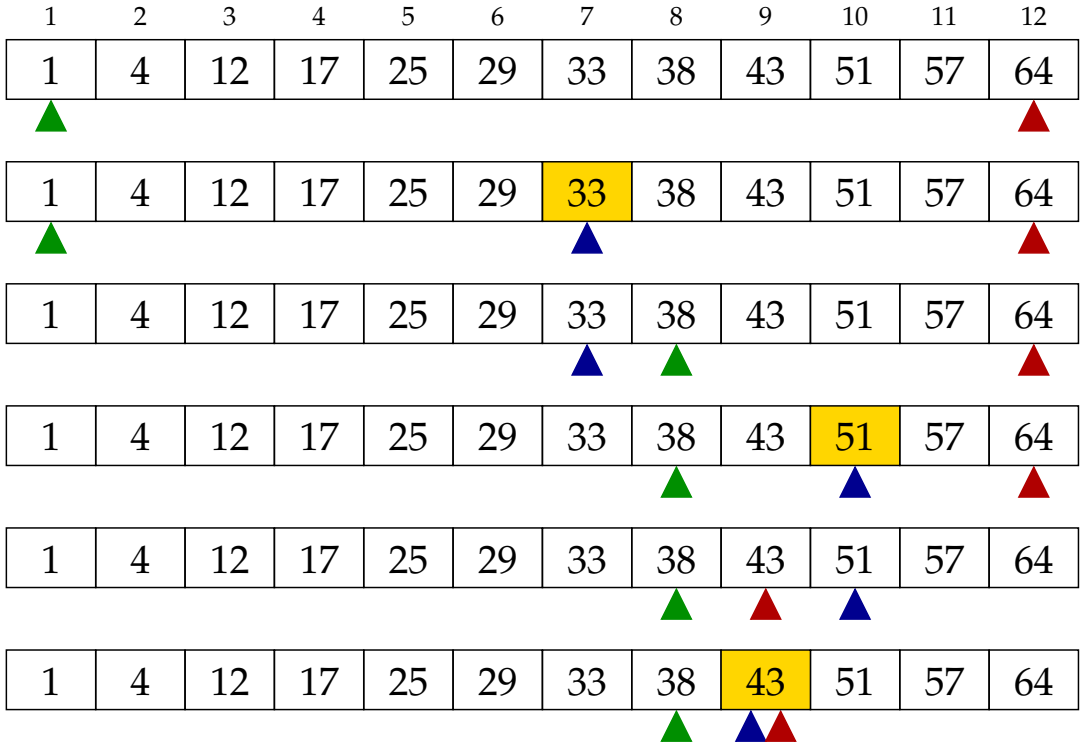
Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent



Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent

Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent

Recherche dans un tableau trié

- Recherche de 41 dans le tableau précédent