CS-170 Exam 1 February 22, 1995
David Wolfe

You should not need to write any code for this exam. Please make your answers as brief and as clear as possible. I highly recommend crossing out mistakes with a few strokes of the pen rather than erasing the work completely in case it is worth partial credit.

The exam consists of six questions, which appear first on the next two pages, and are repeated one-to-a-page on the following pages. Please leave the exam stapled. You can look at the solution set (with the questions repeated) when you leave; you'll get copies in a few days after all make-up exams have occurred.

The estimated time per question is my personal estimate, assuming you don't get too stuck. Surely, most students will get stuck on at least one problem, so don't be concerned if you don't get to the last question. I suspect 80/120 will be a respectable score worth, perhaps, a $B+$ or better.

1.	10 minutes?	/20
2.	10 minutes?	/20
3.	2 minutes?	/10
4.	20 minutes?	/30
5.	20 minutes?	/30
6.	20 minutes?	/20
Total	82 minutes?	/120

1. (20 points) Ms. Jones has an algorithm which she proved (correctly) runs in time $O(2^n)$. She coded the algorithm correctly in C, yet she was surprised when it ran quickly on inputs of size up to a million. What are at least **two** possible explanations of this behavior? (Two rather different plausible explanations will receive full credit. If you give additional explanations, you may lose points for unplausible ones. Be brief in your explanations.)

2. (20 points) Consider the following recurrence, where $a + b < 1$ and $a, b \geq 0$:

$$\begin{aligned} T(n) &= 0 && \text{, if } n \leq 1 \\ T(n) &= T(an) + T(bn) + n && \text{, if } n > 1 \end{aligned}$$

Prove $T(n) = O(n)$.

3. (10 points) For which type of input data is Huffman coding more likely to achieve better compression: random characters or English text. Why? (A one sentence explanation is sufficient.)

4 & 5. **The following text refers to the next two problems:**

You are going on a long trip. You start along the road at mile post 0. Along the road that you will travel there are n hotels at mile posts $a_1 < a_2 < \cdots < a_n$ (a_i is measured from the start of the trip). When you choose to stop, you must stop at one of these hotels (but you can choose which hotels you want to stop at). You must stop at the last hotel, which is your destination. You decide that the ideal distance to travel per day is 300 miles (plus or minus a few is ok); so if x is the number of miles traveled in one day, you assign a cost function of $(300 - x)^2$ that you want to minimize.

Example:

Suppose $n = 4$ and hotels are at miles $a_1 = 250$, $a_2 = 310$, $a_3 = 550$ and $a_4 = 590$. You must stop at mile 590 (since it's the last), and if you also stop at mile 310, you would incur a total cost of

$$(300 - 310)^2 + (300 - (590 - 310))^2 = 100 + 400 = 500$$

In fact, this is the best possible schedule you could arrange.

4. (30 points) Design a dynamic programming algorithm to determine your total cost when you choose to stop at those hotels which minimize the total cost function. (Hint: Let C_i be the minimum cost if you were to start at mile 0 and complete your trip at hotel i.)

   (a) (15 points) Give a recursive rule for computing C_i.

   (b) (10 points) Explain how you can use dynamic programming to compute C_i in polynomial time.

   (c) (5 points) Analyze the running time of your dynamic programming algorithm.

5. (30 points) Propose and discuss a greedy heuristic for finding which hotels to stop at. (Your heuristic need not actually minimize the total cost function.)

   (a) (10 points) Propose a reasonable linear-time greedy heuristic for the problem.

   (b) (15 points) Either prove your greedy heuristic minimizes the total cost function, or give a counterexample demonstrating how your heuristic may fail to give the optimum set of hotels to stop at.

   (c) (5 points) Analyze the running time of your heuristic.

6. (20 points) Consider the problem of finding the largest, second largest and third largest from a collection of 8 elements using comparisons. (The 3 largest elements should be reported in decreasing order.) You may assume the elements are distinct.

   Let u be an upper bound on the number of comparisons required to solve this problem.. (The value of u would be a number, like "17", since there is no parameter such as n in the problem.) Let l be a lower bound on the number of comparisons required.

   (a) (2 points) Is it possible that $u < l$? What would you conclude?

   (b) (2 points) Is it possible that $u > l$? What would you conclude?

   (c) (4 points) Is it possible that $u = l$? What would you conclude?

   (d) (6 points) Find an upper bound, u, on the number of comparisons required.

   (e) (6 points) Find a lower bound, l, on the number of comparisons required. (I recommend an information theoretic bound, since it's simplest.)

   (You'll certainly receive full credit for parts (d) and (e) if $|u - l| \leq 3$. A little worse should be ok, too.)