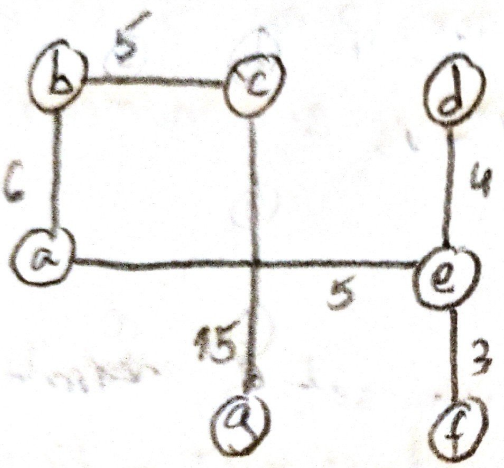
ER2019

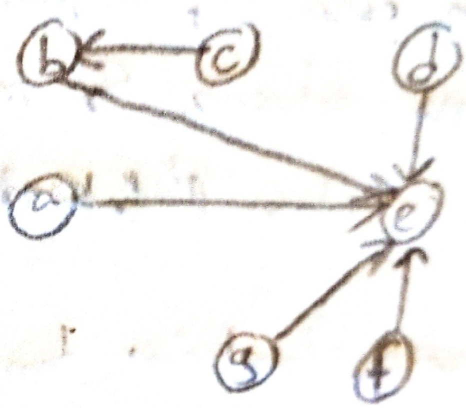
① $a=2$, $b=2$, $c=0$

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

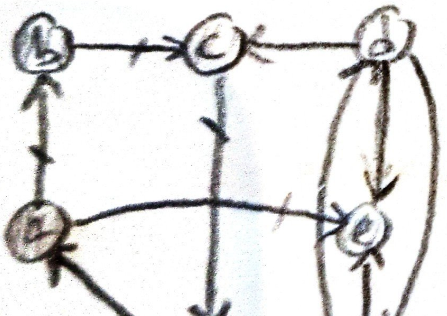
$$\log_b a = \log_2 2 = 1 > 0 = c$$

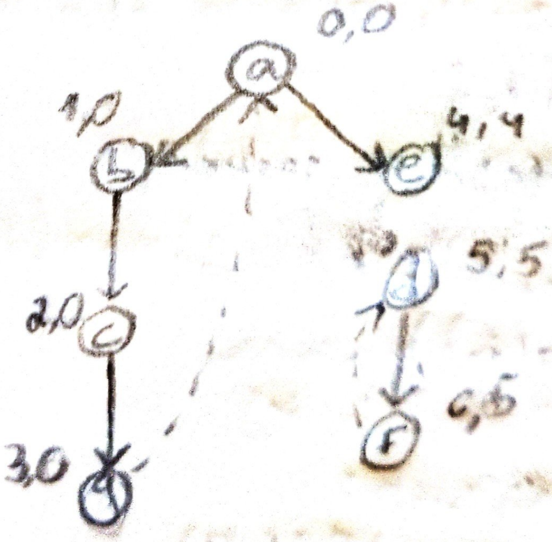
②





③





SCCs = {{e}, {d,f}, {a,b,c,g}}

(4)

(a) (1) Assumindo: S^* é ótimo, usando $l \leq K$ objetos e com valor total v^*, então, se removermos um objeto s de S^*, obtemos a solução ótima para o subproblema com $S \setminus \{s\}$ e com $K-1$ objetos, com valor total $v^* - v_s$.

(2) Negando essa implicação, quer dizer que (para $S \setminus \{s\}$ e $K-1$) existe uma solução S' tal que $v' > v^* - v_s$ e o número total de objetos $l' \leq K-1$.

(3) Ou seja, também tem de existir um set S'', tendo em conta o objeto s, com valor total $v' + v_s > v^*$ e $l \leq K$, o que contradiz (1), portanto (2) não pode ser verdade.

(b) Greedy Algorithm Paradigm

```
function g(S, K):
    sort(all(S))
    out = []
    for i in 0...K:
        out.append(S[i])
    return out
```

$O(n \log n)$

(5)

(a)

```
function T(i,j):
    if dp[i,j] is cached:
        return dp[i,j]
    if i ≤ 1: return dp[i,j] = +∞
    mx = -∞
    for l in j,...,(n-i+2)-1:
        mx = max(mx, min(D[j,l], T(i-1, l)))
    return dp[i,j] = mx
function max_wrap(K,n)
    out = -∞
    for j in 1,...,n:
        out = max(out, T(K,j))
    return out
```

call max_wrap(K,n)