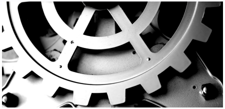


Algorithmique et structures de données
Recherche d'un élément dans un vecteur

Gaël Mahé

slides : Elise Bonzon et Gaël Mahé

Université Paris Descartes

Licence 2

Recherche d'un élément dans un vecteur

Recherche d'un élément dans un vecteur

Vecteur

Vecteur

Soit un ensemble E muni d'un ordre total.
Un vecteur de dimension n est une application :

$$V : [\![1, n]\!] \to E$$

Si $n = 0$, l'intervalle est vide.

Exemple : $V : [\![1, 10]\!] \to \mathbb{R}$

1	2	3	4	5	6	7	8	9	10
5.2	-2.6	3.8	25.3	-9.3	85.1	45.2	5.1	56.1	-7.1

Vecteur trié

Vecteur trié

Le vecteur $V : [\![1, n]\!] \rightarrow E$ est trié si

$$\forall i \in [\![1, n-1]\!],\ \ V(i) \leq V(i+1)$$

Si $n = 0$ ou $n = 1$, le vecteur est trié

Exemple : $V : [\![1, 10]\!] \rightarrow \mathbb{R}$

1	2	3	4	5	6	7	8	9	10
-9.3	-7.1	-2.6	3.8	5.1	5.2	25.3	45.2	56.1	85.1

Recherche d'un élément dans un vecteur

Recherche séquentielle dans un vecteur non trié

- Soit $V : [\![1, n]\!] \rightarrow E$ et $x \in E$.
  On cherche s'il existe un indice $i \in [\![1, n]\!]$ tel que $V(i) = x$
- Parcourir le vecteur : pour tout $i \in [\![1, n]\!]$
  - Si $V(i) = x$, renvoyer vrai
  - Sinon, si $i = n$, renvoyer faux
  - Sinon, $i \leftarrow i + 1$

1	$\dots$	i	$\dots$	n
		$x?$		

$V(j) \neq x$ Non testé

Explication de l'algorithme

1	...	i	...	n
		$x?$		

$\underbrace{\qquad\qquad}_{V(j) \neq x}$ $\qquad$ $\underbrace{\qquad\qquad}_{\text{Non testé}}$

- Hypothèses de la situation générale
  - les $i-1$ premiers éléments sont traités ($1 \leq i \leq n$)
  - $\forall j \in [\![1, i-1]\!]$, $x \neq V(j)$
- Progression vers la solution. Deux cas sont possibles :
  - $V(i) = x$, c'est fini, $x \in V$
  - $V(i) \neq x$, deux cas possibles :
    - $i = n$, c'est fini, $\forall j \in [\![1, n]\!]$, $x \neq V(j)$, $x \notin V$
    - $i < n$, En faisant $i \leftarrow i+1$, on retrouve l'assertion de l'hypothèse : $\forall j \in [\![1, i-1]\!]$, $x \neq V(j)$. On revient au premier point.
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Explication de l'algorithme

1	...	i	...	n
		$x?$		

$V(j) \neq x$ — Non testé

- Hypothèses de la situation générale
  - les $i-1$ premiers éléments sont traités ($1 \leq i \leq n$)
  - $\forall j \in [\![1, i-1]\!]$, $x \neq V(j)$
- Progression vers la solution. Deux cas sont possibles :
  - $V(i) = x$, c'est fini, $x \in V$
  - $V(i) \neq x$, deux cas possibles :
    - $i = n$, c'est fini, $\forall j \in [\![1, n]\!]$, $x \neq V(j)$, $x \notin V$
    - $i < n$, En faisant $i \leftarrow i+1$, on retrouve l'assertion de l'hypothèse : $\forall j \in [\![1, i-1]\!]$, $x \neq V(j)$. On revient au premier point.
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Algorithme

Algorithme 1 : Recherche séquentielle dans un vecteur non trié

début

/* ENTRÉES: Un vecteur V de taille n, un élément x */

/* SORTIE: vrai si $x \in V$, faux sinon */

$i \leftarrow 1$

tant que $i < n$ *et* $V(i) \neq x$ **faire** $i \leftarrow i + 1$

/* sortie de boucle */

si $V(i) = x$ **alors** **retourner** vrai

sinon **retourner** faux

fin

Schéma général...

... pour écrire un algorithme séquentiel efficient par construction :

- Définir l'hypothèse H de la situation générale, pour i quelconque.
- Progression vers la solution : **montrer** qu'à l'étape i, avec H vérifiée,
  - soit le problème est résolu si la condition C_1 ou C_2 ou... est vérifiée ; (note : une des conditions est du type "$i = n$")
  - soit on fait $i \leftarrow i + 1$ et H est vérifiée pour le nouveau i.
- S'assurer que H est vérifiée pour le i initial.

