Calcul du max

vendredi 10 février 2017 14:16

1)
```
def maximum(T):
    if not T:
        return
    valeur = T[0]
    for element in T:
        if element > valeur:
            valeur = element
    return valeur
```

2) n comparaisons

3) Meilleur cas : 1 affectation, pire cas : n affectations

4) Exemple : `t = [1, 2, 3, 4]`
Calculons le coût pour chaque tour de boucle :

Première itération - Possibilités : 1, 2, 3, 4
On paye dans chacun des cas $\Rightarrow$ coût $= 1$

Deuxième itérations - Possibilités :
<u>1 2</u>, <u>1 3</u>, <u>1 4</u>, 2 1, <u>2 3</u>, <u>2 4</u>, 3 1, 3 2, <u>3 4</u>, 4 1, 4 2, 4 3
On paye dans 6 cas sur 12 (Cas soulignés) $\Rightarrow$ coût $= \frac{1}{2}$

Troisième itération - Possibilités :
<u>1 2 3</u>, <u>1 2 4</u>, 1 3 2, <u>1 3 4</u>, 1 4 2, 1 4 3,
<u>2 1 3</u>, <u>2 1 4</u>, 2 3 1, <u>2 3 4</u>, 2 4 1, 2 4 3,
3 1 2, <u>3 1 4</u>, 3 2 1, <u>3 2 4</u>, 3 4 1, 3 4 2,
4 1 2, 4 1 3, 4 2 1, 4 2 3, 4 3 1, 4 3 2
On paye dans 8 cas sur 24 $\Rightarrow$ coût $= \frac{1}{3}$

Probabilité qu'il y aie affectation à la i-ème itération = probabilité que le i-ème élément soit supérieur à ceux qui le précède $= \frac{1}{i}$ (Car le maximum étant positionné au hasard parmi les i éléments, il a 1 chance sur i de se retrouver à la fin)

Finalement :

$$\text{nombre total d'affectations } = \sum_{i=1}^{n} \frac{1}{i}$$

5)
```
def max_et_min(T):
    if not T:
        return
    maximum = minimum = T[0]
    for element in T:
        if element < minimum:
            minimum = element
        elif element > maximum:
            maximum = element
    return maximum, minimum
```

Cet algorithme est peu efficace en nombre de conditionelles !
En effet, on passe toujours dans la première, or elle est peu probable, ce qui nous fait souvent passer dans la deuxième. On a donc intérêt à choisir une première conditionelle plus probable :

```
def max_et_min(T):
    if not T:
        return
    maximum = minimum = T[0]
    for element in T:
        if element < minimum or element > maximum:
            if element > maximum:
                maximum = element
            else:
                minimum = element
    return maximum, minimum
```

Dichotomie

vendredi 10 février 2017 15:17

10)
```
def dichotomie(T, e, a, b):
    c = ((a + b)//2
    if e > T[b] or e < T[a]:
        return False
    if e == T[c]:
        return True
    if e > T[c]:
        return dichotomie(T, e, c, b)
    else:
        return dichotomie(T, e, a, c)
```

11) On fait $O(1)$ comparaisons à chaque itération.
On doit donc compter le nombre d'itérations.
Meilleur cas : $O(1)$
Pire cas : $\log_2 n$