I21: Introduction à l'algorithmique

Cours 6: Algorithmes de recherche

Nicolas Méloni

Licence 1: 2ème semestre
(2017/2018)

Problème de la recherche

- But : trouver un élément donné dans un tableau d'éléments
- Deux cas de figure principaux :
  - les éléments sont rangés aléatoirement ;
  - les éléments sont triés.

Dans le premier cas on ne peut pas faire mieux que $\Theta(n)$ (n la taille du tableau). Si le tableau est trié on peut faire beaucoup mieux.

Problème : Recherche d'un élément

Entrée : tableau d'entiers T de taille n et un nombre x.
Sortie : l'indice de x dans le tableau s'il s'y trouve ou 0 sinon.

Problème de la recherche

```
ALGORITHME RechercheSeq(T,x):
DONNEES
   T: tableau d'entiers de taille n
   x: entier
VARIABLES
   i: entier
DEBUT
   i ← 1
   TQ i ⩽ n ET T[i]≠x FAIRE
      i ← i+1
   FTQ
   SI i = n ALORS
      RENVOYER 0
   SINON
      RENVOYER i
FIN
```

- Arrêt : la suite des valeurs prises par i est strictement croissante
- Validité : $(x \notin T[1 : i-1])$ est un invariant de boucle
- Complexité :
  - Meilleur cas : $\check{C}(n) = \Theta(1)$
  - Pire cas : $\hat{C}(n) = \Theta(n)$
  - Cas général : $C(n) = O(n)$

Problème de la recherche

Problème : Recherche dans un tableau trié

Entrée : tableau d'entiers T de taille n contenant des nombres entiers triés dans l'ordre croissant et un nombre x.
Sortie : l'indice de x dans le tableau s'il s'y trouve ou 0 sinon.

Problème de la recherche

Idée générale :

- Séparer le tableau en deux moitiés ;
- comparer x avec l'élément au milieu du tableau ;
- déterminer à quelle moitié il appartient ;
- recommencer avec le sous tableau choisi.

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99	$x = 50$
7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99	
7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99	

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99
7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99
7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99
7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

Recherche dichotomique

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

$x = 50$

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

7	11	19	23	27	28	31	35	39	40	42	46	50	71	79	99

Recherche dichotomique

```
ALGORITHME Dichotomie(T,x):
DONNEES
  T: tableau d'entiers tries
     de taille n
VARIABLES
  g,d,m: entiers
DEBUT
  g,d←1,n
  TQ g ⩽ d FAIRE
    m←⌊(g+d)/2⌋
    SI T[m] < x ALORS
      d←m−1
    SINON SI T[m] > x ALORS
      g←m+1
    SINON
      RENVOYER m
    FSI
  FTQ
  RENVOYER 0
FIN
```

Recherche dichotomique

```
ALGORITHME Dichotomie(T,x):
DONNEES
  T: tableau d'entiers tries
     de taille n
VARIABLES
  g,d,m: entiers
DEBUT
  g,d←1,n
  TQ g ⩽ d FAIRE
    m←⌊(g+d)/2⌋
    SI T[m] < x ALORS
      d←m−1
    SINON SI T[m] > x ALORS
      g←m+1
    SINON
      RENVOYER m
    FSI
  FTQ
  RENVOYER 0
FIN
```

- Arrêt : la suite des valeurs prises par $d-g$ est strictement décroissante
- Validité : $(T[g] \leqslant x \leqslant T[d])$ est un invariant
- Complexité :
  - Meilleur cas :$\check{C}(n) = \Theta(1)$
  - Pire cas :$\widehat{C}(n) = \Theta(\log(n))$
  - Globale :$C(n) = O(\log(n))$

Définition

On appelle **pic** tout élément $T[i]$ d'un tableau T vérifiant

$$\begin{cases} T[i-1] \leqslant T[i] \geqslant T[i+1] \text{ si } 2 \leqslant i \leqslant n-1 \\ T[i] \geqslant T[i+1] \text{ si } i = 1 \\ T[i-1] \leqslant T[i] \text{ si } i = n \end{cases}$$

Recherche de pic

Définition

On appelle **pic** tout élément $T[i]$ d'un tableau T vérifiant

$$\begin{cases} T[i-1] \leqslant T[i] \geqslant T[i+1] \text{ si } 2 \leqslant i \leqslant n-1 \\ T[i] \geqslant T[i+1] \text{ si } i = 1 \\ T[i-1] \leqslant T[i] \text{ si } i = n \end{cases}$$

- Exemple : $T = [4, 3, 2, 3, 4, 5, 4, 3, 3]$ alors $T[1], T[6]$ et $T[9]$ sont des pics.

