

Recherche du k-ème élément

le problème du «k^ee»

Données:

- Un tableau **T** de taille $n \geq 1$
- Des éléments tous distincts deux à deux
- Un entier **k**

Résultat:

L'élément x de T tel que:

$$|\{y \in T \mid y < x\}| = k-1$$

$$|\{y \in T \mid y > x\}| = n-k$$

la médiane

Données:

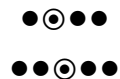
- Un tableau **T** de taille $n \geq 1$
- Des éléments tous distincts deux à deux

Résultat:

l'élément x de T tel que:

$$|\{y \in T \mid y \leq x\}| = \lceil n/2 \rceil$$

$$|\{y \in T \mid y > x\}| = \lfloor n/2 \rfloor$$



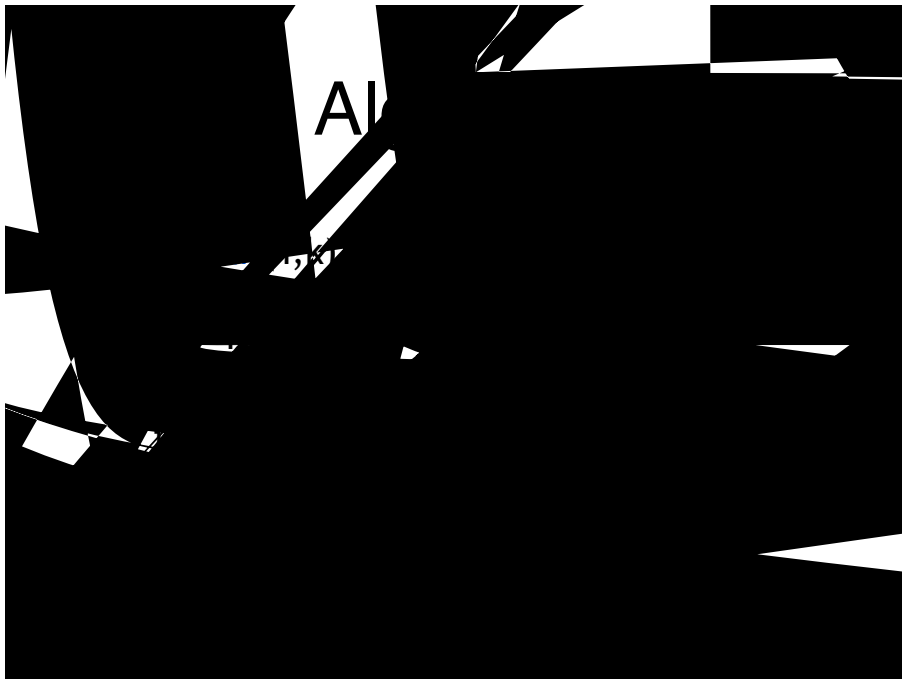
5 algorithmes

- Algo «naif»
- Algo «naif 2»
- Algo «select» (quickselect»)
- Algo «select opt»
- Algo par arbre min

Algo naif

$|T| = n$

```
def algonatif(T,k) :  
    if k > n : erreur  
    for i = 1..k :  
        indmin = indiceMin(T)  
        if i < k : T[indmin] =  $\infty$   
    return T[indmin]
```



Algo naif 2

- Trier $T[1..n]$
- retourner $T[k]$

Algorithme Select (ou quickselect)

Algo naif 2

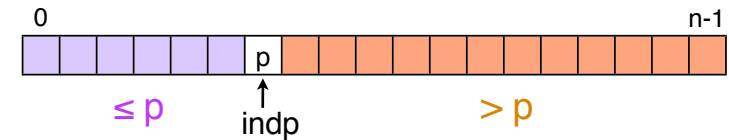
- Trier $T[1..n]$
- retourner $T[k]$

Complexité: $O(n \log n)$

Algorithme select

Basé sur le quicksort («quickselect»).

On **pivote** T selon un pivot p :



On continue en cherchant le...

- k^{eme} élément sur la **partie gauche** si $k < \text{indp} + 1$
- $k - \text{indp}^{\text{eme}}$ élément sur la **partie droite** si $k > \text{indp} + 1$

et on s'arrête si $k = \text{indp} + 1$!

