CS3230 Design & Analysis of Algorithms

Final Examination

National University of Singapore
School of Computing
AY2008/09, Semester 2

Apr/May 2009 **Time Allowed: 2 hours**

Matriculation Number: | | | | | | | | | |

Instructions

1. The examination is **closed book**, but you may have one page of handwritten information. **No other aids are allowed.**
2. The examination paper consists of **11** pages. Make sure that you have all the pages.
3. Read **all** questions before attempting them.
4. Answer all questions within the space provided in this booklet. If you use the back of pages, indicate so clearly.
5. You may quote a well-known algorithm or an algorithm covered in the class without explaining the details of the algorithm, but you must justify why the algorithm is applicable.
6. If a problem asks for "an algorithm in ... time", partial credits are given for slower algorithms. One good way of proceeding is to come up with a working algorithm first and then improve it to achieve the specified running time.
7. Write up your solutions neatly. Graders can only award points if they understand your handwriting. You may use pencils.

Question	Max Points	Points
1	30	
2	30	
3	10	
4	10	
5	10	
6	10	
total	100	

Question 2 (30 points)
For each question below, give the **answer** and a **short justification**.

2.1 When is merge sort preferred over randomized quicksort and vice versa?

2.2 For solving the shortest path problem, when is Dijkstra's algorithm preferred over Bellman-Ford algorithm and vice versa?

2.3 Consider the password checking algorithm discussed in the class and the tutorial. Suppose that the checking algorithm uses 3 hash functions and a 1000-bit table for each hash function and that that there are 100 illegal passwords in the dictionary. What is the probability for a new password to cause an accidental rejection? Justify your answer.

Hint. You may use the formula $(1-x)^n \approx 1 - nx$ for positive integer n and sufficiently small positive real number x.

2.4 The table below contains the frequency counts for each character in a piece of text. Draw a Huffman coding tree for this text. What is the length of the text after Huffman encoding?

character	I	U	B	S	C	H	M	P
frequency count	75	200	25	275	50	100	25	250

2.5 Find an optimal solution to the 0-1 knapsack (knapsack without repetition) problem below. The knapsack has total capacity $C = 8$. Which items are chosen by the optimal solution? Show the dynamic programming table to justify your answer.

i	1	2	3	4
w_i	2	3	4	5
p_i	3	4	5	6

Question 3 (10 points)
Let G be a weighted undirected graph representing a communication network. The edge weight t_{ij} between two vertices i and j of G is the amount of time needed to transmit a signal between i and j. One goal in planning signal transmission is to minimize the transmission time. Furthermore, the communication network is noisy. Every time a signal is transmitted along an edge from one vertex to another, it may get corrupted or lost. So a second goal is to reduce the number of "hops", or edges along a transmission path.

(a) To transmit a signal from a vertex s to a vertex g of G, describe an efficient algorithm that finds a shortest path in G between s and g using at most k edge, where k is a given constant.

(b) Analyze the running time of your algorithm and express your answer using the O-notation.

Question 4 (10 points)
At the Perfect Programming Company, the productivity of a programmer is measure by the number of lines of correct code produced per day. Programmers work in **pairs** in order to ensure the quality of the code produced. The productivity of each pair of programmers is the productivity of the slower programmer.

(*a*) For a team of an even number of programmers, give an efficient algorithm for pairing them up in order to maximize the productivity of the team. The productivity of a team is the sum of the productivity of each pairs of programmers in the team.

(*b*) Analyze the running time of your algorithm and express your answer using the O-notation.

Question 5 (10 points)
Let S be a set of n points. Let $P = \{S_1, S_2, \ldots, S_m\}$ be a collection of point sets, where each point set S_i contains a subset of points in S. In the *point set union* problem, we want to find the smallest subset P' of P such that the *union* of the point sets in P' is exactly S.

(a) State precisely the **decision problem** corresponding to the point set union problem.

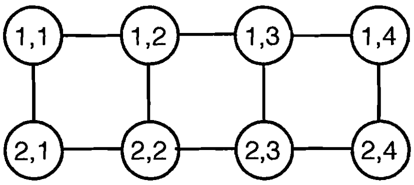
(b) Prove that the point set union problem is in NP.

(c) Prove that the point set union problem is NP-complete.

Hint. Use reduction from the dominating set problem. The dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex in $V - D$ is connected to a vertex in D by an edge in E. The dominating set problem is to determine whether G contains a dominating set of size less than K, for some given positive constant K.

BLANK PAGE

Question 6 (10 points)
In a *weighted 2-tracked* graph T, every vertex is labeled as (i, j) for $i = 1, 2$ and $j = 1, 2, \ldots, n$. There is an edge between (i, j) and $(i, j + 1)$ for $i = 1, 2$ and $j = 1, 2, \ldots, n - 1$. There is also an edge between $(1, j)$ and $(2, j)$ for all $j = 1, 2, \ldots, n$. The weight of a vertex (i, j) is given by w_{ij} Shown below is an example of a 2-tracked graph for $n = 4$.



An independent set I of T is a subset of vertices of T such that there is no edge in T between any two vertices in I. The total weight of I is the sum of the weights of all vertices in I.

(*a*) Give an algorithm that finds an independent set with the maximum total weight for a 2-tracked graph. Make your algorithm as efficient as possible.

(*b*) Justify that your algorithm indeed finds an optimal solution, i.e., an independent set with the **maximum** total weight.

(*c*) Analyze the running time of your algorithm.

BLANK PAGE

BLANK PAGE

END OF PAPER