National University of Singapore

School of Computing

Semester 1 (2015/2016)

CS2010 - Data Structures and Algorithms II

Written Quiz 2 (15%)

Saturday, October 31, 2015, 10.00am-11.30am (90 minutes)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. Written Quiz 2 is conducted at COM1-2-SR1 (for all 178 students).
3. This assessment paper contains THREE (3) sections.
   It comprises TEN (10) printed pages, including this page.
4. This is an **Open Book Assessment**.
5. Answer **ALL** questions within the **boxed space** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
6. Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
   Read all the questions first! Some questions might be easier than they appear.
7. You can use **pseudo-code** in your answer but beware of penalty marks for **ambiguous answer**.
   You can use **standard, non-modified** algorithm discussed in class by just mentioning its name.
8. All the best :).
   After Written Quiz 2, this paper will be collected and graded manually as fast as humanly possible. We will try to return the script to you on Wednesday, 04 November 2015 (Lecture 12). If that is not possible, we will do so by Wednesday, 11 November 2015 (the last Lecture 13).
9. Write your Student Number in the box below:

A	0							

A Analysis (15 marks); Marks = _____

Prove (the statement is correct) or disprove (the statement is wrong) the statements below.

1. We can design an $O(1)$ algorithm to detect if a **connected undirected graph** with V ($V \geq 3$) vertices and E edges contains a simple cycle involving 3 or more vertices.

   Yes. The only connected undirected graph that has no cycle is a tree and a tree has $E = V - 1$ edges. If $E \geq V$, it is definitely not a tree and has at least one cycle. If $E < V - 1$, the graph will not be connected. So we simply check if $E = V - 1$ to say no cycle and say cycle otherwise for all cases where $E \geq V$. This is $O(1)$. Note that the wording of the question is to avoid ambiguity involving bidirectional edge (u, v) as having a simple cycle $u \rightarrow v \rightarrow u$, thus we required 3 vertices or more.

2. This time, the graph is **not necessarily connected**. We can design an $O(V)$ algorithm, independent of E (the number of edges), to detect if an **undirected graph** with V ($V \geq 3$) vertices and E edges contains a simple cycle involving 3 or more vertices.

   Yes. When the graph is not necessarily connected, we can run DFS/BFS (from each unvisited connected component) that terminates with a cycle report as soon as it hits a vertex that has been visited or *it explores more than* $V - 1$ *edges*. If the graph really acyclic, the DFS/BFS will not explore more than $V - 1$ edges overall, so it is $O(V)$. If the graph contains at least one cycle, then at most the combined DFS/BFS call(s) explore V edges, so it is $O(V)$ too.

3. The edge weights of a connected undirected weighted graph G are all different. That means the MST of G may be not unique, i.e. there can be MST_1 of G that is structurally different than MST_2 of G yet both MST_1 and MST_2 have the same minimum weight.

   Impossible. Proof by contradiction. Assume MST_1 and MST_2 are structurally different. Then there must be an edge in MST_1 that is not in MST_2 and vice versa. But because the edge weights in G are all different, one of them must be smaller than the other. That means MST_1 must have different weight than MST_2 and both cannot be MST of G.

4. Suppose that we only run the Bellman Ford's algorithm for one round, i.e. we only relax all E edges once. Such Bellman Ford's variant surely produces wrong answer for *all kind* of graphs.

   Not always. If the graph is a DAG and the E edges $u \rightarrow v$ happens to be processed according to the topological order of u, then this one-pass Bellman Ford's is actually correct.

5. There is only one possible shortest path from source vertex s to a certain destination vertex t in a Directed Acyclic Graph (DAG).

   Not always. The 'Dijkstra's killer case' that is presented to you during Lab Demo 07 shows many (actually exponential) number of possible shortest paths from s to t.

Section A Marks = _____

B Not Yet in VisuAlgo Online Quiz (50 marks)

In this section, Steven design a few questions that *may* become part of VisuAlgo Online Quiz system *in the future*.

B.1 Non Standard DFS Challenge (10 marks)

Suppose that we have a DFS implementation `dfs-avoid-odd` as follows:

```
dfs-avoid-odd(u)
  visited[u] <- 1
  if out-degree(u) is odd, return // additional new line
  for all v adjacent to u // ordered based on increasing vertex number
    if visited[v] = 0
      dfs-avoid-odd(v)
```

Show the sequence of vertex visitation (5 marks) and state which vertices are not reachable (another 5 marks) when we execute `dfs-avoid-odd(0)` and `dfs-avoid-odd(7)` on the graph in Figure 1.

