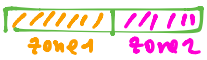
Les algorithmes de tri-

- tri par sélection.
- tri par insertion.
- TRI FUSION:

idée: - Pour trier un tableau de taille n, on :




  - découpe le tabl. en 2 ss-tab de taille $\frac{n}{2}$
  - on appelle récursivt l'algo. sur les 2 ss.tableau
    Lo on obtient alors 2 ss-tab TRIÉS.
  - on fusionne les 2 ss-tab pour obtenir un tabl. complet trié.
    à voir...

```
def TriFusion(T):
    Si |T| > 1 Alors:
        Si |T| = 2 Alors:
            Si T[0] > T[1] Alors T[0] ↔ T[1]
        Sinon:
            m = |T| / 2
            T1 = TriFusion(T[0 .... m-1])
            T2 = TriFusion(T[m ... |T|-1])
            T = Fusion(T1, T2)
    Retourner T.
```

```
def Fusion(T1, T2):
    res = [ ]      : tableau vide.
    i1 = 0 , i2 = 0
    Tant que i1 < |T1| ∧ i2 < |T2| Faire:        (∧ : ET)
        Si T1[i1] < T2[i2] Alors:
            res.append(T1[i1])
            i1 = i1 + 1
        Sinon:
            res.append(T2[i2])
            i2 = i2 + 1
    Si i1 == |T1| Alors:
```

Append : ajoute un élt à la fin de res (T1[i1])

$\lfloor$ res = res + $T_2[i_2 : \dots |T_2|-1]$ : concaténation avec copie.

Sinon :

$\lfloor$ res = res + $T_1[i_1 \dots |T_1|-1]$

Retourner res

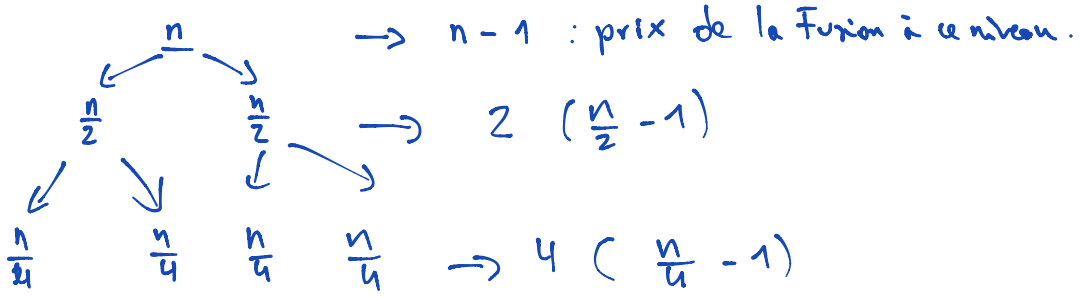
Sur un ex de taille 10 → 21 comparaisons.

Soit $C(n)$ le coût (en nb de comparaisons) de l'algo. dans le pire cas pour trier un tableau de taille $n$.

$C(0) = C(1) = 0$

$C(2) = 1$

$C(n) = \underbrace{2 \cdot C(\frac{n}{2})}_{\text{coût des appels récursifs}} + \underbrace{n-1}_{\text{coût fusion}}$




→ $n-1$ : prix de la fusion à ce niveau.

→ $2\left(\frac{n}{2}-1\right)$

→ $4\left(\frac{n}{4}-1\right)$

Aux feuilles → 1/0

A chaque niveau, on peut majorer le coût des fusions par $n$

il y a $\log_2 n$ niveaux → $\sim n \cdot \log n$

⚠ passer de $n^2$ à $n \cdot \log n$ est très important !