

ER2023

① $a = 1$ (1 rec. call)

$b=2$ (tamanho input $\frac{1}{2}$)

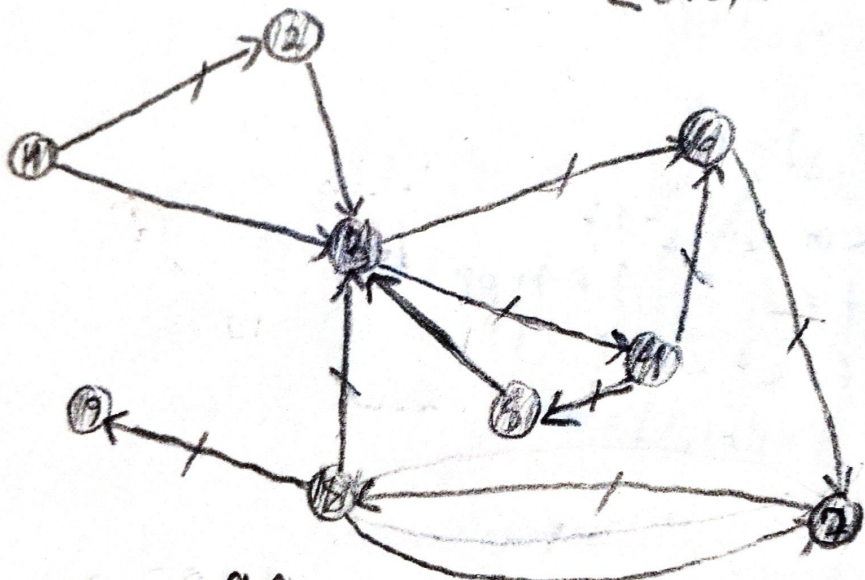
$b=2$ (fatoração implícita 2)
 $c=1$ (trabalho n/ recursivo e loop de $1 \dots \frac{n}{2}$)

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

$$\log_b a = \log_2 1 = 0 < 1 = c$$

$$G(n)$$

$\langle \text{dfs}, \text{low} \rangle$



O, C

SCCs

195

8, 7, 6, 4, 5, 3

425

495

8



6

五

4

2

2



```

graph TD
    3((3)) --> 4((4))
    4 --> 5((5))
    4 --> 6((6))
    5 --> 2((2))
    6 --> 1((1))
    style 3 fill:none,stroke:#000
    style 4 fill:none,stroke:#000
    style 5 fill:none,stroke:#000
    style 6 fill:none,stroke:#000
    style 2 fill:none,stroke:#000
    style 1 fill:none,stroke:#000
  
```

Handwritten decision tree diagram showing a game structure. The root node is 3, which branches to node 4. Node 4 branches to node 5 and node 6. Node 5 branches to node 2, and node 6 branches to node 1. The terminal nodes are 1, 2, 3, 4, 5, and 6. The values at the terminal nodes are: 1: (3,0), 2: (2,0), 3: (1,1), 4: (1,0), 5: (2,2), 6: (3,3). The values at the internal nodes are: 3: (1,1), 4: (1,0), 5: (2,2), 6: (3,3).

⑥ 3.0

⑦ 4,4
9,0

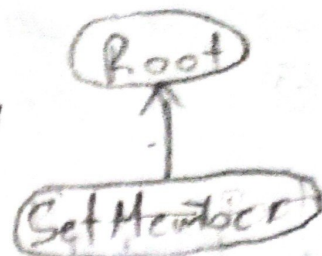
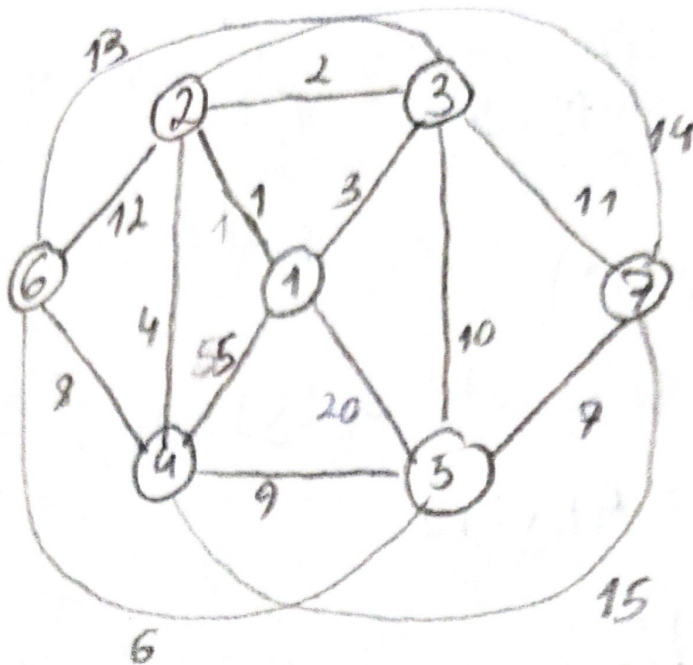
↓ 5,5
⑧ 5,0