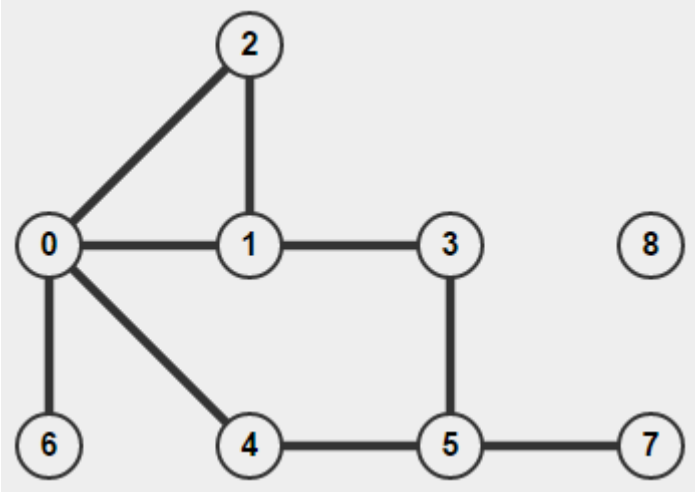

Figure 1: Run `dfs-avoid-odd(0)` and `dfs-avoid-odd(7)` on this graph

`dfs-avoid-odd(0)` will visit: 0, 1, (odd degree, backtrack), 2 (cannot go to 1, backtrack), 4, 5 (odd degree, backtrack), backtrack again, 6 (odd degree, backtrack), done.
Vertex 3, 7, and 8 are not reachable. `dfs-avoid-odd(7)`: 7 and immediately stop, as out-degree(7)

is 1, which is an odd number.
Vertex [0..6] and vertex 8 are all not reachable.

Page 3 Marks = _____

B.2 Gotta Find 'Em All Challenge (10 marks)

The graph in Figure 2 is a connected unweighted graph with $V = 5$ vertices, i.e. all edge weights are ones. Obviously the MST weight of this graph is 4 units, but that is not the question. The question is how many different MST structures exist in Figure 2?

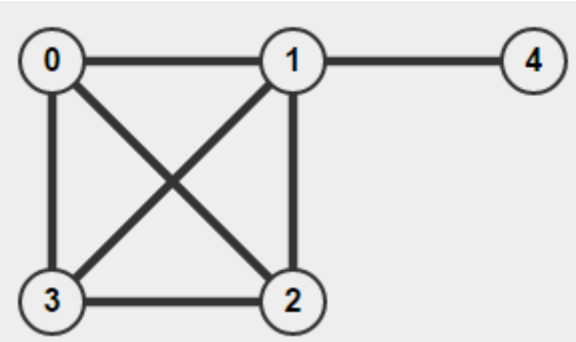

Figure 2: How many different MSTs exist in this picture?

Well, you have 90 minutes to try all this. But if you already know Cayley's formula from Online Quiz 2, this question is just an extension of that. The number of different spanning trees in a complete graph is V^{V-2}. There is a complete subgraph with $V = 4$ vertices involving vertex 0-1-2-3. So there are $4^{4-2} = 4^2 = 16$ spanning (sub)trees in that subgraph. Drawing all those can be very frustrating though but one can utilize rotations and symmetries to help. Now, each of those subtrees must then includes edge 1-4 to be the MST of the input graph. So the answer is still 16.

Alternatively, you can also realize that edge 1-4 has to be inside all MST of the input graph. For the other 6 edges, we will choose 3 of them. So there are ${}_6C_3 = 20$ possible combinations. However, 4 of these 20 are definitely cyclic (a triangle), so we can exclude those 4, i.e. we get the same answer $20 - 4 = 16$.

Page 3+4 Marks = _____

B.3 Virus Infection Challenge (10 marks)

You define vertices that have shortest path weight $-\infty$ from source vertex 0 as 'virus infected'.

Given the directed weighted graph in Figure 3 that **may have negative weight edge(s) and may have negative weight cycle(s)**, please clearly shade all vertices that are infected by virus.

