Problem Set 3: Sorting and Greedy Algortihms

Handed out Friday, September 6. Due at the start of class Friday, September 13.

Problem 1. (25 points) This problem involves the question of computing change for a given coin system. A coin system is defined to be a sequence of coin values $v_1 < v_2 < ... < v_n$, such that $v_1 = 1$. For example, in the U.S. coin system we have six coins with values $\langle 1, 5, 10, 25, 50, 100 \rangle$. The question is what is the best way to make change for a given integer amount A.

(a) Let $c \geq 2$ be an integer constant. Suppose that you have a coin system where there are n types of coins of integer values $v_1 < v_2 < ... < v_n$, such that $v_1 = 1$ and, for $1 < i \leq n, v_i = c \cdot v_{i-1}$. (For example, for $c = 3$ and $n = 4$, an example would be $\langle 1, 3, 9, 27 \rangle$.) Describe an algorithm which given n, c, and an initial amount A, outputs an n-element vector that indicates the minimum number of coins in this system that sums up to this amount. (Hint: Use a greedy approach.)

(b) Given an initial amount $A \geq 0$, let $\langle m_1, ..., m_n \rangle$ be the number of coins output by your algorithm. Prove that the algorithm is correct. In particular, prove the following:

(i) For $1 \leq i \leq n, m_i \geq 0$

(ii) $\sum_{i=1}^{n} m_i \cdot v_i = A$

(iii) The number of coins used is as small as possible.

Prove that your algorithm is optimal (in the sense that it generates the minimum number of coins) for any such currency system.

(c) Give an example of a coin system (either occurring in history, or one of your own invention) for which the greedy algorithm may fail to produce the minimum number of coins for some amount. Your coin system must have a 1-cent coin.

Problem 2.(10 points) You are given an array of n keys, each with one of the values *red*, *white*, and *blue*. Give an $O(n)$ algorithm for rearranging the keys so that all the reds come before all the whites, and all the whites come before all the blues. The only operations permitted on the keys are:

(1) Examine(A, i) - report the color of the i-th element of A.

(2) Swap(A, i, j) - swap the i-th element of A with the j-th element of A

Explain your algorithm and derive its running time.

Problem 3.(15 points) Consider the following algorithm for selection sort. Recall that the algorithm operates in stages, from 1 to n - 1. During the ith stage, the algorithm finds the minimum elements of the subarray $A[i..n]$ and swaps it into its final sorted location in $A[i]$.

```
SelectSort(int n, array A[1..n]) {
    for i = 1 to n-1 do {
        min = i
        for j = i+1 to n do
            if (A[j] < A[min])
                min = j
        swap A[i] with A[min]
    }
}
```

(a) As a function of n, express exactly how many times the test in the "if" statement is executed. Express your answer as a simple expression with no embedded summations or recurrences.

(b) In class we said that selection sort is stable. This is *not* true (thanks to Kevin for pointing this out!). Give an example of an array with some duplicate elements, such that the relative position of a duplicate pair of items is reversed after the sort is finished. To keep things straight, label duplicate elements with subscripts, e.g. 2a, 2b, etc. (To make the grader's life easy, you will receive the maximum credit by giving the smallest counterexample.) Briefly explain your counterexample.

(c) It is natural to consider whether there is some simple change to the algorithm which will make it stable. Suppose we change the $<$ in the algorithm above with $\leq$. Show that the algorithm is still unstable even after this change.

Problem 4.(15 points) Programming Problem: (upload your code to Moodle)
Be sure to include your name as a comment in your code file.

Implement insertion sort and mergesort sorting algorithms in Python. Your program should allow the user to specify which sorting algorithm they would like to use as well as the name of the input file they would like to use. (You may allow for this to be entered from command line or as prompts in your code. Please either supply comments in your code or helpful prompts so that I can easily run your code as you intended. Please also specify if you are using Python 2 or Python 3 in the comments.)

As part of your written homework, please indicate for each sorting algorithm how long your program takes to sort PS3_input.txt (On course website)

See code example "PythonReviewInputTimingParseFile.py" (on Moodle) for help with calculating the run time of your program, parsing your input file, and getting arguments from the command line.

You should assume that all input files are of the same format: 1 integer per line. Your output may be the same format as the input file, or you may output the actual sorted array.