Exercises #codeforlife - introduction

For all challenges, we have considered that all or part of the below inputs are provided. Given a certain **public venue**, it can be split into several areas and mapped onto a graph, with the following properties:

- A maximum number of people allowed into each area;
- An edge capacity, showing how many people can *move* within a given amount of capacity from one area to another;
- Exit(s) position(s) and edge capacity towards the exit showing how many people can be evacuated during 1 iteration.

Prerequisites:

- Weighted bidirectional graph G with number of vertices V, edges E and exit edges: G=(V, E)
- Non negative capacities for each node, route, and capacity per exit(s)
- Danger starting point(s) (for challenges 4 and 5).

Time: 48 hours

Contents

Challenge 1

Name: Evacuation Time (Level: Easy)

Given a graph, with a number of individuals associated to each node, an exit placement and an evacuation capacity of each edge (connecting the nodes to each other and towards the exits), compute the complete evacuation time expressed in a **minimum number of iterations.**

Application of such an algorithm:

A set of cameras can provide information about the number of people in each area of a public venue. Thus, real-time inputs can be read: the number of persons and their distribution by area (node) – this constitutes the first input. An alert can be raised (call to the authorities) when the maximum allowed evacuation time is exceeded for a venue. The maximum allowed evacuation time is expressed in iterations and this constitutes the second input.

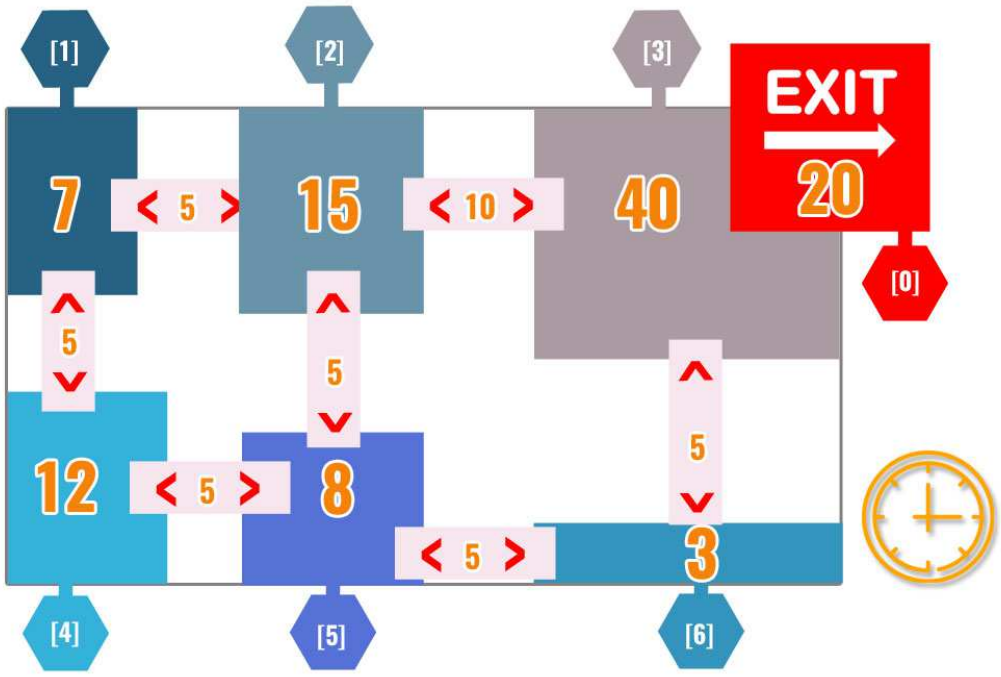
Input:

- Array with capacity for each node, **array of N integer elements.** N is the size of the input array.
- Matrix M(N+1)x(N+1) representing the connections of the nodes and capacities for each edge, and exit(s). Row and column with 0 index contain the capacities towards the exit(s), where there is an exit.

Output: number of iterations needed to evacuate the given number of people **I Integer >=1.**

Example 1:

For the following graph:



Input:
7,15,40,12,8,3
0,0,0,20,0,0,0
0,0,5,0,5,0,0
0,5,0,10,0,5,0
20,0,10,0,0,0,5
0,5,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0

Output: 5

Consider the graph above (with one exit only) represented in a matrix:

Each exit node will be represented by Node 0 with infinite capacity.