Algorithme **select**

inv: $k \in \{1, \dots, bd - bg + 1\}$
et $bg \leq bd$

```
def select(T,k,bg,bd) :  
    indp = indicepivot(T,bg,bd)  
    rangpivot = indp-bg+1  
    if (k<rangpivot) :  
        return select(T,k,bg,indp-1)  
    elif (k>rangpivot) :  
        return select(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

Exercice: vérifier l'invariant...

Algorithme **select**

inv: $k \in \{1, \dots, bd - bg + 1\}$
et $bg \leq bd$

```
def select(T,k,bg,bd) :  
    indp = indicepivot(T,bg,bd)  
    rangpivot = indp-bg+1      rang du pivot dans T[bg..bd]  
    if (k<rangpivot) :  
        return select(T,k,bg,indp-1)  
    elif (k>rangpivot) :  
        return select(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

Algorithme **select**

```
def select(T,k,bg,bd) :  
    indp = indicepivot(T,bg,bd)  
    rangpivot = indp-bg+1  
    if (k<rangpivot) :  
        return select(T,k,bg,indp-1)  
    elif (k>rangpivot) :  
        return select(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

jeudi 18 octobre 2012

11

Algorithme «**Select-opt**»

jeudi 18 octobre 2012

12

Algorithme **select**

```
def select(T,k,bg,bd) :  
    indp = indicepivot(T,bg,bd)  
    rangpivot = indp-bg+1  
    if (k<rangpivot) :  
        return select(T,k,bg,indp-1)  
    elif (k>rangpivot) :  
        return select(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

Complexité: **$O(n^2)$**

cas pire: recherche du plus petit élé^t dans T
trié dans l'ordre décroissant.

jeudi 18 octobre 2012

11

Algo «select-opt»

(proposé par Blum, Floyd, Pratt, Rivest et Tarjan en 1973)

- Si n est petit, utiliser l'algo **select**.
- Sinon:
 - diviser les n éléments en $\lfloor n/5 \rfloor$ blocs B_i de 5 éléments.
 - **trouver le médian** m_i de chaque B_i
 - **trouver le médian** M des m_i
 - **pivoter** avec M comme pivot
 - continuer dans la bonne partie du tableau (comme dans **select**)...

Idée: garantir que la prochaine étape se fera sur
une **fraction** du tableau initial...

jeudi 18 octobre 2012

13

Algo «select-opt»

```
def selectopt(T,k,bg,bd) :  
    n = bd - bg + 1  
    if (n < 50) : return select(T,k,bg,bd)[0]  
  
    Taux = [T[IndexMedianQuintuplet(T,bg+j*5)] for j = 1... ⌊n/5⌋ ]  
  
    m = selectopt(Taux, ⌈|Taux|/2⌉ )  
    indp = indicepivotfixe(T,m,bg,bd)  
    rangpivot = indp-bg+1  
  
    if (k < rangpivot) : return selectopt(T,k,bg,indp-1)  
    elif (k > rangpivot) : return selectopt(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

jeudi 18 octobre 2012

14

Algo «select-opt»

```
def selectopt(T,k,bg,bd) :  
    n = bd - bg + 1  
    if (n < 50) : return select(T,k,bg,bd)[0]  
  
    Taux = [T[IndexMedianQuintuplet(T,bg+j*5)] for j = 1... ⌊n/5⌋ ]  
  
    m = selectopt(Taux, ⌈|Taux|/2⌉ )  
    indp = indicepivotfixe(T,m,bg,bd)  
    rangpivot = indp-bg+1  
  
    if (k < rangpivot) : return selectopt(T,k,bg,indp-1)  
    elif (k > rangpivot) : return selectopt(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

l'ind. du médian de

retourne

Algo «select-opt»

```
def selectopt(T,k,bg,bd) :  
    n = bd - bg + 1  
    if (n < 50) : return select(T,k,bg,bd)[0]  
  
    Taux = [T[IndexMedianQuintuplet(T,bg+j*5)] for j = 1... ⌊n/5⌋ ]  
  
    m = selectopt(Taux, ⌈|Taux|/2⌉ )  
    indp = indicepivotfixe(T,m,bg,bd)  
    rangpivot = indp-bg+1  
  
    if (k < rangpivot) : return selectopt(T,k,bg,indp-1)  
    elif (k > rangpivot) : return selectopt(T,k-rangpivot,indp+1,bd)  
    else :  
        return T[indp]
```

