F(n) ⇝ espace que le tableau va prendre si en place alors n ⇝ $\log n$

```
def fusion_dpr ( t, bg, bd )
    if bg == bd-1
        return
    else
        t1=fusion_dpr (        )
        t2 fusion_dpr (        )
          return fusion(t1,t2)
```

possible de le faire non rcc avec implémentation des tas $C(tas)\ \log kn$.

TD 4 :

Programmation dynamique

[3 10 4 5 6 9 7 8]

xi la longueur max d'une suite croissant en les i premiers cases

y la longueur max d'une sous sequence croissante qui se termine par ti

$y_0 = 1$

$y_i = \max\{ y_j \mid 0 \leq j \leq i, T[i] \geq T[j] \} + 1$
(si il y a 2)
= 2 (si cet ensemble est vide)

Reprenons l'exemple : [3 10 4 5 6 9 7 8] -- 2]

y : 1 2 2 3 4 5 5 6 -- 1

On recommence.

```
def pgssc ( t ):
    y = [None] x len(t)        #n'est pas vide
    y[0] = 1
    for i in range( 1, len(t)):
        max = 0
        for j in range ( 0, i ):
            if t[i] >= T[j]:
                if y[i] > max:
                    max = y[j]
        y[i] = max + 1
    return max(y)
```

⇝ $\Theta(n^2)$

// n'est pas la façon la plus optimal. (version nlogn)

nouveau exemple: [3, 10, 4, 5, 6, 9, 7, 8, 2, 4, 6, 8, 10, 4]

y: 1 2 2 3 4 5 5 6 1 3 5 7 8 4

prec: None 0 0 2 3 4 4 6 None 2 4 7 max 9

```
def pgsse (t)
    y = [None] * len(t)      prec = [None] * len(t)
    y[0] = 1
    for i ____
        max = 0
        for ____
            if ____
                if ____
                    max = y[j]      prec[i] = j
        y[i] = max + 1
    return max(y)
```

⇝ [3 4 5 6 7 8 8 10]