Algorithme

Algorithme 2 : Algorithme issu du schéma général

début
  Initialiser i
  tant que $\mathrm{non}(C_1 \vee C_2 \vee \ldots)$ **faire** $i \leftarrow i+1$
  /* sortie de boucle */
  Selon que C_1 ou C_2 ou ... est vérifiée,
  retourner résultat correspondant
fin

Recherche d'un élément dans un vecteur

Recherche séquentielle dans un vecteur trié

- Soit $V : [\![1, n]\!] \to E$ et $x \in E$.
  On cherche s'il existe un indice $i \in [\![1, n]\!]$ tel que $V(i) = x$
- Deux étapes
  - Chercher un indice i tel que :
    $\forall j \in [\![1, i-1]\!],\ V(j) < x$ et $\forall j \in [\![i, n]\!],\ V(j) \geq x$
  - Vérifier si $x = V(i)$

1	...	i	...	n
		x?		

$\underbrace{\qquad\qquad\qquad}_{V(j)<x}$ $\underbrace{\qquad\qquad\qquad}_{V(j)\geq x}$

Construction de l'algorithme

1	$\dots$	i	$\dots$	n

$\underbrace{\qquad\qquad\qquad\qquad}_{V(j)<x}$

- Hypothèses de la situation générale
  - les $i-1$ premiers éléments sont traités $(1 \leq i \leq n)$
  - $\forall j \in [\![1, i-1]\!],\ V(j) < x$
- Progression vers la solution. Deux cas sont possibles :
  - $V(i) \geq x$, c'est fini, on a trouvé i tel que :
    $\forall j \in [\![1, i-1]\!],\ V(j) < x$ et $\forall j \in [\![i, n]\!],\ V(j) \geq x$.
    Il reste à vérifier si $V(i) = x$
  - $V(i) < x$, deux cas possibles :
    - $i = n$, c'est fini, $\forall j \in [\![1, n]\!],\ V(j) < x,\ x \notin V$
    - $i < n$. En faisant $i \leftarrow i+1$, on retrouve l'assertion de l'hypothèse :
      $\forall j \in [\![1, i-1]\!],\ V(j) < x$. On revient au premier point.
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Construction de l'algorithme

1	...	i	...	n

$V(j) < x$

- Hypothèses de la situation générale
  - les $i-1$ premiers éléments sont traités ($1 \leq i \leq n$)
  - $\forall j \in [\![1, i-1]\!],\ V(j) < x$
- Progression vers la solution. Deux cas sont possibles :
  - $V(i) \geq x$, c'est fini, on a trouvé i tel que :
    $\forall j \in [\![1, i-1]\!],\ V(j) < x$ et $\forall j \in [\![i, n]\!],\ V(j) \geq x$.
    Il reste à vérifier si $V(i) = x$
  - $V(i) < x$, deux cas possibles :
    - $i = n$, c'est fini, $\forall j \in [\![1, n]\!],\ V(j) < x,\ x \notin V$
    - $i < n$. En faisant $i \leftarrow i+1$, on retrouve l'assertion de l'hypothèse :
      $\forall j \in [\![1, i-1]\!],\ V(j) < x$. On revient au premier point.
- Condition initiale : $i = 1$ satisfait les hypothèses de la situation générale

Algorithme

Algorithme 3 : Recherche séquentielle dans un vecteur trié

début

    /* ENTRÉES: Un vecteur V de taille n, un élément x */
    /* SORTIE: i si x apparait au rang i de V, 0 si $x \notin V$ */
    $i \leftarrow 1$
    tant que $V(i) < x$ *et* $i < n$ **faire** $i \leftarrow i + 1$
    /* sortie de boucle */
    si $V(i) = x$ **alors** **retourner** i
    sinon **retourner** 0

fin

Recherche d'un élément dans un vecteur

Recherche dichotomique

- Il est possible d'être beaucoup plus efficace si le vecteur est trié
- Méthode dichotomique :
  - Comparer l'élément x à une valeur située au milieu du vecteur
  - Si cette valeur est différente de x, continuer la recherche sur le demi-vecteur susceptible de contenir x
- Version itérative (nous verrons la version récursive plus tard)

Explication de l'algorithme

Partages successifs sur un vecteur **trié**

- Situation générale

1	...	*inf*	...	*med*	...	*sup*	...	*n*

$V(j) < x$ — A explorer — $V(j) > x$

$1 \leq inf \leq sup \leq n$

- on pose $med = \lfloor \frac{(inf + sup)}{2} \rfloor$
  2 cas possibles :
  - $V(med) = x$: arrêt et retour *med*
  - $V(med) \neq x$:
    - $V(med) > x$, $sup \leftarrow med - 1$
    - $V(med) < x$, $inf \leftarrow med + 1$

  Si $sup < inf$, arrêt et retour 0, sinon on recommence

- Conditions initiales : $inf = 1$, $sup = n$.
  Respectent les hypothèses de la situation générale.

