National University of Singapore

School of Computing

Semester 1 (2011/2012)

CS2010 - Data Structures and Algorithms II

Quiz 2 (20%)

Saturday, October 27, 2012, 10.00am-12.00pm (120 minutes)

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this question paper until you are told to do so.
2. Quiz 2 is conducted at COM1-2-206/SR1.
3. This question paper contains FIVE (5) sections with sub-questions.
   It comprises FOURTEEN (14) printed pages, including this page.
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, Steven's 'Competitive Programming 1/2/2.5' book, or *any other printed material* that you think will be useful. But remember that the more time that you spend flipping through your files implies that you have less time to actually answering the questions.
6. **Please write your Matriculation Number here:** ______________________
   and your Tutorial Group Number (or Tutor Name): ______________________
   But *do not* write your name in order to facilitate unbiased grading.
7. You are encouraged to use Java syntax to answer questions in this paper as it is much clearer. However, as promised, you can still use pseudo-code but beware of penalty marks for **ambiguous answer**. You can also use **standard, non-modified** algorithms discussed in class by just mentioning their names. However, if you need to modify **any part** of the standard algorithm, you have to write the **full algorithm**.
8. All the best :).

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

1 'ADT Table': (Questions → Answers) (14 marks)

The answers for this set of questions can be found in the lecture notes (or mentioned during lecture), tutorial files, PS files, Competitive Programming book, etc.
Can you find them *as fast as you can*? It is $O(1)$ if you already have the answer in your memory.
Please fill in your answers on the space provided, 2 marks per question.
Grading scheme: 0 (zero correct answer), 1 (one correct answer), 2 (two correct answers).

1. Graph A of V vertices is known to be a bipartite graph.
   Thus, the max number of edges that graph A can possibly have is $E =$ ________________ .
   Graph B of V vertices is known to be a tree.
   Thus, the max number of edges that graph B can possibly have is $E =$ ________________ .

2. When the in-degree and out-degree of each vertex of a graph is the same,
   the graph *could be* a/an ____________________ graph.
   When all vertices have in-degree and out-degree $= V - 1$,
   the graph *must be* a/an ____________________ graph.

3. To store a graph with V vertices and E edges, an Adjacency Matrix requires $O(V^2)$ space, whereas an Adjacency List requires __________________ space,
   and an Edge List requires __________________ space.

4. The best graph data structure if we need to list down the edges based on increasing edge weight is an __________________ , whereas the best graph data structure if we need to frequently check the existence of an edge between two vertices is an __________________ .

5. Please mention at least two greedy graph algorithms that you know:
   1). ____________________________________
   2). ____________________________________

6. Bellman Ford's algorithm is too __________________ to be used to solve the SSSP problem on a relatively large directed weighted graph ($V \times E > 100000000$) whereas BFS algorithm (usually) produces __________________ answer when used to solve the SSSP problem on a typical directed weighted graph.

7. The *best* algorithm to solve the SSSP problem when the input graph is a non-negative weighted tree is: __________________ , whereas the best algorithm to solve the SSSP problem when the input graph is a non-negative weighted bipartite graph is __________________ .

2 Basic Graph DS/Algorithm Questions (24 marks)

Grading scheme: 0 (no answer), 1 (the final answer is wrong, **regardless whether the mistake is minor or major**), 4 (the final answer is correct).

Q1. Draw a (Simple) Graph I (4 marks)

Draw a directed graph with $V = 5$ vertices and $E = 10$ directed edges. This directed graph **cannot** has any cycle. The 5 vertices have been given below. Your task is to draw **exactly** 10 **directed** edges.

Figure 1: For Section 2 Question 1

Q2. Draw a (Simple) Graph II (4 marks)

Draw a graph with $V = 5$ vertices such that calling `dfs(0)`/`bfs(0)` (that is, running DFS or BFS from source vertex 0) produces the same DFS/BFS spanning tree. The 5 vertices have been given below. Your task is to draw **at least two undirected edges as you need** to satisfy the requirement.

Figure 2: For Section 2 Question 2

Q3. Bipartite Graph? (4 marks)

Is the graph below (on the left side) a bipartite graph? If yes, please redraw it using the space provided on the right to clearly indicate that it is really a bipartite graph. If no, write a word 'NOT BIPARTITE' as your answer.