l'ind. du médian de

retourne

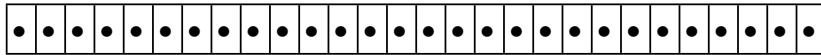
jeudi 18 octobre 2012

14

Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

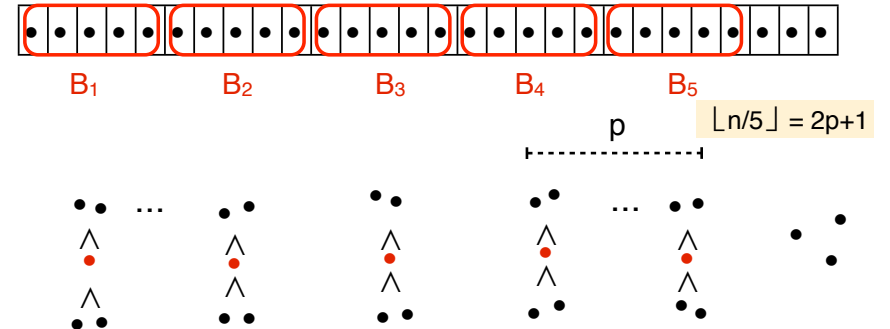
Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?



Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

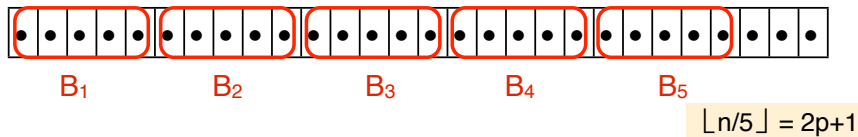
Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?



Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

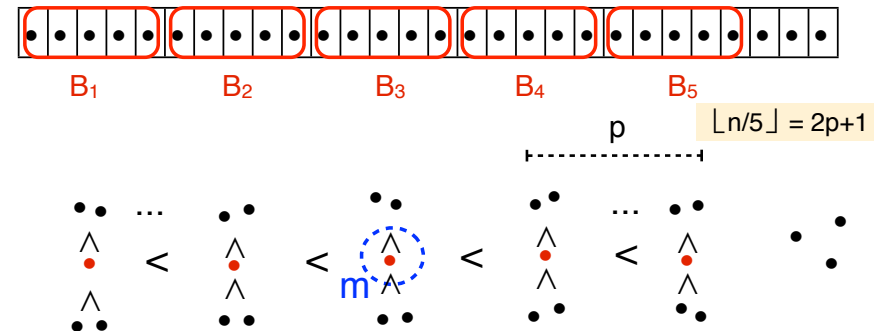
Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?



Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?



Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

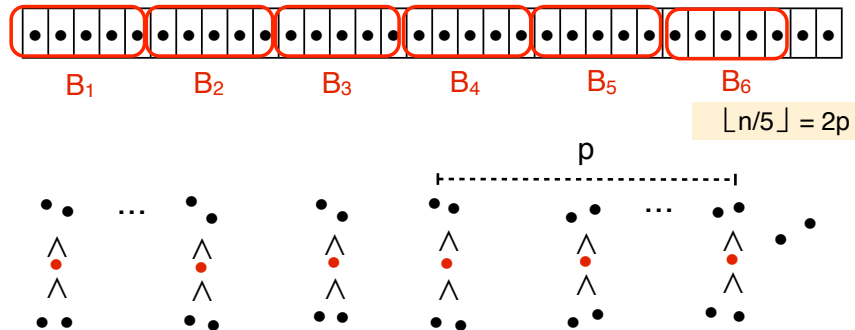
Comment évaluer le nb d'élé^{ts} $\leq m$ et $\geq m$?

| | | | | | | |

Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?

