Advanced Topics on Algorithms

Homework #1: Due 11:59PM, 25th March 2024 (Monday).

Problem #1 (20 points). Programming

Table Sorting

- You are sorting data of a table with multiple rows and columns, the first row includes column names. You are given a list of rules for sorting in the format "COLUMN_NAME SORT_ORDER." The SORT_ORDER can either be ASC (ascending order) or DESC (descending order). Your task is to arrange the rows of the table primarily based on the first rule. In the event of a tie, the secondary rule is used for sorting, and so on. If two rows have identical values for all the rules, their original relative order is maintained.
- Input is read from the text file **input.txt** consisting of
  - The first line contains column names.
  - The second line contains the list of rules.
  - The rest of the input data contains the table. All the words and column names are separated by single spaces. Names of columns and elements are strings containing only uppercase and lowercase Latin letters and digits, having the length between 1 and 10, inclusive.
- Output is written to the text file **output.txt** consisting of the table after sorting.

Example:

input.txt	output.txt
NAME GROUP AGE GROUP ASC, AGE DESC Alex 412 19 Peter 422 19 Sergey 412 18 Andrey 311 18	Andrey 311 18 Alex 412 19 Sergey 412 18 Peter 422 19

Limitation:

- The number of rows and columns is between 1 and 100
- Running time of the program $\leq$ 2 seconds

Notes:

- Using standard C, C++, Java, Python (equivalent)
- Only built-in functions of each language are allowed. External libraries, such as *sklearn in Python*, are prohibited.

- The name of the source code file is **tablesorting.xxx** (The extension **xxx** depends on the used programming language)
- The name and format of input and output files **must follow exactly what was described in the problem**. Students **get zero point if they do not follow the format**

Problem #2 (20 points). Programming

Counting

- There are n people standing in a row, each person is facing either left or right. Each individual **counts the number of people in the direction they are facing**. The **row value** is determined by **adding up the count of each person**.
- As an illustration, take the sequence LRRLL, where L indicates an individual facing the left direction, and R indicates an individual facing the right. The respective counts for each person in this sequence are [0, 3, 2, 3, 4], leading to a row value of 12.
- Given the initial position of all people in the row. Your task is to determine the **maximum row value** if you can change the direction of at most k people.
- Input is read from the text file **input.txt** consisting of:
  - The first line of each test case contains an integer n - the length of the line
  - The following line contains a string consisting of n characters, each of which is either L or R, representing a person facing left or right, respectively
- Output is written to the text file **output.txt** consisting of n space-separated non-negative integers — the maximum row value if you can change the direction of at most k people, for each k from 1 to n.

Example:

input.txt	output.txt
3 LLR	3 5 5

Example explanation:

- k=1: change the direction of 1 person to make the line RLR. The row value is 2+1+0=3
- k=2: change the direction of 2 people to make the line RLL. The row value is 2+1+2=5
- k=3: change the direction of 2 people to make the line RLL. The row value is 2+1+2=5.

Limitation:

- $1 \leq n \leq 2 \times 10^5$

- Running time of the program ≤ 2 seconds

Notes:

- Using standard C, C++, Java, Python (equivalent)
- Only built-in functions of each language are allowed. External libraries, such as *sklearn in Python*, are prohibited.
- The name of the source code file is **counting.xxx** (The extension **xxx** depends on the used programming language)
- The name and format of input and output files **must follow exactly what was described in the problem**. Students **get zero point if they do not follow the format**

Problem #3 (10 points). Suppose we are comparing two sorting algorithms.

a) Suppose that for all inputs of size n, the first algorithm runs in $8n^2$ seconds, while the second algorithm runs in $64n \log_2 n$ seconds. For which values of n does the first algorithm run faster the second algorithm?
b) Use slides 18&19 in the lecture note as an example, illustrate the operations of merge sort on the following array: {**23, 11, 45, 33, 11, 34, 77, 66, 12**}

Problem #4 (15 points). We are sorting an array in ascending order as follows:
In the first iteration, the 1st element is compared with the 2nd element. If it is found to be greater than the 2nd element, they are interchanged. Then, the 2nd element is compared with the 3rd element. If it is found to be greater than the 3rd element, they are interchanged. In the same way, all the elements (excluding the last) are compared with their next element and are interchanged if required. After the 1st iteration of the algorithm, the largest element is placed at the last position and it is not considered in the next iteration. Moving on to the 2nd iteration, the same process is applied to place the second largest element at the second last position. This process continues iteratively, until the list becomes fully sorted.

a) Write pseudocode for this algorithm.
b) Apply the above algorithm to an array L = {**92, 78, 34, 23, 56, 90, 17, 52, 67, 81, 18**}. Show the array after each iteration.
c) Give the best-case and worst-case running times of the algorithm in O-notation. When do these cases happen?

Problem #5 (10 points). Consider the Fibonacci function *F*(*n*), which is defined such that *F*(1) = 1, *F*(2) = 2, and *F*(*n*) = *F*(*n* - 2) + *F*(*n* - 1) for *n* > 2. Prove by induction that *F*(*n*) < 2^n.

Problem #6 (15 points).

a) Write a pseudocode for a divide-and-conquer algorithm for finding values of both the largest and smallest elements in an array of n numbers.

b) For $n = 2^k$, determine and solve the recurrence relation for the number of comparisons made by the above algorithm.

Problem #7 (10 points). For each of the following functions, prove whether $f = O(g), f = \Omega(g), or\ f = \Theta(g)$, and explain your answer by specifying some explicit constants n_0 and $c > 0$ such that the definition of these notations is satisfied.

a) $f(n) = n\log(n^3)$ $\quad g(n) = n\log n$

b) $f(n) = 2^{2n}$ $\quad g(n) = 3^n$

c) $f(n) = \sum_{i=1}^{n} \log i$ $\quad g(n) = n\log n$

What you have to submit:

1) For programming problems: source codes, input data, and output data.
2) For other problems: submit a **Word** file (HW1.docx) including your answers.

◆ Compress all files into one zip file named HW1_StudentID_NAME.zip (ex. HW1_2013711123_홍길동.zip) and submit to iCampus.

NOTICE:

✓ BOTH ORIGINAL AND COPY WILL GET –30 POINTS EACH INSTEAD OF 0S.
✓ ANY SOURCE CODE WITH COMPILE OR RUNTIME ERROR WILL GIVE YOU 0 POINTS.
✓ THERE WILL BE POINTS OFF FOR INAPPROPRIATE SUBMISSION STYLE.
✓ ALL THE HOMEWORK MATERIALS (INCLUDING EMAIL CONTENTS AND DOCUMENTATION) SHOULD BE MADE IN **ENGLISH**.

Good luck!