National University of Singapore

School of Computing

Semester 1 (2011/2012)

CS2010 - Data Structures and Algorithms II

Quiz 2 (20%)

Saturday, November 05, 2011, 10.00am-12.00pm (2 hours)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this question paper until you are told to do so.
2. For Quiz 2, we will only use **ONE** large room in NUS: COM1/SR1 for all 106 students.
   To speed up the grading process of Quiz 2, you will be seated based on full name (in ascending order, as in final exam) so that we do not have to do $(106 \log 106)$ steps of sorting.
3. This question paper contains FIVE (5) sections with sub-questions.
   It comprises FOURTEEN (14) printed pages, including this page.
   The weightage is **a bit different** compared to Quiz 1, please **adjust** your strategy accordingly!
4. Write all your answers in this question paper, **but only in the space provided**.
   You can use either pen or pencil. Just make sure that you write **legibly**!
   Important tips: Pace yourself! Do **not** spend too much time on one (hard) question.
5. This is an **Open Book Examination**. You can check the lecture notes, tutorial files, problem set files (**you have to bring printed version of ps3.pdf, ps4.pdf, and ps5.pdf**), Steven's 'Competitive Programming 2' book, or any other books that you think will be useful. But as Steven has said for Quiz 1, the more time that you spend flipping through your files implies that you have less time to actually answering the questions.
6. When this Quiz 2 starts, **please immediately write your Matriculation Number here: _____ (do not forget the last letter and do not write your name).**
7. All the best :).
   After Quiz 2, this question paper will be collected, graded manually in 2 weeks, and will likely returned to you during study week (definitely *before* CS2010 final exam on 30 November 2011).

–This page is intentionally left blank. You can use it as ‘rough paper’–

1 'ADT Table': (Questions → Answers) – version 2 (18 marks)

Most answers for this set of questions can be found verbatim in the lecture notes, tutorial files, or PS files. But unlike Quiz 1, *some* questions now require a bit of thinking process and may have more than two blank spaces. Please fill in your answers on the blank spaces provided, 2 marks per question. Grading scheme: 0 (no correct answer), 1 (at least one answer is correct), 2 (all answers are correct)

1. A). There are V(V-1) edges in a complete simple graph of V vertices. (complete graph ->every vertex has an edge to every other vertex)
   B). A tree that includes all vertices of a graph G is called the spanning tree tree of G.

2. An Adjacency matrix needs $O(V^2)$ **space** for storing a graph with V vertices and E edges whereas an Adjacency list only needs $O(V+E)$ space to store the same graph.

3. The **time** complexity of Breadth First Search (BFS) graph traversal algorithm is $O($V + E$)$ if the graph is stored as an adjacency list .

4. We can test if a pair of vertices u and v are connected (or not) via a path by running kosaraju's algorithm from vertex u and then check if v is reachable from u .

5. A tree T with V vertices has V - 1 edges and there is/are one unique path/paths between any pair of vertices in T (delete whenever appropriate).

6. Topological sort is a linear ordering of the vertices of a directed acyclic graph graph G, where vertex u comes before vertex v in the ordering if there is a directed edge $u \rightarrow v$ in G.

7. We need at least three data structures to implement Prim's algorithm efficiently (in $O(E \log V)$):
   an Adjacency List to quickly go through neighbours of each vertex ,
   a Priority Queue to quickly select the smallest neighbouring edge ,
   and a Boolean array of size V to quickly keep track if vertex has been visited or not.

8. The SSSP problem is not appropriate if the graph has a negative weight cycle because it will lead to an infinite loop .

9. The shortest path on a graph without negative weight cycle is a minimum path and this implies that the shortest path between the source s and any other vertex v of a graph with V vertices and E edges has *at most* V - 1 edges.

2 Graph DS/Traversal/MST/SSSP (29 marks)

General grading scheme (some questions have additional/different requirement): 0 (blank), 1 (the answer is totally wrong), 3 (the answer has noticeable mistakes but not totally wrong), 4 (the answer contain very minor mistake), 5 (the answer is correct).

Q1. Breadth First Search (BFS) (5+1 = 6 marks)

Run Breadth First Search (BFS) graph traversal algorithm as shown in Lecture 5 on the graph in Figure 1 (note that some – but not all – edges are bi-directional). **Start** BFS **only** from vertex 0. If there are two or more valid next vertices, please always select vertex with the *lowest vertex number*. If there is one (or more) vertex (vertices) that are unreachable from vertex 0, highlight it (them).

