O COMP 215 Algorithms Θ

Lab #6

This work should, if possible, be completed during the lab and submitted electronically before 11:59pm Friday. Submit a .zip file that contains all the file with your code (preferably a single file lab6.cpp, but you can make it multiple files if you prefer).

Finally! we can start implementing algorithms that we actually covered in class. In this lab, you will implement the first few brute force algorithms we have seen in class.

1. Your program should first ask the user if the input file will contain numbers (for the selection sort, bubble sort or sequential search algorithm) or text (for the string matching algorithm).

The program should then proceed in two different paths, depending if the input file contained numbers or text. First let us consider the path if the input file contained numbers.

2. Create a class `pairsAgain` to hold the data from the input file. The class should contain two public integer `val1` and `val2`, a default constructor, a constructor that takes 2 integers and initializes `val1` and `val2` accordingly, and overload the operators ==, <, >, <= >= so that $(a, b) == (c, d)$ if and only if $a = c$ and $(a, b) < (c, d)$ if and only if $a < c$ (so basically, we are comparing the `val1` component of the pairs in the straightforward way, and ignoring the `val2` component in the comparison).
3. Your program should then ask the user for the name of the input file. The first line of the input file will contain the number of pairs in the file, the following lines will contain two integers per line (first and second element of each pair). Read all the data from the file and store it in an array of `pairsAgain`.
4. Implement the Selection Sort, Bubble Sort and Sequential Search algorithms from the book. Make sure that you implement the improvement to Bubble Sort so that the algorithms stops as soon as the array is in ascending order.
5. Ask the user if he wants to use Selection Sort, Bubble Sort or Sequential Search.
   (a) If the user asks for Selection Sort, use your Selection Sort function on the array you just input, ask the user for the name of an output file and output the sorted array to the file (number of lines on the first line, one pair on each line for the other lines)
   (b) If the user asks for Bubble Sort, do the same as the above, but using Bubble Sort instead
   (c) If the user asks for Sequential Search, keep prompting the user for search values until the user enters a value that is not in the array.

If the user said that the input file would contain text, do the following:

6. Your program should ask the user for the name of the input file. The first line of the input file will contain the number `nblines` of text lines in the file, followed by `nblines` of text. Read the entire input file and concatenate all the lines together so that the whole text is stored in a single string (make sure that you remove all the newline characters, and you can assume that all the lines end with a space character, so that you do not have to add spaces).
7. Implement the brute force String Matching algorithm from the book.
8. Keep asking the user for patterns until the user enters a pattern that is not present in the text. At each step, you should print on the screen the index of the first appearance of the pattern.

You will find on onCourse a .zip file that contains a few possible input files. Use them to test your program, or create your own to make additional tests.