$v_1 \to 0$

$v_2 \to \infty$ ~~10(v_4)~~ 7(v_8)

$v_3 \to \infty$ 1(v_1)

$v_4 \to \infty$ ~~12(v_3)~~ 10(v_5)

$v_5 \to \infty$ 8(v_6)

$v_6 \to \infty$ 5(v_3)

$v_7 \to \infty$ ~~10(v_6)~~ 9(v_2)

$v_8 \to \infty$ 4(v_3)

· U C ·

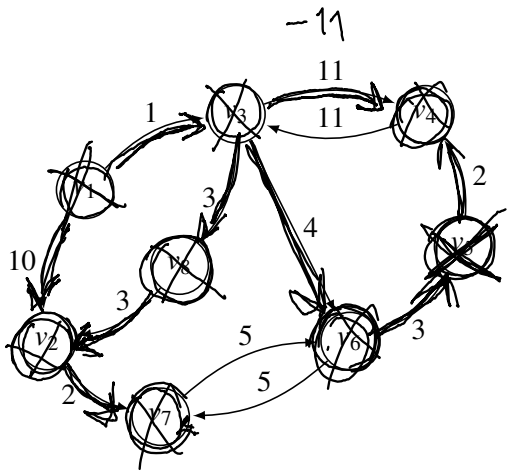
UNIVERSIDADE DE COIMBRA
Faculty of Science and Technology
Department of Informatics Engineering

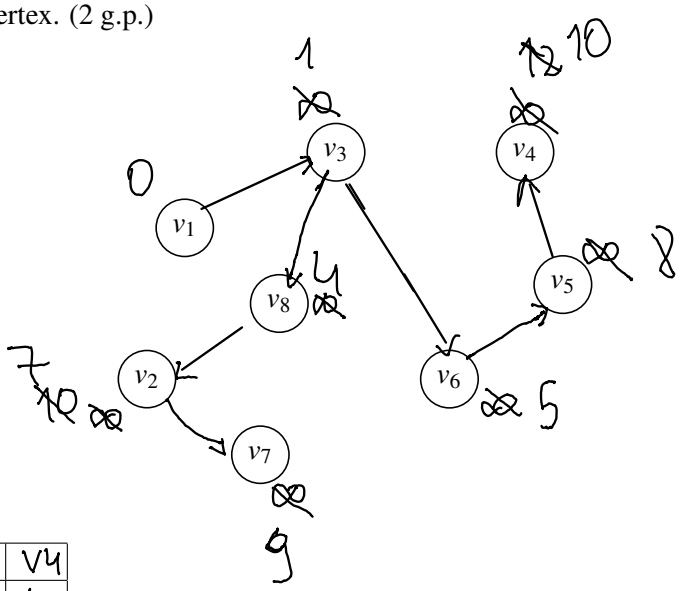
Estratégias Algorítmicas
Normal Exam – June 6 2022

Name: Tiago Jorge Coimbra da Silva Student ID: 2022216219

12 grade points in total, 2h, closed books.

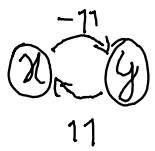
1.a) Find all shortest paths in the left-hand graph between vertex v_1 and every other vertex using Dijkstra algorithm. Draw the arcs that belong to the path in the right-hand graph. Fill in the table with the visited vertices, ordered according to the visiting order of Djikstra algorithm, and with the shortest distance from vertex v_1 to every vertex. (2 g.p.)





