NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
WRITTEN QUIZ 2 (6%)
AY2017/18 Semester 1

CS2010 – Data Structures and Algorithms II

8 November 2017 Time allowed: 45 minutes

INSTRUCTIONS TO CANDIDATES

1. Do **NOT** open the question paper until you are told to do so.
2. This question paper contains 5 questions. It comprises EIGHT (8) printed pages, including this page.
3. Answer all questions only on this **Question Paper**. You can use either pen or pencil. Just make sure that you write **legibly**!
4. This is an **Open Book Quiz**. You can check the lecture notes, tutorial files, problem set files, CP3 book, or any other books that you think will be useful.
5. **Pace yourself!** Do not spend too much time on one question.

TUTORIAL GROUP

STUDENT NUMBER:

A								

For examiners' use only		
Question	***Max***	***Marks***
Q1-3	9	
Q4	16	
Q5	15	
Total	**40**	

Section A – Analysis (9 Marks)

Prove (the statement is correct) or disprove (the statement is wrong) the following statements below. If you want to prove it, provide the proof or at least a convincing argument. If you want to disprove it, provide at least one counter example. 3 marks per each statement below (1 mark for saying correct/wrong, 2 marks for explanation):

1. We can build a connected, directed and positively weighted graph of 4 vertices and 6 edges such that performing bellman ford will result in >= 19 relaxations.

   False. For positively weighted graphs, Bellman ford will run for at most V-1 passes. Thus maximum relaxation will be 3*6 = 18.

2. Given an undirected and connected graph, if the weight of the shortest path from vertex s to vertex x, i.e $s\sim>x$, is smaller than the weight of the shortest path from s to every other vertex, and the weight of the shortest path from vertex t to vertex x, i.e $t\sim> x$ is smaller than the weight of the shortest path from t to every other vertex, then the shortest path from s to t must be $s\sim> x\sim> t$ in **all cases**.

   False. There can be cases where the shortest path from s to t is simply the edge (s,t).

3. There are no graphs where the SSSP spanning tree is also the same as the MST (in terms of the edges used).

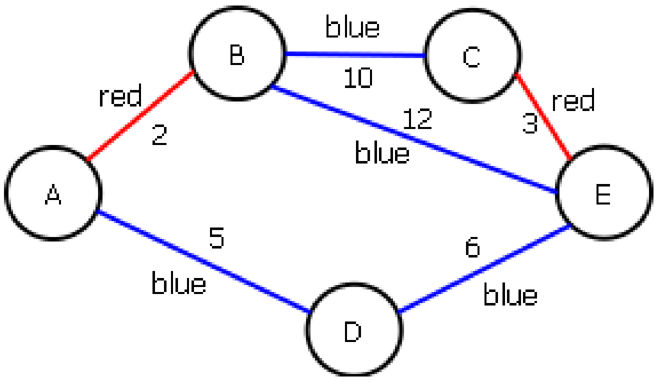
   False. If the graph is a tree then the SSSP spanning tree (from any source vertex) will be the same as the MST.

For Section B, Partial marks will be awarded for correct answers which do not meet the time complexity required (or if the required time complexity of not given, then any correct answer that is less efficient than the lecturer's answer) . You can write in pseudo-code. Provide enough details to all user defined DSes/algorithms/modifications to taught DSes and algorithms so as to show understanding of the solution, but also ***try be as concise as possible, writing more will take more time and possibly make more mistakes!***

Section B – Application (31 Marks)

4. **Colored paths [16 marks]**

City Z is a very unique city. Its roads are painted either red or blue. Junctions and roads of Z can be represented as an undirected weighted graph, where in addition to the weight (positive value), an edge which represents a road also has an extra parameter color which indicates what color the road is. An example is shown below



a) If a person wants to get from some source vertex X to some destination vertex Y, and only wants to take a path where the roads are only of a certain color K, give the best algorithm you can think of to find the shortest path from X to Y given the above restriction. If there is no such shortest path return infinity. Analyze the time complexity of your algorithm in terms of V and E the number of vertices and edges in the graph respectively. **[8 marks]**

Simply perform Dijkstra's (original or modified) from X as source. Then when processing outgoing edges only choose those with color K. Time complexity is O((V+E)*logV).