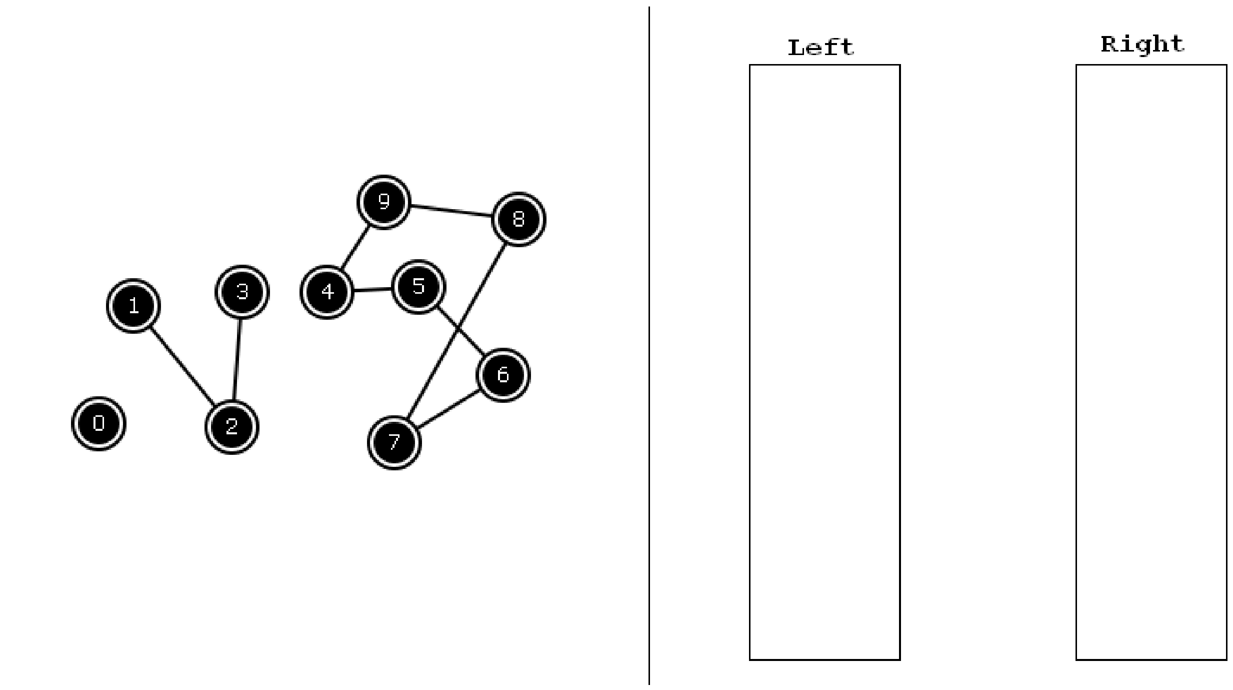


Figure 3: For Section 2 Question 3

Q4. DFS (4 marks)

Run DFS from vertex 0 of this graph below. The neighbors of a vertex are ordered based on increasing vertex number. Your task is to draw the **current (not the final)** DFS spanning tree when DFS visits vertex 8 **for the first time**.

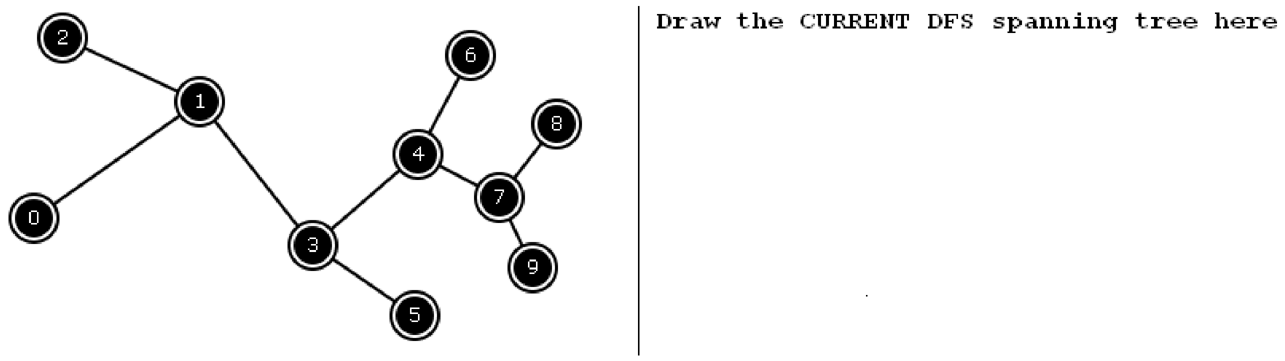


Figure 4: For Section 2 Question 4

Q5. Minimum Spanning Tree (4 marks)

Show the actual Minimum Spanning Tree of this connected, undirected, weighted graph below. You can use either Prim's (you can start from any source) or Kruskal's algorithms. Both algorithms have been tested to arrive at the same unique output. There should not be any case for tie breaking as all edges have different weight (note: edge 0-3 has weight 9).

