ALG2 : recherche d'éléments

loig.jezequel@univ-nantes.fr

Tableaux

Définition

Un tableau de taille n est une structure de donnée indicée de 0 à $n-1$ contenant n éléments (tous d'un même type).

Opérations

Pour un tableau t

- $t[i]$ représente l'élément d'indice i,
- $len(t)$ représente la taille de t.

Exemple

t =

12	32	7	23	9

- les éléments de t sont des entiers,
- $len(t) = 5$,
- $t[0] = 12,\ t[1] = 32,\ t[2] = 7,\ t[3] = 23,\ t[4] = 9$,
- $t[5]$ n'existe pas.

Les problèmes qu'on se pose

Problème : recherche d'élément

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Problème : recherche de minimum

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Problème : sélection d'éléments

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Exemples pratiques

Gestion d'une promotion d'étudiants

- Savoir si un étudiant appartient à un groupe.
- Trouver l'étudiant le plus jeune.
- Lister les étudiants qui ont eu la moyenne.

Gestion d'un ensemble de tâches à réaliser

- Savoir si une tâche a déjà été réalisée.
- Trouver la tâche la plus prioritaire.
- Lister les tâches qu'il reste à réaliser.

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 5$
- $x = 7$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =	12	32	7	23	9

- $x = 5$
- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ retourner $(true, i)$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =	12	32	7	23	9

- $x = 5$

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 5$

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$,
   retourne $(true, 2)$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =				
12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 \neq x$, puis $i = 3$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 \neq x$, puis $i = 3$
4. $t[i] = t[3] = 23 \neq x$, puis $i = 4$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 \neq x$, puis $i = 3$
4. $t[i] = t[3] = 23 \neq x$, puis $i = 4$
5. $t[i] = t[4] = 9 \neq x$, puis $i = 5$

Recherche d'élément

Problème

Étant donné un tableau t et un entier x, dire si x appartient à t et, si oui, donner un indice i tel que $t[i] = x$.

Algorithme

termine l'algorithme

Soit $i = 0$. Tant que $i < len(t)$, si $t[i] = x$ $\boxed{\text{retourner } (true, i)}$, sinon augmenter i de 1. Retourner $(false, -1)$.

Exemple

t =

12	32	7	23	9

- $x = 7$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 = x$, retourne $(true, 2)$

- $x = 5$

1. $t[i] = t[0] = 12 \neq x$, puis $i = 1$
2. $t[i] = t[1] = 32 \neq x$, puis $i = 2$
3. $t[i] = t[2] = 7 \neq x$, puis $i = 3$
4. $t[i] = t[3] = 23 \neq x$, puis $i = 4$
5. $t[i] = t[4] = 9 \neq x$, puis $i = 5$
6. $i \geq len(t)$, retourne $(false, -1)$

Recherche d'élément : nombre d'opérations

Pourquoi se poser la question du nombre d'opérations ?

Permet de comparer des algorithmes entre eux, savoir lequel sera le plus efficace en temps de calcul.

Quelles opérations ?

Affectations de variables, tests, opérations arithmétiques.

Remarque

Pour être vraiment précis il faudrait compter les instructions en langage machine car c'est cela qu'un processeur va exécuter au final.

- Dépend du langage, du compilateur, du processeur.
- Dépend de l'implantation de l'algorithme.

En général on calcule un ordre de grandeur du nombre d'opérations en fonction de la taille de l'entrée.

Recherche d'élément : nombre d'opérations, suite

affectation de variable

tests

Rappel de l'algorithme

Soit $\boxed{i = 0}$. Tant que $\boxed{i < len(t)}$, si $\boxed{t[i] = x}$ retourner $(true, i)$, sinon $\boxed{\text{augmenter } i}$ de 1. Retourner $(false, -1)$.

opération arithmétique

Nombre d'opérations

- 1 affectation de variable à l'initialisation.
- 2 tests et 1 opération arithmétique par tour de boucle.