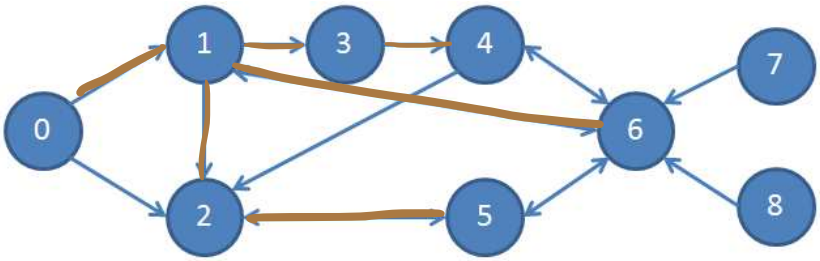


Figure 1: For Section 2 Question 1

Draw the resulting BFS spanning tree below (5 marks):

List down the vertex (vertices) that is (are) unreachable from vertex 0 (if any): 7, 8 – (1 mark).

Q2. MST (Prim's or Kruskal's) (5+1 = 6 marks)

Run *either* Prim's or Kruskal's algorithm on the weighted undirected graph in Figure 2, but **stop the execution** of your chosen MST algorithm after it has taken **the first six (6) edges**. You can choose to draw your answer using *both* algorithms but your marks is capped to **max**(score for Prim's output, score for Kruskal's output). **Therefore to save time, please just draw one.**

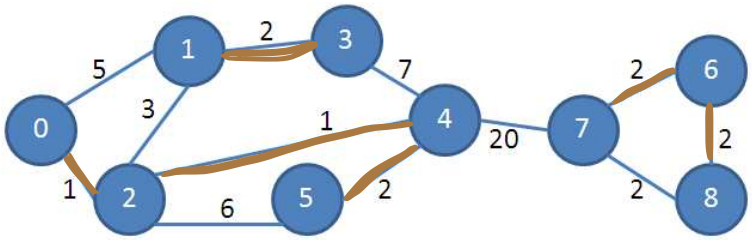


Figure 2: For Section 2 Question 2

If you choose Prim's algorithm, draw the resulting (partial) MST after Prim's algorithm takes the first six edges below (5 marks): (Tie breakers: Order the edge information pair (w, v) based on lower edge weight w first, and if ties, by lower vertex number v).

The (partial) MST weight **so far** (total weight of the first six edges) is: _____ – (1 mark).

If you choose Kruskal's algorithm, draw the resulting Spanning Forests (multiple subtrees/components) after Kruskal's algorithm takes the first six edges below (5 marks): (Tie breakers: Order the edge information triple $(w, (u, v))$ based on lower edge weight w first, and if ties, based on lower vertex number pair (u then v). A pair of vertices (u, v) that defines an undirected edge will always ordered such that $u < v$. Some examples: (1, (0, 2)) < (1, (2, 4)), (2, (6, 7)) < (2, (6, 8)) < (2, (7, 8)), etc).

The Spanning Forests' weight **so far** (total weight of the first six edges) is: 10 – (1 mark).

Q3. SSSP (Bellman Ford's or Original/Modified Dijkstra's) (5 marks)

Run *either* Bellman Ford's, Original/Modified Dijkstra's algorithm on the weighted directed graph in Figure 3. The source vertex is vertex `s = 7` with `D[s] = D[7] = 0` (see the value inside a box next to vertex 7). Note that because all directed edge weights are different and positive, the final answer is unique regardless of which shortest paths algorithm that you use.

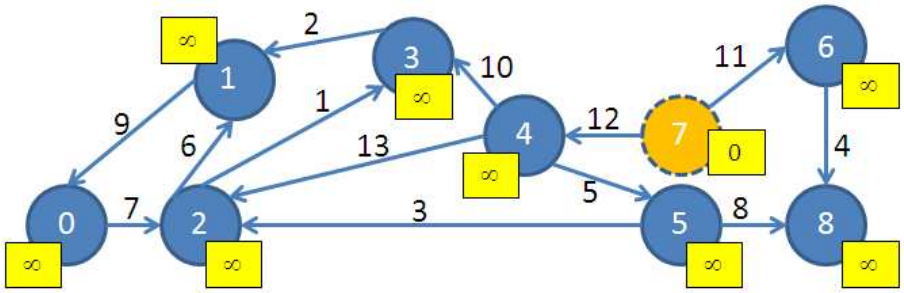


Figure 3: For Section 2 Question 3

Draw the resulting Shortest Paths Spanning Tree (the predecessor array `p`) below (5 marks):
(For each vertex, please also indicate the shortest paths values `D` from source `s = 7` to that vertex).