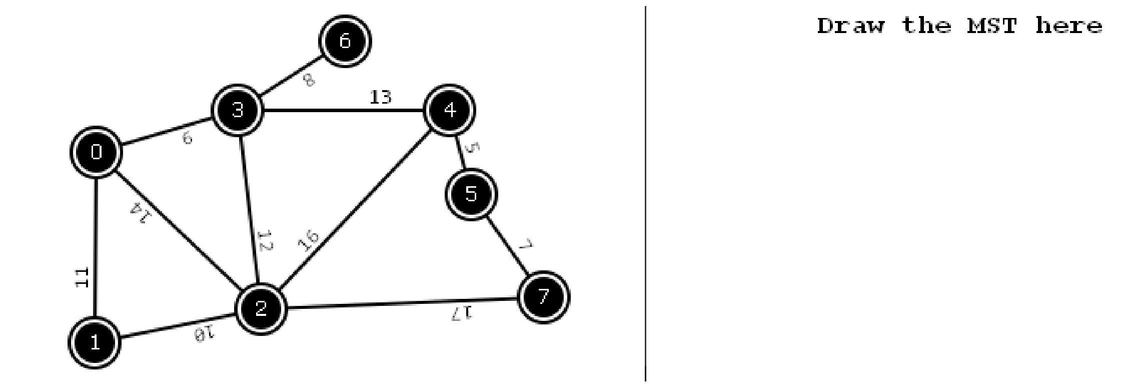


Figure 5: For Section 2 Question 5

Q6. Single-Source Shortest Paths (4 marks)

Show the actual Shortest Paths Spanning Tree of this directed, weighted graph below. Remember: The arrow tip for directed edge is shown as a circle! The source vertex is vertex 0. Please also indicate the shortest path values of each vertex by filling in the table below. You can use either version of Dijkstra's algorithms or even Bellman Ford's algorithm. The output should be unique (note: both edge 0 → 3 and edge 4 → 6 have weight 6; edge 7 → 5 has weight 2).

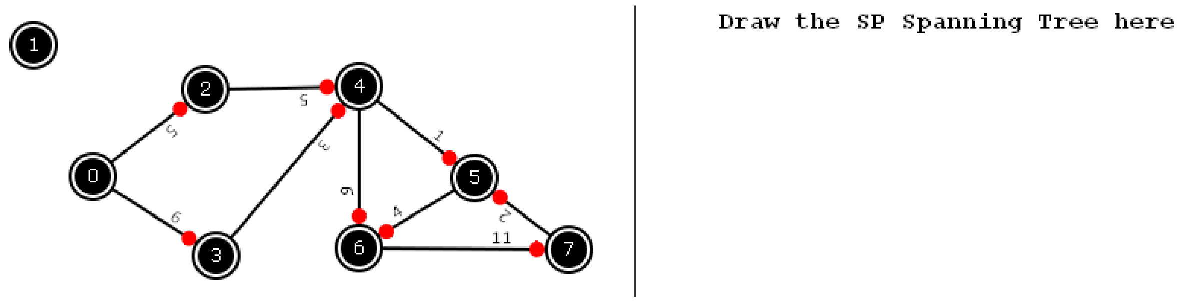


Figure 6: For Section 2 Question 6

	0	1	2	3	4	5	6	7
Initially	0	∞	∞	∞	∞	∞	∞	∞
The actual shortest path values								

3 Analysis (18 marks)

Prove (the statement is true) or disprove (the statement is false) 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. A connected graph have at least $E = V - 1$ edges.

   true. connected graph means it will have a spanning tree and a tree is the smallest connected graph with V - 1 edges

2. A single linked list has V vertices and $E = V - 1$ directed edges. These directed edges go from vertex i to vertex $i + 1$, $\forall i \in [0 \ldots V\text{-}2]$. This graph only has one possible topological sort.

   true. since it is has V - 1 edges, it is a "tree" or a DAG with a unique path between each pair of vertices and only one path of traversal hence only has one possible topo sort

   has linear ordering of vertices 1,2,3,...

