National University of Singapore

School of Computing

EXAMINATION FOR

Semester 1 AY2011/2012

CS2010 - Data Structures and Algorithms II

Nov 2011, Time Allowed: 2 hours

INSTRUCTIONS TO CANDIDATES:

1. Do **NOT** open this question paper until you are told to do so.
2. This examination paper contains FIVE (5) questions with sub-questions.
   It comprises FOURTEEN (14) printed pages, including this page.
3. This is an **Open Book Examination**. You can check the lecture notes, tutorial files, problem set files, or any other books. But remember that the more time that you spend flipping through your files implies that you have less time to actually answering the questions.
4. Answer **ALL** questions within the 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. When this final exam starts, **please immediately write your Matriculation Number here:** _________________________ **(do not forget the last letter and do not write your name).**
7. All the best :).

This portion is for examiner's use only

Question	Maximum Marks	Student's Marks
1	20	
2	15	
3	15	
4	20	
5	30	
Total	100	

–This page is intentionally left almost blank. You can use it as 'rough paper'–
This figure below is for Section 1, Question 1 on page 3, so students do not need to flip the page.

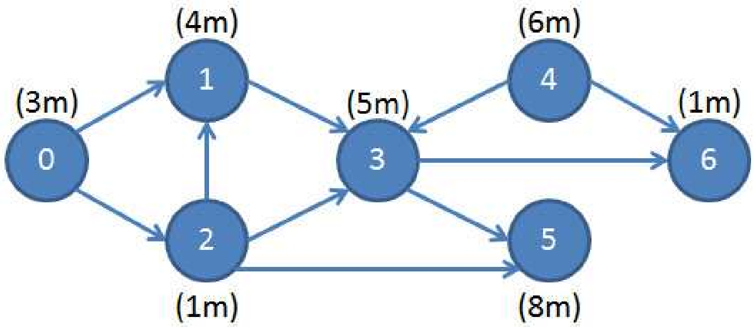


Figure 1: For Section 1, Question 1 (same as Figure 2 in Page 4)

1 Basic Understanding of CS2010 Materials (20 marks)

Please fill in your answers on the blank spaces provided. Each question has different marks.

1. List down **at least four** characteristics/properties/statements about the graph in Figure 1 on page 2 (or page 4) **(2 marks)**:
   1). ____directed____ , 2). ____7 SCCs____ ,
   3). ____cyclic____ , 4). ____unique MST____ .

2. Complete the following table. The first entry is given as an example **(5 marks)**.

Problem	Graph Characteristics	Best Algorithm	Time Complexity
SS Shortest Paths	Unweighted	BFS	$O(V+E)$
Min Spanning Tree	Weighted (positive)	Krukal's/Prim's	O(E log V)
Count Components	Tree	DFS	O(V + E)
SS Shortest Paths	Weighted	Bellman Ford's	O(VE)
Diameter of Graph	Weighted (positive)	Djikstra	O(V^2)
SS Shortest Paths	DAG	DFS/toposort, DP	O(V + E)

3. Suppose the content of distance array is `D` = {`0, 5, 7, 8, 10, 9, 11, 8`} and predecessor (or parent) array is `p` = {`-1, 0, 1, 1, 3, 3, 5, 5`} after running Dijkstra's algorithm (either version) on a small positive-weighted graph with $V = 8$ vertices and a certain source s. We know that `p[s] = -1`. Therefore, the source s must be vertex __0__ and the shortest path from that source s to vertex $t = 6$ is therefore path: __0 - 7-______ - 6 with `D[t] =` __19____ **(3 marks)**.

