Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de datos y algoritmos — 2020-2

Ayudantía 12

Pregunta 1

Dada una matriz de 1s y 0s encuentre la cantidad de sub-cuadrados de 1s existentes.

Ejemplo:

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

En este caso el output debiese ser 15

Hay 10 cuadrados de lado 1

Hay 4 cuadrados de lado 2

Hay 1 cuadrado de lado 3

Total = 10 + 4 + 1 = 15

Pregunta 2

Dado un nodo fuente s, escribe un algoritmo tal que se marque

$$v.distance = -\infty$$

para todo vértice v si es que se encuentra un ciclo negativo en algún camino desde s hasta v

Pregunta 1

```
def countSquares(matrix):
    counter = 0
    for i in matrix[0]:
        if i == 1:
            counter += 1
    for i in range(1, len(matrix)):
        if matrix[i][0] == 1:
            counter += 1
    for i in range(1, len(matrix)):
        for j in range(1, len(matrix[0])):
            if matrix[i][j] == 1:
                matrix[i][j] += min(
                    matrix[i - 1][j - 1],
                    matrix[i - 1][j],
                    matrix[i][j - 1],
                )
            counter += matrix[i][j]
    return counter
```

Pregunta 2

Se partirá con el algoritmo `Bellman-Ford` y se modificará su tercer ciclo con el fin de poder marcar los nodos como es solicitado.

```
1: procedure MODIFIED BELLMAN FORD(G, s)
2:     distance = list of size G.V
3:     predecessor = list of size G.V
4:     marked_nodes = list of size G.V
5:     for each vertex ∈ G.V do
6:         vertex.d = ∞
7:         vertex.π = null
8:     end for
9:     s.d = 0
10:    i = 0
11:    while i ≤ |G.V| − 1 do
12:        for each edge (u,v) in G.E do
13:            w = edge.weight
14:            if u.d + w ≤ v.d then
15:                v.d = u.d + w
16:            end if
17:        end for
18:        i = i + 1
19:    end while
20:    for each edge (u,v) in G.E do
21:        w = edge.weight
22:        if u.d + w ≤ v.d then
23:            add V to marked_nodes
24:        end if
25:    end for
26:    for each vertex in marked_nodes do
27:        Follow_Node(vertex)
28:    end for
29:    for each vertex in marked_nodes do
30:        DFS(vertex)
31:    end for
32: end procedure
33: procedure FOLLOW_NODE(v)
34:    if v ≠ null ∧ v.d ≠ −∞ then
35:        v.d = −∞
36:        Follow_node(v.π)
37:    else
38:        return
39:    end if
40: end procedure
```