You are given an undirected weighted `graph`, which is represented as an adjacency matrix. Find the shortest path between a `start` node and a `finish` node in the `graph`. You are allowed to add at most one edge of a given `weight`between any two nodes that are not directly connected to each other.

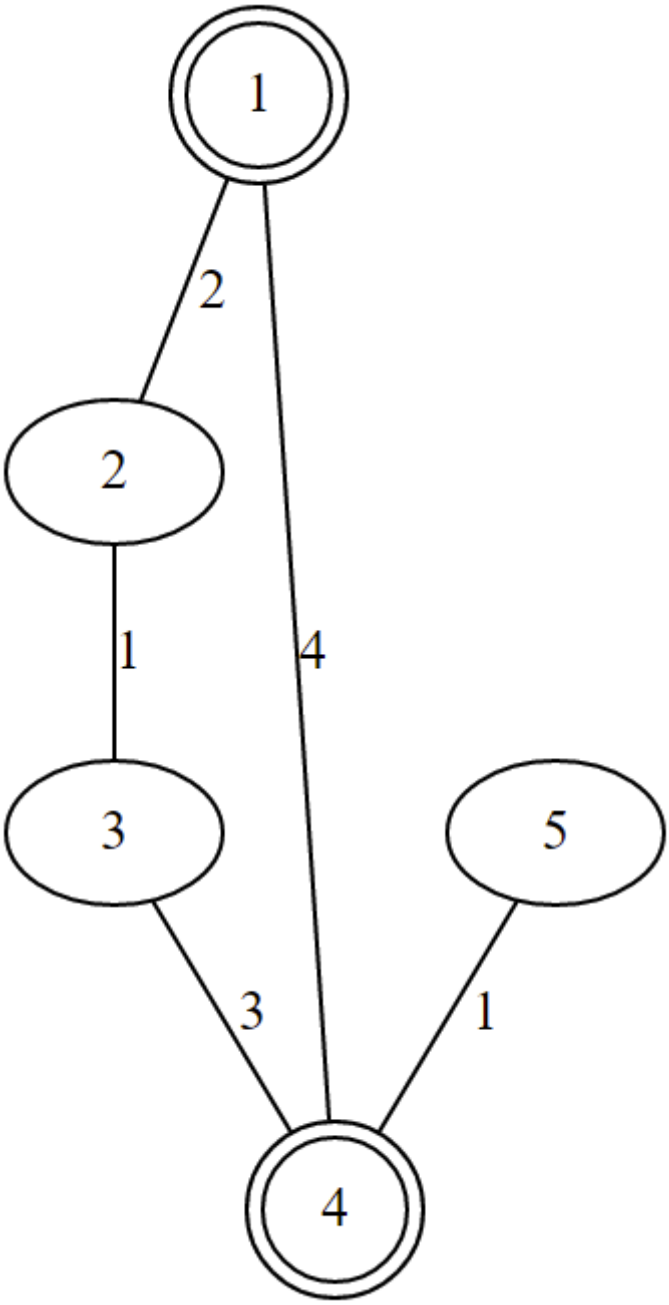
Example

For `start = 1`, `finish = 4`, `weight = 2` and

```
graph = [[0, 2, 0, 4, 0],
         [2, 0, 1, 0, 0],
         [0, 1, 0, 3, 0],
         [4, 0, 3, 0, 1],
         [0, 0, 0, 1, 0]]
```

the output should be
`shortestPathWithEdge(start, finish, weight, graph) = 3`.

In the original graph, the shortest distance between nodes `1` and `4` is equal to `4`. But you can add an edge of weight `2` between nodes `1`and `5`, making the resulting distance `3`.



Input/Output

- **[time limit] 4000ms (py3)**
- **[input] integer start**

  Constraints:
  `1 ≤ start ≤ graph.length.`
- **[input] integer finish**

  Constraints:
  `1 ≤ finish ≤ graph.length.`
- **[input] integer weight**

Constraints:
`1 ≤ weight ≤ 10^5`.

- **[input] array.array.integer graph**

  Constraints:
  `1 ≤ graph.length ≤ 1000`,
  `graph[i].length = graph.length`,
  `0 ≤ graph[i][j] ≤ 10^5`.

- **[output] integer**

  The shortest path from `start` to `finish` with the possibility of adding an extra edge with the given `weight`.