4. Complete the *comparison* table below. The first entry is given as an example **(10 marks)**.

	Similarities	Differences
Balanced BST vs Binary Heap (Max/Min Heap)	1). The underlying structures are the same: *Binary Tree* 2). ________	1). Delete(i) is $O(\log n)$ in balanced BST; $O(n)$ in heap when i != root 2). ________
Original Dijkstra's vs Prim's	1). ________ 2). ________	1). ________ 2). ________
Shortest Paths on DAG vs Longest Paths on DAG	1). ________ 2). ________	1). ________ 2). ________

2 Algorithms on (Explicit/Implicit) DAG (15 marks)

Q1. Earliest Possible Completion Time (8 marks)

Figure 2 shows a dependency graph of an important project that contains $V = 7$ vertices (sub-projects) and $E = 10$ directed edges (dependency among sub-projects). The brackets above each vertices in Figure 2 show the estimated time to complete each sub-project (in months). A sub-project can only be started when all its dependent sub-projects have been processed, e.g. in Figure 2, sub-project 1 can only be started when both sub-project 0 and sub-project 2 have been processed. You are the manager of this important project and you want to determine its earliest possible completion time. You do not care about the cost and you will make sure all sub-projects that are independent to be processed in parallel (by hiring more workers) in order to save time.

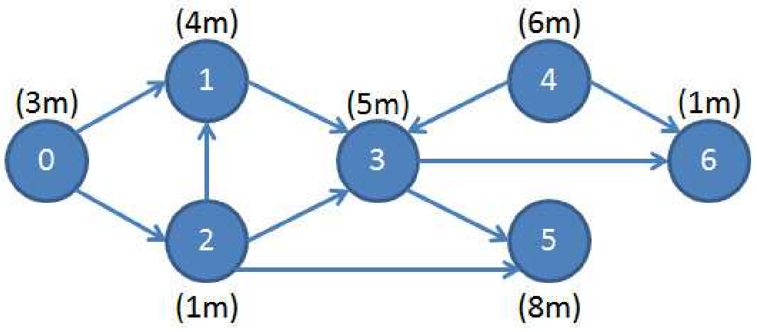


Figure 2: A Dependency Graph of an 'Important Project' (same as Figure 1 in Page 2)

A). Give *any* valid topological order of the DAG above **(1 mark)**: 0, 2, 1, 4, 3, 5, 6.

B). Determine the earliest possible completion time `T` for each vertex (sub-project) in the DAG below and also *highlight* (with **bold lines**) the *critical path* of this DAG **(4 marks)**:

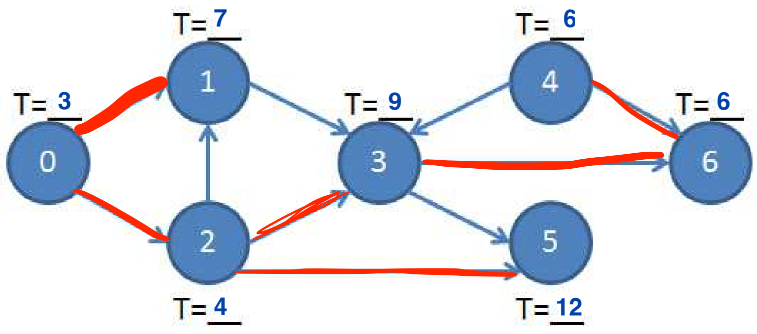


Figure 3: Write your answers directly on this DAG

C). The earliest possible completion time of this important project is thus 12 months **(1 mark)**.

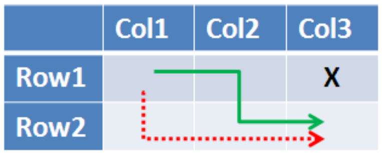
D). Which sub-project(s) that is (are) 'non-critical'? 9, 5, 6 **(2 marks)**.
A non-critical sub-project is defined as a sub-project that can be delayed (use more time) *up to two months* without changing your answer in sub-question C). above.

Q2. Walking on a Grid (7 marks)

Given a grid of N rows and M columns, count how many possible distinct paths are there from cell $(1,1)$ to cell (N,M) if some cells are **forbidden** (cannot be used in the path) and we can only go **south** (one cell below the current cell) and **east** (one cell to the right of the current cell)!

