Problemas con el Profe

$$f_n = \begin{cases} 0 & n=0 \\ 1 & n=1 \\ f_{n-1} + f_{n-2} & n>1 \end{cases}$$

fibonacci

```
def fib(n)
    if == 0
        return 0
    else if n == 1
        return 1
        else
        return fib(n-1) + fib(n-2)
```

$t(n) = t(n-1) + t(n-2) + c$

$t(n) \in \Theta(\phi^n)$

```
def fib_men(int m)
        ⋮
```

$A \in M_{2\times 2}$ $\quad A^n = \underbrace{A \cdot \ldots \cdot A}_{} = A^{n/2}\, A^{n/2}$ par

$A\, A^{\frac{n-1}{2}}\, A^{\frac{n-1}{2}}$ impar

$A^0 = I$

```
def dyV(n):
    if n == 0
        return I
    else if n%2 == 0
        r = dyV(n/2)
        return r x r
    else if n%2 == 1
        r = dyV((n-1)/2)
        return A·r·r;
```

$n = 2^k$

$t(n) = t(n/2) + \Theta(1) \rightarrow t(n) \in \Theta(\log n \mid n=2^k)$

$f_n = f_{n-2} + f_{n-1}$

$$\underbrace{\begin{pmatrix} f_n \\ f_{n-1} \end{pmatrix}}_{F_n} = \underbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} f_{n-1} \\ f_{n-2} \end{pmatrix}}_{F_{n-1}} = \begin{pmatrix} f_{n-1} + f_{n-2} \\ f_{n-1} \end{pmatrix}$$

casos base

$$F_1 = \begin{pmatrix} f_1 \\ f_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$F_n = A \cdot F_{n-1} = A^{n-1} \cdot F_1$

$F_n = \underbrace{A^{n-1} \cdot F_1}_{} = A^{n-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

$A^2 \cdot F_{n-2} \;\; * \;\; A^3 F_{n-3} \;\; * \;\; A^4 F_{n-4} \ldots\ldots$

Algoritmo →

```
def fibodyV(n)
    if n == 0
        return 0
    else
        A ± (1 1 / 1 0)   ← Fn-1
        return dyV(A, n-1)[0][0]
```

$\Theta(\log n)$

$[p_0 \rightarrow p_1 \; p_{n-1} \; p_n]$

$A_0, A_1, \dots A_{n-1}$ $\quad A_0 \dots A_{n-1}$ $\quad A\,B \rightarrow i, k$ $\quad (i,j)(j,k)$

$A_0 \rightarrow p_0 \times p_1$ $\quad (A_0 (A_1) A_2)$

$A_1 \rightarrow p_1 \times p_2$ $\quad A_0 \; 10 \times 30$ $\quad (A_0 A_1) A_3 \rightarrow 4500$

A_2 $\quad A_1 \; 30 \times 5$ $\quad A_0 (A_1 A_2) \rightarrow 27000$

$\vdots$ $\quad A_2 \; 5 \times 60$

$m(i,k)$ $\quad m(k+1, j)$

$A_i \dots A_k \; A_{k+1} \dots A_j$

$m(i,j)$ = nº mínimo de multipl desde $A_i - A_j$

$p[i] \times p[k+1]$ $\quad p[k-1] \cdot p[j+1]$

$$m(i,j) = m(i,k) + m(k+i,j) + p[i]\,p[k+1]\,p[j+1]$$

Divide y venceras

Subarray sum

nums = $[-2, 1, -3, \underbrace{4, -1, 2, 1}_{sum = 6}, -5, 4]$

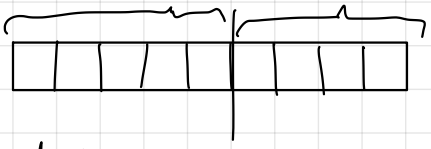
a) fuerza bruta → $O(n^2)$

b) dyV → $O(algo)$

c) PD → $O(n)$

Fuerza bruta

```
def max_sum_n(array, n)
    ma_sumA = -∞
    for i in 0 ... n          O(n^2)
        Csum = 0
        for j in i, ... n-1
            csm += 0
        max_sum = max(csum, max_sum)
    retur max_sum
```



dyv

si pequeño → array

resolver → an[i]

{ dividir y resolver }

{ convinar }

```
def sub_array (arr, left, right)
    if (left == right)
        return arr[left]
    mid = (left + right) / 2
    lm = sub_dyv (arr, left, mid)
    rm = sub_dyv (arr, mid+1, left)
    return max (_ , _ , _ )
```

frontera

izq {
- left_sum = -∞
- Csum = 0
- for i in mid, mid-1, ... left
  - csum += arr[i]
  - left_sum = max (Csum, left_sum)

righ {
- righ_sum = -∞

return left_sum + right_sum

$$t(n) = 2t(n/2) + \Theta(n)$$

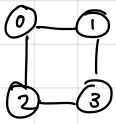
① $\log_2 2 = 1$

$\Theta(n \log n)$

Backtracking

Grafos aristas

$|V| = n \quad G = (V, E)$



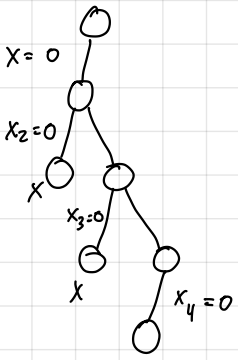
$\chi(G) = 2$

$S = (x_1, x_2, \dots x_n)$

$x_i \to 0, 1, \dots, max\_1 \sim O(A(G)+1)^n$

Tipo de arbol → K - ario

construimos el arbol



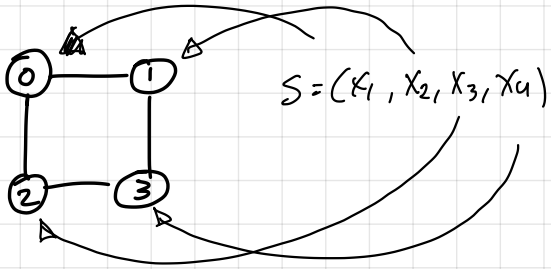
def backtracking (G) { // implementado Repositorio.