Weight of the nodes: infinite 7 15 40 12 8 3	0	1	2	3	4	5	6
0	0	0	0	20	0	0	0
1	0	0	5	0	5	0	0
2	0	5	0	10	0	5	0
3	20	0	10	0	0	0	5
4	0	5	0	0	0	5	0
5	0	0	5	0	5	0	5
6	0	0	0	5	0	5	0

For each iteration, this is how node capacities are moved to the Exit:

				20 (exit)
1st	7	15	33	
iteration	2	5	3	

				40 (exit)
2nd	4	12	26	
iteration	0	0	3	

				60 (exit)
3rd	0	6	19	
iteration	0	0	0	

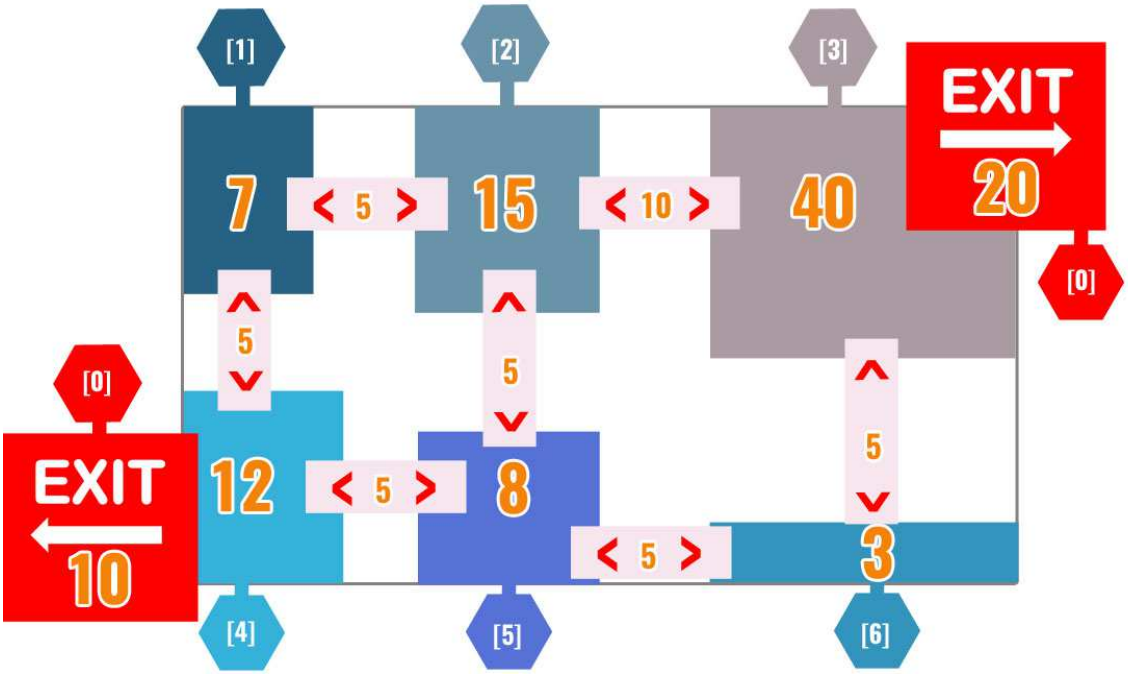
				79 (exit)
4th	0	0	6	
iteration	0	0	0	

				85 (exit)
5th	0	0	0	
iteration	0	0	0	

Initial state	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Exit[3] = 0	Exit[3] = 20	Exit[3] = 40	Exit[3] = 60	Exit[3] = 79	Exit[3] = **85**
N[1] = 7	N[1] = 7	N[1] = 4	N[1] = 0	N[1] = 0	N[1] = 0
N[2] = 15	N[2] = 15	N[2] = 12	N[2] = 6	N[2] = 0	N[2] = 0
N[3] = 40	N[3] = 33	N[3] = 26	N[3] = 19	N[3] = 0	N[3] = 0
N[4] = 12	N[4] = 2	N[4] = 0	N[4] = 0	N[4] = 6	N[4] = 0
N[5] = 8	N[5] = 5	N[5] = 0	N[5] = 0	N[5] = 0	N[5] = 0
N[6] = 3	N[6] = 3	N[6] = 3	N[6] = 0	N[6] = 0	N[6] = 0

