CENG 315 - ALGORITHMS - TAKE HOME EXAM 3

Compiled as:

```
g++ test.cpp the3.cpp -Wall -std=c++11 -o test
./test
```

Specifications

- There is **1 task** to be solved in **12 days** in this take home exam.
- You will implement your solutions in **the3.cpp** file.
- You are free to add other functions to **the3.cpp**.
- Do not change the first line of **the3.cpp**,which is ***#include "the3.h"***
- Do not include any other library or write *include* anywhere in your *lab3.cpp* file (not even in comments)
- *<iostream>, <vector>, <numeric>, <utility> , <climits>* are included for your convenience.

- You are given a test.cpp file to **run** and **debug** your work. You can and you are encouraged to modify this file to add different test cases.
- Clicking **run** or **debug** will compile your solutions with **test.cpp** file.
- **evaluate** will be done **after the deadline**. You will not be able to evaluate yourself.

- The system have the following limits:
  - a maximum execution time of 1 minute
  - a 256 MB maximum memory limit
  - a stack size of 64 MB for function calls (ie. recursive solutions)

- Each task has a complexity constraint explained in respective sections.
- Solutions with longer running times will be heavily **penalized**. You will receive 20% of the total grade.

- If you are sure that your solution works in the expected complexity constrains but your evaluation fails due to limits in the lab environment, the constant factors may be the problem.
- If your solution is correct, the time and memory limits may be adjusted to accept your solution after the lab. Please send an email if that is the case for you.

TAKE HOME EXAM - IMPORTANT NODES

```
int Important (int n, const int**& edgeList, int*& scores);
```

In this exam you need to find the order of importance of nodes in a given (non-negative) weighted undirected graph where n indicates the number of nodes whereas $edgeList$ represents the edges given as adjacency matrix.

The importance score of nodes are defined as, the sum of ratio of shortest-path passing through the given node to shortest-path over all node pairs. To be precise, given a graph $G = (V, E)$:

$$scores(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

where $v \in V$, $\sigma(s,t)$ is the shortest path from s to t and $\sigma(s,t|v)$ is the shortest path from s to t passing through v. Also, if $s = t$, $\sigma(s,t) = 1$ and if $v = s$ or $v = t$, $\sigma(s,t|v) = 0$.

Note that, the given graph may contain disconnected components so scores for the nodes in each component should be computed only with respect to other nodes in that component: you don't need to compute shortest path between nodes that reside in different components. In other terms, simply ignore the term in summation if any one of s, t or v is disconnected.

Your code should return the number of disconnected components and fill the given array $scores$ with importance scores of nodes in the indices of their ids.

