EA CNTD 1 — DUCHI ENRICA duchi@irif.fr — 17.09.20

NCC : en TD, pas d'examen (pour la 1ère session)

- CC 1 : semaine du 28 sept
- CC2 : ——— 12 oct
- CC3 : ——— 09 nov
- CC 4 : ——— 30 nov → sur tout le pgrm

Note 1ère session = moyenne des contrôles TD

SESSION 2 : examen terminal

note : max (ET, moy(CC1, CC2, CC3, ET))

---

BONUS : chaque semaine, coder un exo du TD en Java et l'envoyer à la prof.
↳ dépôt Moodle

# EA CM+TD 1

17.09.20

## I/ Définitions

- **algorithme** : séquence d'instructions pour résoudre un problème.
  - ↳ données en entrée et en sortie

  ex : minimum entre deux entiers
  - entrée : $i, j \in \mathbb{Z}$ ; sortie : le ⊜ petit des deux
  - algo : if $i > j$ return $j$ else return $i$

- **instance** d'un problème : un ensemble de données sur lequel on va exécuter l'algo.

  ex : $i = 3$ ; $j = 5$

- algo **correct** : renvoie tjrs la bonne sortie, pour n'importe quelle instance.

- **efficacité** : 2 types de complexité
  - ESPACE
    - ↳ mesure la place mémoire occupée pdt l'exécution de l'algo
    - ↳ lié à la structure de données utilisée
  - TEMPS : lié à l'écriture de l'algo
    - ↳ compter le nb d'opérations élémentaires effectuées
    - ↳ lié à la structure de données choisie.

Pb : Recherche d'un min ds un tableau

algo : RechercheMin (tab : tableau d'entiers de taille n)

```
min ← tab[0]                          1
for (i from 1 to n-1)
    if (tab[i] < min)                 2 · n-1
        min ← tab[i]
return min
```

dans le pire des cas $1 + 2(n-1)$ opérations

on cherche à voir la dépendance à la taille du pb

COMPLEXITÉ LINÉAIRE

- **opérations élémentaires** :
  - * assignations
  - * comparaisons

Pb : Recherche d'un élément dans un tableau

algo : RechercheElTableau (tab : tableau d'entiers de taille n, el : entier)

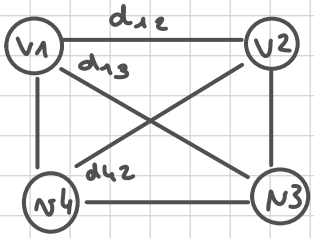
```
for (i from 0 to n-1)
    if (tab[i] == el)
        return i;
```

$\approx n$ opérations élémentaires

| complexité \ $n$ | 20 |
|---|---|
| $n^2$ | 1/2500 ms |
| $n^5$ | 3 ms |
| $e^n$ | |
| $n^n$ | $(3,3\,10)^7$ siècles |

## Problème du voyageur de commerce

PAS DE SOLUTION EFFICACE CONNUE




$v_i$ : villes
$d_{i,j}$ : distance entre $v_i$ et $v_j$

Ne passer qu'une seule fois par chaque ville, distance min

**Problème de l'arrêt** : pour n'importe quel algo, est ce qu'il termine ?

ON NE PEUT PAS RÉSOUDRE

# EXERCICES

## Exercice 1

1) $\{1,1,1\} \rightarrow \{1,2,3\}$ $\{0,1,2\} \rightarrow \{0,1,3\}$

2) $T = \{0, 0, \ldots, i\}$ , $i \in \mathbb{N}$

3)
```
Fonct2(int[] t){
    for(int i = n-1; i = 0; i --){
        tab[i] = tab[i] - tab[i-1];
    }
}
```

## Exercice 2

1) incorrect : trouver un contre exemple

correct : preuve mathématique (+/- une récurrence)

2) $\{1, 2, 3\}$

→ max = 2
max2 = 1

→ i = 2 ; t[i] = 3 ⇒ 3 > max donc max = 3

→ renvoie 1 au lieu de 2.

