CS140 - Assignment 6

Due: Friday, Mar. 10 at 8pm

http://www.smbc-comics.com/index.php?db=comics&id=2217

For this assignment, you may (and are encouraged to) work with a partner.

1. **[5 points]** Suppose the symbols a, b, c, d, e occur with frequencies 30, 20, 15, 40, 60 respectively, in a file.

   (a) What is the Huffman encoding of the alphabet? Note, follow the algorithm in class, where the smallest value is the left branch and the next smallest is the right branch. For encoding, the left branch will be 0 and the right branch 1.

   (b) If this encoding is applied to the file, what is the length of the encoded file in bits?

2. **[13 points]** A thief robbing a bulk food store finds n items worth $v_1, v_2, ..., v_n$ dollars and weigh $w_1, w_2, ..., w_n$ pounds, where v_i and w_i are integers. The thief can carry at most W pounds in the knapsack. Because it's a bulk food store, the thief may take all of an item *or* only some fraction of any item (e.g. half of item i, getting value $v_i/2$ with weight $w_i/2$). The goal of the thief is to select the items so as to maximize the profit while staying under the weight constraint.

   (a) **[5 points]** Describe an optimal greedy heuristic and a high-level algorithm for selecting which items (and how much) to select. You don't need to state the low-level details of the algorithm, just how you will construct your solution.

   (b) **[5 points]** Prove that your algorithm is correct. You may either use a "stays ahead" proof or a proof by contradiction.

   (c) **[3 points]** State your algorithm more precisely and, based on this, state what the running time of your algorithm will be in terms of n the number of items.

3. **[10 points]** Given a set of points $x_1, x_2, ..., x_n$ on the real line, describe a greedy algorithm that determines the smallest set of unit-length (i.e., length=1) closed intervals that contains all of the given points. State the worst case running time and prove that your algorithm is correct. You do not need write pseudo-code, but make your description clear.

4. **[15 points]** Hashtable fun

   (a) **[3 points]** Show the result of inserting 5, 28, 19, 15, 20, 10, 33, 12, 17 into a hashtable with collision resolution by chaining. The table should have 9 slots and use $h(k) = k \bmod 9$ for the hash function.

   (b) **[3 points]** Show the result of inserting the first 6 of these into another hashtable using open addressing and linear probing.

   (c) **[1 points]** For the these insertions, what was the largest number of collisions you had before finding an open slot and what key was it?

   (d) **[8 points]** Open addressed hashtables can fill up. If you want to be able to support hashtables that are not of a fixed size, when an insertion happens and the table is full (or, more often, something like half full) a common approach is to create a new larger hashtable and insert the existing keys into this larger hashtable.

i. Can you simply copy the entries from the old hashtable into the new hashtable? If so, justify why this still results in a correctly functioning hashtable. If not, describe what the problem is.

ii. Argue that increasing the table size by a constant amount when it fills up will end up with a insertion being amortized $\Omega(n)$.