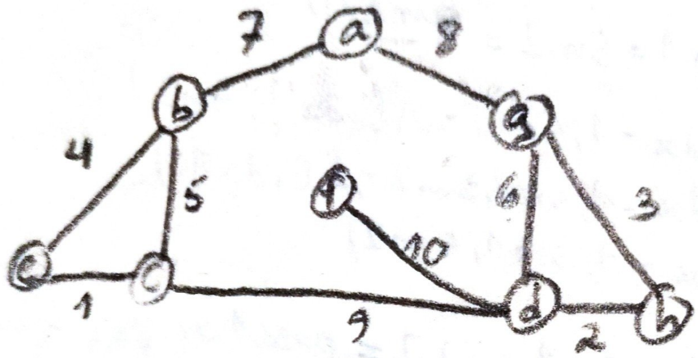
EN 2019

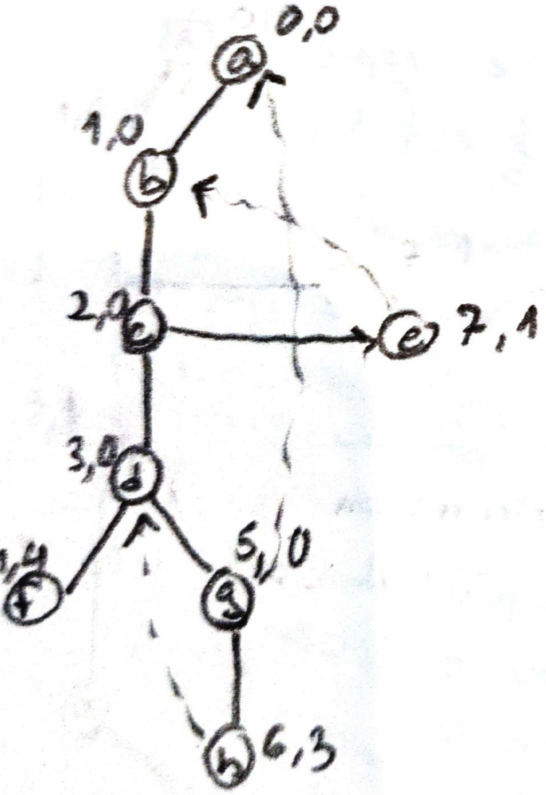
① $T(n) = \begin{cases} T(\frac{n}{2})+1, & n>1 \\ 1, & n=1 \end{cases} \Rightarrow T(n) = \Theta(\log n)$

$a=1$
$b=2$
$c=0$

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

② ⓐ

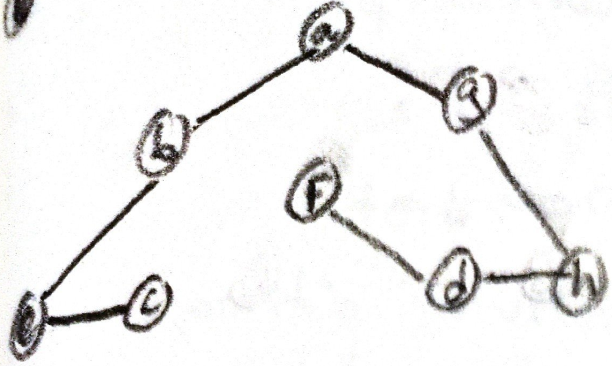


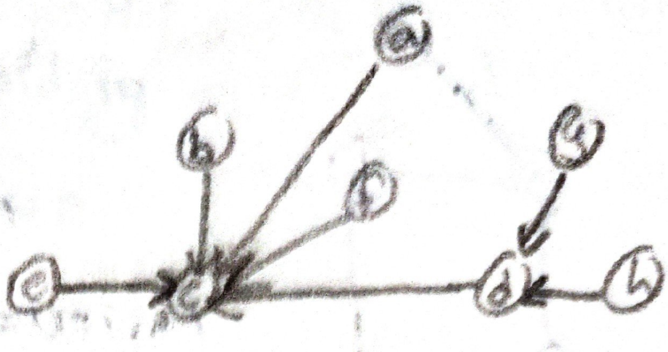


$APs = \{d\}$

$low[f] \geq dfs[d]$

1





3

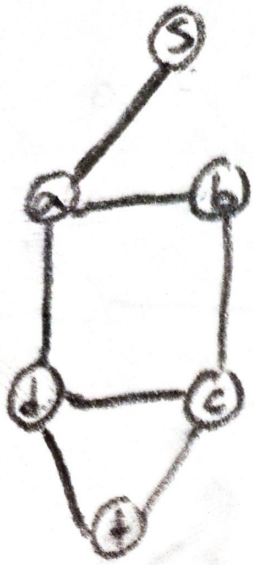
a

```
vis = [false for every index 0...|V|]

function f(cur, vis, t, G):
    if cur = t:
        return 0
    vis[cur]=True
    mx_from_cur = -∞
    for each {cur, w} ∈ E:
        if vis[w] = False
            from_w = f(w, vis, t, G)
            if from_w ≠ -∞:
                mx_from_cur = max(max_from_cur,
                                  1+from_w)
    vis[cur]=False
    return max_from_cur
```

$O(|V|!)$, aproximadamente, dado que estamos a permutar a ordem de escolha dos vertices

(b)



Resposta correta p/ problema s → t:

s → a → b → c → d → t

Assumindo substrutura ótima, a resposta ao sub-problema b → t é b → c → d → t, com comprimento = 3, no entanto, a resposta real a esse subproblema é b → a → d → c → t, com comprimento = 4

(4)

(a)

```
function M(i, j):                                call M(1,n)
    if dp[i,j] is cached
        return dp[i,j]
    if i = j
        return 0

    mn = +∞
    for K in i,...,j-1:
        mn = min(mn, M(i,K) + M(K+1, j) + p[i,K,j])
    return dp[i,j] = mn.
```

(b)

for i in 0...n:
    dp[i, i] = 0 } ← Redundante

for i in ~~0...n~~ n-1, ..., 0 :
    for j in 0...n:
        ~~for~~ if i=j: dp[i,j] = 0
        dp[i,j] = ∞
        for K in i,...,j-1:
            dp[i,j] = min(dp[i, j], dp[i,K] + dp[K+1, j] + p[i, K, j])

return dp[1, n]