We consider two paths $p1$ and $p2$ to be distinct if there is a vertex v in $p1$ that is not used in $p2$.

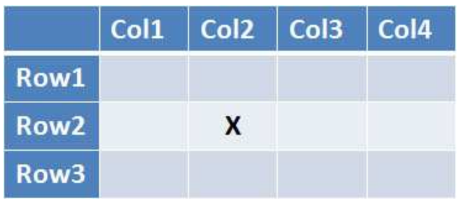
In Figure 4, we have a 2×3 grid with cell $(1,3)$ forbidden. In this example, it is clear that there are only two possible distinct paths according to the rules above: $p_1 = (1,1) \rightarrow (1,2) \rightarrow (2,2) \rightarrow (2,3)$ and $p_2 = (1,1) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (2,3)$. So, the answer is 2.



	Col1	Col2	Col3
Row1			X
Row2			

Figure 4: A Sample Grid

Now, you are given a grid of size $N = 3$ and $M = 4$ with 1 forbidden cell: $(2,2)$. How many possible distinct paths are there from $(1,1)$ to $(3,4)$ obeying the rules above?

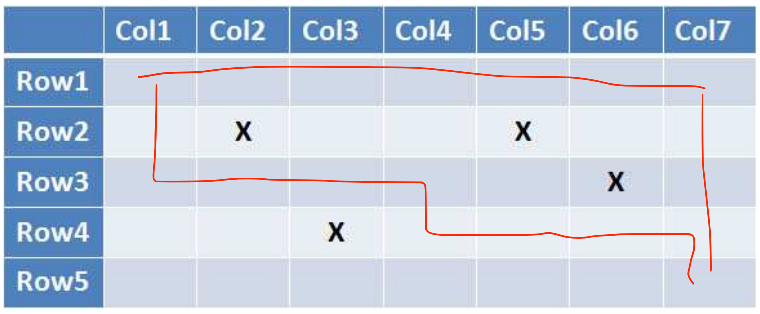


	Col1	Col2	Col3	Col4
Row1				
Row2		X		
Row3				

Figure 5: A Medium-Sized Grid

There are 2 possible distinct path(s) in this grid according to the rules above **(3 marks)**.

Next, you are given a larger grid of size $N = 5$ and $M = 7$ with 4 forbidden cells: $(2,2)$, $(2,5)$, $(3,6)$, and $(4,3)$. How many possible distinct paths are there from $(1,1)$ to $(5,7)$ obeying the rules above?



	Col1	Col2	Col3	Col4	Col5	Col6	Col7
Row1							
Row2		X			X		
Row3						X	
Row4			X				
Row5							

Figure 6: A Larger Grid

There are ____________ possible distinct path(s) in this grid according to the rules above **(4 marks)**.

Hint: Model this problem as a DAG and count the number of paths in the DAG.
Note: Please indicate your intermediate workings directly in Figure 5 and Figure 6 as partial marks *may be given* even if your final answer is wrong.

3 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, a convincing argument, or an example (to show existence). 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. There exists DAGs that only have *one unique* topological sort.

   true, DAGs with V vertices and V-1 edges

2. Dijkstra's algorithm (either the original or modified version) is the best algorithm to solve the Single-Source Shortest Paths problem given *any kind of graph*.

   false. if there are -ve weight edges without negative cycle then there are extreme cases when the same vertices are repeatedly re-processed

   for tree djikstra not best. just use dfs/bfs
   or if unweighted just do bfs

3. There exists a better algorithm than Floyd Warshall's algorithm to solve the All-Pairs Shortest Paths problem on a *weighted tree*.

