Programming Design, Spring 2016
Homework 7

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Please upload one PDF file for Problem 1 and two CPP files for Problems 2 and 3 to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***2:00 am, April 11, 2016***. Please answer in either English or Chinese.

The TA who generates the testing data and grades this homework is ***Parker Chiang***.

Problem 1

(20 points; 4 points each) In this problem, we will introduce the concept of *graphs* in the field of Computer Science to you. Intuitively, a graph is an abstract representation of a realistic map. In a graph, there are *nodes* connected by *edges*. For example, in the graph presented in Figure 1, there are six nodes and seven edges.

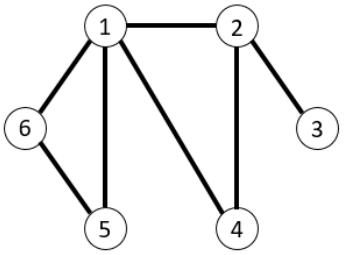

Figure 1: A graph

We say that two nodes are *adjacent* if there is an edge (i, j) between nodes i and j. In Figure 1, node 1 is adjacent to nodes 2, 4, 5, and 6, but nodes 2 and 6 are not adjacent. A *path* from node i to node j is an ordered collection of edges (i, n_1), (n_1, n_2), ..., and (n_k, j). This means that we may go from node i to node j by traveling through nodes i, n_1, n_2, ..., n_k, and finally j. There are $k + 1$ edges on this path. In Figure 1, there is a path $\{(6, 1), (1, 2), (2, 3)\}$ from node 6 to node 3. We often abbreviate the representation of a path as $(i, n_1, n_2, ..., n_k, j)$. For the above path from node 6 to node 3, it can be represented as $(6, 1, 2, 3)$.

Given a graph, one way (among many) to store the node and edge information is to build an *adjacency matrix*. Suppose that we have n nodes, we will build an $n \times n$ matrix A which satisfies

$$A_{ij} = \begin{cases} 1 & \text{if nodes } i \text{ and } j \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases}.$$

If we build an adjacency matrix to represent the map in Figure 1, we will have $A_{12} = 1$, $A_{13} = 0$, $A_{14} = 1$, ..., and $A_{56} = 1$. Note that A must be symmetric. By definition, $A_{ii} = 0$ for all $i \in \{1, 2, ..., n\}$.

(a) Consider the graph in Figure 1. Construct its adjacency matrix A as an 6×6 matrix.

(b) Given A, we may multiply it by itself to obtain A^2. Let's define

$$A^{(2)}_{ij} = \begin{cases} 1 & \text{if } A^2_{ij} \geq 1 \\ 0 & \text{otherwise} \end{cases}.$$

For the A you find in Part (a), find $A^{(2)}$.

(c) If you treat the $A^{(2)}$ matrix you find in Part (b) as another adjacency matrix, it should represent a particular graph. Draw the graph and then intuitively explain what $A^{(2)}$ tells us regarding the original graph in Figure 1.

(d) Given an adjacency matrix A, propose a method to identify the number of nodes that can be reached from a given node i by traveling through *no more than* two edges.

(e) Given an adjacency matrix A, propose a method to identify the number of nodes that can be reached from a given node i by traveling through *no more than* k edges, where $k > 0$ is a given integer. Use pseudocodes to present your method.

Problem 2

(40 points) The IEDO city's subway system has four lines and 53 stations. Figure 2 is the map of the subway system. The green line, from station 1 to station 12, is basically a north-south one. The blue line, starting from station 13 to station 26, is basically an east-west one. The yellow line, from station 27 to station 50 (and then connecting back to station 27), is a circular one. Finally, the orange one, from station 32 to station 53, serves citizens at the north-east corner. Stations 4, 6, 9, 14, 22, and 32, which are colored as red, are transfer stations. Passengers need to transfer to another line at one of the six stations. Note that all passengers traveling from stations 51, 52, and 53 to anywhere excluding station 32 must transfer at station 32.