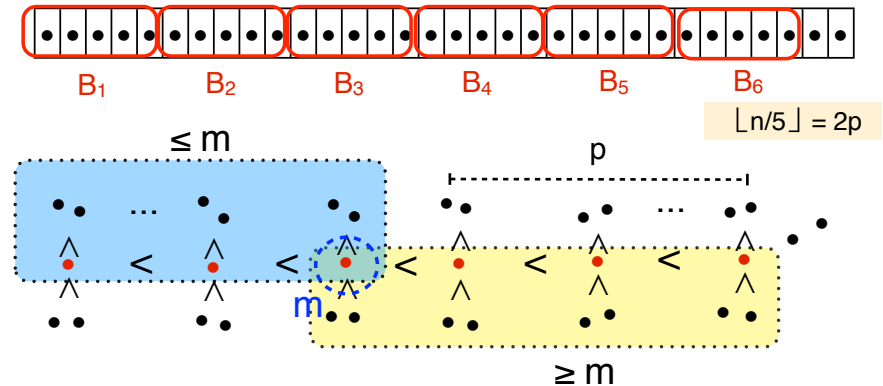


$$\lfloor n/5 \rfloor = 2p$$

Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?

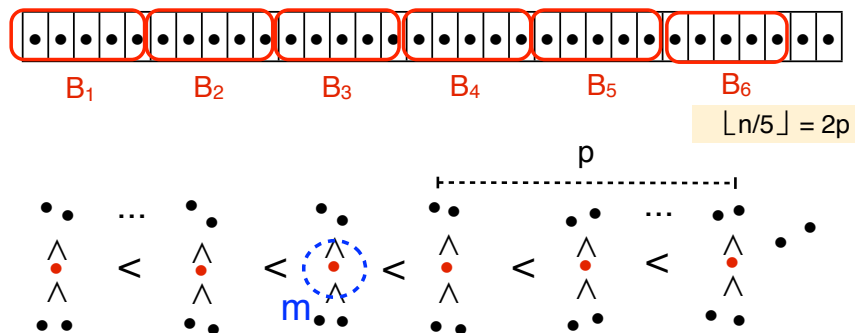


$$\lfloor n/5 \rfloor = 2p$$

Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?

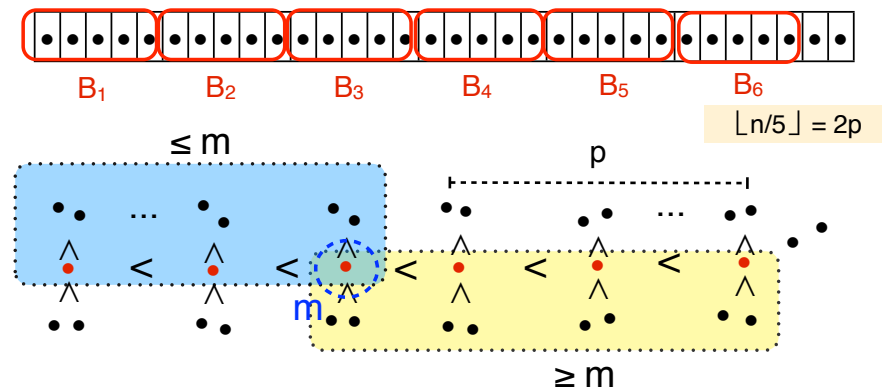


$$\lfloor n/5 \rfloor = 2p$$

Algo «select-opt» - complexité

La complexité de l'algorithme est bonne car le pivot m choisi n'est jamais «trop mauvais»...

Comment évaluer le nb d'éléts $\leq m$ et $\geq m$?



$$\lfloor n/5 \rfloor = 2p$$

$$nb_{\leq m} = 3p \quad nb_{\geq m} = 3(p+1)$$

Algo «select-opt» - complexité

$$1) \lfloor n/5 \rfloor = 2p, \quad nb_{\leq m} = 3p \quad nb_{\geq m} = 3(p+1)$$

$$2) \lfloor n/5 \rfloor = 2p+1, \quad nb_{\leq m} = nb_{\geq m} = 3(p+1)$$

Algo «select-opt» - complexité

$$1) \lfloor n/5 \rfloor = 2p, \quad nb_{\leq m} = 3p \quad nb_{\geq m} = 3(p+1)$$

$$p = \frac{1}{2} \lfloor n/5 \rfloor = \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

$$p+1 = \lceil \frac{1}{2}(2p+1) \rceil = \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil$$

$$2) \lfloor n/5 \rfloor = 2p+1, \quad nb_{\leq m} = nb_{\geq m} = 3(p+1)$$

$$p+1 = \lceil \frac{1}{2}(\lfloor n/5 \rfloor) \rceil$$

$$p+1 = \frac{1}{2}(\lfloor n/5 \rfloor + 1) = \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil$$

