

9.3 - Complexité?

Supposons $b = 2^n \Leftrightarrow \log_2 b = n$

à chaque itération y est divisé par 2
 \Rightarrow on fait n itérations.

donc Multiplier2 est en $O(\log_2 b)$

$$\exists n \text{ tq } 2^n \leq b < 2^{n+1}$$

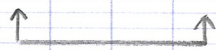
Exo 11 :

$$T[1..n]$$
$$1 \leq i < j \leq n$$

$$e(i, j) = j - i - 1 ; \text{emin}(T) ?$$

11.1 -

1	2	4	5	2	7	8	1	5	3
---	---	---	---	---	---	---	---	---	---



$$\text{emin}(T) = e(2, 5) = 2$$

11.2 -

* $I(k)$: écart min contient l'écart minimum entre les valeurs du tableau $T[1..k]$ et l'occurrence suivante de ces valeurs dans T .

* $I(k)$ est invariant. Validité?
à la fin de la dernière itération:
on a:

d'après $I(n)$: écart min contient l'écart min entre les valeurs du Tableau $T[1..n] = T$ et l'occurrence suivante dans T .

11.3 - la complexité pire-cas correspond au cas où toutes les valeurs n'apparaissent qu'une fois dans T

* on effectue dans le pire cas :

$$C(n) = \sum_{i=1}^{n-1} (n-i-1) \text{ iterations}$$

$$\begin{cases} k = n-i-1 \\ i=1 \Rightarrow k = n-2 \\ i=n-1 \Rightarrow k=0 \end{cases}$$

$$\Rightarrow C(n) = \sum_{k=0}^{n-2} k = \frac{(n-2)(n-1)}{2} \in \mathcal{O}(n^2)$$

$$\begin{array}{r} 0 + 1 + 2 + \dots + (n-2) \\ + \quad (n-2) + (n-3) + \dots + 0 \\ \hline (n-2) + (n-2) + \dots + (n-2) \\ \underbrace{\hspace{10em}}_{n-1 \text{ fois}} \end{array}$$

11.4 - elle est effectivement atteinte (preuve dans la question précédente)

11.5 : On suppose que T contient des valeurs entières entre 1 et k avec $k \leq n$
- stocker le dernier indice où on a rencontré la valeur dans un tableau D.

ex. prenons le tableau donné dans l'exo :

1	2	3	4	5	6	7	8
1	2	10	3	4	-1	6	7

dernière occurrence \rightarrow

8 5

9

D

$e_{\min} = 2$

D est initialisé à (-1)
 - on parcourt le tableau T avec i
 si $D[T[i]] = -1$ alors $D[T[i]] \leftarrow i$
 sinon $\left(\begin{array}{l} e_{\min} \leftarrow \min(e_{\min}, i - D(T[i]-1)) \\ D[T[i]] \leftarrow i \end{array} \right.$

TD 2:

* Rappels :

Algo(n):

Si $\langle \text{cas de base} \rangle$ alors
retourner $\langle \dots \rangle$

Sinon

Algo(n/b)

Algo(n/b)

...

a appels

on divise en sous-problèmes

recombinaison des sous-problèmes

$$T(n) = a \times T(n/b) + \underbrace{\Theta(n^d)}_{\text{coût recombinaison}}$$