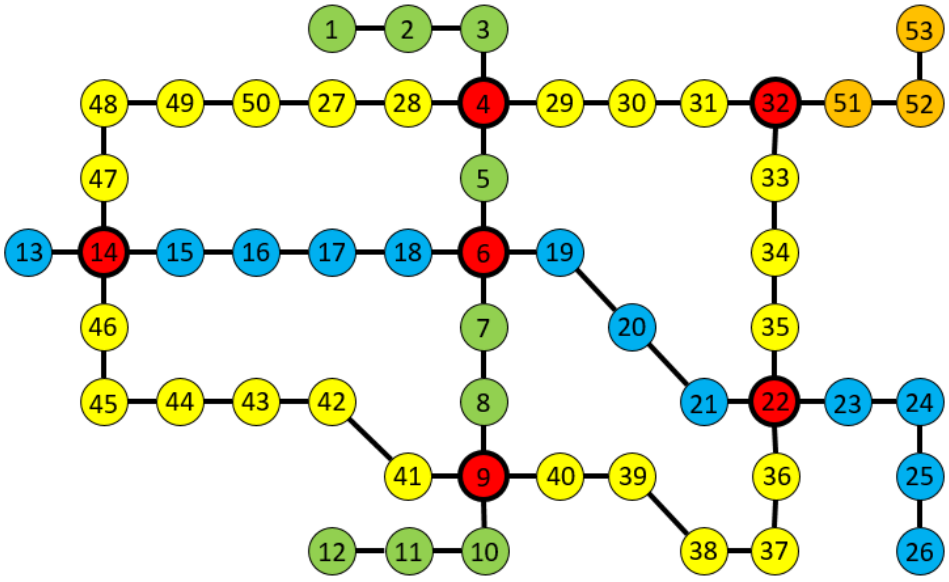

Figure 2: A subway map

You will be given a station ID and an integer k. Your program needs to find all stations that may be reached by starting from the given station in at most k steps. In other words, the number of stations traveled, including the starting and ending stations, can be at most $k + 1$. Your program also needs to find out the number of stations to travel for each of these reachable stations.

Input/output formats

There are 15 input files. In each file, there are two integers i and k, separated by a white space. It is known that $i \in \{1, 2, 3, ..., 53\}$ and $1 \leq k \leq 53$. Given this input, your program should list the IDs of the stations that is reachable from station i in no more than k steps (it is said that station i can be reached

from station i in 0 step). Station IDs should be listed in the ascending order. Each two consecutive integers should be separated by a white space.

For example, for the input

```
6 2
```

the output should be

```
4 5 6 7 8 17 18 19 20
```

If the input is

```
51 3
```

the output should be

```
30 31 32 33 34 51 52 53
```

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use only techniques covered so far. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 10 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

Problem 3

(40 points) Continue from Problem 2. Given a starting station i and a destination station j, we now want to find a best route from i to j. The quality of a route is determined by two things: the number of stations that need to be traveled (including the starting and destination stations) and the number of transfers. When one gives you a pair of stations i and j, she/he will also indicate her/his preference $p \in \{0, 1\}$, where $p = 0$ means minimizing the number of transfers and $p = 1$ means minimizing the number of stations traveled. Your program should find the best route and then print out the number of transfers and number of stations traveled.

Hint. There are multiple ways to solve this problem. In particular, recursion may be a good one.

Input/output formats

There are 15 input files. In each file, there are three integers i, j, and p. Two consecutive integers are separated by a white space. It is known that $i \in \{1, 2, ..., 53\}$, $j \in \{1, 2, ..., 53\}$, and $p \in \{0, 1\}$. Given this information, your program should find the best route from station i to station j given the preference indicated by p. For example, the following input

```
9 53 0
```

says that one wants to find the least-transfer route from station 9 to station 53. The output should be

```
1 14
```

where the best route $(9, 40, 39, 38, 37, 36, 22, 35, 34, 33, 32, 51, 52, 53)$ needs one transfer at station 32. On the contrary, the input

```
9 53 1
```

looks for the least-station route from station 9 to station 53. The output should be

```
2 13
```

where the best route $(9, 8, 7, 6, 5, 4, 29, 30, 31, 32, 51, 52, 53)$ needs two transfers at stations 4 and 32.

When $p = 0$, if there are multiple routes whose numbers of transfers are all the smallest, choose the route with the fewest stations traveled. When $p = 1$, if there are multiple routes whose numbers of stations traveled are all the smallest, choose the route with the fewest transfers.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use any technique. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 10 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.