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

http://www.cs.rochester.edu/~stefanko/Teaching/07CS282

1 Schedule

Problem sessions:
Mon, Oct. 29, 7pm-8pm
Wed, Oct. 31, 8:15pm-9:15pm
Mon, Nov. 5, 7pm-8pm
Wed, Nov. 7, 8:15pm-9:15pm
Mon, Nov. 12, 7pm-8pm

The quiz will be on Tuesday, Nov. 13.

2 List of algorithms covered in the class

(B-basic, I-intermediate, A-advanced):

B: Huffman encoding (p.139, DSV).
I: Coin-change problems (handout).
I: Longest increasing subsequence (p.157, DSV).
I: Edit distance (p.159, DSV).
I: Knapsack (p.164, DSV).
A: Chain matrix multiplication (p.171, DSV).
A: Independent sets in trees (p.176, DSV).

3 Basic material

Important concepts, problems, theorems, and algorithms:

- Two basic greedy algorithms: Huffman encoding, coin change.
- Two basic dynamic programming algorithms: knapsack, longest increasing subsequence.

Testing method:

- Trace all of the four algorithms above/compute answer to example problems.

Example test problems (homework):

7.1 (**due Nov 6, 2007**) Suppose that symbols a, b, c, d, e, f, g, h occur with frequencies $1/36, 1/36, 1/12, 1/9, 5/36, 1/6, 7/36, 1/4$, respectively. Construct the Huffman encoding of the alphabet.

7.2 (**due Nov 6, 2007**) Consider the coin change problem with coin values $1, 3, 5$. 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.

7.3 (**due Nov 6, 2007**) 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.

7.4 (**due Nov 6, 2007**) 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.

7.5 (**due Nov 6, 2007**) Consider the knapsack problem with knapsack limit 136 and the following items (weight,value):

$$(41, 4), (48, 4), (44, 4), (37, 3), (29, 2), (26, 2), (22, 2), (40, 4), (46, 4).$$

Find an optimal solution of the problem.

7.6 (**due Nov 6, 2007**) Find the longest increasing subsequence of

$$5, 3, 4, 1, 6, 10, 7, 11, 12, 8, 9.$$

4 Additional homework

7.7 (**due Nov 6, 2007**) [Problem rating: I.] 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 $O(n^2)$.

7.8 (**due Nov 6, 2007**) [Problem rating: I.] 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.

7.9 (**due Nov 8, 2007**) [Problem rating: I.] 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.

7.10 (**due Nov 8, 2007**) [Problem rating: I.] We are given n coin values $c_1, c_2, \ldots, c_n$ and an amount P (the c_i and P are positive integers). Unlike in the original coin change problem (where we had an unlimited supply of each coin value) we now have only 2 of each coin value. We would like to figure out whether we can pay P, and if we can, what is the minimal number of coins that can be used to pay P. Give an efficient algorithm for the problem.

For example if the coin values are $1, 2, 5, 6$ and $P = 15$ then the answer is yes - use 5 coins (since $15 = 6+6+2+1$ or $15 = 6 + 5 + 2 + 2$). (Note that we cannot pay $15 = 5 + 5 + 5$, since we have only 2 coins of value 5.)

7.11 (**due Nov 6, 2007**) [Problem rating: I.] 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.

7.12 (**due Nov 8, 2007**) [Problem rating: I.] We are given a sequence of n positive numbers $a_1, \ldots, a_n$. Give an algorithm which finds the increasing subsequence 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 $1, 2, 3, 100$.)

5 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, 5.26, 5.32,
- 6.7, 6.8, 6.11, 6.26, 6.27, 6.29.