National University of Singapore

School of Computing

CS2010 - Data Structures and Algorithms II

(Semester 1: AY2015/16)

Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this assessment paper until you are told to do so.
2. This assessment paper contains FOUR (4) sections.
   It comprises ELEVEN (11) printed pages, including this page.
3. This is an **Open Book Assessment**.
4. Answer **ALL** questions within the **boxed space** in this booklet.
   You can use either pen or pencil. Just make sure that you write **legibly**!
5. 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.
6. 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.
7. Write your Student Number in the box below:

A	0							

This portion is for examiner's use only

Section	Maximum Marks	Your Marks	Remarks by Grader
A	15		
B	15		
C	60		
D	10		
Total	100		

A Basics (15 marks)

Please fill in your answers on the blank spaces provided.

1. **(5 marks)** List down **five more** *special graphs* that you are aware of and state what is so special about them that can be used to simplify a certain graph data structure and/or algorithm. All graphs must be simple graphs, i.e. no self-loop nor multiple edges between the same pair of vertices. One example has been shown below.

   (a) Complete Graph. Reason: It has V vertices and $E = V \times (V-1)/2$ (undirected) edges that connects every pair of vertices in the graph. This fact allows the entire graph structure to be stored by just memorizing a single integer, the number of vertices V.

   (b)

   (c)

   (d)

   (e)

   (f)

Page 2 Marks = _____

2. **(5 marks)** Let n_h be the minimum number of vertices in an AVL Tree of height h. Here, the height of an AVL Tree is defined as the number of edges from the root to the deepest leaf of that AVL tree.

   (a) **(1 mark)** What is the value of n_h when $h = 0$ and $h = 1$?

   (b) **(1 mark)** What is the formula to compute n_h when $h \geq 2$?

   (c) **(3 marks)** What is the value of n_h when $h = 10$, $h = 15$, and $h = 20$, respectively?

3. **(5 marks)** This question is based on the following 5×5 grid of **base-25 number**, i.e. the digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (for 10), B (for 11), ..., N (for 23), and O (for 24):

I	J	K	5	4
H	**O**	L	6	3
G	N	M	7	2
F	C	B	8	1
E	D	A	9	0

   The source vertex s is the cell with the highest value, i.e. cell with character **'O' ('O' for Orange)**, highlighted with **bold** in the grid above. The target vertex t is the cell with the lowest value, i.e. cell with integer 0 (zero), highlighted with underline in the grid above.

   We can go from one cell with value X to its North/East/South/West cell with value Y if $X > Y$.

   Answer the following (**1 mark** each):

   (a) Is there a cycle reachable from the source vertex s? Mention such cycle, if any! _____

   (b) There are _____ cells that are reachable from the source vertex (inclusive).

   (c) The **length of** the shortest **unweighted** path between s and t is _____ edges.

   (d) The **length of** the longest **unweighted** path between s and t is _____ edges.

   (e) The **number of** paths between s and t is _____.

Section A Marks = _____

B Analysis (15 marks)

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

1. Let {0, 1, 2, 3} be the topological order of a Directed Acyclic Graph (DAG) with 4 vertices labeled with [0..3]. We compute the shortest paths from **source vertex 1**. Without knowing the actual edge weights, we can conclude that the shortest path value $D[0] = \delta(1, 0) = \infty$.

   true. since 0 comes before 1 and we are computing from 1 onwards, there is no directed edge going to 0 hence the path is never traversed and its weight is inf

2. The last vertex in a topological order of a DAG must have **in**-degree 0.

   false. if the graph is connected then the last vertex has a directed edge pointing towards it from the previous edge which makes the in degree 1

3. We can use $O(VE)$ Bellman Ford's algorithm to detect if there exists a **positive weight cycle** reachable from source vertex 0 in a general directed weighted graph.

   true. using a visited array. if running bellman ford's

4. **Only** $O(VE)$ Bellman Ford's algorithm can be used to detect if a directed weighted graph G has a **negative weight cycle** reachable from source vertex 0.

   false, floyd warshalls will work too by just checking the adjacency matrix

5. An **undirected weighted tree** is stored in an Adjacency List. Each undirected edge (a, b) with weight w is stored twice as two directed edges $a \rightarrow b$ and $b \rightarrow a$ with the same weight w. **All edge weights are negative**. We run $O(V^2)$ Bellman Ford's algorithm as $E = 2 * (V - 1)$ and will get a report that there is **no negative weight cycle found** as the input is a tree.

   false, the bidirectional edges themselves from a negative weight cycle where there edges are infinitely relaxed
   i.e. 0 -> 1 : -1
   1 -> 1: - 1
   0 -> 1: - 2
   1 -> 0: - 3

Section B Marks = _____

C Applications (60 marks)

Background Story: Robot Again

You are given a 2D square grid of size $(N+1) \times (N+1)$ and $1 \leq N \leq 100\,000$. You have a robot and want to programme the robot to traverse this grid.