c.6

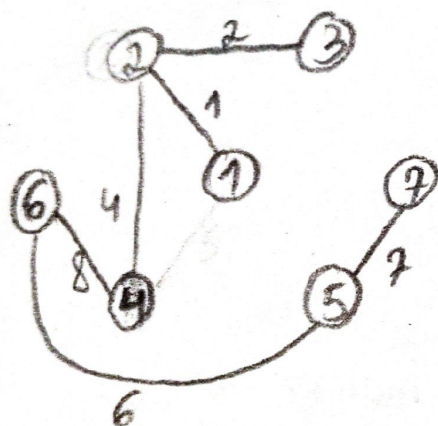
7.7

② 8,8

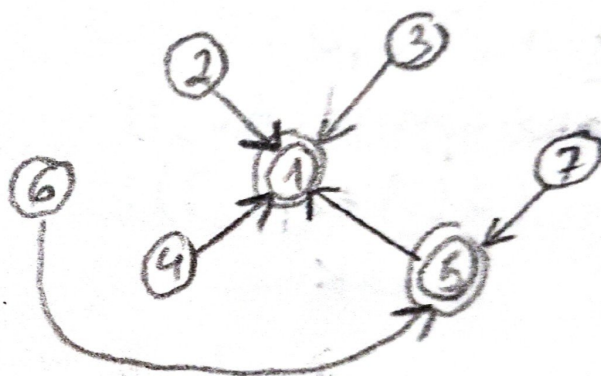
③



MST



DSU



④
$$\bar{L} = \frac{1}{n} \sum_{i=1}^n L_i$$

Base Case: $n=1$

$$\text{mean}(L, 1) = L_1$$

$$\bar{L} = \frac{1}{1} L_1 = L_1$$

Inductive Hypothesis:

$$\frac{L[K]}{K} + \left(\text{mean}(L, K-1) \times \frac{K-1}{K} \right) =$$

$$\text{mean}(L, K) = \frac{1}{K} \sum_{i=1}^K L_i$$

Inductive Step:

Para $K+1$ elementos:

$$\frac{L[K+1]}{K+1} + \left(\text{mean}(L, K) \times \frac{K}{K+1} \right) = \frac{L[K+1]}{K+1} + \left(\frac{K}{K(K+1)} \sum_{i=1}^K L_i \right) =$$

$$= \frac{L[K+1]}{K+1} + \frac{1}{K+1} \sum_{i=1}^K L_i = \frac{1}{K+1} \sum_{i=1}^{K+1} L_i //$$

⑤

-2	1	-3	4	-1	2	1	-5	4
dp = ②	①	-2	④	3	5	⑥	1	5

↑
Largest

→ Find $\max_{0 \leq i < n} (dp(i))$

→ Go through i to 0 until $dp[j] = A[j]$

→ Max subarray = $A[j:i]$

$i = \max_{0 \leq i < n} (dp(i))$

ans = []

for j in $i, \dots, 0$:

ans.append($A[j]$)

if ($A[j] = dp[j]$):

break

return ans.reversed

⑥ $dp[i][sum1][sum2] = \text{False}$ para todos os índices

↳ "Os primeiros i elementos podem formar 2 subconjuntos cujas somas são $sum1$ e $sum2$?"

function check(N, n, S):

$dp[0][0] = \text{True}$

for i in $1, \dots, n$:

cur = $N[i]$

for $s1$ in $0, \dots, \frac{S}{2}$:

for $s2$ in $0, \dots, \frac{S}{2}$:

$dp[i][s1][s2] = dp[i-1][s1][s2]$

if ($s1 \geq \text{cur}$ and $dp[i-1][s1-\text{cur}][s2] = \text{True}$):

$dp[i][s1][s2] = \text{True}$

if ($s2 \geq \text{cur}$ and $dp[i-1][s1][s2-\text{cur}] = \text{True}$):

return $dp[n, \frac{S}{2}, \frac{S}{2}]$ $dp(i, s1, s2) = \text{True}$

Time Complexity: $O(n S^2)$

$n \times \frac{S}{3} \times \frac{S}{3}$ estados preenchidos na tabela dp.

Space Complexity: $O(n S^2)$

tabela dp de tamanho

$$n \times \frac{S}{3} \times \frac{S}{3}$$

A solução está correta
abordagem correta pois:

Caso base: $dp(0,0,0) = \text{True}$ está correto pois num set de 0 elementos, podemos formar 2 subsets cuja soma dos elementos é 0

Relações dentro da tabela: Em cada passo, são consideradas 3 opções para o elemento $N[i]$:

→ Colocar no terceiro subset:

$dp[i, s1, s2]$ permanece igual pois os estados da tabela dependem dos 2 primeiros conjuntos

→ Colocar no primeiro subset:

se $dp[i-1, s1 - N[i], s2]$ é verdade, então colocando $N[i]$ no 1º conjunto faz com que $dp[i, s1, s2]$ seja verdade

→ Colocar no segundo subset (mesma explicação)

⇓
Então, o elemento $dp[n, \frac{S}{3}, \frac{S}{3}]$ guarda a resposta para se os primeiros n elementos conseguem criar 2 subsets cujas somas dos elementos são $\frac{S}{3}$ (o terceiro terá, automaticamente, $\text{sum} = \frac{S}{3}$)