Greedy algorithms/dynamic programming (part 4 of CSC 282),

http://www.cs.rochester.edu/~stefanko/Teaching/11CS282

1 Schedule

"Homework" problem sessions are in CSB 601, 6:15-7:15pm on Nov. 2 (Wednesday), and on Nov. 9 (Wednesday); held by Rintaro Kuroiwa and Josh Pawlicki.

Homework is **due Nov 10** (Thursday).

"Exam" problem session is in CSB 601, 4:45-5:45pm on Nov. 14 (Monday).

EXAM #3 will be on **Tuesday, Nov. 15**.

2 Basic Homework - solve and turn in

4.1 (**due Nov 10**) Consider the coin change problem with coin values $1, 4, 6$. Does the greedy algorithm always find an optimal solution? If the answer is no, provide a counterexample. If the answer is yes, give a proof.

4.2 (**due Nov 10**) Consider the coin change problem with coin values $1, 4, 7$. Does the greedy algorithm always find an optimal solution? If the answer is no, provide a counterexample. If the answer is yes, give a proof.

4.3 (**due Nov 10**) We are given an $n \times n$ array A of zeros and ones. Give an algorithm to find the size of the largest contiguous all-ones square. Your algorithm must run in time $O(n^2)$.

4.4 (**due Nov 10**) We are given n positive numbers $a_1, \ldots, a_n$ (the numbers are not necessarily integers). 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.

4.5 (**due Nov 10**) We are given n positive integers $a_1, \ldots, a_n$ and another positive integer M. We want to figure out if we can select a subset of the integers which sums to M. Give an $O(Mn)$-time algorithm for the problem.

4.6 (**due Nov 10**) Write a dynamic programming algorithm which for a given number n finds the smallest number of squares which sum to n (for example for $n = 7$ we need 4 squares ($7 = 2^2 + 1^2 + 1^2 + 1^2$), whereas for $n = 13$ we only need 2 squares ($13 = 3^2 + 2^2$)). Implement your algorithm and find all numbers from $\{1, 2, \ldots, 100\}$ which need 4 squares. Use "The On-Line Encyclopedia of Integer Sequences" to find a formula for the numbers which need 4 squares.

3 Advanced Homework - solve and turn in

4.7 (**due Nov 10**) We are given a sequence of n positive numbers $a_1, \ldots, a_n$. Give an algorithm which finds the increasing subsequence[1] of $a_1, \ldots, a_n$ with the maximal sum. (For example on input $1, 101, 2, 3, 100, 4, 5$ your algorithm should output $1, 2, 3, 100$.)

4.8 (**due Nov 10**) 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 an expression (or a piece of code) for $T[i, j]$ in terms of already computed values in T.

4.9 (**due Nov 10**) 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
- 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.

4.10 (**due Nov 10**) A **shuffle** of two strings A, B is formed by interspersing the characters into a new string, keeping the characters of A and B in the same order (for example, '**sev**e**ral**' is a shuffle of 'seal' and 'evr'). Given three strings $A = a_1 \ldots a_n$, $B = b_1 \ldots b_m$, and $C = c_1 \ldots c_{m+n}$, we would like to verify whether C is a shuffle of A and B. Give a dynamic programming algorithm for the problem

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

Try to solve the following problems. A few of them will 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.

[1] A subsequence of $a_1, a_2, \ldots, a_n$ is any $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. Thus, e. g., $1, 3, 5$ is a subsequence of $1, 2, 3, 4, 5$.