Algorithms, Winter 2010-11, Homework 3
due Friday 01/07/11, 4:00pm

Problem 1

In a little town there are n houses on the main street. This street runs along the x-axis and the coordinates of the houses are $x_1, x_2, \ldots, x_n$. You decided to open a candy shop and you want to find the most profitable coordinate x_{best}. The most profitable coordinate will be one that minimizes the sum of the distances to every house. Give an $O(n)$ algorithm that finds x_{best} such that $dist_{best} := \sum_{i=1}^{n} |x_{best} - x_i|$ is as small as possible.

Your submission should include:

- Pseudocode of your algorithm and its running time estimate.
- Argument that indicates that your algorithm is correct.
- Java/C/C++ implementation.

Problem 2

An alphabet contains letters A, B, C, D, E, F. The frequencies of the letters are 35%, 20%, 15%, 15%, 8%, and 7%. We know that the Huffman algorithm always outputs an optimal prefix-free code. However, this code is not always unique (obviously we can, e.g., switch 0's with 1's and get a different code – but, for some inputs, there are two optimal prefix-free codes that are significantly different). For the purposes of this exercise, we consider two Huffman codes to be different if there exists a letter for which one of the codes assigns a shorter codeword than the other code.

- Trace the Huffman algorithm and construct two different optimal Huffman codes for the above input.
- Compute the expected number of bits per character (i.e., the expected codeword length) for both codes.

Problem 3

Implement the two algorithms for the indivisible KNAPSACK problem (slides 24 and 27); both algorithms output the maximum cost one can get with a backpack of size W.

- Create inputs for $n \in \{5, 10, 20, 50, 100\}$, where each item costs 1 and weights 1, and $W = n/2$. Run both algorithms on these inputs. Write a paragraph summarizing your findings.
- Modify the second algorithm so that it prints the list of items that achieve the maximum cost, using overall running time $O(nW)$. Include a pseudocode of your algorithm and a short explanation of how the algorithm works.