Greedy algorithms/dynamic programming (part 1 of CSC 282),
http://www.cs.rochester.edu/~stefanko/Teaching/14CS282

1 Schedule

Homework is due **Thursday, Sep. 11**.

"Exam" problem session is in CSB 209, 6pm-7pm on **Wednesday, Sep. 17**.

EXAM #1 will be on **Thursday, Sep. 18**.

2 Homework - solve and turn in

1.1 (**due Sep. 11**) We are given an $n \times n$ array A of zeros and ones. We want to find the size of the largest contiguous all-ones square. Give an $O(n^2)$-time algorithm for the problem.

1.2 (**due Sep. 11**) We are given n positive numbers $a_1, \ldots, a_n$. The goal is to select a subset of the numbers with maximal sum and such that no three consecutive numbers are selected. Here are three example inputs together with optimal solutions (the numbers in boxes are selected):

$$\boxed{5}\,\boxed{5}\,8\,\boxed{5}\,\boxed{5}$$

$$\boxed{5}\,5\,\boxed{12}\,5\,\boxed{5}$$

$$1\,\boxed{2}\,\boxed{2}\,1\,\boxed{2}\,\boxed{1}\,2\,\boxed{5}\,\boxed{5}$$

Give an $O(n)$-time algorithm for the problem.

1.3 (**due Sep. 11**) Given a string $\overline{x} = x_1 x_2 \cdots x_n$ we would like to find the length of the longest palindromic subsequence[1] of $\overline{x}$ (a sequence is palindromic if it is the same as its reverse). Let $T[1..n, 1..n]$ be an array where $T[i, j]$ is the length of the longest palindromic subsequence of $x_i, \ldots, x_j$ (note that, $T[i, j]$ is undefined for $j < i$). Give a piece of code to efficiently compute T. (Hint: start with $T[i, j]$ where $j = i$, then compute $T[i, j]$ where $j = i + 1$, then compute $T[i, j]$ where $j = i + 2$, etc.)

3 Bonus Homework - solve and turn in

1.4 (**due Sep. 11**) Let $a_1, \ldots, a_n$ and $b_1, \ldots, b_m$ be two sequences of numbers. We would like to find **the number of occurrences** of $b_1, \ldots, b_m$ as a **subsequence**[1] of $a_1, \ldots, a_n$

For example, if $a_1 = a_2 = \cdots = a_n = 1$, $m = 2$, $b_1 = b_2 = 1$ then the answer is $n(n-1)/2$. Give a polynomial-time dynamic programming algorithm for the problem.

1.5 (**due Sep. 11**) We are given a sequence A of n numbers $a_1, \ldots, a_n$. We would like to count **the number of different subsequences** of A. For example, $A = 1, 2, 3$ has 8 different subsequences (one of length 3, three of length 2, three of length 1, and one of length 0); on the other hand $A = 1, 1, 1$ has 4 different subsequences (one of length 3, one of length 2, one of length 1, and one of length 0). Give a polynomial-time dynamic programming algorithm for the problem.

1.6 (**due Sep. 11**) We have an $a \times b$ bar of chocolate (where a, b are integers). By breaking the bar we can either

- create two bars $a_1 \times b$ and $a_2 \times b$ where a_1, a_2 are integers and $a_1 + a_2 = a$, or

[1] What is a subsequence? We say that $b_1, \ldots, b_m$ is a subsequence of $a_1, \ldots, a_n$ if by erasing some of the a_i's from $a_1, \ldots, a_n$ we can obtain the sequence $b_1, \ldots, b_m$; for example $1, 1$ is a subsequence of $1, 2, 1$.

- create two bars $a \times b_1$ and $a \times b_2$ where b_1, b_2 are integers and $b_1 + b_2 = b$.

We can further break the resulting bars. Our goal is to 1) end up with bars that are square (that is, have size 1×1, or 2×2, or 3×3, and so on) and 2) minimize the total number of breaks.

For example, if $a = 2$ and $b = 3$ then we use 2 breaks:

$$\begin{aligned} 2 \times 3 \rightarrow \\ 2 \times 2 \text{ and } 2 \times 1 \rightarrow \\ 2 \times 2 \text{ and } 1 \times 1 \text{ and } 1 \times 1. \end{aligned}$$

For example, if $a = 2$ and $b = 4$ then we use 1 break:

$$\begin{aligned} 2 \times 4 \rightarrow \\ 2 \times 2 \text{ and } 2 \times 2. \end{aligned}$$

Give a dynamic programming algorithm which computes a table $T[1..a, 1..b]$ where $T[x, y]$ contains the minimal number of breaks to "squareize" an $x \times y$ bar.

4 Additional problems from the book (do not turn in)

Try to solve the following problems. A few of them can be on the quiz. We will go over the ones that you choose in the problem sessions.

- 5.14, 5.15, 5.16, 5.20, 5.21, 5.26, 5.32,
- 6.1, 6.2, 6.6, 6.8, 6.9, 6.11, 6.14, 6.20, 6.21, 6.25, 6.26, 6.27, 6.29.

Check Jeff Erickson's (UIUC) notes for additional problems:
`http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/notes/05-dynprog.pdf`