CSE 3101 Final exam
Summer 2007
August 7, 2007
Instructor: S. Datta

Name (LAST, FIRST): ______________________________

Student number: ______________________________

Instructions:

1. If you have not done so, put away all books, papers, cell phones and pagers. Write your name and student number NOW!
2. Check that this examination has 13 pages. There should be 6 questions together worth 120 points.
3. Feel free to detach the last sheet of the exam – it contains formulae that you may refer to.
4. If you use an algorithm from the book to solve a problem, you need NOT describe or analyze it. E.g., you can state that you are using the Floyd-Warshall algorithm for some problem and that its complexity is $\Theta(n^3)$ as proved in the book.
5. Unless otherwise specified, a formal proof of correctness of any algorithm you design is not required, but you MUST provide arguments and/or intuition to show that your algorithm is correct.
6. In case you use the Master Theorem, **please prove that it is applicable**, i.e., the preconditions hold.
7. You have 180 minutes to complete the exam. Use your time judiciously.
8. Show all your work. Partial credit is possible for an answer, but only if you show the intermediate steps in obtaining the answer.
9. If you need to make an assumption to answer a question, please state the assumption clearly.
10. Points will be deducted for vague and ambiguous answers.
11. Your answers MUST be LEGIBLE.
12. Good luck!

Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

DO NOT WRITE ANYTHING ON THIS PAGE.

Q	part a	part b	part c	TOTAL
1				
2				
3				
4				
5				
6				

Aggregate score =

1. (a) (5 points) With a small example illustrate how radix sort really needs a stable sort as a subroutine in order to work correctly.

   (b) (5 points) Suppose $T(1) = 1$ and for $n > 1$,

   $$T(n) = 4T\left(\frac{n}{2}\right) + n^2\sqrt{n}.$$

   Solve this recurrence using the Master Theorem. You are not permitted to use results or theorems from any book other than our text for this question.

(c) (10 points) Given n files of lengths $m_1, m_2, \ldots, m_n$, the optimal tape storage problem is to find which order is the best to store them on a tape, assuming that each retrieval takes time equal to the length of the preceding files in the tape plus the length of the retrieved file, and that files are to be retrieved in reverse order (i.e. the last file is retrieved first). Design a greedy algorithm to put all of the files on tape and prove the greedy choice property holds.

2. (20 points) Given a set of n numbers, we wish to find the i largest in sorted order using a comparison-based algorithm. Find the algorithm that implements each of the following methods with the best worst-case running time, and analyze the running times of the algorithms in terms of n and i.

   (a) (4 points) Sort the numbers, and list the i largest.

   (b) (8 points) Build a max-priority queue from the numbers and call EXTRACT-MAX i times.

   (c) (8 points) Use an order-statistic algorithm to find the i^{th} largest number, partition around that number and sort the i largest numbers. When is this better than (a) above?

3. (20 points) You are given a rectangular m x n matrix with integer entries. Design a dynamic programming algorithm to compute the biggest square sub-matrix that has all non-zero entries. The sub-matrix has to be contiguous (i.e. no gaps are allowed). Analyze the running time and the space required by your algorithm.

4. (a) (8 points) Let T be a minimum spanning tree of the graph $G = (V, E)$ and let V' be a subset of V. Let T' be the subgraph of T induced by V'. Show if T' is connected, then T' is a minimum spanning tree of G'.

   (b) (12 points) Given a graph G and a minimum spanning tree T, suppose we decrease the weight of one of the edges not in T. Give an algorithm for finding the minimum spanning tree in the modified graph. You get no credit if your algorithm has the same asymptotic running time as Prim's or Kruskal's algorithm.

5. (20 points) You are given a sorted array $A[1 \ldots n]$ of floating point numbers. Define $A[i+1] - A[i] (1 \leq i \leq n-1)$ to be the i^{th} gap of A. The average gap of A is the average of these $n-1$ gaps.

   (a) (5 points) Design a constant-time algorithm for finding the *average* gap of A. Argue that your algorithm is correct.

   (b) (9 points) Find a sub-linear time algorithm to find a gap of A that is no larger than the average gap. You will get no credit for the trivial linear-time algorithm that looks at each gap and computes the minimum. Analyze the running time of your algorithm.

(c) (6 points) Compute a Huffman code for a file with the following 8 letters. You must show the steps involved.
a: 1 occurrences
b: 1 occurrences
c: 2 occurrences
d: 3 occurrences
e: 5 occurrences
f: 8 occurrences
g: 13 occurrences
h: 21 occurrences

6. (20 points) Bonnie and Clyde have just robbed a bank. They have a bag of money and want to divide it up. For each of the following scenarios, either give a polynomial time algorithm (and justification about its correctness and running time) or prove the problem is NP-complete. The input in each case is a list of the n items in the bag, along with the value of each.

   (a) (4 points) Explain in your own words the phenomenon of NP-completeness and its implication on the tractability of problems.

   (b) (8 points) There are n checks, each made out to "cash" (i.e. either Bonnie or Clyde can cash them). They wish to divide the checks so that they each get the exact same amount of money.

(c) (8 points) There are n checks, each made out to "cash" but Bonnie and Clyde are willing to accept a split in which the difference is no more than 100 dollars. Also indicate how to detect if such a split is not possible.

Use this page if you need extra space. Mark the question number clearly.

List of useful formulae and facts.

1. $\log ab = \log a + \log b$
2. $\log a^n = n \log a$
3. $\log_b a = \log a / \log b$
4. $n = 2^{\lg n}$
5. $\frac{dx^k}{dx} = kx^{k-1}$
6. $$\frac{d}{dx}\frac{u}{v} = \frac{v\frac{du}{dx} - u\frac{dv}{dx}}{v^2}$$
7. $$\sum_{i=1}^{n} i = n(n+1)/2$$
8. $$\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$$
9. $$\sum_{i=1}^{n} i^3 = n^2(n+1)^2/4$$
10. $$\sum_{i=0}^{n} x^k = \frac{x^{n+1}-1}{x-1}$$
11. $$\sum_{i=0}^{n} x^k = \frac{1}{1-x}, \text{ when } |x| < 1.$$
12. For a monotonically increasing function $f(x)$,
$$\int_{m-1}^{n} f(x)dx \le \sum_{k=m}^{n} f(k) \le \int_{m}^{n+1} f(x)dx$$
13. Master Theorem : Let $a \ge 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence
$T(n) = aT(n/b) + f(n)$,
where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$/ Then $T(n)$ can be bounded asymptotically as follows.
    (a) If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
    (b) If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
    (c) If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \le cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.