Soit $3k + 1$ opérations, où k est le nombre de tours de boucle.

Nombre d'opérations dans le pire cas

On ne peut pas savoir à l'avance le nombre de tours de boucle effectués, on considère ce qui va se passer au pire : $k = len(t)$.

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

1. $t[i] = t[1] > t[0] = t[i_{min}]$, i_{min} ne change pas, puis $i = 2$,

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

1. $t[i] = t[1] > t[0] = t[i_{min}]$, i_{min} ne change pas, puis $i = 2$,
2. $t[i] = t[2] < t[0] = t[i_{min}]$, donc $i_{min} = 2$, puis $i = 3$,

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

1. $t[i] = t[1] > t[0] = t[i_{min}]$, i_{min} ne change pas, puis $i = 2$,
2. $t[i] = t[2] < t[0] = t[i_{min}]$, donc $i_{min} = 2$, puis $i = 3$,
3. $t[i] = t[3] > t[2] = t[i_{min}]$, i_{min} ne change pas, puis $i = 4$,

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

1. $t[i] = t[1] > t[0] = t[i_{min}]$, i_{min} ne change pas, puis $i = 2$,
2. $t[i] = t[2] < t[0] = t[i_{min}]$, donc $i_{min} = 2$, puis $i = 3$,
3. $t[i] = t[3] > t[2] = t[i_{min}]$, i_{min} ne change pas, puis $i = 4$,
4. $t[i] = t[4] > t[2] = t[i_{min}]$, i_{min} ne change pas, puis $i = 5$,

Recherche de minimum

Problème

Trouver le plus petit entier dans un tableau t, c'est-à-dire trouver un indice i_{min} tel que pour tout $i \in [0, len(t)[$ on a $t[i_{min}] \leq t[i]$.

Algorithme

Soit $i = 1$ et soit $i_{min} = 0$. Tant que $i < len(t)$, si $t[i] < t[i_{min}]$ poser $i_{min} = i$, dans tous les cas augmenter i de 1. Retourner i_{min}.

Exemple

t =

12	32	7	23	9

1. $t[i] = t[1] > t[0] = t[i_{min}]$, i_{min} ne change pas, puis $i = 2$,
2. $t[i] = t[2] < t[0] = t[i_{min}]$, donc $i_{min} = 2$, puis $i = 3$,
3. $t[i] = t[3] > t[2] = t[i_{min}]$, i_{min} ne change pas, puis $i = 4$,
4. $t[i] = t[4] > t[2] = t[i_{min}]$, i_{min} ne change pas, puis $i = 5$,
5. $i = 5 = len(t)$, on retourne $i_{min} = 2$.

Recherche de minimum : nombre d'opérations

Rappel de l'algorithme

Soit $\boxed{i = 1}$ et soit $\boxed{i_{min} = 0}$. Tant que $\boxed{i < len(t)}$, si $\boxed{t[i] < t[i_{min}]}$ poser $\boxed{i_{min} = i}$, dans tous les cas $\boxed{\text{augmenter } i}$ de 1. Retourner i_{min}.

Nombre d'opérations

- 2 opérations à l'initialisation.
- 3 opérations à chaque tour de boucle.
- 1 opération par tour de boucle où la conditionnelle a lieu.

Soit $3 \times (len(t) - 1) + k + 2$ opérations, avec k le nombre de tours de boucle où la conditionnelle a lieu.