O(VE) solution: Use Bellman Ford and relax only edges of color K. -> 5 marks

Wrong answers -> 1 mark

b) Now another person wants to get from some source vertex X to some destination vertex Y, but she only wants to travel a path where the edges **alternate** in color starting from a red edge (i.e red,blue,red,blue,…). Now give the best algorithm you can think of to find the shortest path from X to Y given the above restriction. If there is no such shortest path return infinity. Analyze the time complexity of your algorithm in terms of V and E the number of vertices and edges in the graph respectively. **[8 marks]**

Use modified Dijkstra with the following modification. When enqueuing a vertex v into the PQ, other than the shortest path estimate d, add a new parameter c, where c describes the color of the outgoing edges that must be used when v is dequeued. So the distance array is now a distance matrix of size N*2, where the second dimension is the color parameter. Thus at the start we will enqueue (X,0,Red) for the source vertex X. When X is dequeued, only process all outgoing edges (X,X') which are red (if there are any). Now enqueue each relaxed vertex but with the opposite color (i.e blue). Keep alternating the colors until the algorithm ends. The final cost to get from A to B will now be min(D[Y][red],D[Y][blue]). If both is –ve infinity then there is no alternating path from A to B.

Time complexity is still O((V+E)*logV).
You can imagine this algorithm is similar to transforming the original graph to another graph where each vertex in the original is attached the additional color parameter. Thus there are 2 copies of the original vertex in the transformed graph, one for red and another for blue (except the start vertex which must be red).
A vertex (X,C) will be linked to another vertex (Y,C') if there is an outgoing edge from X to Y of color C in the original graph and C' is the opposite color of C. Thus this new graph still has O(V) vertices and O(E) edges, and you basically run Dijkstra without changes on this transformed graph to find shortest path from A to B.

Another O((V+E)*logV) solution: Explicitly building the transformed graph (which is directed but not acyclic) and apply standard Dijkstra -> full marks

O(VE) solution: Build the transformed graph and run bellman ford → 5 marks

Wrong answer -> just say alternate between looking at red and blue outgoing edges when dequeuing from PQ. Color is not a global variable it must be a parameter associated with each vertex. → 3 marks

Other wrong answers -> 1 mark.

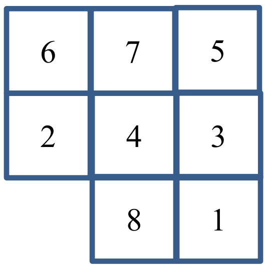
5. **Portal [15 marks]**

Dhell wakes up to find himself trapped in a strange building which consists of only 1 floor. He suddenly realizes that he has knowledge of the layout of the building. The building consists of M rooms where $3 \leq M \leq 100{,}000$. Adjacent rooms are connected to each other by a door, through which Dhell can move between the two room. The time to move from a room r to an adjacent room r' and vice versa is the time for the automatic door connecting r and r' to open (the door will immediately close once Dhell has moved through it). There are N doors in all, where $M - 1 \leq N \leq 1{,}000{,}000$. There is also always some way to get from one room to another.

Dhell is stranded in room A and he needs to get to the lobby B (also a room) which contains a portal to exit the building. However with his mysterious knowledge of the layout of the building, he knows there is a sequence of rooms $R = \{A, r_1, \ldots r_k, B\}$, where $1 \leq k \leq M - 2$, which he needs to visit before the portal will open in lobby B, and he can escape.

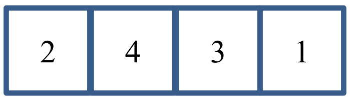
Dhell can go through other rooms as many times as he wants as long as the rooms in R are visted **exactly once** and **in the order they are given**.

In example 1 below, if $R = \{2,3,4,1\}$ Dhell (who will start at 2) can escape by going through the rooms **2**,6,7,5,**3**,**4**,8,**1** in order to escape. However, **2**,**4**,**3**,**4**,8,**1** is not a valid escape path since 4 is first visited before 3 and then it is visited again after 3. Note this is just an example, the rooms might not be square!