ERREUR : l13 ↪ { max2 = max
max = t[i]

## Exercice 3

1)
```
static int tabMax (int [] t){
    int max = t[0];
    for(int i = 1; i < t.length ; i++){
        if (t[i] > max) { max = t[i]; }
    }
    return max;
}
```

$n-1$ comparaisons pour un tableau de taille $n$.

2)
```
static int[] minMax (int [] t){
    int max = t[0];
    int min = t[0];
    for (int i = 1; i < t.length ; i++){
        if (t[i] > max)  max = t[i];
        if (t[i] < min)  min = t[i]
    }
    return {min, max};
```

$2(n-1)$ comparaisons.

$T = \{2, 1, -3, 5, 4, 8\}$

| $i=1$ | $i=2$ | $i=3$ | $i=4$ | $i=5$ |
|---|---|---|---|---|
| min = 2 | min = 1 | " | min = -3 | " | min = -3 |
| max = 2 | " | " | max = 5 | " | max = 8 |

3)
```
static int [] minMaxEfficace (int [] t){   // On suppose t de longueur paire
    int min = t[0];
    int max = t[0];
    for (int i = 0; i < t.length/2; i++){
        int petit = t[i];
        int grand = t[i+1];
        if (t[i] > t[i+1]) {
            petit = t[i+1];
            grand = t[i];
        }
        if (petit < min) min = petit;
        if (grand > max) max = grand;
    }
    return {min, max};
}
```

$n/2$ $\rightarrow$ $2(n/2)$

$= \frac{3}{2} n$ comparaisons

$T = \{2, 1, -3, 5, 4, 8\}$

| | | | | |
|---|---|---|---|---|
| min | 2 | 1 | -3 | -3 |
| max | 2 | 2 | 5 | 8 |
| i | 0 | 0 | 1 | 2 |
| petit | 0 | 1 | -3 | 4 |
| grand | 0 | 2 | 5 | 8 |

4) Il faut trouver l'ordre entre les 3 valeurs : a, b et c.

```
if (a > b){                                   ← n/3
    grand = a;
    petit = b;
    if (c > a) grand = c;
    if (c < b) petit = c;
}
```

$2\left(\frac{n}{3}\right)$

puis comparaisons entre | gd / max | petit / min | $2\left(\frac{n}{3}\right)$

$\frac{n}{3} + 2\left(\frac{n}{3}\right) + 2\left(\frac{n}{3}\right)$

$= \frac{5}{3}n$ comparaisons

↳ Donc NON, ça n'est pas plus efficace.

## Exercice 4

1)
```
static int inf(char[] a, char[] b){
    for (int i = 0; i < min(a.length, b.length); i++){
        if (a[i] < b[i]) return -1;
        if (a[i] > b[i]) return 1;
    }
    if (a.length < b.length) return -1;
    if (a.length > b.length) return 1;
    return 0;
```

2)
```
static char[] min(char[][] t){
    char[] min = tab[0];
    for (int i = 1; i < t.length; i++){
        if (inf(min, t[i]) == 1){   // t[i] > min
            min = t[i];
        }
    }
    return min;
}
```

## Exercice 5

1) 
```
static int compteInf (int[] t, int x){
    int inf = 0;
    for (int i=0; i < t.length ; i++){
      if(t[i]<x) inf += 1;   // n comparaisons
    }
    return inf;
}
```

2) 
```
static int mediane (int[] t){          // on fait l'hypothèse que toutes les valeurs sont différentes.
    int plancher = (t.length -1) /2;
    for (int i=0; i<t.length ; i++){
      if (compteInf(t, t[i]) == plancher) return t[i];
    }                                  // ↑ n.n = n² comparaisons
}
```

3) C'est $n^2$ comparaisons dans le pire des cas.

$T = \{1, 2, 5, 6, 4\}$ atteint la borne (il faut tester chaque valeur pour savoir si c'est la médiane).