Q4. Depth First Search (DFS)/Topological Sort (5+2 = 7 marks)

Run topological sort algorithm as shown in Lecture 5 on the graph in Figure 4. If there are two or more valid next vertices, please always select vertex with the *lowest vertex number*. Note that the topological sort algorithm will explore *all* connected components of the graph.

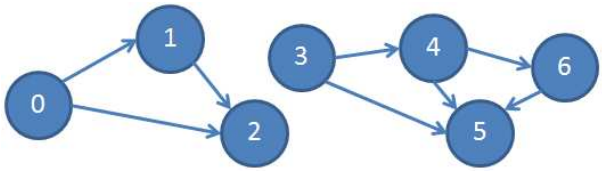


Figure 4: For Section 2 Question 4 and 5

Draw the resulting DFS spanning tree (actually forest) below (5 marks):

Then, write down the topological order found *by that algorithm* (already reversed) – (2 marks):

---------- .

Q5. Graph DS (Adjacency Matrix/Adjacency List/Edge List) (5 marks)

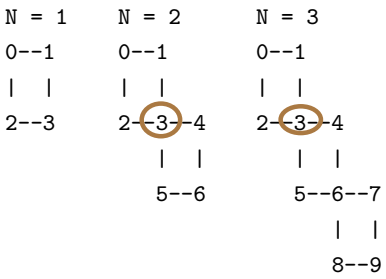
Draw *either* Adjacency Matrix, Adjacency List, or Edge List of the graph shown in Figure 4.
For **Adjacency Matrix**, you can leave a cell blank to indicate that it has value 0 (no edge).
For **Adjacency List**, use a list of Integers (vertex number of neighbors) as all edge weights are 1. Sort the neighbors based on increasing vertex number.
For **Edge List**, you do not need to use the weight column as all edge weights are 1, but make sure the direction of edges are clear: (u, v) means $u \rightarrow v$. Sort the Edge List based on increasing u, then if ties, by increasing v.
You can choose to draw *all three* data structures but your marks is **capped to 5 marks**.
Therefore to save time, please just draw one.

3 PS++ (31 marks)

PSes are the core of CS2010 as students will probably spend more than the three recommended hours per week to deal with these problems. These group of questions are variants of the existing PSes. Can you use the knowledge that you have gained when you solved these PSes to solve the new variants?

3.1 PS3++, please refer to `ps3.pdf` that you have printed (8 marks)

Suppose that the given graph (hospital building) is always an artistic building with special shape defined by these rules: Let N be a parameter of the graph (hospital building) ($1 \leq N \leq 1000000$). There are $1 + 3 \times N$ vertices (hospital rooms) in this artistic building, numbered from $[0 \ldots 3 \times N]$. There are edges *only* between the following vertices (hospital rooms) $(i, i+1)$, $(i, i+2)$, $(i+1, i+3)$, $(i+2, i+3)$, for all $i \in [0, 3, 6, ..., 3*(N-1)]$. The rating score of each room is stored as *vertex weight* as with PS3. An example of these special graphs (hospital buildings) with various small N are shown below (rating scores of the rooms are not shown):



impt rooms are rooms which if you remove, you disconnect the graph

qn: find articulation points with the lowest rating score

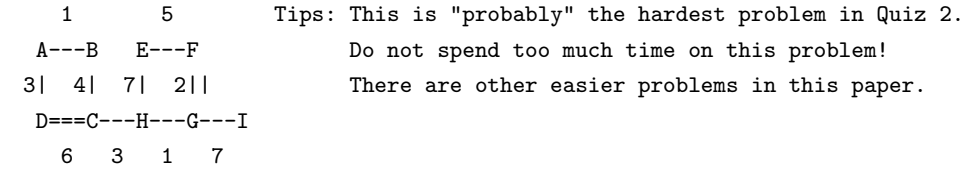
As with PS3, devise the *best possible* solution to get the lowest rating score among the important rooms of this special graph (you can assume that *no* other graph type will be asked) and analyze its time complexity *per query* w.r.t N. Just outline your ideas with pseudo codes to answer this question. Note 1: To make this problem fair for all students, Steven *forbids* the usage of the $O(V+E)$ Hopcroft-Tarjan's algorithm for finding articulation point/bridges (which is the solution for PS3, bonus points). Note 2: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

n = 1, no articulation point
n > 2, articulation point are all multiples of 3 < 3n

just go through all the vertex stored and get the lowest weight amongst the impt room(value is <3N)

3.2 ***PS4++, please refer to ps4.pdf that you have printed (13 marks)

Steven wants to make the problem of finding the best path for Grace in PS4 *a little bit more realistic*. Suppose that now Grace *has to* visit certain important corridor(s) along the path from her starting point to her target point. For example, see the diagram below for the same test case as shown in `ps4.pdf`, but slightly modified at three edges: Edge D=C, edge F=G, and an addition of edge G-I.



