

## 1 Schedule

Homework is **due Nov 9, 2010**.  
The **QUIZ** will be on **Thursday, Nov. 11, 2010**.

## 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.

**Basic problems (solve, do NOT hand in):**

**4.1** 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.

**4.2** 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.

**4.3** 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.

**4.4** 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.

**4.5** Find the longest increasing subsequence of

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

## 4 Homework

**4.6 (due Nov 9)** 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  $O(n^2)$ .

**4.7 (due Nov 9)** We are given  $n$  positive numbers  $a_1, \dots, a_n$  (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):

$$\begin{array}{cccccc} \boxed{5} & \boxed{5} & 8 & \boxed{5} & \boxed{5} & \\ & \boxed{5} & 5 & \boxed{12} & 5 & \boxed{5} & \\ 1 & \boxed{2} & \boxed{2} & 1 & \boxed{2} & \boxed{1} & 2 & \boxed{5} & \boxed{5} \end{array}$$

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

**4.8 (due Nov 9)** We are given  $n$  positive integers  $a_1, \dots, a_n$  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.

**4.9 (due Nov 9)** We are given  $n$  coin values  $c_1, c_2, \dots, c_n$  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.)

**4.10 (due Nov 9)** 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  $\{1, 2, \dots, 100\}$  which need 4 squares. Use “The On-Line Encyclopedia of Integer Sequences” to find a formula for the numbers which need 4 squares.

**4.11 (due Nov 9)** We are given a sequence of  $n$  positive numbers  $a_1, \dots, a_n$ . Give an algorithm which finds the increasing subsequence<sup>1</sup> of  $a_1, \dots, a_n$  with the maximal sum. (For example on input 1, 101, 2, 3, 100, 4, 5 your algorithm should output 1, 2, 3, 100.)

**4.12 (due Nov 9)** Given a string  $\bar{x} = x_1x_2 \dots x_n$  we would like to find the length of the longest palindromic subsequence<sup>1</sup> of  $\bar{x}$  (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  $x_i, \dots, x_j$  (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  $T$ .

---

<sup>1</sup>A subsequence of  $a_1, a_2, \dots, a_n$  is any  $a_{i_1}, a_{i_2}, \dots, a_{i_k}$ , where  $1 \leq i_1 < i_2 < \dots < i_k \leq n$ . Thus, e. g., 1, 3, 5 is a subsequence of 1, 2, 3, 4, 5.

**4.13 (due Nov 9)** 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 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.

## 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, 5.20, 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.