CSCI E-124–Spring, 2003 Homework 3 Out: March 11, 2004
Due: March 26, 2004

For all homework problems where you are asked to give an algorithm, you must prove the correctness of your algorithm and establish the best upper bound that you can give for the running time. You should always write a clear informal description of your algorithm in English. You may also write pseudocode if you feel your informal explanation requires more precision and detail. As always, try to make your answers as clear and concise as possible.

1. Prove that there is a unique minimum spanning tree on a connected undirected graph when the edge weights are unique.

2. Give a family of set cover problems where the set to be covered has n elements, the minimum set cover is size $k = 3$, and the greedy algorithm returns a cover of size $\Omega(\log n)$. That is, you should give a description of a set cover problem that works for a set of values of n that grows to infinity – you might begin, for example, by saying, "Consider the set $X = \{1, 2, \ldots, 2^b\}$ for any $b \geq 10$, and consider subsets of X of the form...", and finish by saying "We have shown that for the example above, the set cover returned by the greedy algorithm is of size $b = \Omega(\log n)$."

3. Consider the following scheduling problem: we have two machines, and a set of jobs $j_1, j_2, j_3, \ldots, j_n$ that we have to process one at a time. To process a job, we place it on a machine. Each job j_i has an associated running time r_i. The load on the machine is the sum of the running times of the jobs placed on it. The goal is to minimize the completion time, which is the maximum load over all machines.

   Suppose we adopt a greedy algorithm: each job j_i is put on the machine with the minimum load after the first $i-1$ jobs. (Ties can be broken arbitrarily.) Show that this strategy yields a completion time within a factor of 3/2 of the best possible placement of jobs. (Hint: Think of the best possible placement of jobs. Even for the best placement, the completion time is at least as big as the biggest job, and at least as big as half the sum of the jobs.) Give an example where a factor of 3/2 is achieved.

   Suppose now instead of 2 machines we have m machines. What is the performance of the greedy solution, compared to the optimal, as a function of m? Give a *family* of examples (that is, one for each m – if they are very similar, it will be easier to write down!) where the factor separating the optimal and the greedy solutions is as large as you can make it.

4. Consider an algorithm for integer multiplication of two n-digit numbers where each number is split into three parts, each with $n/3$ digits.

   (a) Design and explain such an algorithm, similar to the integer multiplication algorithm presented in class. Your algorithm should describe how to multiply the two integers using only six multiplications on the smaller parts (instead of the straightforward nine).

   (b) Determine the asymptotic running time of your algorithm. Would you rather split it into two parts or three parts?

   (c) Suppose you could use only five multiplications instead of six. Then determine the asymptotic running time of such an algorithm. In this case, would you rather split it into two parts or three parts?

   (d) Challenge problem– this is optional, and not worth any points. Solving it will simply impress the instructors. Find a way to use only five multiplications on the smaller parts. Can you generalize to when the two initial n-digit numbers are split into k parts, each with n/k digits? Hint: aslo consider multiplication by a constant, such as 2; note that multiplying by 2 does not count as one of the five multiplications. You may need to use some linear algebra.

5. Suppose we have an array A containing n numbers, some of which may be negative. We wish to find indices i and j so that

$$\sum_{k=i}^{j} A[k]$$

is maximized.

   (a) Give a trivial $O(n^2)$ algorithm.

   (b) Show how using divide and conquer, we can obtain an $O(n \log n)$ algorithm. (Hint: consider carefully what information you want your divide and conquer algorithm to compute.)

   (c) Think about the solution to your divide and conquer algorithm. Now try to come up with an $O(n)$ algorithm for the problem.

6. A challenge that arises in databases is how to summarize data in easy-to-display formats, such as a histogram. A problem in this context is the minimal imbalance problem. Again suppose we have an array A containing n numbers, this time all positive. Consider k indices $j_1, j_2, \ldots j_k$ that partition the array into $k+1$ subarrays $A[1, j_1], A[j_1+1, j_2], \ldots, A[j_k+1, n]$. The weight $w(i)$ of the ith subarray is the sum of its entries. The *imbalance* of the partition is

$$\max_i |w(i) - (\sum_{\ell=1}^{n} A[\ell])/(k+1)|.$$

   That is, the imbalance is the maximum deviation any partition has from the average size.

   Give an algorithm for determining the partition with the minimal imbalance. (This corresponds to finding a histogram with $k+1$ bars as close to equal as possible, in some sense.)

   Explain how your algorithm would change if the imbalance was redefined to be

$$\sum_i |w(i) - (\sum_{\ell=1}^{n} A[\ell])/(k+1)|.$$

7. Suppose we want to print a paragraph neatly on a page. The paragraph consists of words of length $\ell_1, \ell_2, \ldots, \ell_n$. The maximum line length is M. (Assume $\ell_i \leq M$ always.) We define a measure of neatness as follows. The extra space on a line (using one space between words) containing words ℓ_i through ℓ_j is $M - j + i - \sum_{k=i}^{j} \ell_k$. The penalty is the sum over all lines **except the last** of the **cube** of the extra space at the end of the line. This has been proven to be an effective heuristic for neatness in practice. Find a dynamic programming algorithm to determine the neatest way to print a paragraph. Of course you should provide a recursive definition of the value of the optimal solution that motivates your algorithm.

   For this problem, besides explaining/proving your algorithms as for other problems on the set, you should also code up your algorithm to print an optimal division of words into lines. The output should be the text split into lines appropriately, and the numerical value of the penalty. You can use any coding language you wish. You should assume that a *word* in this context is any contiguous sequence of characters not including blank spaces.

   After coding your algorithm, determine the minimal penalty for the following review of the Season 1 Buffy DVD, apparently written by Ryan Crackell for the Apollo Guid, for the cases where $M = 40$ and $M = 72$. We will try to put the text of the review on the class page as well.

   Buffy the Vampire Slayer fans are sure to get their fix with the DVD release of the show's first season. The three-disc collection includes all 12 episodes as well as many extras. There is a collection of interviews by the show's creator Joss Whedon in which he explains his inspiration for the show as well as comments on the various cast members. Much of the same material is covered in more depth with Whedon's commentary track for the show's first two episodes that make up the Buffy the Vampire Slayer pilot. The most interesting points of Whedon's commentary come from his explanation of the learning curve he encountered shifting from

blockbuster films like Toy Story to a much lower-budget television series. The first disc also includes a short interview with David Boreanaz who plays the role of Angel. Other features include the script for the pilot episodes, a trailer, a large photo gallery of publicity shots and in-depth biographies of Whedon and several of the show's stars, including Sarah Michelle Gellar, Alyson Hannigan and Nicholas Brendon.