Solution:
The number of people evacuated per iteration = 20
The total volume to evacuate = 85
Total transport capacity = 40
From the initial data, the time needed to evacuate = **Output = 5** (iterations)

Example 2:



Input:
7,15,40,12,8,3
0,0,0,20,10,0,0
0,0,5,0,5,0,0
0,5,0,10,0,5,0
20,0,10,0,0,0,5
10,5,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0

Output: 3

1st iteration		3	4	33	**20 (exit)**
	10 (exit)	12	0	3	

2nd iteration		0	0	20	**40 (exit)**
	20 (exit)	5	0	0	

3rd iteration		0	0	0	**60 (exit)**
	25 (exit)	0	0	0	

2 exits
Weight of the nodes: infinite 7 15 40 12 8 3

	0	1	2	3	4	5	6
0	0	0	0	20	10	0	0
1	0	0	5	0	5	0	0
2	0	5	0	10	0	5	0
3	20	0	10	0	0	0	5
4	10	5	0	0	0	5	0
5	0	0	5	0	5	0	5
6	0	0	0	5	0	5	0

The number of people evacuated per iteration = 30
The total volume to evacuate = 85
Total transport capacity = 40
From the initial data, the time needed to evacuate = **Output** = 3 (iterations)

For each iteration, explain how node capacities are moved to the exit nodes:

Initial state	**Iteration 1**	**Iteration 2**	**Iteration 3**
Exit[3] = 0	**Exit[3] = 20**	**Exit[3] = 40**	**Exit[3] = 60**
Exit[4] = 0	**Exit[4] = 10**	**Exit[4] = 20**	**Exit[4] = 25**
N[1] = 7	N[1] = 3	N[1] = 0	N[1] = 0
N[2] = 15	N[2] = 4	N[2] = 0	N[2] = 0
N[3] = 40	N[3] = 33	N[3] = 20	N[3] = 0
N[4] = 12	N[4] = 12	N[4] = 5	N[4] = 0
N[5] = 8	N[5] = 0	N[5] = 0	N[5] = 0
N[6] = 3	N[6] = 3	N[6] = 0	N[6] = 0

Challenge 2

Name: Exit Placement (Level: Intermediate)

Given a graph, with a number of individuals associated to each node, a number of exits, an evacuation capacity of each edge (connecting the nodes to each other and towards the exits) and a maximum evacuation time expressed in iterations, you will have to determine where the exits would be best placed. The number of people evacuated per exit(s) will be the same.

Application of such an algorithm:

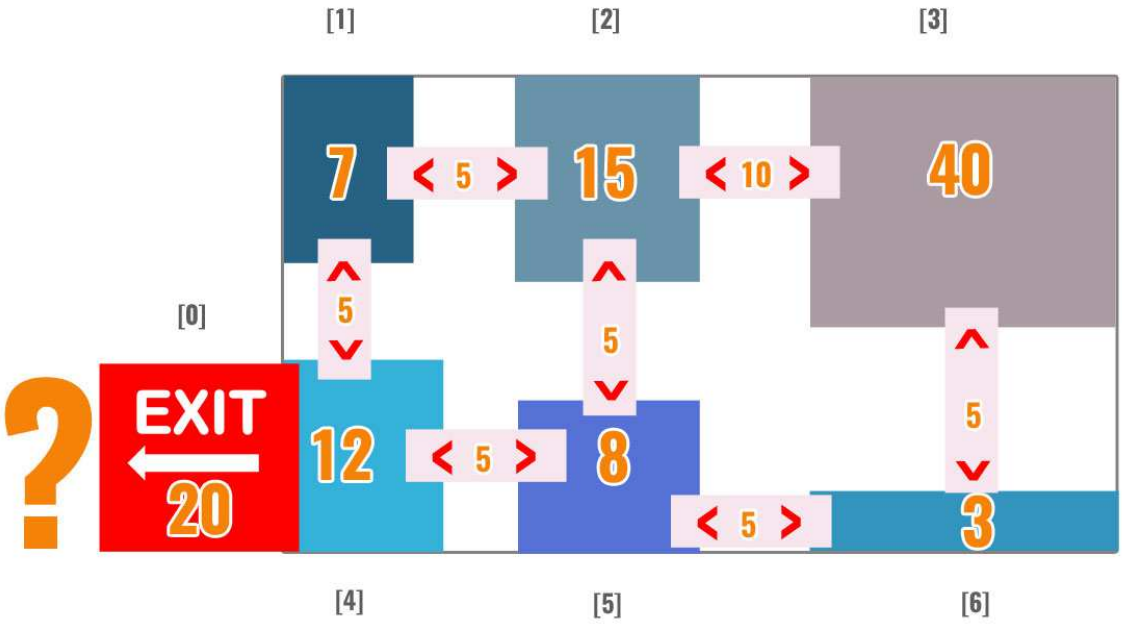
An algorithm of this type can be used for simulation software in order to determine the exit placement based on the mapping of the public venue. Such software would be very useful in the design phase of the public venues in order to comply with the existing rules and regulations and simulate various evacuation scenarios.