$\chi(G) \leq A(G) + 1$

$\chi(G) \leq \text{voraz}(G) \leq \Delta(G) + 1$

$2^n \quad 5^n$

7 mayo 2025



$\chi(G) \leq \text{voraz}(G) \leq BT(G)$

```
def backtracking(G):
    nivel = 1
    max_color = voraz(G)
    S = [-1, ..., -1]  # nº nodos
    Sol = None
    voa = ∞
    Usados = [0, ..., 0]  # max_color
    while nivel ≠ 0
        generar
```

```
generar 1
S[nivel-1] += 1
if S[nivel-1] == 0:
    usados[S[nivel-1]] += 1
else:
    usados[S[nivel-1]-1] -= 1
    usados[S[nivel-1]] += 1
    usados[0] -= 1
    usados[1] += 1
```

```
sol 2
nodo = nivel-1
color = S[nivel-1]
for vecino in vecinos(nodo):
    if S[vecino] == color:
        return False
return nivel == n
```

```
valor(S) 3
cont = 0
for c in usados:
    if c > 0:
        cont += 1
return cont
```

if sol valor(S) < voa
    sol, voa
if criterio y valor(S) < voa
else
    while nivel > 0 y ¬ + H
        retroceder

```
criterio 4
if nivel == n:
    return False
else:
    nodo = nivel-1
    color = S[nivel-1]
    for vecino in vecinos(nodo):
        if S[vecino] == color:
            return False
    return True
```

```
+ Her. 5
S[nivel-1] < max_color - 1

retroceder
usados[S[nivel-1]] -= 1
S[nivel-1] = -1
nivel -= 1
```

nº nodos x tiempo por nodo

$O((\Delta(G)+1)^n)$ $O(\Delta(G))$

$\chi(G) \leq voraz(G) \leq \delta(G)+T$

Final $O(\Delta(G)(1+\Delta(G))^n)$

Borazes

$\{2, 3, 4\}$

$14 = 5+9$	$7+9 = 16$	$6+9 = 15$
2 3 4 → 5 → 9	2 3 4 → 7 → 9	2 4 3 → 6 → 9

cogemos el más pequeño

$P(F)$

1. x_i, x_j de F
2. $f = merge(x_i, x_j)$
3. $P(F - \{x_i, x_j\} \cup \{f\})$

sol:

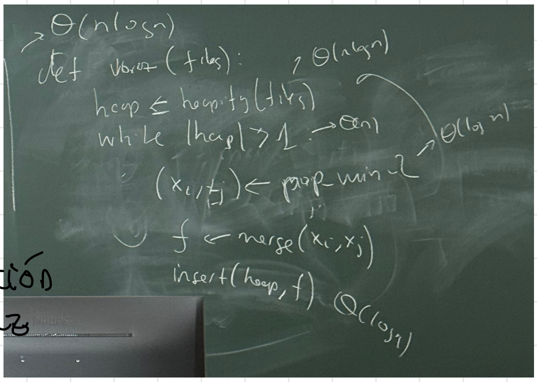
```
def voraz(files):                          → O(n)
    while |files| > 1 :
        O(n) (xi, xj) ← cogemos los 2 más peq.
        f ← merge(xi, xj)
        files ← files - {xi, xj} ∪ {f}
```

$O(n^2)$

¿Cual es el orden? $O(n^2)$

insertar $O(\log n)$
find_min $O(1)$
eliminar $O(\log(n))$

no entra optimización del voraz



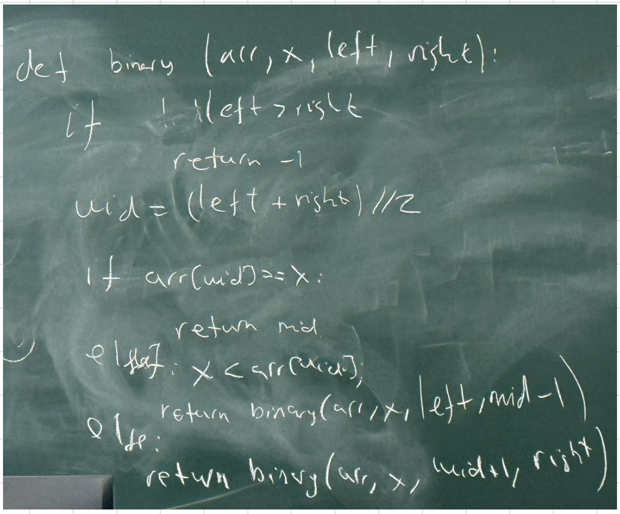
fuerza bruta

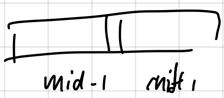
$S = [0, 2, 27, 34]$

$X = 27 \Rightarrow i = 2$

fuerza bruta < $O(1)$ mejor caso. / $O(n)$ peor caso.

Divide y vencerás (búsqueda binaria)





$t(n) = t(n/2) + C_1$

teorema maestro { $\log_2 1 = 0$; no ; $O(\log n)$ }