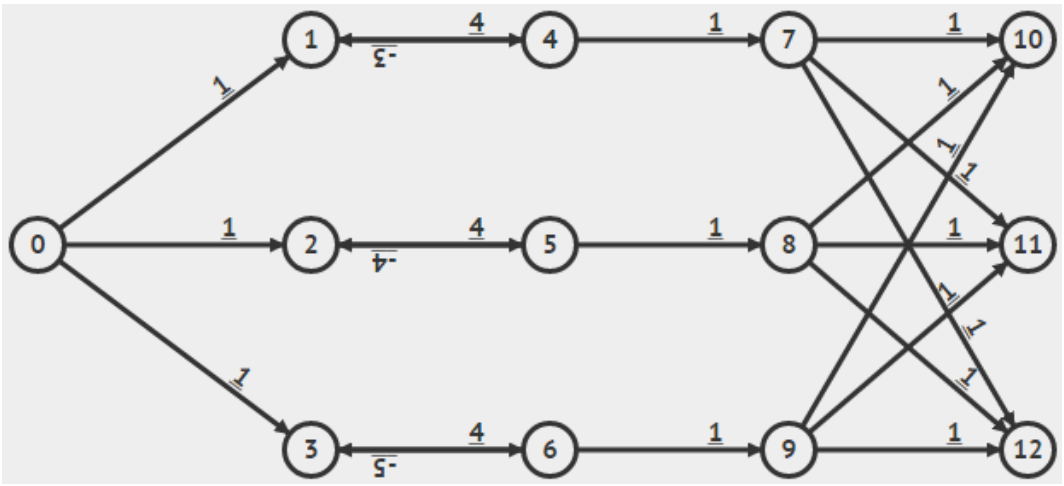

Figure 3: Please clearly shade all vertices that are infected directly in this picture

Cycle 1-4 and Cycle 2-5 are positive/zero weight cycles, respectively. It will not cause a problem. Cycle 3-6 is a negative weight cycle so at least 3-6 are infected. Vertex 9 and then 10-11-12 are also infected. The other vertices are not infected.

B.4 Save Face Challenge (10 marks)

You mistakenly implement BFS algorithm instead of Dijkstra's algorithm to solve an SSSP problem for a graph with $V = 7$ vertices (the vertices are labeled from [0..6]) and $E = 11$ directed weighted edges (with 11 different integer edge weights $\in$ [1..11]). The source vertex is vertex 0. We have placed the vertices on Figure 4. Your job is to draw 11 directed weighted edges on this picture as per instruction so that your BFS algorithm still happen to produce correct answer on the graph that you drawn and thus safe your own face (for now).

Figure 4: Draw your answer directly in this picture

Simply draw 6 edges that goes out from vertex 0 (source) to the other 6 vertices with the smallest 6 weights, i.e. 1 to 6. Then put any other 5 edges with weight 7 to 11 randomly. An example of such drawing is Figure 5. As BFS(0) will immediately update $D[v]$, $\forall v \in [1..6]$ on it's first hop, BFS happens to be correct for such test case. Obviously you need to remember that BFS will generally produce wrong answer for many other directed weighted graphs.

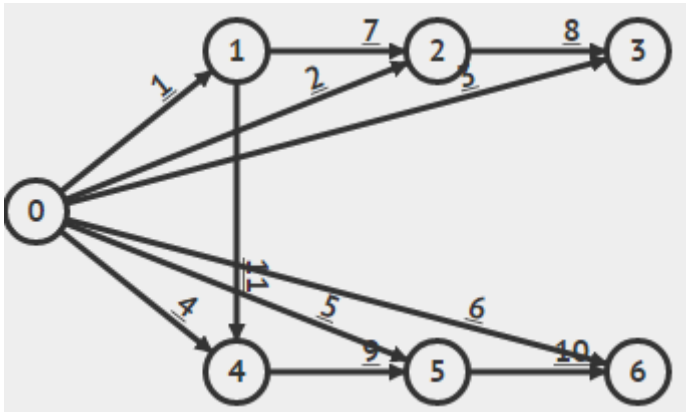

Figure 5: One possible answer

Page 3+4+5+6 Marks = _____

B.5 Save Face Challenge, Extreme? (10 marks)

After solving the previous Save Face Challenge in Question B.4, you think further...

You have learned about DFS, BFS, Prim's, and Dijkstra's algorithm in the second part of CS2010. You were told that all four algorithms requires a source vertex to begin with and all four algorithms terminate with their own spanning trees of the graph.

Instead of just comparing BFS and Dijkstra's as in Question B.4, you also want to include DFS and Prim's (an MST algorithm). You now want to challenge yourself to construct one single undirected weighted (each edge is treated as having bidirectional edges of same weight by Dijkstra's algorithm) graph with $V = 7$ vertices (the number of edges and the choice of edge weights is now entirely up to you) so that `dfs(0)`, `bfs(0)`, `prim(0)`, and `dijkstra(0)` somehow happen to produce exactly the same spanning tree.

Figure 6: Draw your answer directly in this picture

This is a trick question and actually easier than Question B.4. Just draw an unweighted (weight 1) tree at vertex 0 with vertex [1..6] as vertex 0's children. This way, we have $E = 6$ edges. `dfs(0)` = `bfs(0)`. The input graph itself is the MST and the input graph itself is the SSSP spanning tree :).

Section B Marks = _____

C Think Outside the Box (35 marks)

C.1 Graph Traversal (10 marks)

In Written Quiz 1, you were asked to store graphs with characteristics like the example shown in Figure 7 below: Near complete graphs of V vertices ($5 < V < 100\,000$) with only up to k edges missing ($0 \leq k \leq 7$). To further clarify the question, the graphs are unweighted, the vertices are numbered from [0..V-1], and no special information is stored in each vertex other than vertex number.

