Data Structures	Assignment 4
CCO 2103-01, Spring 2024	Due at 5:00pm, Tuesday, June 4

Hand in your solutions electronically using LearnUs. As was announced in class, you are not allowed to use any data structure libraries, including the linked lists, stacks, queues, trees, binary search trees, priority queues, heaps, and graphs provided by JDK. When in doubt, contact the course staff.

This assignment is *not* a completion points assignment.

Submit your source code(s), zipped as `yourStudentID.zip`. For example, if your student ID is `2023000000`, then you must zip all your source code(s) into `2023000000.zip` and submit this file. Each class should have its own `.java` file, of which the filename is the same as the class name. Do *not* include your student ID as part of the class names.

This assignment consists of one programming task.

(1) (100 points) Write a Java method that, given a simple weighted undirected graph and its two vertices, finds the length of a shortest path between the two given vertices.

Your code must have a class named `SP` that has a ***static*** method called `getSPLen()`. Its header is as follows:

```
public static int getSPLen(int n, int m, int[] x, int[] y,
                           int[] c, int s, int t)
```

- The parameter `n` is the number of vertices. The vertices are numbered from 0 to $\texttt{n} - 1$.
- The parameter `m` is the number of edges. The edges are numbered from 0 to $\texttt{m} - 1$.
- The parameters `x` and `y` specify the edges. For each `i` ($0 \leq \texttt{i} \leq \texttt{m} - 1$), `x[i]` and `y[i]` are (the numeric IDs of) the two endpoints of Edge `i`.
- The parameter `c` specifies the edge costs. For each `i` ($0 \leq \texttt{i} \leq \texttt{m} - 1$), `c[i]` is the cost of Edge `i`.
- The parameters `s` and `t` are (the numeric IDs of) the two vertices between which you need to find a shortest path.
- If a shortest path between `s` and `t` exists, your method `getSPLen()` must return the length of a shortest path between `s` and `t`.
- If a shortest path between `s` and `t` does not exist, your method `getSPLen()` must return `-1`.

You can assume that $2 \leq \texttt{n} \leq 2,000,000$, $\texttt{m} \leq 2,000,000$, and $\texttt{s} \neq \texttt{t}$. You can assume that the cost of every edge is ***nonnegative*** and no greater than $1,050$. Your `getSPLen()` method must return within 5 seconds on the TA's computer.

You are free to add your own methods, fields, and classes as you see fit, but you need to implement all of those.

To ease your testing, we will provide a skeleton code. The skeleton code reads the input from `input.txt` in the current working directory, and outputs the answer reported by your code, as is, to `output.txt` in the current working directory.

The first line of `input.txt` contains `n` and `m`, separated by a space. The second line contains `s` and `t` separated by a space. Each of the following `m` lines of the file contains (the numeric IDs of) the two endpoints of each edge and its cost, separated by spaces. The input therefore consists of $m + 2$ lines in total.

The output file consists of a single line, containing the answer reported by your method.

The entry point of the skeleton code is `As4.main()`. To help your testing, the skeleton code will output the total amount of time elapsed to the standard output. This time, of course, does not include the file I/O time. (The skeleton code provided does not terminate your code even if it spends more than 5 seconds.)

Submit all your source files, including `SP.java` and any other source files you created. Note that `As4.java` is the skeleton code and you do not need to submit this file. In other words, you must ***not*** modify this skeleton code. When we grade your code, we will use a different driver to test your solution.

Example 1

`input.txt`

```
3 3
0 1
0 1 4
1 2 1
0 2 2
```

`output.txt`

```
3
```

Example 2

`input.txt`

```
3 0
0 1
```

`output.txt`

```
-1
```