   FALSE, if do krukal's/prims on every pair of vertices -> V^2 * Elog(V) which is worse than V^3

   true. if we have weighted tree run bfs/dfs with each vertex as a source so it is O(v^2)

4. Floyd Warshall's algorithm can be used to detect if a graph has negative weight cycle(s).

   true. run floyd warshall, and check main diagonal. if < inf and >= 0 it is positive cycle else if < 0 there are negative cycle

5. Floyd Warshall's algorithm *can be made* to run in $O(V^2 \log V)$ with help of a priority queue.

   false.

   we dont do any kind of ordering by weight/cost of shortest paths that have been explored so no use of pq

4 Vehicle Monitoring System (20 marks)

Up to this year (2011), Singapore has hosted a Formula One race four times (2008-2011). The race is held on a 5.067 km long street circuit, consisting of 14 left hand turns and 10 right hand turns. In the run up to the F1 race, the number of illegal night street racing activities have been on the rise. Such races consist of several rounds around a designated street circuit.

The authorities would like to deploy a new vehicle monitoring system in order to catch these illegal racers. The system consists of a number of cameras mounted along various roads. For the system to be effective, there should be *at least one camera along each of the possible circuits*.

The Singapore road system can be represented as a series of junctions and connecting bidirectional roads (see Figure 7). A possible racing circuit consists of a start junction followed by a path consisting of *three or more roads* that eventually leads back to the start junction. Each road in a racing circuit can be traversed only in one direction, and only once.

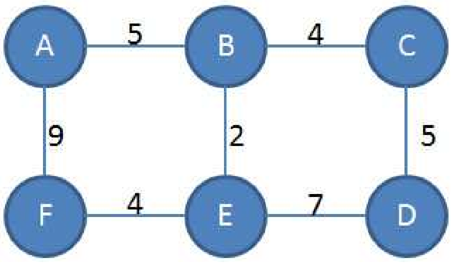


Figure 7: An example of (small) Singapore road system

You will be provided with a description of a *connected* road network to be monitored in terms of the roads and junctions. The junctions are identified by the alphabets in Figure 7. A camera can be deployed *only on the roads* (and not the junctions). There can be up to $3 \leq n \leq 10000$ junctions and $2 \leq m \leq 100000$ roads in Singapore road system. It is always possible to reach one junction from another junction in Singapore road system. The cost of deploying a camera depends on the road on which it is placed (the cost is an integer that ranges between `[1..1000]` hundreds SGD). The numbers along the roads in Figure 7 indicate the cost of deploying a camera on that road.

connect component

Your task is to write an algorithm that computes the optimal placement of the vehicle-monitoring cameras, that is, you have to select *a set of roads* that *minimizes* the total cost C of camera(s) deployment while ensuring that there is at least one camera along every possible racing circuit (i.e. loop in the road network). Please output an integer, the value of C.

In Figure 7 above, the *basic form* of the three possible racing circuits are circuit 1: {`A,B,E,F,A`}, circuit 2: {`B,C,D,E,B`}, and circuit 3: {`A,B,C,D,E,F,A`}. Note that circuit {`B,E,F,A,B`} is essentially the same as circuit 1, i.e. we can start from any junction of this circuit. We need at least two cameras: One located at road `B-E` with cost 2 (hundreds SGD) and another one at road `B-C` (or road `E-F`) with cost 4 (hundreds SGD), with total cost of $C = 6$ (hundreds SGD). With these two cameras, any illegal racing activities along any of the three possible racing circuits will be detected by the authorities. Therefore the output is $C = 6$.

Credits to Melvin Zhang Zhiyong, the author of the original version of this problem.

Sub-questions A)., B)., C)., and D). are designed to guide you to understand this problem.