Algo «select-opt» - complexité

$$1) \lfloor n/5 \rfloor = 2p, \quad nb_{\leq m} = 3p \quad nb_{\geq m} = 3(p+1)$$

$$p = \frac{1}{2} \lfloor n/5 \rfloor = \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

$$p+1 = \lceil \frac{1}{2}(2p+1) \rceil = \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil$$

$$2) \lfloor n/5 \rfloor = 2p+1, \quad nb_{\leq m} = nb_{\geq m} = 3(p+1)$$

Algo «select-opt» - complexité

$$1) \lfloor n/5 \rfloor = 2p, \quad nb_{\leq m} = 3p \quad nb_{\geq m} = 3(p+1)$$

$$p = \frac{1}{2} \lfloor n/5 \rfloor = \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

$$p+1 = \lceil \frac{1}{2}(2p+1) \rceil = \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil$$

$$2) \lfloor n/5 \rfloor = 2p+1, \quad nb_{\leq m} = nb_{\geq m} = 3(p+1)$$

$$p+1 = \lceil \frac{1}{2}(\lfloor n/5 \rfloor) \rceil$$

$$p+1 = \frac{1}{2}(\lfloor n/5 \rfloor + 1) = \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil$$

Conclusion:

$$nb_{\leq m} = 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

$$nb_{\geq m} = 3 \lceil \frac{1}{2}(\lfloor n/5 \rfloor + 1) \rceil \geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

Algo «select-opt» - complexité

$$nb_{\leq m}, nb_{\geq m} \geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

comme $\lfloor n/5 \rfloor \geq (n-4)/5$, on:

$$\begin{aligned} nb_{\leq m}, nb_{\geq m} &\geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil \geq 3/2 \cdot \lfloor n/5 \rfloor \geq (3n-12)/10 \\ &\geq 3n/10 - 2 \end{aligned}$$

Algo «select-opt» - complexité

$$nb_{\leq m}, nb_{\geq m} \geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

comme $\lfloor n/5 \rfloor \geq (n-4)/5$, on:

$$\begin{aligned} nb_{\leq m}, nb_{\geq m} &\geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil \geq 3/2 \cdot \lfloor n/5 \rfloor \geq (3n-12)/10 \\ &\geq 3n/10 - 2 \end{aligned}$$

Les appels récursifs se font donc sur des sous-tableaux de taille $\leq n-3n/10+2$, donc $\leq 7n/10+2$.

$$C(n) = C(\lfloor n/5 \rfloor) + C(7n/10+2) + O(n)$$

Algo «select-opt» - complexité

$$nb_{\leq m}, nb_{\geq m} \geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil$$

comme $\lfloor n/5 \rfloor \geq (n-4)/5$, on:

$$\begin{aligned} nb_{\leq m}, nb_{\geq m} &\geq 3 \lceil \frac{1}{2} \lfloor n/5 \rfloor \rceil \geq 3/2 \cdot \lfloor n/5 \rfloor \geq (3n-12)/10 \\ &\geq 3n/10 - 2 \end{aligned}$$

Les appels récursifs se font donc sur des sous-tableaux de taille $\leq n-3n/10+2$, donc $\leq 7n/10+2$.

Algo «select-opt» - complexité

Proposition:

Si $t(n) = \sum_i a_i t(n/b_i) + c \cdot n$ avec $\sum_i a_i/b_i < 1$
alors $t(n) = \Theta(n)$

Corollaire: $C(n) = \Theta(n)$

L'algorithme select-opt est en temps linéaire !

Algo «select-opt» - complexité

$$C(n) = C(\lfloor n/5 \rfloor) + C(7n/10+2) + O(n)$$

Proposition:

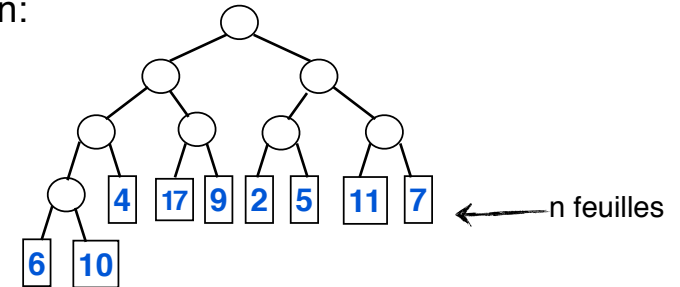
Si $t(n) = \sum_i a_i t(n/b_i) + c.n$ avec $\sum_i a_i/b_i < 1$
alors $t(n) = \Theta(n)$