Vertices:	v_1	v_3	v_8	v_6	v_2	v_5	v_7	v_4
Distance:	0	1	4	5	7	8	9	10

1.b) Consider that length of the arc from vertex v_3 to v_4 changed to -11. Are the shortest paths that you have found above still valid after this modification? Would you be able to use Dijkstra algorithm? Justify your answer. (1 g.p)

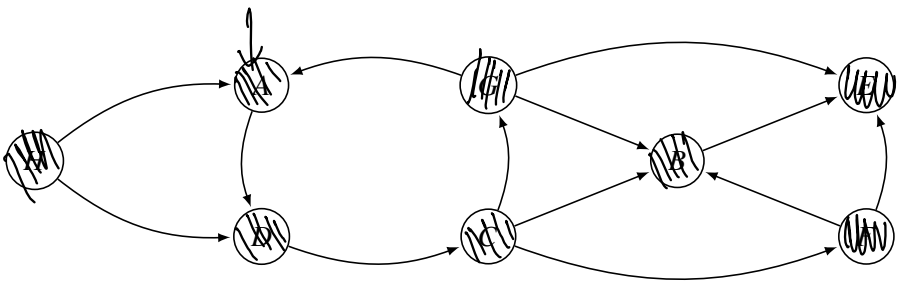


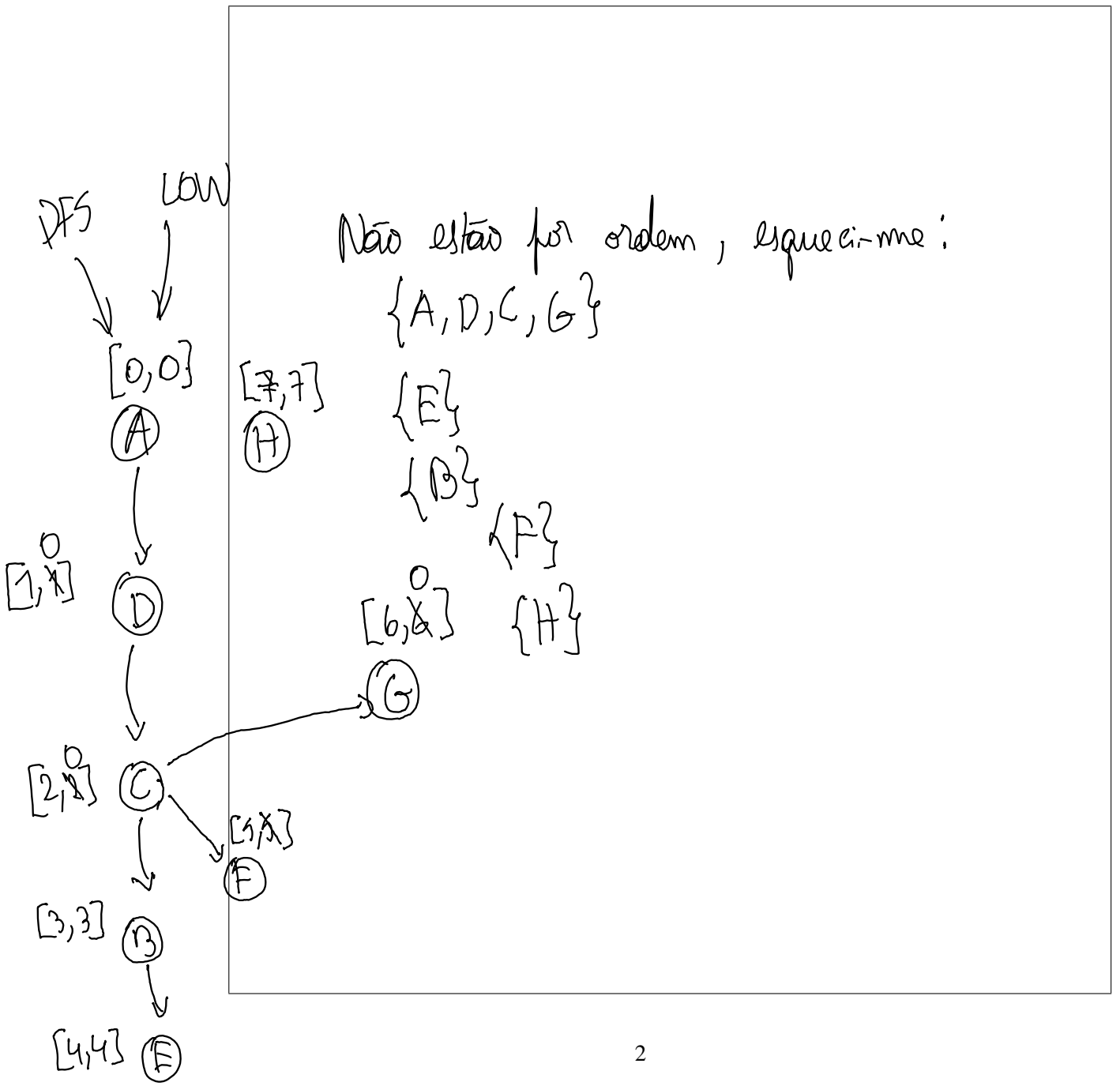
$dy \leq (d(x) - 11)$

$d(x) \geq (dy + 11)$

Não serviam válidos, não posso usar Dijkstra com pesos negativos. A aproach guloso deteta que é vantajoso ficar preso no ciclo.

2. Find the strongly connected components in the following graph using Tarjan Algorithm. Justify your answer by reporting the DFS tree starting from node *A*, choosing the vertices for traversal in alphabetic order of the labels, and by explicitly writing the final values for `dfs` and `low` at each vertex. In addition, report the strongly connected components, ordered by the time they are found in the DFS tree traversal. (2 g.p.)





3. Consider the following recurrence relation. Let $v_1, \ldots, v_\ell$ be a sequence of ℓ positive integers. We define $M(i,j)$, $0 \leq i \leq m$, $0 \leq j \leq c$, as follows

$$M(i,j) = \begin{cases} 0 & \text{if } i \leq 0 \\ 0 & \text{if } j \leq 0 \\ \max \begin{cases} v_1 + M(i-v_1, j-1) \\ v_2 + M(i-v_2, j-1) \\ \ldots \\ v_\ell + M(i-v_\ell, j-1) \end{cases} & \text{if } i > 0 \text{ and } j > 0 \end{cases}$$

3.a) Give the pseudo-code of a top-down dynamic programming algorithm that explores the recurrence above to find the value for $M(m,c)$. (2 g.p.)

```
Function M(i,j)
    If i<=0
        return 0
    If j<=0
        return 0
