

## TD2 : Algorithmique:

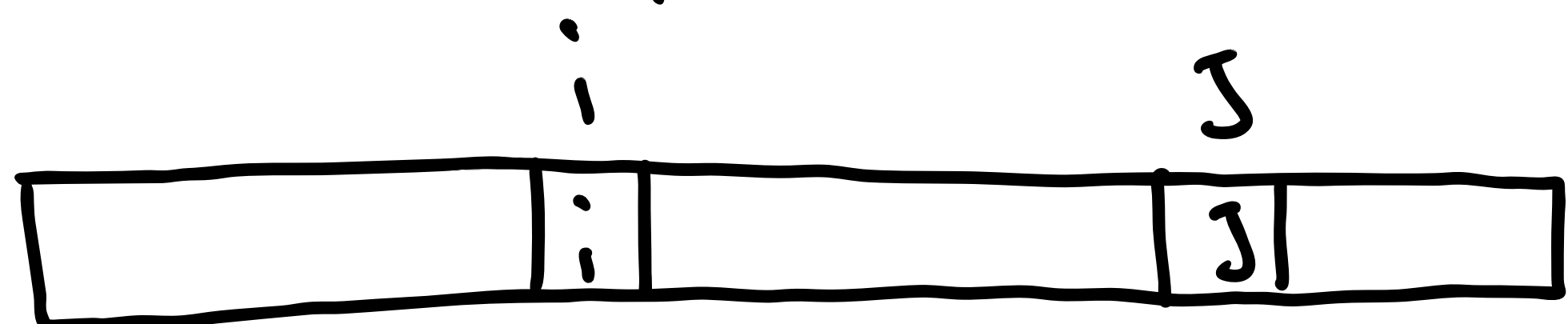
### Exercice 1,

algo naïf,

```
Pour i=0 à len(T)-1
  Si T[i] == i Alors retourner i.
Retourner  $\emptyset$ 
```

Complexité  $\mathcal{O}(n)$   
 $n = \text{len}(T)$

Propriété remarquable :



→ Si il y a des pts fixes, ils sont sur des "cases" adjacentes

exemple:   
A horizontal rectangle representing an array is divided into nine equal-width cells. Above the first eight cells are the numbers 1 through 8. Above the ninth cell is the number 9. The seventh cell from the left contains the number 7. Below the seventh cell is the number 7, and below the eighth cell is the number 8.

$v = 7 \Rightarrow \text{bingo}$

$v < 7 \rightarrow$  rechercher dans 8 --- 14

sinon ( $v > 7$ )  $\rightarrow$  rechercher dans 0 --- 6

def algorec( $T, g, d$ ),

Si  $g > d$  Alors return -1

$m = (g+d)/2$

Si  $T[m] = m$  Alors return  $m$

sinon si  $T[m] < m$  return algorec( $T, m+1, d$ )

sinon return algorec( $T, g, m-1$ )

Complexité:

$$T(n) = T\left(\frac{n}{2}\right) + 2 \quad n > 0$$

$$T(0) = O(1)$$

Master Theorem

$$\rightarrow \begin{matrix} a=1 \\ b=2 \end{matrix} \quad \log_b a = \log_2 1 = 0$$

$$\text{Cas 2} \rightarrow T(n) = O(\log n)$$

Algo  $k(T)$ :

$$g = 0$$

$$d = \text{len}(T) - 1$$

while ( $g \leq d$ )

$$m = (g + d) / 2$$

$$\text{if } T[m] = m, \text{ return } m$$

sinon

$$\text{if } T[m] < m :$$

$$g = m + 1$$

sinon

$$d = m - 1$$

return -1

Complexité  $O(\log n)$

Exercice 2:

$$x_1 \quad \widehat{\quad} \quad x_2 \quad x_3 \quad \widehat{\quad} \quad x_4 \quad x_5$$

$$x_1 < x_2$$

$$n \text{ él.}$$

$n-1$  comparaisons au minimum

3.

$$4. n = 2^k$$

$$C(n) = \begin{cases} 2 C(\frac{n}{2}) + 2 & n \geq 2 \\ 1 & n = 2 \\ 0 & n = 1 \end{cases}$$

coût pour  $n = 2^k$

$$d_k = \begin{cases} 2 d_{k-1} + 2 & \text{si } k \geq 2 \\ 1 & \text{si } k = 1 \\ 0 & \text{si } k = 0 \end{cases}$$