3. We want to run DFS on a *general graph* that is currently stored in an Edge List. There is no way we can run DFS faster than $O(VE)$ because for every vertex (there are $O(V)$ such vertices), we need another $O(E)$ time to determine the neighbors.

   true. for each vertex we will need to run through edge list again to find its next neighbour

   false. we can convert edge list to adjacency list in O(V + E) time and then perform DFS on the adj list in O(V + E) time

4. We can make Kruskal's algorithm runs faster than $O(E \log V)$ if the input graph is a connected undirected weighted graph where all edge weight is within small integer range [1 .. 100].

   false, as edges are already sorted in ascending order depending on the sorting algorithm used and not on the weights of the graph and the main selection of edges has a very slow growth rate so the sorting algorithm dominates the time complexity

   true if its small range that is bounded we can do radix sort in O(E) time and going through the edge list gives us O(E alpha(V)) time which is the new dominating time complexity

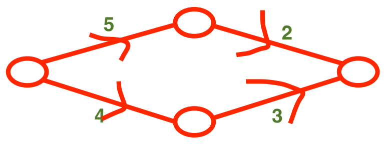
5. The Minimum Spanning Tree of a connected undirected weighted graph where all edges weights are distinct is always unique.

   true, since all edge weights are distinct there are no alternative edges that could be taken in the MST

6. The Shortest Paths Spanning Tree of a directed weighted graph where all edges weight are distinct is always unique (the source is always vertex 0).

   true, there is only one way to relax all the edges

   false this tree had unique edge weights but two shortest paths with the same value



4 Intermediate Graph DS/Algorithm Question (12 marks)

4.1 IsTree (12 marks)

Please implement the following function `IsTree` that takes in a graph stored in an Adjacency List and returns true if the graph is a tree, or return false otherwise.

The format of the Adjacency List is as outlined in Lecture 05: A Vector of V lists, one for each vertex. List i contains another Vector of IntegerPairs where we store list of IntegerPair information: (the vertex ID, the corresponding edge weight) of neighbors of i. Note that bidirectional edge (i, j) will be stored *twice* in this format, i.e. edge $(i \to j)$ is stored in list i whereas the other edge $(j \to i)$ is stored in list j.

A tree is as outlined in Lecture 05: A connected (undirected) graph with V vertices and $E = V - 1$ (undirected) edges. There is only one unique path between any pair of vertices in the tree.

```
Boolean IsTree(Vector < Vector < IntegerPair > > AdjList) {
```

adj list cant diferentiate if edges are bidrected or undirected

```
1) check first property of tree: E = V - 1 -> means it is unique path alr

    go through each index and sum up size of neighbour list.
    if(sum / 2 != v - 1)
       return false

2) check if the graph is a single connected component
    if no of cc != 1
       return false
```

```
}
```

5 Escape Plan (32 marks)

5.1 The Story

You works in a maze. Unfortunately, portions of the maze have caught on fire, and the owner of the maze neglected to create a fire escape plan. You need to escape QUICKLY.

Given your location in the maze and which squares (can be zero, one, or more than one) of the maze that are initially on fire (when you first realized that there is a fire), you must determine **whether** you can exit the maze before the fire reaches you; and if you can, **how fast** you can do it.

Both you and any of the fire each move one square per minute, vertically or horizontally (**not diagonally**). The fire spreads all four directions from each square that is on fire. You may exit the maze *from any square that borders the edge of the maze.* Neither you nor the fire may enter a square that is occupied by a wall.

You are given two integers R and C ($1 \leq R, C \leq 1000$) and a 2D character array M with R rows and C columns that describes the starting condition of the maze. Each cell can contain either:

- '#', a wall, neither you nor the fire may enter this cell.
- '.', a passable square (for you and the fire).
- 'Y', your initial position in the maze, which is a passable square.
  There will be **exactly one** 'Y' cell only in the test case.
- 'F', a square that is currently on fire at this point of time.
  There can be **zero, one, or more than one** 'F' cell(s) in the test case.

Your task is to implement the following function `EscapePlan` that takes in R, C, and M and simply returns -1 if you *cannot* exit the maze before the fire reaches you, or a positive integer that gives the earliest time that you can safely exit the maze, in minutes.

5.2 Example 1: R1 = 4, C1 = 4, maze M1 is as shown below

```
####     ####     ####     ####
#YF#     #FF#     #FF#     #FF#
#..#     #YF#     #FF#     #FF#
#..#     #..#     #YF#     #FF#
                            Y
t=0      t=1      t=2      t=3 (you manage to escape)
```

The function `EscapePlan(R1, C1, M1)` will return 3 because by running downwards **three steps (in three minutes)**, you can escape the maze while the fire cannot catch you.