Corollaire: $C(n) = \Theta(n)$

L'algorithme select-opt est en temps linéaire !

k^{eme} élément et arbres min

Arbre min:



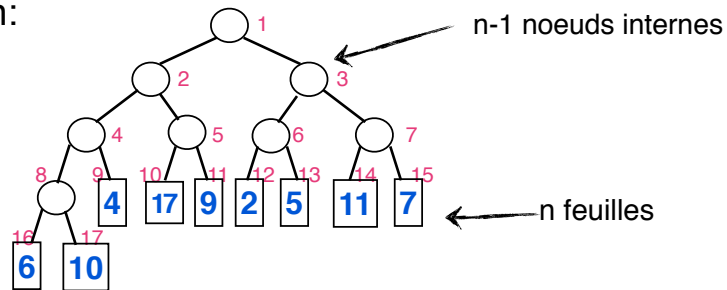
arbre binaire parfait...
9 feuilles + 8 noeuds internes

Algorithme avec des arbres Min

Voir «*The Art of Computer Programming*», volume 3,
D. Knuth.

k^{eme} élément et arbres min

Arbre min:

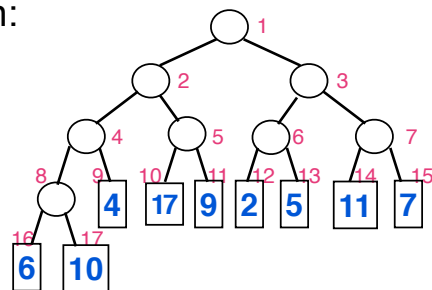


arbre binaire parfait...
9 feuilles + 8 noeuds internes

**calcul: indiquer le n° de la feuille
de valeur min dans le sous-arbre.**

k^{eme} élément et arbres min

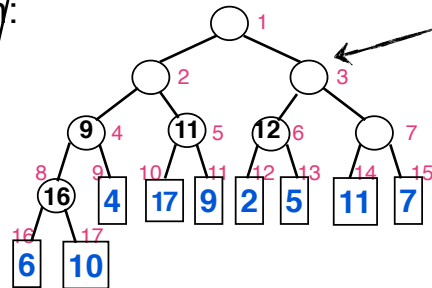
Arbre min:



arbre binaire parfait...
9 feuilles + 8 noeuds internes

k^{ème} élément et arbres min

Arbre min:



n-1 noeuds internes

n feuilles

arbre binaire parfait...
9 feuilles + 8 noeuds internes

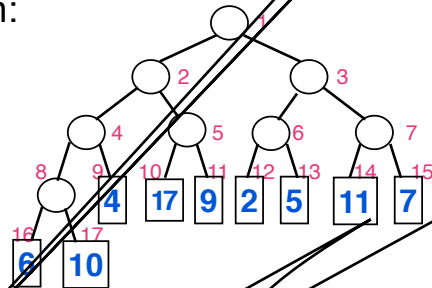
**calcul: indiquer le n° de la feuille
de valeur min dans le sous-arbre.**

jeudi 18 octobre 2012

21

k^{ème} élément et arbres min

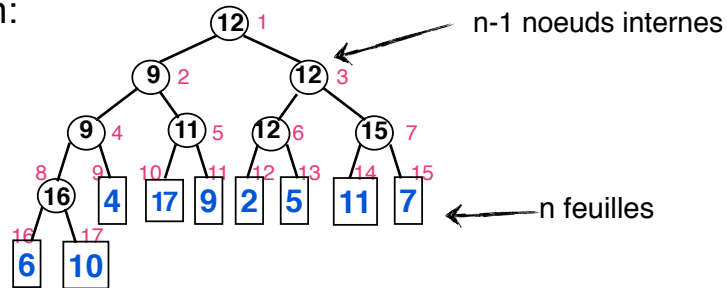
Arbre min:



arbre binaire parfait...
9 feuilles + 8 noeuds internes

k^{eme} élément et arbres min

Arbre min:

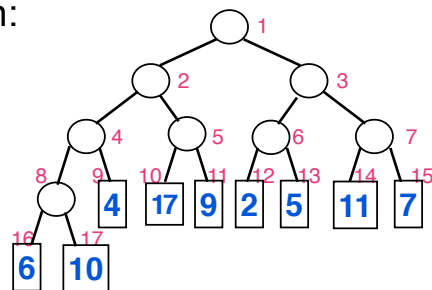


arbre binaire parfait...
9 feuilles + 8 noeuds internes

**calcul: indiquer le n° de la feuille
de valeur min dans le sous-arbre.**

k^{eme} élément et arbres min

Arbre min:



arbre binaire parfait...
9 feuilles + 8 noeuds internes

k^{eme} élément et arbres min

Arbre binaire parfait: hauteur $\leq \lceil \log(n) \rceil$

Calculer les n° de feuilles des noeuds internes : **CalculArbre**
n-1 noeuds internes... n-1 comparaisons : $O(n)$!

Changer une valeur d'une feuille et recalculer : **MaJ**

k^{eme} élément et arbres min

```
def CalculArbre(T,F,n) :  
  for i = n-1..1 :  
    if (F[T[2i]] <= F[T[2i+1]]) : T[i] = T[2i]  
    else : T[i] = T[2i+1]
```

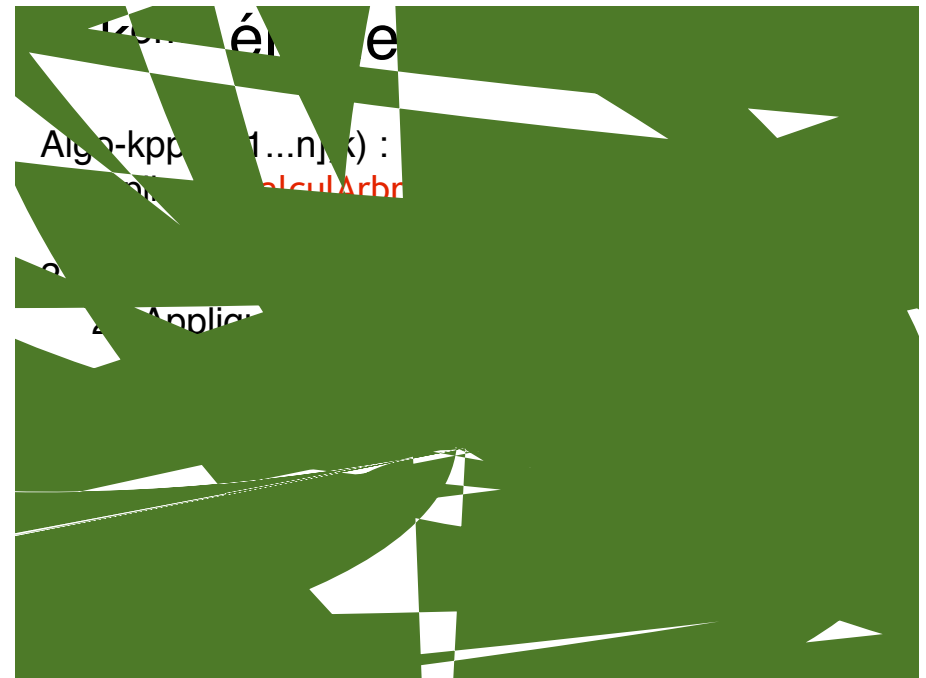
```
def MaJ(T,v,nf,F) :  
  F[nf] = v  
  i = nf  
  while (i/2 >= 1) :  
    i = i/2  
    if (F[T[2i]] <= F[T[2i+1]]) : T[i] = T[2i]  
    else : T[i] = T[2i+1]
```

k^{eme} élément et arbres min

Arbre binaire parfait: hauteur $\leq \lceil \log(n) \rceil$

Calculer les n° de feuilles des noeuds internes : **CalculArbre**
n-1 noeuds internes... n-1 comparaisons : $O(n)$!

Changer une valeur d'une feuille et recalculer : **MaJ**
 $O(\log n)$!



k^{eme} élément et arbres min

Complexité:

$$O(n-k + (k-1) \log(n-k+2))$$

CalculArbre

$$k-2 \text{ MaJ} + \text{MaJ}(, \infty,)$$