Suppose Grace wants to go from point `D` to point `F` and she also has to visit a toilet that is located in between corridor D=C and wants to buy certain snack that is only located somewhere along corridor G=F. In the *original* problem, Grace will choose this best path: $D \rightarrow A \rightarrow B \rightarrow C \rightarrow H \rightarrow G \rightarrow F$ with max effort rating of 4 at edge B-C. However, for this new problem variant, she will choose this best path: $D \Rightarrow C \rightarrow H \rightarrow G \Rightarrow F$ with max effort rating of 6 at edge D=C. This way, she will pass edge D=C and edge G=F along her best path. There is no other path that has max effort rating less than 6 yet still ensures that Grace visits both important edges D=C and edge G=F. Answer is 6.

If Grace wants to go from point `D` to point `I`, the best path is: $D \Rightarrow C \rightarrow H \rightarrow G \Rightarrow F \Rightarrow G \rightarrow I$ (she *has to* visit edge D=C and G=F!) with max effort rating of 7 at edge G-I. Answer is 7.

Please come up with the *best possible* solution to handle this new requirement and analyze its time complexity per query. On top of the requirements in PS4, you now have access to a Boolean function `isImportantCorridor(u, v)` that returns `true` if edge (u, v) (or (v, u)) is an important corridor; or `false` otherwise. You only need to report a number, *the max effort rating* along the best path that visits all important corridors. Just outline your ideas with pseudo codes to answer this question. Note: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

preliminary:

find the path that minimises the maximum difficulty rating -> minimax path, could be longer but the max value in the path is smaller
to solve it just get mst do bfs/dfs from start to end and track the largest edge along the path

O(E log V) because it is standard kruskals

qn wants how to find minimax given a set of edges that MUST be crossed

it means that the edges in the set must be in the MST, so first we add them into the UFDS, and run kruskal's to find the remaining edges in the sorted edge list to be inserted into in the MST

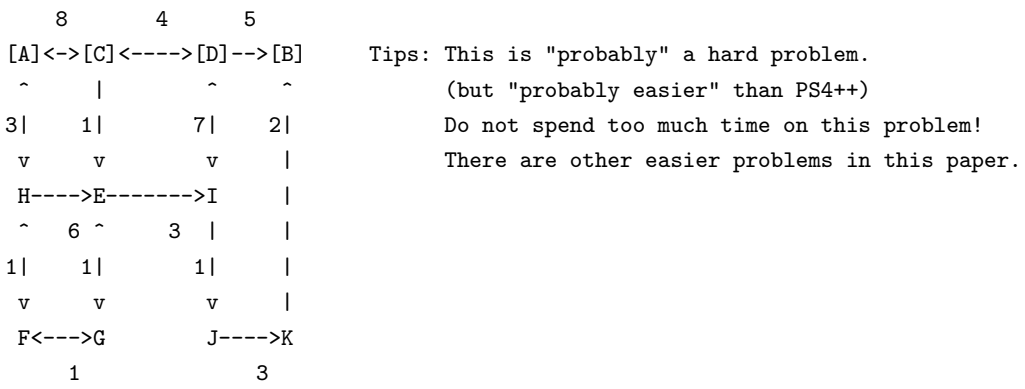
2nd part: how to find max edge along the path that she must take
in result tree to dfs/bfs starting from s and created a variable to keep track of max edge weight,
if it hits one of the required edges, check the largest edge along current path and compare it to max edge weight
if it is larger than max weight, update max weight

O(V) because we just do traversal of the tree

total time complexity is O(E log V)

3.3 *PS5++, please refer to `ps5.pdf` that you have printed (10 marks)

Steven wants to make the problem of finding the shortest path from Steven and Grace's home to hospital in PS5 *a little bit more realistic*. Suppose that the scenario is as follows: Steven is currently not at home (vertex A), but at his office (vertex C). He also does not have enough cash so he will need to make a quick visit to his bank (vertex D). Steven is wondering what is the *fastest total estimated traveling time* if he starts from his office (vertex C), ends at hospital (vertex B), 'pass through' (assume that visitation time is negligible/0 minute) home (vertex A) **and** bank (vertex D), in any order. See an example figure below which is *different* from the one in `ps5.pdf` (please use this version instead).



