CS3230 Design & Analysis of Algorithms

Final Examination

National University of Singapore
School of Computing
AY2010/11, Semester 2

Apr 2011 **Time Allowed: 2 hours**

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

Instructions

1. The examination is **closed book**, but you may have one page of handwritten notes. **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. 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 better 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	6	
2	10	
3	10	
4	12	
5	12	
6	12	
7	12	
8	16	
total	90	

Question 1 (6 points)
For each of the following statements, answer whether it is **true** or **false**. Give **short** justifications to your answers.

(*a*) $2^{n+1} = O(2^n)$

(*b*) $2^{2n} = O(2^n)$

Question 2 (10 points)
Give a tight asymptotic upper bound (in O-notation) for $T(n)$ in the recurrence relation below and prove that it is an upper bound by induction. You may assume that the base cases $T(0), T(1), \ldots$ are all constant.

$$T(n) = T(n/3) + T(2n/3) + 2n^2.$$

Question 3 (10 points)
The following algorithm sorts an array $A[1:n]$ of n numbers. It is similar to the standard insertion sort, but uses binary search to determine where to insert an element $A[i+1]$ into $A[1:i]$.

---

for i from 2 to n **do**
  p :=BinarySearch($A[1:i-1], A[i]$)
  for j from i to $p+1$ **do**
    Swap($A[j], A[j-1]$)

Is the worst-case asymptotic running time of this algorithm better than that of the standard insertion sort? Why?

Question 4 (12 points)
The table below contains the frequency counts for each character in a piece of text.

character	I	B	S	C	H	M
frequency count	5	7	10	15	20	45

(*a*) Draw a Huffman coding tree for this text.

(*b*) What is the compression ratio for your Huffman coding of the text, compared with the standard ASCII coding? Assume that the ASCII coding uses a fixed-length format of 8 bits per character.

Question 5 (12 points)
Let $G = (V, E)$ be an weighted, undirected graph. For each of the following statements, answer whether it is **true** or **false**. If true, give a **short** justification (formal proof not required). If false, give a counter-example.

(*a*) If G has more than $|V| - 1$ edges and there is a unique heaviest edge e, then e cannot be part of any MST.

(*b*) If e is any edge of minimum weight in G, then e must be part of some MST.

(*c*) Suppose that we create a new graph G' by increasing the edge weight of every edge in G by 1. A shortest path between two nodes in G is also a shortest path between the two corresponding nodes in G'.

(*d*) The shortest path between two nodes is necessarily path of some MST.

Question 6 (12 points)
For each of the following statements, answer whether it is **true** or **false**. Give **short** justifications to your answers.

(*a*) It is possible that P = NP.

(*b*) It is possible that P = NP and NP = EXP.

(*c*) If any NP-complete problem lies in P, then every NP-complete problem lies in P.

(*d*) A formula is in 3-DNF if it is a disjunction of conjunctive clauses, each containing exactly three literals, for example, $(x_1 \wedge \overline{x_1} \wedge x_4) \vee (x_2 \wedge x_3 \wedge \overline{x_4}) \vee (x_1 \wedge x_2 \wedge x_4)$. Note that 3-DNF is different from 3-CNF, which we studied in the class. 3-CNF is a conjunction of disjunctive clauses. The problem of deciding whether there is an assignment of truth values to the variables of a 3-DNF formula to make it **false** is NP-complete.

Question 7 (12 points)
A string-processing language offers a primitive operation which splits a string $s[1:n]$ into two pieces. Since this operation involves copying the original string, it takes n units of times for a string of length n, regardless of the location of the cut. Now suppose that we want to break a string into many pieces. The order in which the breaks are made can affect the total running time. For example, if we want to cut a 20-character string at position 3 and 10, then making the first cut at position 3 incurs a total cost of 20+17=37, while doing position 10 first has a better cost of 20+10=30.

(*a*) Given the locations of m cuts $c_1, c_2, \ldots, c_m$ in a string of $s[1:n]$, give a dynamic programming algorithm that finds the minimum cost of breaking the string into $m+1$ pieces. Make sure that you include the recurrence relation when you describe your dynamic programming algorithm.

(*b*) What is the worst-case asymptotic running time (O-notation) of your algorithm? Justify your answer.

Question 8 (16 points)
We are given a set of n activities, each having a start time s_i and an end time e_i for $i = 1, 2, \ldots, n$. We wish to participate in as many activities as possible; however, at any point in time, we can only be present in a single activity. Formally, we wish to select the largest subset S of activities such that no two activities in S clashes in time.

Here is a greedy algorithm for solving the activity selection problem. For each activity a_i, count the number of other activities that clashes with a_i. Sort the activities in increasing number of clashes and store them in a list A. We first choose the activity a_1 that has the smallest number of clashes and remove from A the activity a_1 and all activities that clash with a_1. We then repeatedly apply the same procedure to the list of remaining activities in A, until A is empty.

(*a*) Assume that each activity clashes with at most 10 other activities. Describe an efficient algorithm that determines the number of clashes for each activity in a (unsorted) set of n activities. Analyze the worst-case asymptotic running time (O-notation) of your algorithm. Justify your answer.

(*b*) Does the above greedy algorithm produce an optimal solution to the activity selection problem, *i.e.*, a solution containing a maximum number of activities? If yes, justify your answer. If no, give a counter-example.

BLANK PAGE

BLANK PAGE

END OF PAPER