

CSCE 235: Introduction to Discrete Structures
Programming assignment 2 (80 points)
Assigned Tuesday, April 10, 2007
Due 11:59 p.m., Sunday, April 22, 2007

You are to implement Dijkstra’s algorithm for undirected graphs, using good design, documentation, and programming style. See Section 9.6 of the textbook for details about this algorithm.

Description of input

The input to your program will contain zero or more graph descriptions; each graph description will be followed by zero or more test cases.

A graph description, which describes a connected weighted undirected graph, will consist of several integers, separated by spaces, on one line. It will be of the form

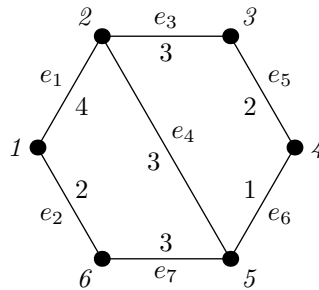
$$N \ V \ E \ u_1 \ v_1 \ w_1 \ u_2 \ v_2 \ w_2 \ \dots \ u_E \ v_E \ w_E,$$

where N is a graph identifier (explained below), V is the number of vertices in the graph, E is the number of edges in the graph, and the following $3E$ integers specify the edges of the graph and their weights (the i th edge of the graph connects vertices u_i and v_i and has weight w_i). The vertices of the graph are labeled with consecutive positive integers from 1 to V .

For example, the graph description

$$1 \ 6 \ 7 \ \underbrace{1 \ 2 \ 4}_{e_1} \ \underbrace{1 \ 6 \ 2}_{e_2} \ \underbrace{2 \ 3 \ 3}_{e_3} \ \underbrace{2 \ 5 \ 3}_{e_4} \ \underbrace{3 \ 4 \ 2}_{e_5} \ \underbrace{4 \ 5 \ 1}_{e_6} \ \underbrace{5 \ 6 \ 3}_{e_7}$$

describes the graph below, where the italic numbers are labels for the vertices and the roman (upright) numbers are weights of edges.



The first number in a graph description is a graph identifier, which is simply an integer between 1 and 1000000 (one million), inclusive. The value of this number is not important, except that you must include it in your output (see the next section). In particular, the graph identifiers do not need to come in any order, and they may be repeated in an input file; in fact they may be entirely random. The end of the input will be specified by a single zero where otherwise there would be a graph identifier.

Your program should be able to handle all valid graph descriptions where $1 \leq V \leq 32$, $0 \leq E \leq 2000$, and $0 \leq w_i \leq 5000$ for all i . Be aware that the input describes *undirected* graphs, so an edge may be specified by giving its endpoints in either order. Also be aware that a graph description may describe a graph that is not simple; you must figure out how to handle the possibility of nonsimple graphs. However, you are guaranteed that the graphs given as input will be connected.

Following each graph description will be one line which will contain zero or more test cases. This line will be of the form

$$n \ a_1 \ b_1 \ a_2 \ b_2 \ \dots \ a_n \ b_n,$$

where n is the number of test cases and the following $2n$ integers specify pairs of vertices. For each pair (a_i, b_i) , your program should find the length of a shortest path between the vertices a_i and b_i in the graph.

Thus, the input consists of a description of a graph, followed by several “distances” for your program to compute. Then comes a description of another graph, followed by another list of distances to compute, and so on. The input ends when there is a zero in a position that would normally be a graph identifier.

The input will be available to your program through *standard input*. This means that you can use `cin` to read the input in C++, `scanf` in C, `System.in` in Java, and the corresponding functions in other programming languages. Please ask me if you have questions about the mechanics of reading the input.

Description of output

The output of your program should be presented one graph description per line. Each line should begin with the graph identifier; this is to make it easier for me to match up the lines of your output with the lines of the input. For each of the test cases for this graph description, your program should output one space (to separate the output of this test case from that of previous test cases or the graph identifier), followed by the length of a shortest path in the specified graph between the two vertices given in the test case. After all of the test cases have been completed for the graph description, your program should output a single newline. The lines of your output should not contain leading or trailing spaces.

The output of your program should be sent to *standard output*. This means that you should use `cout` to produce the output in C++, `printf` in C, `System.out` in Java, and the corresponding functions in other programming languages.

Do not output any prompts, debugging information, or anything else other than the results I ask for above. When your program is graded, a large input file will be piped to your program, and the output your program produces for this input file will be saved to an output file. This output file will be automatically compared with a file containing the correct answers. If your output contains prompts or other extraneous information, this comparison step will indicate that your program produced incorrect output.

What to hand in

Hand in all source files and a README file that explains how to compile and run your program. It is not necessary to hand in a binary executable. Use the Web handin system at

<http://cse.unl.edu/~cse235/handin/>

to hand in your files. Instructions for using the Web handin system were included on the back of the handout for the first programming assignment; if you've lost this handout, you can get a copy from the course Web page.

Sample input

```
1 6 7 1 2 4 1 6 2 2 3 3 2 5 3 3 4 2 4 5 1 5 6 3
5 1 4 1 2 3 6 4 2 3 4
77777 6 9 1 2 2 3 1 8 2 3 6 4 6 6 4 5 5 6 5 8 1 6 14 2 5 5 3 4 3
12 4 5 6 2 4 4 2 3 1 6 6 1 1 4 2 5 1 5 2 4 3 4 1 2
235 8 7 1 2 1 2 3 6 3 4 15 4 5 20 5 6 15 6 7 6 7 8 1
4 1 8 3 6 7 5 2 3
1138 3 5 1 2 791 2 1 603 2 3 240 2 2 82 1 3 926
5 1 2 1 3 2 3 2 2 3 1
0
```

In this example, the first line is the graph description used as an example earlier. The second line asks for five distances in this graph: the length of a shortest path between vertices 1 and 4, the length of a shortest path between vertices 1 and 2, and so on. The third line is a description of a different graph (with an arbitrary graph identifier 77777), and the fourth line asks for 12 distances in this graph. This pattern continues; the end of the input is specified by a single zero in place of a graph description.

Sample output

```
1 6 4 6 4 2
77777 5 13 0 6 14 14 11 5 7 9 3 2
235 64 50 21 6
1138 603 843 240 0 843
```

The first line of this output begins with the graph identifier 1, from the first line of the input. Then come the five distances requested in the second line of the input. For example, the length of a shortest path between vertices 1 and 4 is 6 (this path begins at vertex 1, passes through vertices 6 and 5, and ends at vertex 4).

Grading

Your grade for this programming assignment will consider several aspects of your program, including but not limited to correctness (this is important), your choice of graph representation (see Section 9.3 of the textbook for some ideas), adherence to these specifications, and program design and documentation.