There are K battery charging stations in the grid located numbered from station $[1..K]$ and $4 \leq K \leq 1000$. The charging station is always located at integer coordinates. Charging station 1 is located at cell $(0, 0)$ and that is where your robot currently located. Charging station K is located at cell (N, N) and that is the final destination of your robot. The other $K - 2$ charging stations are located in various cells in the grid. No two charging stations are located at the same cell.

Moving from one charging station i located at cell (a, b) to another charging station j located at cell (c, d) consumes $(a - c)^2 + (b - d)^2$ energy units from your robot's battery. For example, moving from charging station 1 at $(0, 0)$ to charging station 2 at $(0, 2)$ and from charging station 4 at $(4, 2)$ to charging station 7 at $(3, 4)$ in Figure 2 consume $(0 - 0)^2 + (2 - 0)^2 = 4$ and $(3 - 4)^2 + (4 - 2)^2 = 5$ energy units, respectively.

	0	1	2	3	4	5
0	1		2		5	
1						6
2		3				
3					7	8
4			4			
5						9

Figure 1: Example 2D square grid with $N = 5$ and $K = 9$

C.1 Data Structure (5 marks)

How are you going to store the information of this $(N + 1) \times (N + 1)$ 2D square grid and the coordinates of the K charging stations? Use any data structure(s) that is/are suitable and state the space complexit(ies).

store the grid as an adjacency list -> O(V + E)
store the charging stations as IntegerPair(row, col) -> these are vertices
calculate the energy consumed between all pairs of charging stations -> this are the edges and energy is edge weight

Page 5 Marks = _____

C.2 Minimum Charging Actions (15 marks)

Suppose that your robot has battery capacity of M energy units, $0 \leq M \leq 2 \times 100\,000^2$. Your robot can only move from one charging station to another if its battery capacity allows it to do so. Each time your robot is at a charging station, it uses a 'kiasu' strategy whereby it always re-charge itself to **full** M energy units. At the beginning, your robot has empty battery capacity and will do its first full charging action at charging station 1 so its battery is now at full M energy units.

Design the best algorithm so that your robot can move from charging station 1 to charging station K with **minimum charging actions**. If it is impossible to reach charging station K with battery capacity of M energy units, report "Impossible, buy larger battery". What is the time complexity of your algorithm?

Let's use Figure 2 for illustration. If $M = 5$, then your robot requires at minimum 5 charging actions at charging station 1, 3, 4, 7, and 9. If $M = 3$, then the answer is "Impossible, buy larger battery" as your robot cannot move from charging station 1 to any other charging station without running out of battery midway.

```
run prim's on the adjacency list starting at (1,1) ending at K with slight modification

1.enqueue edges connected to starting vertex into min heap pq
2.while(!pq.isEmpty() || pq.offer==K) {
    Vertex v = pq.poll();
    if(edge e > M) {
      print "impossible"
    }

    if(!taken[v]) {
       add it to the spanning tree
       enqueue each edge adjacent to v in the pq if its not already in the spanning tree
    }
  }
O(E log V)
```

Page 5+6 Marks = _____

C.3 Minimum Battery Capacity (15 marks)

Suppose you do not care about the minimum charging actions. What is the **minimum battery capacity** M_{min} so that it guarantees that your robot can go from charging station 1 to charging station K. Design the best algorithm to answer this question and analyze its time complexity.

Let's use Figure 2 again for illustration. The answer is $M_{min} = 4$. Your robot moves from charging station 1, 2, 5, 6, 8, 9, i.e. 6 charging actions, but the energy consumed when going from one station to the next never exceeds the battery capacity of $M_{min} = 4$ units.