Nombre d'opérations dans le pire cas

La conditionnelle a lieu tout le temps : $k = (len(t) - 1)$

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t = | 12 | 32 | 7 | 23 | 9 | et $x = 10$

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t = | 12 | 32 | 7 | 23 | 9 | et $x = 10$

12	32	7	23	9

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t = | 12 | 32 | 7 | 23 | 9 | et $x = 10$

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,
2. $t[i] = t[1] > 10$, E ne change pas, puis $i = 2$,

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t = | 12 | 32 | 7 | 23 | 9 | et $x = 10$

12	32	7	23	9

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,
2. $t[i] = t[1] > 10$, E ne change pas, puis $i = 2$,
3. $t[i] = t[2] \leq 10$, $E = \{7\}$, puis $i = 3$,

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t = | 12 | 32 | 7 | 23 | 9 | et $x = 10$

12	32	7	23	9

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,
2. $t[i] = t[1] > 10$, E ne change pas, puis $i = 2$,
3. $t[i] = t[2] \leq 10$, $E = \{7\}$, puis $i = 3$,
4. $t[i] = t[3] > 10$, E ne change pas, puis $i = 4$,

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t =

12	32	7	23	9

et $x = 10$

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,
2. $t[i] = t[1] > 10$, E ne change pas, puis $i = 2$,
3. $t[i] = t[2] \leq 10$, $E = \{7\}$, puis $i = 3$,
4. $t[i] = t[3] > 10$, E ne change pas, puis $i = 4$,
5. $t[i] = t[4] \leq 10$, $E = \{7, 9\}$, puis $i = 5$,

Sélection d'éléments

Problème

Étant donné un tableau t et un entier x, trouver tous les éléments de t qui sont plus petits que x. On veut construire l'ensemble E tel que y appartient à E si et seulement si y appartient à t et $y \leq x$.

Algorithme

Soit $i = 0$ et soit $E = \emptyset$. Tant que $i < len(t)$, si $t[i] \leq x$ ajouter $t[i]$ dans E, dans tous les cas augmenter i de 1. Retourner E.

Exemple

t =

12	32	7	23	9

et $x = 10$

1. $t[i] = t[0] > 10$, E ne change pas, puis $i = 1$,
2. $t[i] = t[1] > 10$, E ne change pas, puis $i = 2$,
3. $t[i] = t[2] \leq 10$, $E = \{7\}$, puis $i = 3$,
4. $t[i] = t[3] > 10$, E ne change pas, puis $i = 4$,
5. $t[i] = t[4] \leq 10$, $E = \{7, 9\}$, puis $i = 5$,
6. $i = 5 = len(t)$, on retourne $E = \{7, 9\}$.

Sélection d'éléments : nombre d'opérations

affectation ?

Rappel de l'algorithme

Soit $i = 0$ et soit $\boxed{E = \emptyset}$. Tant que $i < len(t)$, si $t[i] \leq x$ $\boxed{\text{ajouter } t[i] \text{ dans } E}$, dans tous les cas augmenter i de 1. Retourner E.

affectation ?

Nombre d'opérations et nombre d'opérations dans le pire cas

Similaires à la recherche de minimum à condition que les opérations sur les ensemble soient simples.

Peut-on faire mieux ?

Bilan des nombres d'opérations nécessaires

Recherche d'élément. $3 \times len(t) + 1$

Recherche de minimum et sélection d'éléments. $4 \times len(t) - 2$

Ordres de grandeur

Les constantes peuvent varier dans ces nombres d'opérations (en fonction de l'implantation de l'algorithme, du compilateur, du processeur) mais ils restent linéaires en la taille du tableau.

Des algorithmes pour réduire l'ordre de grandeur

On ne peut pas faire mieux sur des tableaux quelconques, mais si on sait que les tableaux qu'on aura en entrée ont certaines propriétés c'est différent :

- borne sur la valeur maximum qu'ils contiennent,
- valeurs triées,
- etc

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

12	32	7	23	9	11	29

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

7	9	11	12	23	29	32

- Comment trouver un élément en regardant strictement moins de 7 cases ? (pire cas de l'algorithme vu précédemment)

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

7	9	11	12	23	29	32

- Comment trouver un élément en regardant strictement moins de 7 cases ? (pire cas de l'algorithme vu précédemment)
- Recherche dichotomique : exemple pour $x = 11$