Recherche de pic

Définition

On appelle **pic** tout élément $T[i]$ d'un tableau T vérifiant

$$\begin{cases} T[i-1] \leqslant T[i] \geqslant T[i+1] \text{ si } 2 \leqslant i \leqslant n-1 \\ T[i] \geqslant T[i+1] \text{ si } i = 1 \\ T[i-1] \leqslant T[i] \text{ si } i = n \end{cases}$$

- Exemple : $T = [4, 3, 2, 3, 4, 5, 4, 3, 3]$ alors $T[1], T[6]$ et $T[9]$ sont des pics.

Propriété

- Un tableau d'entiers contient toujours au moins un pic.

- Par récurrence on montre que si un sous tableau $T[1:i]$ ne contient pas de pic alors $T[1:i]$ est trié dans l'ordre croissant.

Recherche de pic

```
ALGORITHME PicSequentiel(T):
DONNEES
  T: tableau d'entiers de taille n⩾2
VARIABLE
  i: entier
DEBUT
  SI T[1] ⩾ T[2] ALORS
    RENVOYER 1
  SI T[n] ⩾ T[n−1] ALORS
    RENVOYER n
  i ← 2
  TQ i ⩽ n−1 FAIRE
    SI T[i−1]⩽T[i]⩾T[i+1] ALORS
      RENVOYER i
    FSI
    i ← i+1
  FTQ
FIN
```

- Arrêt : la suite des valeurs prises par i est strictement croissante
- Validité : ($T[1:i-1]$ ne contient pas de pic) est un invariant de boucle
- Complexité :
  - Meilleur cas : $\check{C}(n) = \Theta(1)$
  - Pire cas : $\hat{C}(n) = \Theta(n)$
  - Cas général : $C(n) = O(n)$

Approche par dichotomie

- on divise le tableau en deux parties ;
- on cherche un critère pour s'assurer qu'il existe un pic dans un des deux sous-tableaux.

Approche par dichotomie

- on divise le tableau en deux parties ;
- on cherche un critère pour s'assurer qu'il existe un pic dans un des deux sous-tableaux.

Posons $m = \lfloor (n+1)/2 \rfloor$ l'indice de milieu de talbeau, si $T[m]$ n'est pas un pic alors

Approche par dichotomie

- on divise le tableau en deux parties ;
- on cherche un critère pour s'assurer qu'il existe un pic dans un des deux sous-tableaux.

Posons $m = \lfloor (n+1)/2 \rfloor$ l'indice de milieu de talbeau, si $T[m]$ n'est pas un pic alors

- soit $T[m] < T[m+1]$ et il existe un pic dans $T[m+1 : n]$;

Approche par dichotomie

- on divise le tableau en deux parties ;
- on cherche un critère pour s'assurer qu'il existe un pic dans un des deux sous-tableaux.

Posons $m = \lfloor (n+1)/2 \rfloor$ l'indice de milieu de talbeau, si $T[m]$ n'est pas un pic alors

- soit $T[m] < T[m+1]$ et il existe un pic dans $T[m+1:n]$;
- soit $T[m] < T[m-1]$ et il existe un pic dans $T[1:m-1]$.

Recherche de pic

```
ALGORITHME PicDichotomie(T):
DONNEES
  T: tableau d'entiers de taille n
VARIABLES

DEBUT
  g,d←1,n
  TQ g ⩽ d FAIRE
    m← ⌊(g+d)/2⌋
    SI m=1 ou m=n ALORS
      RENVOYER m
    SINON SI T[m]<T[m+1] ALORS
      g←m+1
    SINON SI T[m]<T[m−1] ALORS
      d←m−1
    SINON
      RENVOYER m
  FTQ
FIN
```

Recherche de pic

```
ALGORITHME PicDichotomie(T):
DONNEES
  T: tableau d'entiers de taille n
VARIABLES

DEBUT
  g,d←1,n
  TQ g ⩽ d FAIRE
    m← ⌊(g+d)/2⌋
    SI m=1 ou m=n ALORS
      RENVOYER m
    SINON SI T[m]<T[m+1] ALORS
      g←m+1
    SINON SI T[m]<T[m−1] ALORS
      d←m−1
    SINON
      RENVOYER m
  FTQ
FIN
```

- Arrêt : la suite des valeurs prises par $d-g$ est strictement décroissante
- Validité : ($T[g:d$ contient un pic) est un invariant
- Complexité :
  - Meilleur cas : $\check{C}(n) = \Theta(1)$
  - Pire cas : $\hat{C}(n) = \Theta(\log(n))$
  - Cas général : $C(n) = O(\log(n))$