Recherche dichotomique : algorithme

Algorithme 4 : Recherche dichotomique dans un vecteur trié

début

    /* ENTRÉES: Un vecteur V de taille n, un élément x */
    /* SORTIE: i si x apparait au rang i de V, 0 si $x \notin V$ */
    $inf \leftarrow 1$, $sup \leftarrow n$, $i \leftarrow 0$, $trouve' \leftarrow faux$
    tant que $inf \leq sup$ *et* $non(trouve')$ **faire**
        $med \leftarrow (inf + sup)$ **div** 2
        si $V(med) = x$ **alors**
            $i \leftarrow med$
            $trouve' \leftarrow vrai$
        sinon
            si $V(med) > x$ **alors** $sup \leftarrow med - 1$
            sinon $inf \leftarrow med + 1$
    retourner i

fin

Recherche dichotomique de la première occurrence

Si l'on cherche la **première occurrence** d'un élément x, l'algorithme doit renvoyer une position i telle que :

- $V(i) = x$;
- $\forall j < i, \quad V(j) \neq x$.

ou 0 si $x \notin V$.

Explication de l'algorithme

Partages successifs sur un vecteur **trié**

- Situation générale

1	...	*inf*	...	*med*	...	*sup*	...	*n*

$\underbrace{\qquad\qquad}_{V(j)<x}$ $\underbrace{\qquad\qquad\qquad\qquad}_{\text{A explorer}}$ $\underbrace{\qquad\qquad}_{V(j)\geq x}$

$1 \leq inf \leq sup \leq n$

- 2 cas possibles :
  - $inf = sup$: retourne inf si $V(inf) = x$, 0 sinon.
  - $inf < sup$: on pose $med = \lfloor \frac{(inf+sup)}{2} \rfloor$
    - $V(med) \geq x$, $sup \leftarrow med$
    - $V(med) < x$, $inf \leftarrow med + 1$
- Conditions initiales : $inf = 1$, $sup = n$.
  Respectent les hypothèses de la situation générale.

Recherche dichotomique de la 1ère occurrence : algorithme

Algorithme 5 : Recherche dichotomique d'une 1ère occurrence dans un vecteur trié

début

/* ENTRÉES: Un vecteur V de taille n, un élément x */
/* SORTIE: i si x apparait au rang i de V, 0 si $x \notin V$ */
$inf \leftarrow 1$, $sup \leftarrow n$, $i \leftarrow 0$
tant que $inf < sup$ **faire**
  $med \leftarrow (inf + sup)$ **div** 2
  si $V(med) \geq x$ **alors** $sup \leftarrow med$
  sinon $inf \leftarrow med + 1$
si $V(inf) = x$ **alors**
  retourner inf
sinon
  retourner 0

fin

Démonstration - partie A (1)

Montrons que l'algorithme se termine :

- Montrons qu'à chaque itération, $inf < sup$ ou on sort
  - Hypothèse de récurrence H_k : $inf_k < sup_k$ ou on sort
  - H_0 vraie
  - Supposons H_k
  - Donc $H_k \Rightarrow H_{k+1}$

Démonstration. - partie A (2)

- Montrons que la suite des $(sup_k - inf_k)$ est strictement décroissante
  - Si $V(med_k) \geq x$:
  - Si $V(med_k) < x$:
- Conclusion : la suite des $(sup_k - inf_k)$ est strictement décroissante et positive, de valeur initiale n, donc elle converge vers 0 en moins de n itérations.

Démonstration - partie B

Montrons que si $x \in V$, l'algo renvoie bien :

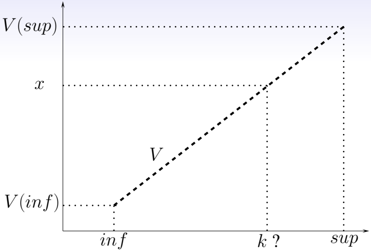
$$i \quad \text{tel que} \quad V(i) = x \quad \text{et} \quad \forall j < i, V(j) \neq x$$

- Hypothèse de récurrence H_k : $inf_k \leq i \leq sup_k$
- H_0 vraie (avec $inf_0 = 1$ et $sup_0 = n$)
- Supposons H_k vraie pour $k < p =$ numéro de la dernière itération
  - Si $V(med_k) \geq x$:
  - Si $V(med_k) < x$:
- Donc $H_k \Rightarrow H_{k+1}$
- Donc H_k vraie de 0 à p. Or pour $k = p$, on sort du while avec $inf = sup = i$. On retourne inf donc i.

Recherche d’un élément dans un vecteur

Recherche par interpolation

- La recherche par dichotomie cherche x autour de l'indice médian
- Si les éléments sont des nombres, chercher x autour de l'indice k tel que :



$$k = inf + ((sup - inf)(x - V(inf))\textbf{div}(V(sup) - V(inf)))$$

→ Même algorithme en remplaçant *med* par cette expression de k

- Comparaison interpolation / dichotomie :
  - Moins d'itérations nécessaires (voir suite)
  - Plus de calculs
  - Intéressante seulement si les éléments augmentent à peu près linéairement

Recherche d'un élément dans un vecteur

Complexité des algorithmes de recherche

Pour un vecteur trié,
Nombre de comparaisons moyen C pour un vecteur de taille n

- Recherche séquentielle : $C \propto n$
- Recherche par dichotomie : $C \propto \log_2(n)$
- Recherche par interpolation : $C \propto \log_2(\log_2(n))$
  (mais plus de calculs)