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

7	9	11	12	23	29	32

- Comment trouver un élément en regardant strictement moins de 7 cases ? (pire cas de l'algorithme vu précédemment)
- Recherche dichotomique : exemple pour $x = 11$
  1. indices $[0, 7[$, le milieu est 3 et $t[3] = 12 > x$, x ne peut être trouvé que dans les indices $[0, 2]$,

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

7	9	11	12	23	29	32

- Comment trouver un élément en regardant strictement moins de 7 cases ? (pire cas de l'algorithme vu précédemment)
- Recherche dichotomique : exemple pour $x = 11$
  1. indices $[0, 7[$, le milieu est 3 et $t[3] = 12 > x$, x ne peut être trouvé que dans les indices $[0, 2]$,
  2. indices $[0, 3[$, le milieu est 1 et $t[1] = 9 < x$, x ne peut être trouvé qu'à l'indice 2,

Recherche d'élément dans un tableau trié

Tableau trié

Étant donnés un tableau t et un ordre total $\leq$ sur les éléments du tableau, on dit que t est un tableau trié si et seulement si $\forall i < j \in [0, len(t)[, t[i] \leq t[j]$.

Remarque

Dans ce cours on utilise des tableaux d'entiers, et on prendra la relation *plus petit ou égal* pour les trier.

Recherche efficace dans un tableau trié

t =

7	9	11	12	23	29	32

- Comment trouver un élément en regardant strictement moins de 7 cases ? (pire cas de l'algorithme vu précédemment)
- Recherche dichotomique : exemple pour $x = 11$
  1. indices $[0, 7[$, le milieu est 3 et $t[3] = 12 > x$, x ne peut être trouvé que dans les indices $[0, 2]$,
  2. indices $[0, 3[$, le milieu est 1 et $t[1] = 9 < x$, x ne peut être trouvé qu'à l'indice 2,
  3. $t[2] = 11 = x$, on a trouvé.

Recherche dichotomique d'élément dans un tableau trié

Algorithme, recherche dichotomique de x dans t trié

Soit $debut = 0$, soit $fin = len(t)$.
Tant que $debut < fin$,
  soit $milieu = (debut + fin)/2$,
  si $t[milieu] = x$, retourner $(true, milieu)$,
  sinon si $t[milieu] > x$, poser $fin = milieu$,
  sinon (si $t[milieu] < x$), poser $debut = milieu + 1$.
Retourner $(false, -1)$.

Exemple

7	9	11	12	23	29	32

on recherche $x = 11$.

Recherche dichotomique d'élément dans un tableau trié

Algorithme, recherche dichotomique de x dans t trié

Soit $debut = 0$, soit $fin = len(t)$.
Tant que $debut < fin$,
  soit $milieu = (debut + fin)/2$,
  si $t[milieu] = x$, retourner $(true, milieu)$,
  sinon si $t[milieu] > x$, poser $fin = milieu$,
  sinon (si $t[milieu] < x$), poser $debut = milieu + 1$.
Retourner $(false, -1)$.

Exemple

7	9	11	12	23	29	32

on recherche $x = 11$.

1. $milieu = (0 + 7)/2 = 3$, $t[3] = 12 > x$, donc *debut* ne change pas et $fin = 3$.

Recherche dichotomique d'élément dans un tableau trié

Algorithme, recherche dichotomique de x dans t trié

Soit $debut = 0$, soit $fin = len(t)$.
Tant que $debut < fin$,
  soit $milieu = (debut + fin)/2$,
  si $t[milieu] = x$, retourner $(true, milieu)$,
  sinon si $t[milieu] > x$, poser $fin = milieu$,
  sinon (si $t[milieu] < x$), poser $debut = milieu + 1$.
Retourner $(false, -1)$.

Exemple

7	9	11	12	23	29	32

on recherche $x = 11$.

1. $milieu = (0 + 7)/2 = 3$, $t[3] = 12 > x$, donc $debut$ ne change pas et $fin = 3$.
2. $milieu = (0 + 3)/2 = 1$, $t[1] = 9 < x$, donc $debut = 1 + 1 = 2$ et fin ne change pas.