Input:

- Array with capacity for each node, **array of N integer elements.** N is the size of the input array.
- Matrix M(N+1)x(N+1) representing the connections of the nodes and capacities for each edge, and exit(s). Row and column with 0 index contain the capacities towards the exit(s), where there is an exit.
- Number of iterations needed to evacuate people **I2 Integer >=1.**

Output: where the exit should be placed so the evacuation can be done in the given number of iterations (the volume of evacuation per exit(s) will be the same) = **the exact coordinates of the exit(s) E = x,0** where x = the node where the exit should be moved and 0.

Example:



Input:

7,15,40,12,8,3
0,0,0,0,20,0,0
0,0,5,0,5,0,0
0,5,0,10,0,5,0
0,0,10,0,0,0,5
20,5,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0
6

Output: 3,0

Solution:

Considering graph above:

You have to find out if the existing exit(s) are optimally placed so the evacuation can be done in the given number of iterations – I2. Basically, you should compare the input given for the number of iterations with the previous output (I1) and determine, depending on the output, if the exit(s) should be changed.
If:
I2 > I1, then you have nothing to optimize concerning the exit(s)
I2 < I1, then look for another exits ()

First, calculate the number of iterations (using the code from the first challenge).
For each iteration, explain how node capacities are moved to the Exit.
The number of people evacuated per iteration = 12 (because the maximum number of nodes before the exit is 12).
The total volume to evacuate = 85
Total transport capacity = 40
From the initial data, the time needed to evacuate = 8 iterations

For the given number of iterations = I2 = 6 < 8 => an optimization is required.

You can verify if the capacity of one node is bigger than the capacity of the exit.

Because CapacityNode[3] > 20, then the exit should be placed next to Node[3] => Node **3,0**.

Challenge 3

Name: Edge Capacity Increase (Level: Intermediate)

Given a graph, with a number of individuals associated to each node, a number of exits with known placement, an evacuation capacity of each edge (connecting the nodes to each other and towards the exits) and a maximum evacuation time expressed in iterations, determine which *edge capacities* to increase in order to achieve evacuation within the given time.

The total transport capacity (sum of all edge capacities) can increase by maximum 50%.

The edge capacity increase has to be minimal.

Application of such an algorithm:

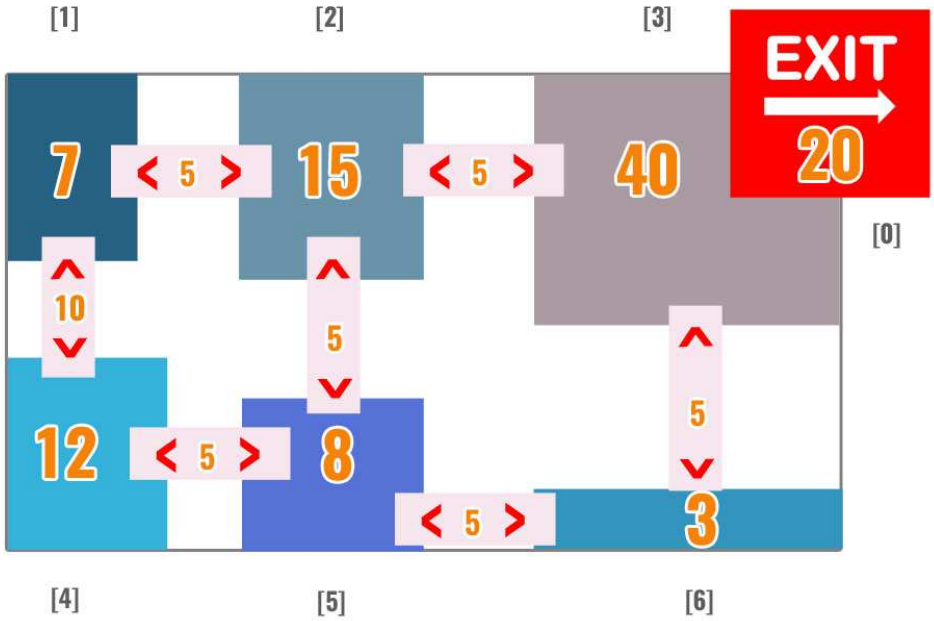
Public venues are often faced with the challenge of re-designing their interior, in order to comply with safety norms and regulations. This type of algorithm could also be used for simulation software, in order to determine exactly which spaces and evacuation paths should be favored in case a faster evacuation is necessary. Consider that additional lighting can be installed in order to highlight the desired evacuation paths (like in airplanes), depending on the outputs of this algorithm.

Input:

- Array with capacity for each node, **array of N integer elements.** N is the size of the input array.
- Matrix **M(N+1)x(N+1)** representing the connections of the nodes and capacities for each edge, and exit(s). Row and column with 0 index contain the capacities towards the exit(s), where there is an exit.
- Number of iterations to evacuate people **I2 Integer >=1.**

Output: with which edge capacities to increase in order to achieve evacuation in the given amount of time (the total transport capacity can increase by maximum 50%). The return values will be the exact matrix with the new values for the edges.

Example:



Input:

7,15,40,12,8,3
0,0,0,20,0,0,0
0,0,5,0,10,0,0
0,5,0,5,0,5,0
20,0,5,0,0,0,5
0,10,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0
5

Output:

0,0,0,20,0,0,0
0,0,5,0,10,0,0
0,5,0,9,0,5,0
20,0,9,0,0,0,5
0,10,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0

M[2,3] = 10

Solution:

Increase the capacities, keeping in mind that the total transport capacity can increase by 50%.

Number of iterations (6) = 5

Output:
M[2,3] = 10

You want to find out what edge capacities can be increased so the evacuation can be done in 5 iterations.

First calculate the number of iterations needed for the initial input. For each iteration, explain how node capacities are moved to the Exit.
The volume of evacuations per iteration = 20
The total volume to evacuate = 85
Total transport capacity = 40
Total transport capacity towards Node[3] = 10
From the initial data, the time needed to evacuate = 7 iterations:
I1 = 20 (20+8 =28), I2 = 40 (8+8), I3 = 56 (0+8), I4 = 64 (0+8), I5 = 72 (0+8), I6 = 80 (0+5), I7 = 85

The restrictions are related to:
- the transport capacity from the neighboring nodes of Node[3] - Node[2] and Node[6]
- the maximum capacity of Node[2] and Node[6]

Initial state	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**	**Iteration 6**	**Iteration 7**
Exit = 0	Exit = 20	Exit = 40	Exit = 56	Exit = 64	Exit = 72	Exit = 80	Exit = 85
N[1] = 7	N[1] = 12	N[1] = 5	N[1] = 0	N[1] = 0	N[1] = 0	N[1] = 0	N[1] = 0
N[2] = 15	N[2] = 15	N[2] = 15	N[2] = 15	N[2] = 10	N[2] = 5	N[2] = 0	N[2] = 0
N[3] = 40	N[3] = 28	N[3] = 16	N[3] = 8	N[3] = 8	N[3] = 8	N[3] = 5	N[3] = 0
N[4] = 12	N[4] = 2	N[4] = 0	N[4] = 0	N[4] = 0	N[4] = 0	N[4] = 0	N[4] = 0
N[5] = 8	N[5] = 5	N[5] = 6	N[5] = 0	N[5] = 0	N[5] = 0	N[5] = 0	N[5] = 0
N[6] = 3	N[6] = 3	N[6] = 3	N[6] = 3	N[6] = 3	N[6] = 0	N[6] = 0	N[6] = 0