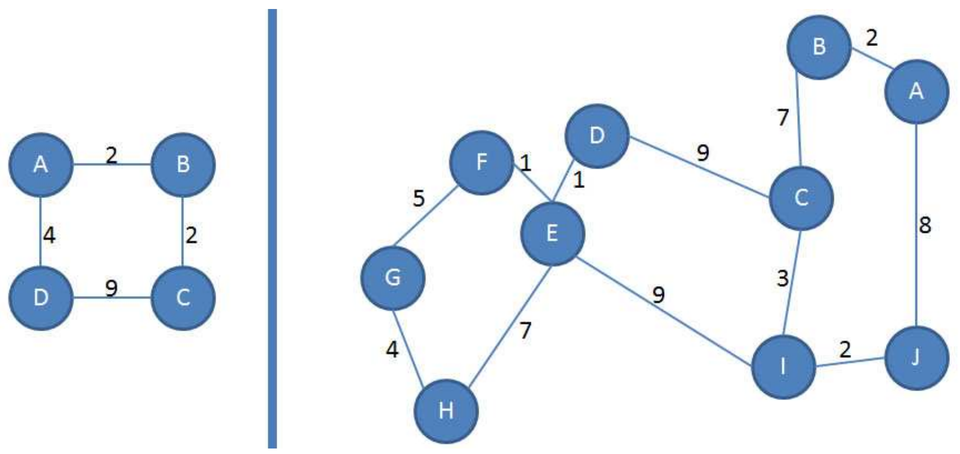


Figure 8: Left: Test case 1 (simple); Right: Test case 2 (larger, similar to the original circuit)

A). What is the answer C for the smaller test case in Figure 8 (left)? **(2 marks)**

C = 2

B). What is the answer C for the larger test case in Figure 8 (right)? **(4 marks)**
Please briefly explain your answer by listing the possible racing circuits, where is (are) the optimal camera placement(s), before you give the answer C!

possible circuits:

E,F,G,H,E
E,I,C,D,E
C,I,J,A,B,C

E,F,G,H,E,I,C,D,E

optimal placements:
E - F
E - D
either I - J or B - A

C - 1 + 1 + 2 = 4

C). Is this a Single-Source (or All-Pairs) Shortest Paths Problem? **(1 mark)**

Select either (Yes) or [✓ (No)]!

D). If you delete the road(s) that you select as the answer(s) (for placing the camera(s)) in sub-question A). and sub-question B). above, what will you get? **(1 mark)**

the maximum spanning tree of each possible circuit

E). Outline an algorithm that solves this 'Vehicle Monitoring System' problem **(10 marks)**
Please do not be too long winded although you are given nearly one page of blank space!

run kruskal's in decreasing order to select max weight edges and
find the max spanning tree. The edges which are not in
max spanning tree and the edges where cameras should be placed and add the cost of these edges

-> add the cost of the min edge in each cycle

O(E logV) cause it is still Kruskal's

F). Analyze the time complexity of your solution in sub-question E). above! **(2 marks)**

5 Cut The List (30 marks)

There is a little girl by the name of Esther who likes to play a particular game. She would start with a list L of N numbers ($1 \leq N \leq 100$, each number is a positive number less than 1000, the numbers in L are not necessarily pairwise distinct) and cut the list up into K sub-list ($1 \leq K \leq N$). Cutting a list is formally defined as follows: Suppose $L = \{l_1, l_2, \ldots, l_i, l_{i+1}, \ldots, l_N\}$. Then, cutting the list L at index i will split L into two *non-empty* sub-lists $L_1 = \{l_1, l_2, \ldots, l_i\}$ and $L_2 = \{l_{i+1}, l_{i+2}, \ldots, l_N\}$. Esther will keep cutting the list until she has K sub-lists in total.

For example, if she starts with the list $L = \{1, 2, 3, 4\}$ with $N = 4$ numbers, then it is possible to cut up L into $K = 3$ sub-lists: $\{1, 2\}, \{3\}, \{4\}$. However, it is **not** possible to cut up L into $\{1, 3\}, \{2\}, \{4\}$ as it is **impossible** to cut out $\{1, 3\}$ from L.

A). Is it possible to cut $L = \{8, 1, 5, 4, 7\}$ into $K = 3$ sub-lists: $\{8\}$, $\{1, 4, 7\}$, and $\{5\}$? **(1 mark)**

Select either (Yes) or [✓ (No)]!