    If cached then return dp[i][j]
    for k from 1 to l:
        dp[i][j] = max(v[k] + M(i-v[k], j-1), dp[i][j])
    return dp[i][j]
```

3.b) Give the pseudo-code of a bottom-up dynamic programming algorithm that explores the recurrence above to find the value for $M(m,c)$ and discuss its time complexity. (2 g.p.)

```
Function M(i,j):
    for i from 1...m
        dp[i][0] <- 0
    for j from 1...c
        dp[0][j] <- 0
    for i from 1...m
        for j from 1...c
            for k from 1...l
                dp[i][j] =
```

4. Consider a square binary matrix of size *nxn* that is filled up with ones and zeros. You want to count how many *islands* exist in this matrix. An island is a set of cells in the matrix containing only ones that has the following properties:

   (a) There exists a sequence of adjacent cells with ones between every pair of cells in the island;

   (b) It is not possible to add more cells to the island without breaking Property (a).

   Two cells are adjacent if the absolute difference between the two row indices and between the two column indices is at most 1. Note the following particular case: a cell with one is an island if all adjacent cells contain 0. For example, in the following matrix, you can find three islands.

0	1	0	0	0	1	0
0	1	0	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0

   Describe a pseudo-code that allows to count how many islands exist in a matrix of size *nxn*. Assume that the matrix is stored in a data structure M, where $M(i, j)$ allows to access the number in the matrix at row i and column j. Give the time complexity of your approach and show (informally) that is correct. (3 g.p.)