CS2010 PS8 - Bundle of Joy

Released: Thursday, 27 October 2011
Due: Tuesday, 08 November 2011, 8am

Collaboration Policy. You are encouraged to work with other students on solving this problem set. However, you **must** write up your solution **by yourself**. In addition, when you write up your solution, you **must** list the names of every collaborator, that is, every other person that you talked to about the problem (even if you only discussed it briefly). Any deviation from this policy will be considered cheating, and will be punished severely, including referral to the NUS Board of Discipline. It is not worth it to cheat just to get 15% when you will lose out in the other 85%.

The Story. Steven runs out of original idea for PSes. Since he is busy with his *'Bundle of Joy'* (a term for 'a new baby'), he cannot spent too much time to come up with new PS (such new questions will go to Quiz 2 or Final Exam :O)...

Moreover, he remembers that his students will soon have CS2010 final exam. So, he decides to use the last CS2010 PS of this semester to give his students a 'Bundle of Joy' too: One of the question from past CS2020 final exam :O.

The Actual Problem. ***k*-Reliable Shortest Path**
When transmitting data in a computer network, it is generally more reliable to use a path with a *smaller number of hops* to avoid packet (data) loss. That is, we want each packet to visit a small number of intermediate routers.

Assume you are given a network of n routers as an undirected weighted graph. Each edge in the graph has a positive weight representing the time it takes for a message to travel across the link (i.e., the latency of that link). For a given value of k, the goal is to find the minimum weight path between two nodes that contains at most k intermediate vertices. We call such a path the *k-reliable shortest path*. We denote the weight of the k-reliable shortest path between s and t as: $RP(s, t, k)$. If there is no k-reliable path between s and t, then we define $RP(s, t, k) = \infty$ (in practice, you have to output 1000000000, or 1 Billion).

The following is a simple example of the k-reliable shortest path problem. Assume you are given the following network where $n = 3$:

- $RP(0, 1, 0) = 2$: That is, the 0-*reliable shortest path* from source router 0 to target router 1 has weight 2, via the direct edge $0 \rightarrow 1$.
- $RP(0, 1, 1) = 2$: The minimum weight path doesn't change although k is 1 larger in this case.
- $RP(0, 2, 0) = 1000000000$: there is no path from 0 to 2 that has zero intermediate vertices.
- $RP(0, 2, 1) = 5$: the path $0 \rightarrow 1 \rightarrow 2$ uses one intermediate vertex has weight 5.
- $RP(1, 2, 0) = 3$: the edge $1 \rightarrow 2$ has weight 3.
- $RP(1, 2, 1) = 3$: the minimum weight path is unchanged, even though k is increased by 1.

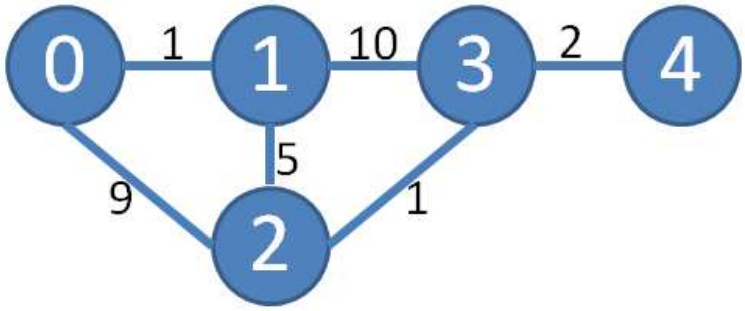
The original goal of this question (in CS2020 final exam) is to design an algorithm that calculates $RP(s, t, k)$ as efficiently as possible for any value of s, t, and k. Your solution will consists of two parts: (i) a pre-processing algorithm, that takes a graph and creates a data structure to store the graph; and (ii) a query algorithm that given nodes s and t, and parameter k, calculates $RP(s, t, k)$. Since there will be a very large number of queries, we want the query time to be *as small as possible*. In this PS, you will give Steven a working code that implements that algorithm (since you have much more time than 2 hours in CS2020 final exam).