B). Show *all* possible ways to cut $L = \{8, 1, 5, 4, 7\}$ into $K = 2$ sub-lists!
One of the way is already shown below! **(2 marks)**
1). $L_1 = \{8, 1\}$, $L_2 = \{5, 4, 7\}$.
2). **{8}, {1,5,4,7}**
... **{8,1,5}, {4,7}**
{8,1,5,4}, {7}

Then for each of the sub-list L_i, she would find out the difference between the maximum value (M_i) and minimum value (m_i) of L_i and call it d_i, i.e. $d_i = (M_i - m_i)$. Then she will sum out all the d_i. Her goal in this game is to *minimize* the sum i.e. $\sum_{i=1}^{K} d_i$.

For example, let the initial list $L = \{7, 2, 1, 5, 3\}$ and $K = 3$. She can cut it up into $\{7\}$, $\{2, 1\}$, and $\{5, 3\}$. So, $\sum d_i = (7 - 7) + (2 - 1) + (5 - 3) = 0 + 1 + 2 = 3$. This so happens to be the minimum possible value of $\sum d_i$ and is the answer she is looking for.

C). If Esther cuts $L = \{8, 1, 5, 4, 7\}$ into $K = 2$ sub-lists: $\{8, 1\}$, and $\{5, 4, 7\}$, what is the value of $\sum_{i=1}^{K} d_i$? **(1 mark)**

(8 - 1) + (7 - 4) = 7 + 3 = 10

Esther heard that you have taken CS2010 - Data Structures and Algorithms II and have learned 'Dynamic Programming'. She seeks your help to solve this problem. So, given the initial list L with N numbers and an integer K, please answer Esther's query as explained above, i.e. implement:

```
int Query(int[] L, int N, int K)
```

D). What is the minimum $\sum_{i=1}^{K} d_i$ that Esther can get with $L = \{8, 1, 5, 4, 7\}$ and $K = 2$? **(2 marks)**
Hint: Look at your answers for sub-question B). and C). above.

6

E). Important sub-question: What is the minimum $\sum_{i=1}^{K} d_i$ that Esther can get with $L = \{8, 1, 5, 4, 7\}$ and $K = 3$ instead? **(2 marks)**

{8}, {1,5,4}, {7} 5 - 1 = 4

F). Important sub-question: What is the minimum $\sum_{i=1}^{K} d_i$ that Esther can get with $L = \{8, 1, 5, 4, 7\}$ and $K = 4$ instead? **(2 marks)**

{8}, {1}, {5,4} ,{7}

5 - 4 = 1

G). Subtask 1, $1 \leq N \leq 100$, K is always N **(1 mark)**
That is, you will only be asked to output the value of `Query(L, N, N)`
What is the time complexity of your solution?

O(1)

H). Subtask 2, $1 \leq N \leq 100$, K is always 1 **(2 marks)**
That is, you will only be asked to output the value of `Query(L, N, 1)`
What is the time complexity of your solution?

O(1)

I). Do you observe *sub-problems* when you attempt sub-question D)., E)., and F). above?
Is the optimal solution for a sub-problem part of the solution of the original problem?
Are those sub-problems *overlapping*?
Do you notice the potential *base cases* when you attempt sub-question G). and H). above?
Can you generalize these phenomenon into *a few English sentences*? **(5 marks)**
Please do not be too long winded although you are given nearly one page of blank space!
Do not write any pseudo-code here as such pseudo-codes are only asked in sub-question J).

sub problem

base cases:

when k = N, the min diff = 0
when k = 1, min diff = N

J). Subtask 3, the general form: $1 \leq N \leq 100$, $2 \leq K \leq N-1$ **(10 marks)**
Note: You just need to outline your algorithm with pseudo codes to answer this sub-question. There are several possible solutions and marks will be given according to its performance.

split into all possible subsets of K splits -> K^2

let all K possible sublists of splits be vertices in a graph where the edges are undirected and unweighted

K). Analyze the time complexity of your solution in sub-question J). above! **(2 marks)**

Credits to Victor Loh, the author of the original version of this problem.

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

– End of this Paper –