$$d_k = ?$$

$$d_k = 2 d_{k-1} + 2$$

$$= 2(2 d_{k-2} + 2) + 2$$

$$= 2(2(2 d_{k-3} + 2) + 2) + 2$$

$$= 2^{k-1} d_1 + \sum_{i=1}^{k-1} 2^i \leadsto \frac{1-2^{k-1}}{1-2} \times 2$$

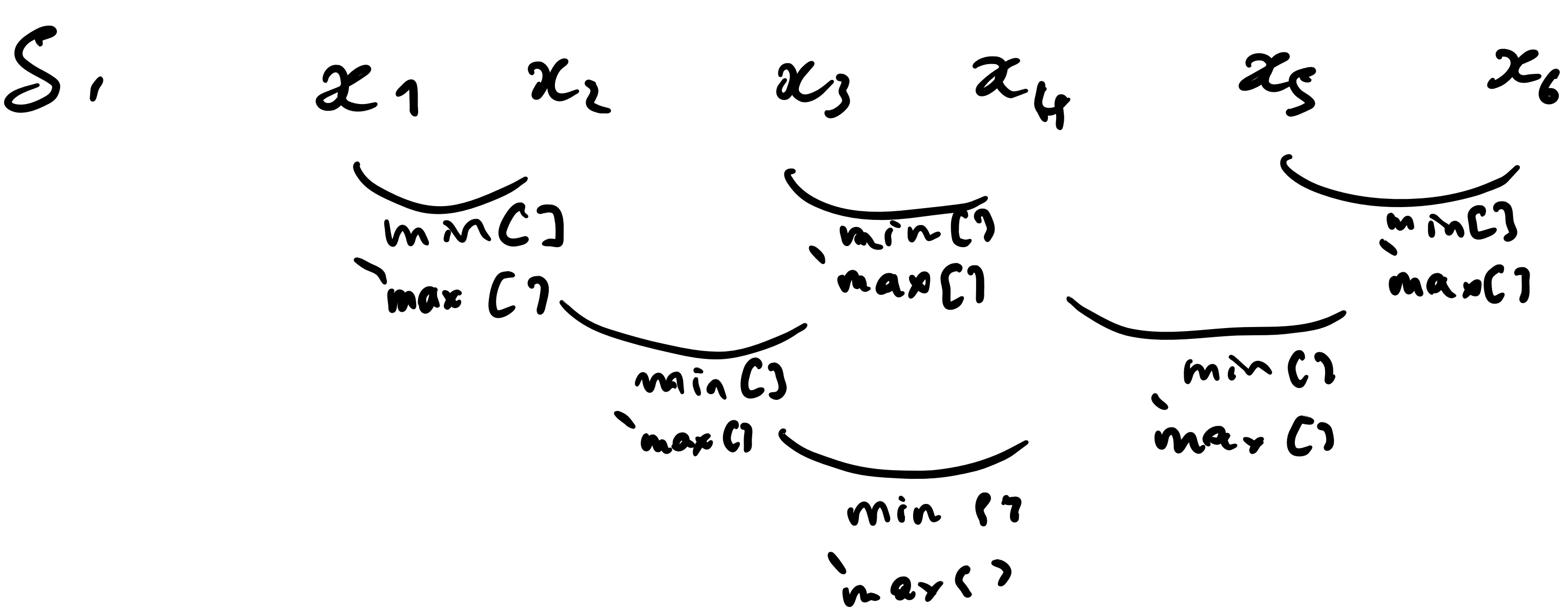
coût pour  $n = 2^k$

$$d_k = 2^{k-1} + 2^k - 2$$

$$= \frac{3}{2} 2^k - 2$$

$$\boxed{C(n) = \frac{3}{2} n - 2}$$

Bis: Montrer que c'est optimal



$$\frac{n}{2}$$

$$\frac{n}{2} - 1 \rightarrow \min$$

$$\frac{n}{2} - 1 \rightarrow \max$$

ajoute les elements  
dans une liste de  
min / max

- on debut =  $P_{\{1, \dots, n\}, \{1, \dots, n\}} = \text{tout le monde est candidat à tout}$

- l'idée : arriver à  $P_{\{i\}, \{j\}} = i \text{ est le min}$   
 $j \text{ est le max}$

optimal car au  
bout d'un moment  
obligé de séparer  
en deux parties

$\Theta(n)$  Master theorem

Exo 3:

$$1) A(n) = A\left(\frac{9n}{10}\right) + n$$

$$\log_5 9 = \frac{\ln 9}{\ln 5}$$

$$a = 1$$

$$b = \frac{10}{9}$$

$$\log_b a = \log_{\frac{10}{9}}(1) = 0$$

$$f(n) = n$$

$$\in \Omega(n^\epsilon) \text{ pour } 0 < \epsilon < 1$$

$$c < 1 \text{ tq. } a \cdot f\left(\frac{n}{b}\right) \leq c \cdot f(n) ?$$

$$\frac{9n}{10} \leq c \cdot n \text{ on a}$$

$$2) a = 2$$

$$b = 4$$

$$\log_4 2 = \frac{1}{2}$$

$$f(n) = \sqrt{n} = n^{\frac{1}{2}}$$

$$\text{CAS 2: } B(n) = \Theta(\sqrt{n} \cdot \log(n))$$

Finir C, D, E, F

↳ Exo 4