				20
	12	15	28	
1st	2	5	3	

				40
	5	15	16	
2nd	0	6	3	

3rd				**56**

	0	15	8	
	0	3	3	

				64
	0	10	8	
4th	0	0	3	

				72
	0	5	8	
5th	0	0	0	

				80
	0	0	5	
6th	0	0	0	

				85
	0	0	0	
7th	0	0	0	

Challenge 4

Name: Evacuation Time in a Given Scenario (Level: Complex)

Given a graph with a number of individuals associated to each node, an exit placement and an evacuation capacity of each edge (connecting the nodes to each other and towards the exits), compute the evacuation time for a full evacuation, in number of iterations, knowing that a set of nodes vanish from the graph. The vanishing nodes at each iteration are known.

Application of such an algorithm:

This scenario can be used to simulate real life situations: fire, assault or flood in a public venue. By using various inputs for the vanishing nodes (hence, various real life situations simulated), decisions can be made regarding the highlighting of the exit which should be used and the path towards that exit.

A node is fully evacuated when its capacity is 0.
A node is vanishes when its capacity is 0 and the associated edges become 0.

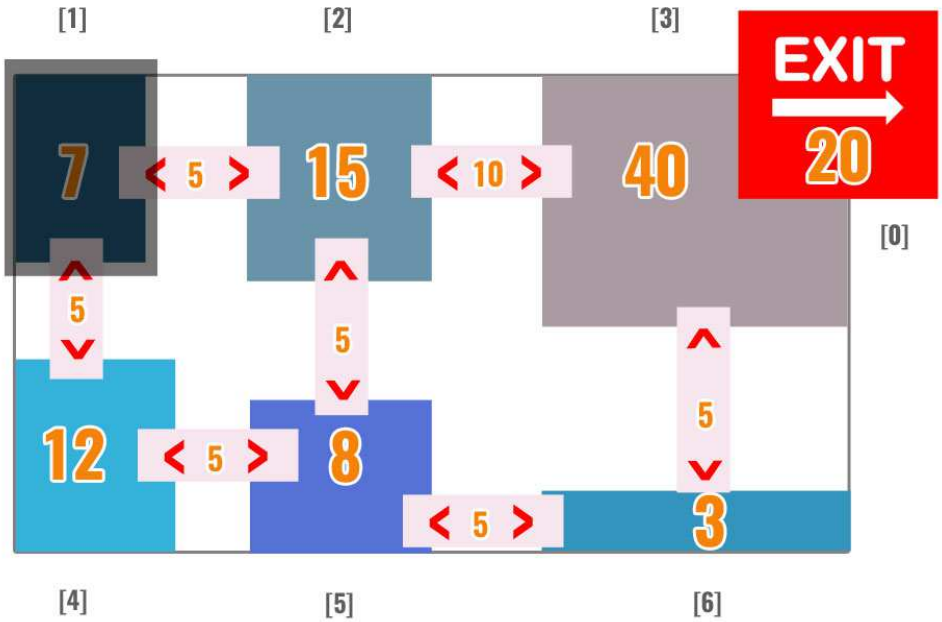
Input:

- Array with capacity for each node, **array of N integer elements**. N is the size of the input array.
- Matrix M(N+1)x(N+1) representing the connections of the nodes and capacities for each edge, and exit(s). Row and column with 0 index contain the capacities towards the exit(s), where there is an exit.

- Node(s) that vanish followed by the iteration(s) when they vanish - a node that vanishes + the iteration(s) when they vanish are represented on different lines in the entry file, separated by comma.

Output: number of iterations needed to evacuate the given graph: **Integer >=1.**

Example:



Input:
7,15,40,12,8,3
0,0,0,20,0,0,0
0,0,5,0,5,0,0
0,5,0,10,0,5,0
20,0,10,0,0,0,5
0,5,0,0,0,5,0
0,0,5,0,5,0,5
0,0,0,5,0,5,0
1,2

Node 1 will vanish at the 2nd iteration.

