COSC3101 Design and Analysis of Algorithms
Jeff Edmonds & Andy Mirzaian
Fall 00-01 Exam

Family Name:

Given Name:

Student #:

Email:

Section: A: Andy Mirzaian B: Jeff Edmonds

1 Short Answers	(32 marks)	
2 Topological Sort	(16 marks)	
3 Minimum Spanning Tree	(16 marks)	
4 Greedy	(19 marks)	
5 Dynamic Programming	(17 marks)	
Total	(100 marks)	

- Five problems, 180 minutes.
- You may use a crib sheet with your name and student number on it as explained in the course.
- Submit your crib sheet together with your exam booklet, when the exam is over.
- Do not use any electronic/mechanical computation devices.
- Read all questions before deciding in what order to answer them.
- You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat and organized.
- Use the back of the pages for scratch work.
- This booklet consists of 6 pages, including this cover page.

1. Short Answers

(a) (10 marks) For each function f, circle which classes it is in and then evaluate the sum.

$f(n) = 4n\log^8 n + 7\sqrt{n}$	$\mathcal{O}(n^2)$	$\Theta(n^2)$	$n^{\Theta(1)}$	$2^{\Theta(n)}$	$\Theta(\sum_{i=1}^{n} f(i)) =$
$f(n) = 4n^2\log^8 n$	$\mathcal{O}(n^2)$	$\Theta(n^2)$	$n^{\Theta(1)}$	$2^{\Theta(n)}$	$\Theta(\sum_{i=1}^{n} f(i)) =$
$f(n) = 4n^5\log^8 n$	$\mathcal{O}(n^2)$	$\Theta(n^2)$	$n^{\Theta(1)}$	$2^{\Theta(n)}$	$\Theta(\sum_{i=1}^{n} f(i)) =$
$f(n) = 3^{4n}n^8$	$\mathcal{O}(n^2)$	$\Theta(n^2)$	$n^{\Theta(1)}$	$2^{\Theta(n)}$	$\Theta(\sum_{i=1}^{n} f(i)) =$
$f(n) = 3^{4n\log n}$	$\mathcal{O}(n^2)$	$\Theta(n^2)$	$n^{\Theta(1)}$	$2^{\Theta(n)}$	$\Theta(\sum_{i=1}^{n} f(i)) =$

(b) (6 marks) Recall the recursive divide-and-conquer algorithm for multiplying two n digit integers x and y that works by solving subproblems of size $n/2$. We gave both a slow and an asymptotically faster version of this algorithm.

i. Give the recurrence relation for the slow algorithm. Solve this recurrence relation.

ii. Give the recurrence relation for the fast algorithm. Solve this recurrence relation.

(c) (4 marks) Does Dijkstra's single-source-shortest-paths algorithm work correctly even when some edges of G have negative weights? Y – N

How about Prim's MST algorithm? Y – N

(d) (6 marks) What order do the following algorithms handle the nodes?

i. Dijkstra's single-source-shortest-paths: _____.

ii. Depth First Search: _____.

iii. Breadth First Search: _____.

(e) NP-complete problems:
The problem Circuit-SAT is: given a $AND/OR/NOT$ combinatorial circuit, return whether there is an input for the circuit that outputs 1.
The problem 3-COL is: given an undirected graph, return whether there is a colouring of the nodes of the graph with three colours so that no edge is given the same colour on both of its incident nodes.

i. (2 marks) What can you say about the running time of the fastest **known** deterministic algorithm for Circuit-SAT?

ii. (4 marks) What is the relationship between the question of whether there is a deterministic polynomial-time algorithm for Circuit-SAT and whether there is such an algorithm for 3-COL?

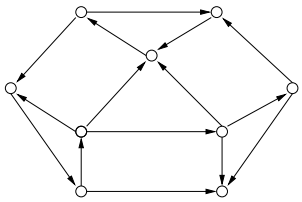
2. Topological Sort:

(a) (4 marks) A topological sort of a directed acyclic graph $G = (V, E)$ and a total order of a partial order are the same. Define this concept.

(b) (4 marks) Give a brief description of the linear-time topological sort algorithm given in class.

(c) (4 marks) A *Hamiltonian path* in a directed graph $G = (V, E)$ is a directed simple path that passes through each vertex of G exactly once. (Note that we are referring to a Hamiltonian *path*, not a cycle.) In general, the problem of deciding whether a given digraph has a Hamiltonian path is known to be NP-complete.

Does the graph shown below contain a Hamiltonian path? If yes, show one by highlighting the edges on the path.

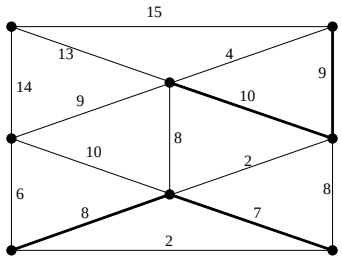


(d) (4 marks) If the graph G happens to be a DAG, how does finding a topological sort of G help in finding a Hamiltonian path of G?

3. Minimum Spanning Tree:

This question concerns a modified version of the minimum spanning tree (MST) problem. Let $G = (V, E)$ be a weighted connected undirected graph, and let F be a subgraph of G that is a forest (i.e., F does not contain any cycles). A *constrained spanning tree* of G with respect to F, denoted $CST(G, F)$ for short, is a spanning tree of G that contains all the edges of F. A *constrained minimum spanning tree* of G with respect to F, denoted $CMST(G, F)$ for short, is a $CST(G, F)$ that has minimum total weight possible.

(a) (3 marks) Show $CMST(G, F)$ of the graph G below, where F is indicated by heavy edges.



(b) (6 marks) Describe in a few sentences how Kruskal's MST algorithm can be modified to compute $CMST(G, F)$ efficiently.

(c) (7 marks) Describe in a few sentences how the loop invariant for this algorithm would differ from the loop invariant for Kruskal's MST algorithm.

4. Greedy:

We are given n data items in arbitrary order, each a real number between 0 and 1. The problem is to put these n items in a **minimum** number of boxes so that:
(a) Each box contains no more than two items, and
(b) The sum of the items in each box is no more than 1.

(a) (3 marks) Give an optimal solution for the following instance: $\{.05, .35, .36, .45, .53, .62, .82, .91\}$.

(b) (7 marks) Describe an efficient greedy algorithm for this problem.

(c) (7 marks) State a loop invariant for your algorithm and explain why your algorithm does what it is supposed to.

(d) (2 marks) Analyze the running time of your algorithm.

5. Dynamic Programming:

Given the n-by-n boolean adjacency matrix of a directed graph, compute the two n-by-n boolean matrices P and Q such that P_{ij} is true iff there is a path of **odd** length from vertex i to j, and Q_{ij} is true iff there is a path of **even** length from vertex i to j, where the length of a path is the number of edges on it.

(a) (7 marks) Describe an efficient algorithm for this problem. [Hint: use ideas similar to Floyd-Warshall's algorithm.]

(b) (5 marks) Explain the correctness of your algorithm.

(c) (4 marks) Analyze the time complexity of your algorithm.