Questions for Self Reflection

You do not have to answer these questions as comments in your Java code. But attempting these 'guiding questions' will be helpful for you to arrive at the solution.

If $RP(s, t, k) = v$, is it always the case that $RP(s, t, k + 1) \leq v$?

Now consider the following network:



- What is the value of $RP(0, 4, 0)$?
- What is the value of $RP(0, 4, 1)$?
- What is the value of $RP(0, 4, 2)$?
- What is the value of $RP(0, 4, 3)$?
- What is the value of $RP(1, 4, 2)$?
- What is the value of $RP(2, 4, 1)$?
- What is the value of $RP(3, 4, 0)$?

Java Skeleton (You have to complete and submit this).

Suppose the network has been stored in an `AdjMatrix` as shown in lecture. The weight of edge (u, v) is stored in `AdjMatrix[u][v]`. As the graph is undirected, `AdjMatrix[u][v] = AdjMatrix[v][u]`. If there is no edge `(u, v)`, then `AdjMatrix[u][v] = AdjMatrix[v][u] = 1000000000`.

Below is the skeleton Java code. Complete it, implementing your algorithm for solving the general case of the k-reliable shortest path problem. (Do not modify the main method; it is given only for reference). Submit this file to our game system. Your TA will check it during latter part of Week13/early part of study week. We really cannot afford late submission this time.

```
import java.util.*;

// write your matric number here:
// write your name here:
// write list of collaborators here:

//-----------------------------------------------------------------
// If you need additional Java API, import them here:




//-----------------------------------------------------------------
public class Joy {
  private static int INF = Integer.MAX_VALUE;
  private static int N, M;
  private static int[][] AdjMatrix = new int[80][80];
  //-----------------------------------------------------------------
  // If you need global variables/additional functions, use this space



  //-----------------------------------------------------------------
  // Implement your pre-processing code here.
  // If you do not need this function, simply leave it blank
  //-----------------------------------------------------------------
  public static void preprocessing() {


  }

  //-----------------------------------------------------------------
  // Implement your query code here:
  //-----------------------------------------------------------------
  public static int RP(int S, int T, int K) {


     return 0; // replace this dummy value with your answer
  }

  //-----------------------------------------------------------------
  // main function: for your reference only
  // Do not modify.
  //-----------------------------------------------------------------
 public static void main(String[] args) {

/*
// Sample graph for this problem
5 6
0 1 1
0 2 9
1 2 5
1 3 10
```

```
2 3 1
3 4 2
*/

    Scanner sc = new Scanner(System.in);

    N = sc.nextInt();
    M = sc.nextInt();

    AdjMatrix = new int[N][N];
    for (int i = 0; i < N; i++)
        Arrays.fill(AdjMatrix[i], INF);
    while (M-- > 0) {
        int A = sc.nextInt();
        int B = sc.nextInt();
        int T = sc.nextInt();
        AdjMatrix[A][B] = AdjMatrix[B][A] = T;
    }

    preprocessing();

    // Try attempting Subtask 1 first :)
    for (int S = 0; S < N; S++)
        for (int T = 0; T < N; T++)
            System.out.println(RP(S, T, 0));

/*
    // Subtask 2
    for (int S = 0; S < N; S++)
        for (int T = 0; T < N; T++)
            System.out.println(RP(S, T, N - 2));

    // Subtask 3
    for (int S = 0; S < N; S++) // issue all possible RP(S, T, K) queries
        for (int T = 0; T < N; T++)
            for (int K = 0; K <= N - 2; K++)
                System.out.println(RP(S, T, K));
*/
  } // end main
} // end class
```

Subtask 1 (20 points). $n \leq 64$, but $k = 0$ for every query. That is, a query returns a 0-reliable shortest path.

Subtask 2 (20 points). $n \leq 64$, but $k = n - 2$ for every query.

Subtask 3 (60 points). $n \leq 64$, solve the full problem!

Note: You are only given `Sample.txt` and `Sample-ans.txt`. You are not given any other official test data. However, you are allowed to check your program's output with your friend's as usual.