Output: 5

Solution:

For each iteration, explain how the node capacities are moved to Node 0.
The number of evacuations per iteration = 20
The total volume to evacuate = 85
Total transport capacity = 40

Prioritize Node[1], the time needed to evacuate = **Output** = 5 (iterations)

Initial state	**Iteration 1**	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**
Exit = 0	Exit = 20	Exit = 40	Exit = 60	Exit = 79	Exit = 85
N[1] = 7	N[1] = 7	N[1] = 4	N[1] = 0	N[1] = 0	N[1] = 0
N[2] = 15	N[2] = 15	N[2] = 12	N[2] = 6	N[2] = 0	N[2] = 0
N[3] = 40	N[3] = 33	N[3] = 26	N[3] = 19	N[3] = 0	N[3] = 0
N[4] = 12	N[4] = 2	N[4] = 0	N[4] = 0	N[4] = 6	N[4] = 0
N[5] = 8	N[5] = 5	N[5] = 0	N[5] = 0	N[5] = 0	N[5] = 0
N[6] = 3	N[6] = 3	N[6] = 3	N[6] = 0	N[6] = 0	N[6] = 0

				20
	2	15	33	
1st	7	5	3	

				40
	0	16	26	
2nd	0	0	3	

				60
	0	6	19	
3rd	0	0	0	

				79
	0	0	6	
4th	0	0	0	

				85
	0	0	0	
5th	0	0	0	

Challenge 5

Name: Edge Capacity Increase in a Given Scenario (Level: Complex)

Given a graph, with a number of individuals associated to each node, a number of exits with known placement, an evacuation capacity of each edge (connecting the nodes to each other and towards the exits) and a maximum evacuation time expressed in iterations, determine which edge capacities to increase in order to achieve evacuation in the given amount of time, knowing that a set of nodes vanish from the graph at each iteration. The vanishing nodes at each iteration are known.

The scenario is known in advance, so you have to plan in advance whom you should evacuate and how. The total transport capacity (sum of all edge capacities) can increase by maximum 50%.

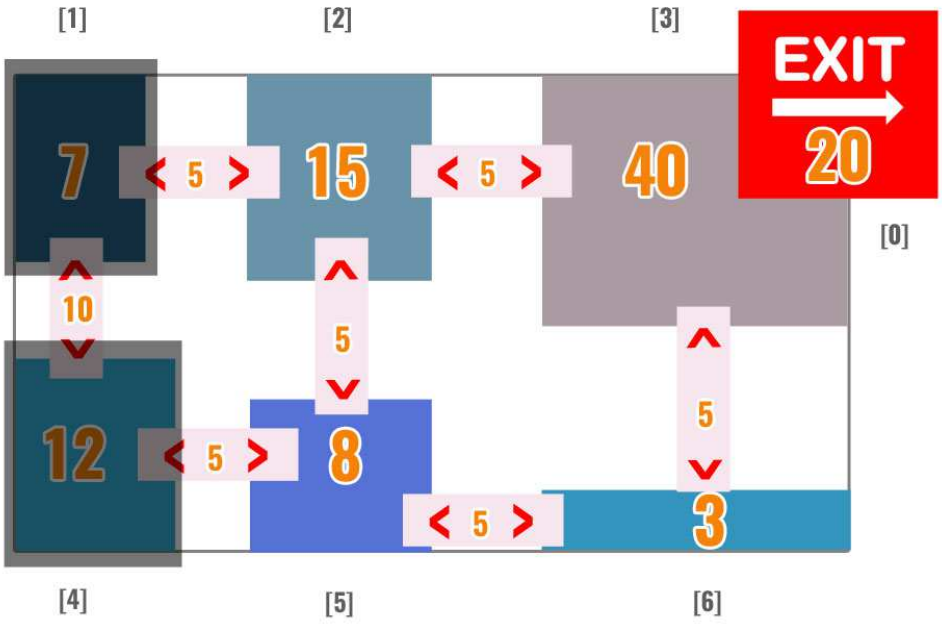
The edge capacity increase has to be minimal.

Input:

- Array with capacity for each node, **array of N integer elements.**
- Matrix M(N+1)x(N+1) representing the connections of the nodes and capacities for each edge, and exit(s). Row and column with 0 index contain the capacities towards the exit(s), where there is an exit.
- Node(s) that vanish followed by the iteration(s) when they vanish – a node that vanishes + the iteration(s) when they vanish are represented on different lines in the entry file, separated by comma.
- Number of iterations to evacuate people **I Integer >=1.**

Output: which edge capacities to increase in order to achieve evacuation within the available time and based on the known vanishing nodes (the total transport capacity can increase by maximum 50%).
The return values will be the exact matrix with the new values for the edges.