5.3 Example 2: R2 = 3, C2 = 6, maze M2 is as shown below

```
######     ######
#Y....     #YF..F
#.F..F     #FFFFF


t=0        t=1 (you are already trapped)
```

The function `EscapePlan(R2, C2, M2)` will return -1 because after 1 minute, the nearest fire already surrounds you... Note that if you and the fire reach the same cell at the same time..., you die...

5.4 Example 3: R3 = 3, C3 = 3, maze M3 is as shown below

```
###     ###     ###
#Y.     #.Y     #..Y
###     ###     ###


t=0     t=1     t=2
```

The function `EscapePlan(R3, C3, M3)` will return 2.

5.5 Example 4: R4 = 5, C4 = 5, maze M4 is as shown below

```
F...F     FF.FF     FFFFF     FFFFF
.....     F...F     FF.FF     FFFFF
..Y..     ....F     F..FF     FFFFF
....F     ..YFF     ..FFF     FFFFF
.....     ....F     ..YFF     ..FFF
                                Y
t=0       t=1       t=2       t=3 (a narrow escape)
```

The function `EscapePlan(R4, C4, M4)` will return ~~4~~, a **narrow escape** as shown above.

5.6 Graph Modeling (6 marks)

What do the vertices and the edges of the graph represent? (2 marks)

What is the graph problem that we want to solve? (2 marks)

What is the most appropriate graph algorithm to solve this problem? (2 marks)

5.7 Helper Functions (8 marks)

```
private static Boolean InsideMaze(int row, int col, int R, int C) { // 1 mark
  // returns true if cell (row, col) is inside the RxC maze, false otherwise
  // row and col are using 0-based indexing
  // e.g. InsideMaze(0, 2, 3, 3) = true, InsideMaze(0, 3, 3, 3) = false



}


private static Boolean OnMazeBorder(int row, int col, int R, int C) { // 1 mark
  // returns true if cell (row, col) is on the N/E/S/W border of the maze
  // returns false otherwise (again, row and col are using 0-based indexing)
  // e.g. OnMazeBorder(0, 2, 3, 3) = true, OnMazeBorder(~~0~~, 1, 3, 3) = false



}


private static IntegerPair You(int R, int C, char[][] M) { // 3 marks
  // returns the coordinates of you (one cell that contains 'Y') in M
  // e.g. if R = 1, C = 5, M = "..Y..", You(R, C, M) = (0, 2)
  // Note: This function must be at most O(R * C)



}


private static Vector < IntegerPair > Fire(int R, int C, char[][] M) { // 3 marks
  // returns the LIST of coordinates of the fire(s) (cell that contains 'F') in M
  // sort the list based on increasing row, and if ties, by increasing col
  // e.g. if R = 1, C = 5, M = "F.Y.F", Fire(R, C, M) = {(0, 0), (0, 4)}
  // Note: This function must be at most O(R * C)



}
```

5.8 Your Solution (18 marks)

Now, write your solution below. You are **encouraged to** use four helper functions that you have defined earlier. You can define additional helper functions if you want (use the space in Page 13/14).

```
int EscapePlan(int R, int C, char[][] M) { // up to 18 marks

}
```

There are different possible solutions for this non-original problem (the problem source will be revealed after Quiz 2). All possible solutions just require CS2010 knowledge taught so far. Your solution will be awarded different marks based on the criteria below:

Maximum Marks	Requirements
1	Incomplete solution (NOTE: Partial Marks are in Section 5.7)
7	An $O(R \times C)$ solution, but only works if there is no 'F' in the maze
13	An $O((k+1) \times R \times C)$ solution, where k is the number of 'F' in the maze
18	An $O(R \times C)$ solution, independent of k

5.9 Bonus Marks (3 marks)

Do **not** attempt this time consuming task unless you have finished all other questions.
What is the return value of `EscapePlan(10, 10, M)`?
`M` of size 10×10 is shown below:

```
FFFFFFFFFF
.########.
.#Y#......
.#...F..#.
...#...#F.
#....#FFF#
#......FF#
#..#....F#
#.#......#
#......#F#
```

– End of this Paper –

Candidates, please do not touch this table!

Question	Maximum Marks	Student's Marks
1	14	
2	24	
3	18	
4	12	
5	32+3	
Total	103	