Example 1

In example 2 below, if $R = \{2,3,4,1\}$ again, there is no way for Dhell to escape since he has to reach room 4 before room 3.



Example 2

Assume Dhell has knowledge of all the rooms in the building, which rooms are adjacent to each other and the time taken for each of the doors to open. Assume he also has knowledge of the sequence of designated rooms R that must be visited before he can escape.

a) Now model this as a graph problem and give the best algorithm you can think of to compute the **shortest time** for Dhell to escape the building. If there is no way to escape the building, return -1. **[8 marks]**

Vertices are rooms and doors are undirected edges connecting adjacent rooms. Edge weight is the time taken for the door represented by the edge to open.

Let A = r_0 and B = r_{k+1}.

Since Dhell has to go through the rooms in R in their order given, shortest path to escape is SP(A,r_1)+SP(r_1,r_2)+…+SP(r_k,B).

However in the shortest path from r_0 to r_1 we cannot go through any of the rooms {r_2,r_3,.., r_{k+1}}. Similarly in the shortest path from r1 to r2 we cannot go through any of the rooms {r_0,r_3,.., r_{k+1}}, and so on.

Algorithm:

1. Hash all rooms in the sequence r_0 to r_{k+1} into a hashmap H.
2. Keep a shortest path cost C initialized to infinity.
3. Now run Dijkstra for i from 0 to K times:
   - At iteration i, remove r_i and r_{i+1} from H and run from r_i as source vertex. Whenever we relax an edge (u,v) check whether v is in H. If it is do not relax it (v is not allowed along the path). Otherwise relax it.
   - After Dijkstra's finish, add D[r_{i+1}] to C.
   - Before going to next iteration add r_i and r_{i+1} back into the hashmap.

C will contain the shortest path cost to get Dhell from A to B such that the portal will open.

Time complexity is O(K*(V+E)*logV) since we run Dijkstra's K+1 times.

O(KVE)/O(KV^3) solution ->

Run Bellman ford/Floyd Warshall K times each time finding the shortest path from

r_i to r_{i+1} but switching off the other vertices in the sequence. -> 5 marks

Wrong solution -> Almost same as model answer but forget to switch off other vertices in the sequence when finding SP from r_i to r_{i+1}. -> 3 marks

Other wrong solutions -> 1 mark

After escaping the building (so there is way to escape ...), Dhell has put this strange experience behind him. However, one day he suddenly woke up to find himself back in a room of the building! He once again mysteriously has knowledge of the layout of the building, and he realizes the rules to escape has changed.

The layout is still the same as before, but this time, in order to get from his starting room A to the lobby B, he only has to go through one particular room C. Dhell can move through other rooms as many times as he wants but can only move through rooms A,C,B once and in that order. In addition, the longest time taken by any other door he opens while trying to move from A to B must be minimized.

b) Using the same graph modeling as for a) give the best algorithm you can think of to compute **a path** he needs to take to get from A to B such that the portal will open. You may assume there is at least one such path. **[7 marks]**

Now instead of shortest path, it is the path that minimizes the longest time taken by a door to open. This is basically a minimax (minimize maximum edge along path) problem.

1. Similar to a) remove B from consideration, and compute MST G' of graph (Kruskal's/Prim's). → O(ElogV)
2. In G' we can perform DFS from A to get the path A~>C which minimizes the largest edge but does not include B. → O(V)
3. Now add back B and remove A from consideration, compute MST G''. →O(ElogV)
4. This time performing DFS from B we have a path B~>C which minimizes the largest edge and does not include A. →O(V)
5. Concatenate the two paths (reverse B~>C) and we have A~>C~>B which minimizes the largest edge in the path and pass through A,C,B in that order once. →O(V)

Total time is O(ElogV).

Wrong solution -> 3 marks

Run MST then find path from A to C and path from C to B. This does not ensure that A~>C does not include B and also that C~>B does not include A.

Another wrong solution (O(V^3) solution): → 2 marks

Run Floyd Warshall's minimax variant on the graph + variant 2 to get the minimax path between every pair of vertices. Valid path that goes from A to B through C is simply the path from A to C + the path from C to B given in the predecessor matrix. Again the same problem as above.

Other wrong solutions → 1 mark

== END OF PAPER ==