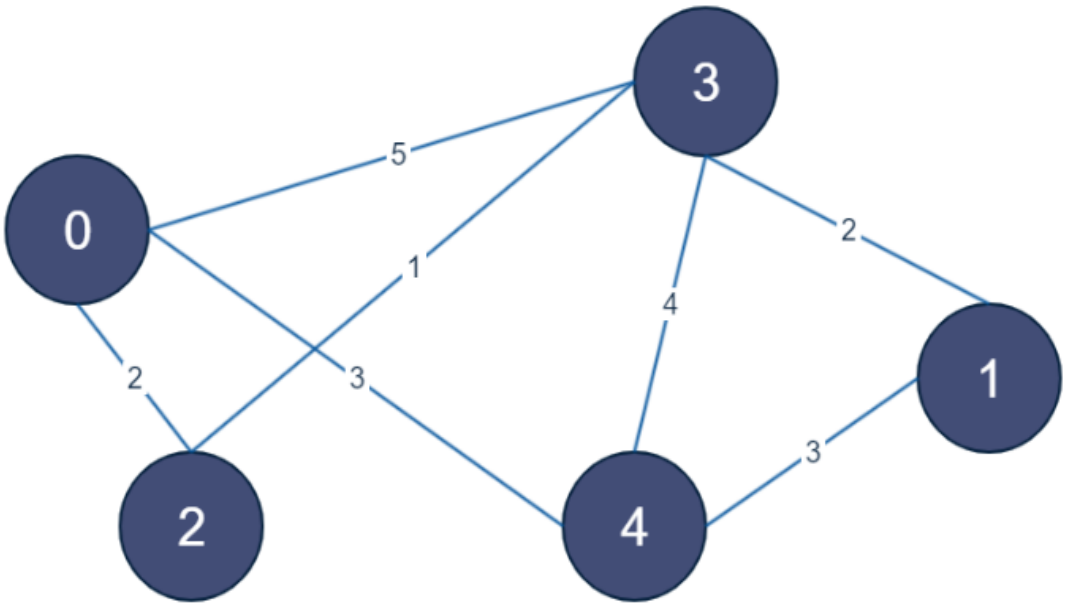
- *Constraints :*

Array sizes are less than 10^3 (and > 0).

Your code should work at most in $O(n^3)$ complexity.

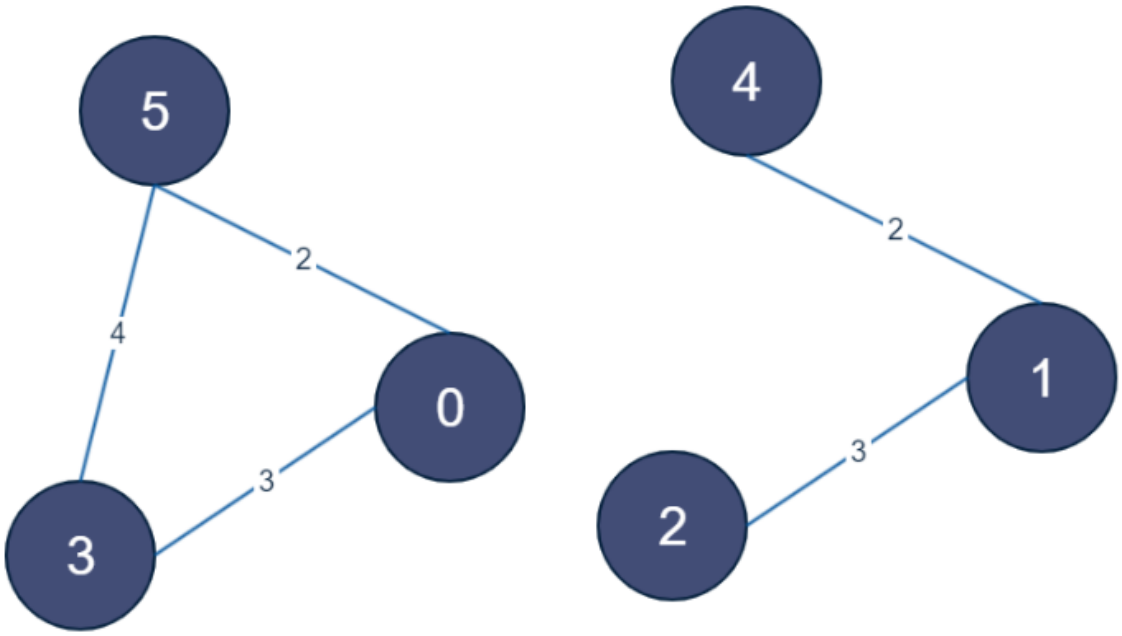
- *Example I/O :*

- ex. 1:




```
n = 5, edgelist =
0 0 2 5 3
0 0 0 2 3
2 0 0 1 0
5 2 1 0 4
3 3 0 4 0
      return 1
```

- ex. 2:




```
n = 6, edgelist =
0 0 0 3 0 2
0 0 3 0 2 0
0 3 0 0 0 0
3 0 0 0 0 4
0 2 0 0 0 0
2 0 0 4 0 0
       return 2
```