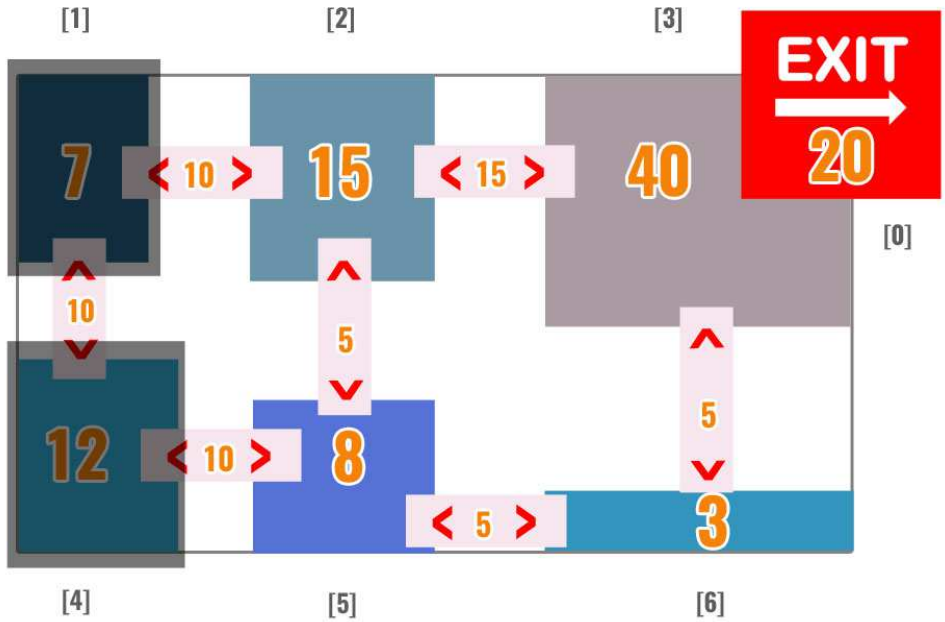
Example:



Input:
7,15,40,12,8,3
0,0,0,**20**,0,0,0
0,0,**5**,0,**10**,0,0
0,**5**,0,**5**,0,**5**,0
20,0,**5**,0,0,0,**5**
0,**10**,0,0,0,**5**,0
0,0,**5**,0,**5**,0,**5**
0,0,0,**5**,0,**5**,0
1,2
4,3
5

Nodes 1 and 4 will vanish at the 2nd iteration and 3rd iteration respectively.
The total transport capacity - 40 - can be increased to maximum 60.

Output:
0,0,0,20,0,0,0
0,0,7,0,10,0,0
0,7,0,10,0,5,0
20,0,10,0,0,0,5
0,10,0,0,0,6,0
0,0,5,0,6,0,5
0,0,0,5,0,5,0



M[1,2] = 10
M[2,3] = 15
M[4,5] = 10

Solution:

- Set of sensors or nodes represented in a weighted graph as below: [7, 15, 40, 12, 8, 3] (1)
- How nodes are connected (2)
- Capacity for each edge (3)
- The maximum capacity for each node (4)
- Exit placement & capacity (5) = Node[0]
- Number of iterations (6) = 5
- Node[1] will vanish at iteration 2.
- Node[4] will vanish at iteration 3.

For the graph above, you want to find out what edge capacities can be increased so the evacuation can be done in 5 iterations.

First calculate the number of iterations needed for the initial input. For each iteration, explain how node capacities are moved to Node 0.
The volume of evacuations per iteration = 20
The total volume to evacuate = 85
Total transport capacity = 40
Total transport capacity towards Node[3] = 10
From the initial data, the time needed to evacuate = 7 iterations (as calculated in Challenge 3)

Knowing that:

- Iterations = 5
- Node[1] will vanish at iteration 2 and Node[4] will vanish at iteration 3.

The following edges can be increased:

				20
	0	12	38	
1st	4	8	3	

				40
	0	5	33	
2nd	0	4	3	

				60
	0	4	21	
3rd	0	0	0	

				80
	0	0	5	
4th	0	0	0	

				85
	0	0	0	
5th	0	0	0	