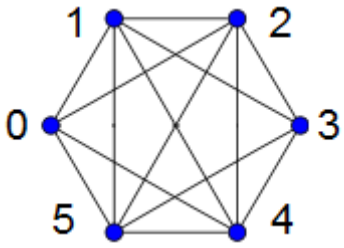

Figure 7: Near Complete Graph with $V = 6$ Vertices (One Edge (0, 3) is Missing)

Now in this Written Quiz 2, you were asked to check if two different vertices i and j are connected in such graph above. What is your best algorithm and analyze it's time complexity?

Notice the constraints... near complete graph with $5 < V < 100\,000$ vertices and up to $0 \leq k \leq 7$ missing edges. If $V \geq 9$, the answer is obviously true. Because in a near complete graph with $V \geq 9$ vertices, we cannot disconnect any vertex i from this graph if we can only 'delete' at most $k \leq 7$ edges. Only when $5 < V \leq 8$ then we may have a case where a vertex i has no edge to the other $V - 1$ vertices due to $k \leq 7$ missing edges.

So for $V = [6..8]$, or according to the constraint of the problem description, we just run a 'modified DFS/BFS' that instead of exploring neighbors of Edge List, the modified DFS/BFS explores neighbors that are NOT listed in the Reversed Edge List to decide if vertex i and j are connected. As we are only dealing with small number of vertices here, it is OK to say that this is an $O(K^2)$ algorithm or even a 'constant factor' algorithm as $K \leq 7$.

Page 8 Marks = _____

C.2 Connected Components (25 marks)

You are given a connected undirected unweighted graph G with V vertices and E undirected edges ($1 \leq V, E \leq 200\,000$). If we delete a vertex, like in PS3, the number of Connected Components (CCs) in G may change. You are given K vertices that you will delete one after another from G. All these K vertices to be deleted are given to you upfront. Now your job is to report the number of CCs in G every time a vertex is deleted from G.

For example, if you are given the connected undirected graph as shown below, deleting $K = 3$ vertices {1, 2, 7} in that order produces 4 CCs ({0}, {4, 5}, {2}, {3, 6, 7, 8}), 3 CCs ({0}, {4, 5}, {3, 6, 7, 8}), 4 CCs again ({0}, {4, 5}, {3, 6}, {8}), respectively.

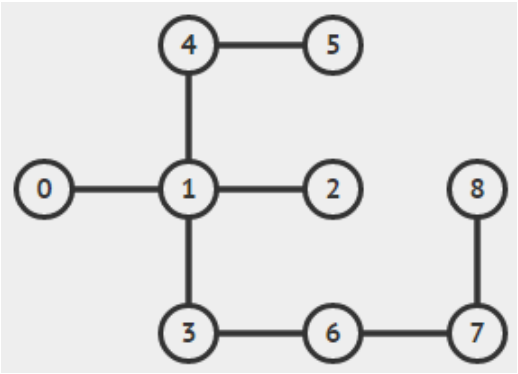

Figure 8: An Example Graph G

C.2.1 $K = 1$ (10 marks)

How to compute the number of CCs in G if there is only $K = 1$ deleted vertex only?
What is the time complexity of such algorithm?

Just run $O(V + E)$ DFS/BFS from each unvisited components, same as outlined in Lecture 06 or used as part of PS3. This is a huge $O(K \times (V + E))$ algorithm but K is just 1 for this part, so saying that this is an $O(V + E)$ algorithm is also correct.

Page 8+9 Marks = _____

C.2.2 $K \leq V$ (15 marks)

How to compute the number of CCs in G if there can be up to $K \leq V$ deleted vertices? Note that $1 \leq V, E \leq 200\,000$ and any computation above $100M$ steps is considered slow. To get full marks, you need to state your algorithm, analyze it's time complexity, and conclude that it runs in $O(V + E)$.

Since you are given the list of K vertices to-be-deleted upfront, we can simply reverse the list (yes, reverse thinking... again). We start from graph G minus these all K vertices and their associated edges, we use UFDS to count number of disjoint sets/connected components. When we add a vertex, we union the end points of all edges involved and store the new number of disjoint sets/CCs. We do so until we have re-added all vertices back. We report the answers in reverse order. This way, we process all V vertices and E edges once, with all UFDS operations treated as $O(1)$. Thus we have an $O(V + E)$ solution.

– End of this Paper, All the Best –

Candidates, please do not touch this table!

Section	Maximum Marks	Your Marks	Comments from Grader
A	15		
B	50		
C	35		
Total	100		