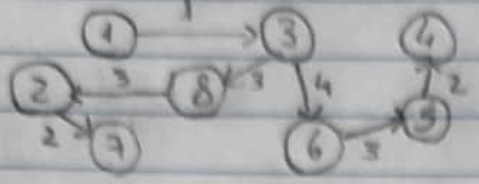
EN
22

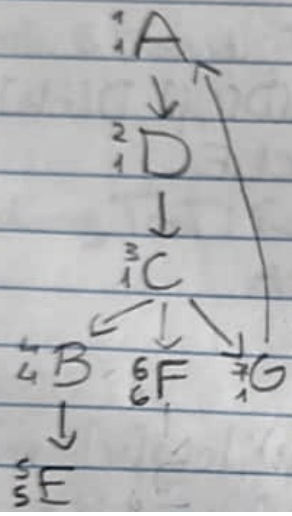
1a)



v	1	3	8	6	2	5	7	4
d	0	1	4	5	7	8	9	10

b) The shortest paths found are no longer valid, v4 would have its distance reduced from 10 to -10. In this particular case, Djikstra's algorithm would spot this, but negative values often cause the need for several iterations to find the optimal distances, and Djikstra's assumption that we can simply choose the closest node to the root over 1 iteration becomes no longer valid. Bellman-Ford's algorithm can deal with negative numbers and cycles by running several iterations.

2.



1. E
2. B
3. F
4. A D C G
(?) 5. H

dp(m+1, c+1)

3) a) DP(i,j):
    if dp[i,j] is cached:
        return dp[i,j]
    if i≤0 or j≤0:
        dp[i,j] = 0
    for v in V:
        dp[i,j] = max(dp[i,j], v + DP[i-v, j-1]
    return d

call DP(m,c)

b) DP(m,c):
    for i = 0 → m:
        dp[i,0] = 0
    for j = 0 → c:
        dp[0,j] = 0
    for i = 1 → m:
        for j = 1 → c:
            dp[i,j] = 0
            for v in V:
                if i-v ≥ 0 and j-1 ≥ 0:
                    dp[i,j] = max(dp[i,j], v + dp[i-v, j-1]
    return dp[m,c]

(estou a assumir que quando vai out of bounds é inválido.))

4) islands(M, n):
    count = 0
    for i = 0 → n-1:
        for j = 0 → n-1:
            if M[i,j] = 1 and !visited[i,j]:
                visit(i,j)
                count++
    return count

visit(i,j):
    visited[i,j] = true
    dir = [0,1,1,1,0,-1,-1,-1]
    for k = 0 → 7:
        if 0 ≤ i+k < n and 0 ≤ j + dir((k+2) % 8) < n:
            if M[i,j] = 1:
                visit(i,j)