Upon receiving an emergency call from Grace, Steven's best path on the example figure above is this: Steven takes a taxi from his office (vertex C), go home first (vertex A), takes Grace, go to bank (vertex D), gets some cash, and then go to hospital (vertex B). The detailed path is: $[C] \rightarrow E \rightarrow G \rightarrow F \rightarrow H \rightarrow [A] \rightarrow C \rightarrow [D] \rightarrow [B]$ with total estimated traveling time of: 1+1+1+1+3+(0)+8+(0)+4+5 = 24. Note that in this best path, Steven's office (vertex C) is revisited twice (it is OK).

Please come up with the *best possible* solution to handle this new requirement and analyze its time complexity per query. On top of the requirements in PS5: Vertex 0/'A' (home) and Vertex 1/'B' (hospital), you are now given two more special vertices: Vertex 2/'C' (Steven's office) and Vertex 3/'D' (Steven's bank). Just outline your ideas with pseudo codes to answer this question. Note: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

If you really need more space for PS3++/4++/5++ answers, use the extra space below.

4 Analysis (15 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 (preferred) or at least a convincing argument.
If you want to disprove it, provide at least one counter example.
Three marks per each statement below (1 mark for saying correct/wrong, 2 marks for explanation):
Note: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

1. Every bipartite graph does not contain an odd cycle (a cycle with odd number of edges).

   true -> bipartite graph is two colourable, every vertex has to be adjacent to an edge of a diff colour,
   if there are odd cycles it is not two colourable -> there exists an edge between two vertices in the same set

2. Every bipartite graph is also a tree.

   false -> tree is connected graph with V-1 edges but for bipartite graph with v-1 edges, if we add
   one more edge it is still bipartite as long as it remains 2 colourable and it does not contain odd cycles.

   alternatively: bipartite graph can have cycles as long as they are not odd, but tree cannot have a cycle as
   there is only one unique path between every pair of vertices?

   every tree is a bipartite graph because there is no odd length cycles

3. Every graph with V vertices and $V - 1$ edges is always a tree.

   false. A disconnected graph could have V vertices and V - 1 edges which doesnt make it a tree. i.e.
   graph with 4 vertices with 3 edges connecting 3 vertices(to form a cycle) and the 4th vertex being
   disconnected

4. Every connected graph G has a spanning tree.

   true. A spanning tree is a tree that connects all of the vertices in a graph so if the graph is connected,
   there is a path to every vertex which results in a spanning tree

   if the graph is a tree, spanning tree of the tree is the tree itself

5. If we remove an articulation point (also known as cut vertex or 'important room' in PS3) of an undirected graph, we will always break the graph into *two* connected components.

   false, cut vertex could be connected to more than one components which results in more than 2 connected
   components being formed. ie graph with 8 vertices where 6 are in a cycle and 7 and 8 are each connected to
   the cycle by the cut vertex(and are not connected to each other). deleting the cut vertex creates 3 connected components
   the cycle with 5 vertices and each vertex 7 and 8

   bridge is equivalent of cut vertex -> if we remove it we get a disconnected graph

5 **Money Changer (10 marks)

Given n currencies and m exchange rates, determine if we can start with a certain amount of money in one currency, exchange this amount for other currencies, and end up at the same currency but with *more money* than what we had at first.

Example 1: Suppose the money changer has $n = 3$ currencies and $m = 3$ exchange rates: 1 USD gives us 0.8 Euro; 1 Euro gives us 0.8 GBP (British pound sterling); and 1 GBP gives us 1.7 USD. So if we start with 1 USD, we can exchange it for 0.8 Euro, which can then be exchanged for 0.64 GBP, and if converted back to USD we have 1.088 dollars. We have just make a profit of 88 cents :O.

Example 2: If the money changer has the following $n = 2$ currencies and $m = 2$ exchange rates: 1 USD gives us 0.8 Euro; and 1 Euro gives us 1.25 USD, then there is no way we can make a profit.

Can you model the n currencies and their m exchange rates as a graph problem and give an algorithm to report whether it is possible to start with any currency, exchange it with one (or more) other currencies, end with the same starting currency, and make a profit?

Just outline your ideas with pseudo codes to answer this question.

Note 1: You are only given a small amount of space below (i.e. do **not** write too long-winded answer)!

Note 2: Do **not** attempt this question unless you have completed the other questions.

– End of this Paper –

Candidates, please do not touch this table!

Question	Maximum Marks	Student's Marks
1	18	
2	29	
3	31	
4	15	
5	10	
Total	103	