due Wednesday, 22 October 2014, at 5:00 PM

[Total points: 33]

Part 1: Dynamic Programming. *In your solution to any of these problems you are required to give a recursive formula that indicates how a problem instance reduces to one or more smaller instances. This accounts for 3 of the 6 points. There are 2 points for the table filling algorithm (really a program) and its time bound, and 1 point for the algorithm that retrieves the actual choices.*

1. [worth 6 points]

[From Baase and Van Gelder, Computer Algorithms, Addison Wesley, 2000]

Suppose you have three strings $X = x_1 \cdots x_m$, $Y = y_1 \cdots y_n$, and $Z = z_1 \cdots z_{m+n}$. String Z is said to be a *shuffle* of X and Y if it can be formed by interspersing characters of X and Y while maintaining the original order of each string. For example `blmuondeasy` and `mbolnudeasy` are shuffles of `monday` and `blues`, but `blmuondeysa` is not (the `a` and `y` of `monday` are in the worng order). Give an algorithm that determines, given inputs X, Y, and Z, whether or not Z is a shuffle of X and Y. Illustrate how your algorithm works on $X =$ `my`, $Y =$ `dog` and $Z =$ `domyg`; also do this with $Z =$ `dmogy`. What is the asymptotic runtime of your algorithm as a function of m and n?

2. [worth 6 points]

[Also from Baase and Van Gelder]

Suppose you have inherited the rights to 500 previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of 5 CD's (numbered 1 through 5) with a selection of these songs. Each disk holds a maximum of 60 minutes of music and each song must appear in its entirety on one disk. Since you are a classical music fan and have no way of judging the artistic merits of the songs, you decide to use the following criteria: the songs will be recorded in order by the date they were written, and the number of songs included will be maximized. Suppose you have a list $\ell_1, \ell_2, \ldots, \ell_{500}$ of the lengths of the songs, in order by the date they were written (no song is more than 60 minutes long). Give an algorithm to determine the maximum number of songs that can be included using the given criteria. Hint: Let $T[i][j]$ be the minimum amount of time needed for recording any i songs from among the first j songs. You should interpret T to include blank time at the end of a completed disk. For example, if a selection of songs uses all of the first disk plus 15 minutes of the second disk, the time for that selection should be 75 minutes even if there is blank space at the end of the first disk. Generalize your solution to a situation where there are m CD's, n songs, and a capacity c for each of the CD's. What is the asymptotic runtime in terms of m and n (treating c as a constant)?

3. [worth 7 points – one extra because the time bound requires additional analysis]

Suppose n jobs, numbered $1, \ldots, n$ are to be scheduled on a single machine. Job j takes t_j time units, has an integer deadline d_j, and a profit p_j. The machine can only process one job at a time and each job must be processed without interruption (for its t_j time units). A job j that finishes before its deadline d_j receives profit p_j, while a tardy job receives no profit (and might as well not have been scheduled). Give an efficient algorithm to find a schedule that maximizes total profit. You may assume that the t_j are integers in the range $[1, n]$. You should be able to come up with an $\mathrm{O}(n^3)$ algorithm if you make the right observation about the maximum relevant deadline.

Part 2: Greedy Algorithms. *For any greedy algorithm you give you must be clear about how a choice reduces an instance of the problem to a smaller one and prove the greedy choice property.*

4. [worth 6 points]

Problem 16-2(a) on page 447. Scheduling to minimize average completion time.

5. [worth 8 points]

[from Brassard and Bratley, Fundamentals of Algorithmics, Prentice Hall, 1996]

Suppose n programs are stored on a magnetic tape (remember those?). Let $s(i)$ be the size of program i and $f(i)$ the frequency of use for program i. The time it takes to access a program is proportional to the sizes of all programs on the tape up to and including it. The goal is to minimize the total access time for all programs by finding an ordering of the programs on the tape. More formally, let π be a permutation of $1, \ldots, n$ that describes the ordering: so if $\pi(1) = i$ then program i is the first on the tape. The total cost is:

$$\sum_{i=1}^{n} f(\pi(i)) \sum_{k=1}^{i} s(\pi(k))$$

There are three possible greedy algorithms for minimizing this cost:

1. select programs in order of increasing $s(i)$ (nondecreasing if we allow for equal sizes – I use increasing to avoid confusion)
2. select programs in order of decreasing $f(i)$
3. select programs in order of decreasing $f(i)/s(i)$

For each algorithm, either find a counterexample to show that it is not optimal or prove that it always gives an optimum solution.