Recherche dichotomique d'élément dans un tableau trié

Algorithme, recherche dichotomique de x dans t trié

Soit $debut = 0$, soit $fin = len(t)$.
Tant que $debut < fin$,
    soit $milieu = (debut + fin)/2$,
    si $t[milieu] = x$, retourner $(true, milieu)$,
    sinon si $t[milieu] > x$, poser $fin = milieu$,
    sinon (si $t[milieu] < x$), poser $debut = milieu + 1$.
Retourner $(false, -1)$.

Exemple

7	9	11	12	23	29	32

on recherche $x = 11$.

1. $milieu = (0+7)/2 = 3$, $t[3] = 12 > x$, donc $debut$ ne change pas et $fin = 3$.
2. $milieu = (0+3)/2 = 1$, $t[1] = 9 < x$, donc $debut = 1+1 = 2$ et fin ne change pas.
3. $milieu = (2+3)/2 = 2$, $t[2] = 11 = x$, donc on retourne $(true, 2)$.

Recherche dichotomique : nombre d'opérations

Rappel de l'algorithme

Soit $\boxed{debut = 0}$, soit $\boxed{fin = len(t)}$.

Tant que $\boxed{debut < fin}$,

soit $\boxed{milieu = (debut + fin)/2}$,

si $\boxed{t[milieu] = x}$, retourner $(true, milieu)$,

sinon si $\boxed{t[milieu] < x}$, poser $\boxed{fin = milieu}$, pas au même tour

sinon (si $t[milieu] > x$), poser $\boxed{debut = milieu + 1}$.

Retourner $(false, -1)$.

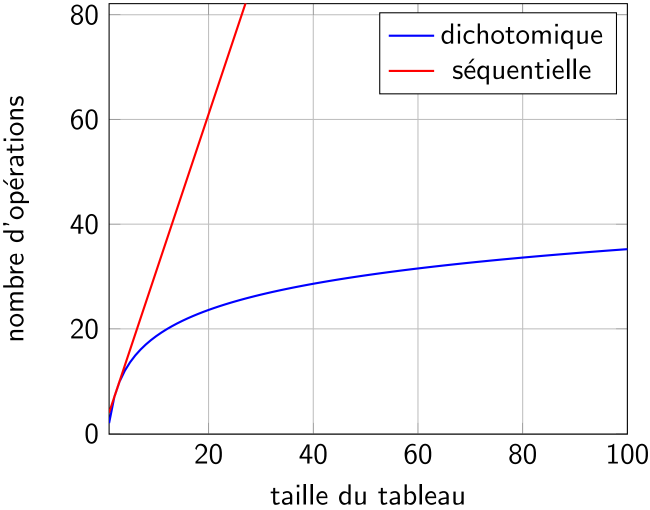
Nombre d'opérations dans le pire cas

$5 \times k + 2$ où k est le nombre de tours de boucle dans le pire cas.

Tours de boucle dans le pire cas

À chaque tour on divise par 2 la taille de l'intervalle $[debut, fin[$ et sa taille initiale est $len(t)$. Donc on fait $\log_2(len(t))$ tours.

Recherche dichotomique/séquentielle dans un tableau trié



Remarques finales

Recherche de minimum dans un tableau trié

Simplement prendre la valeur de la première case du tableau.

Sélection d'éléments dans un tableau trié

Chercher le plus grand élément à sélectionner par dichotomie, puis prendre tous ceux qui sont avant lui dans le tableau.

Trier un tableau

Pour vraiment comparer le nombre d'opérations nécessaires à la recherche séquentielle et à la recherche dichotomique, il faut prendre en compte le tri du tableau :

- pour chercher une fois un élément ce n'est pas rentable de trier,
- si on doit très souvent chercher des éléments ça devient rentable.

Les algorithmes de tri seront le sujet d'un prochain cours.