Use modified floyd warshalls to find the minimax path from 1 to K. Select the maximum edge along the path this will be the min battery required to travel from 1 to K

O(V^3)

Page 5+6+7 Marks = _____

C.4 How Many Fun Ways? (15 marks)

Suppose that your robot has battery capacity of ∞ energy units. You can forget about charging the robot's battery when it traverses the 2D square grid from charging station 1 to charging station K. However, you write a simple additional constraint inside your robot so that it can only move from charging station i at (a, b) to charging station j at (c, d) if and only if $c > a$ and $d > b$ (it never needs to charge itself at charging station though). Again, using Figure 2 for illustration, if your robot is now at charging station 3, it **cannot** go to charging stations {1, 2, 5, 6} but your robot can go to any other charging stations {4, 7, 8, 9}.

Now you want to count **how many fun ways** that your robot can perform its traversal task **excluding** the direct path $1 \rightarrow K$ (as you declare that such direct path is 'not fun'). Design the best algorithm to answer this question and analyze its time complexity. There are **5** such ways on the example Figure 2:

1. $1 \rightarrow 3 \rightarrow 4 \rightarrow 9$
2. $1 \rightarrow 3 \rightarrow 7 \rightarrow 9$
3. $1 \rightarrow 3 \rightarrow 9$
4. $1 \rightarrow 4 \rightarrow 9$
5. $1 \rightarrow 7 \rightarrow 9$ (note that $1 \rightarrow 9$ is not counted as this is the direct path which is not fun)

Page 5+6+7+8 Marks = _____

C.5 Too Much Fun? (10 marks)

Draw any valid grid of size $(N+1) \times (N+1)$ with K charging stations so that the answer for question C.4 is more than **1 Billion (10^9) fun ways**. Remember that $1 \leq N \leq 100\,000, 4 \leq K \leq 1\,000$. You can also explain your answer in words if that is more convenient.

Section C Marks = _____

D Think Outside the Box (10 marks)

Warning: This is the last question in this assessment paper and also the hardest question.
You are encouraged to only attempt this question when you have completed all other questions.

The Problem Description

Given a directed weighted graph G with V vertices and E edges, source vertex $s = 0$, a target vertex $t = V - 1$, we want to know if we update the weight of a directed edge (a, b) from previous weight w to a **smaller** new weight w', will the shortest path value from s to t improves (becomes shorter) or not (remains the same)? There are Q queries and each query is independent. All edge weights in G are non-negative. Vertex $s = 0$ can always reach vertex $t = V - 1$. An algorithm is considered 'fast enough' for this problem if it performs not more than 100 Million steps.

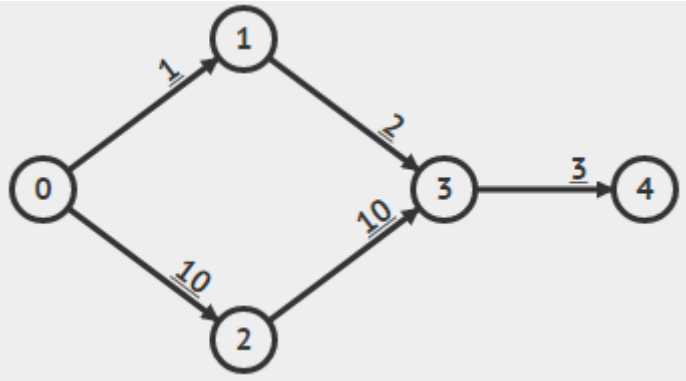

Figure 2: Sample directed weighted graph G with $V = 5$ vertices and $E = 5$ edges, shortest path value between $s = 0$ to $t = 4$ is currently 6.

Consider the graph in Figure 3.

1. If we update $(1, 3)$ from 2 down to 1, the shortest path value from 0 to 4 improves (becomes shorter, from 6 down to 5).

2. If we update $(2, 3)$ from 10 down to 0, the shortest path value from 0 to 4 does not improve (remains the same, which is 6).

D.1 Textbook Answer 1 (1 mark)

Give an algorithm that works for these constraints: $1 \leq V, E \leq 200\,000$, $1 \leq Q \leq 1$.
Analyze its time complexity!

BFS -> O(V + E)

Page 10 Mark = _____

D.2 Textbook Answer 2 (2 marks)

Give an algorithm that works for these constraints: $1 \leq V \leq 200$, $0 \leq E \leq 10\,000$, $1 \leq Q \leq 1\,000\,000$. Analyze its time complexity!

modified Dijkstra

use floyd warshall, we do a preprocessing step -> O(V^3) is within 100 million steps

D.3 Out of the Box Answer (7 marks)

Give an algorithm that works for these constraints: $1 \leq V, E \leq 200\,000$, $1 \leq Q \leq 1\,000\,000$. Analyze its time complexity!

– End of this Paper, All the Best –

Section D Marks = _____