Design and Analysis of Computer Algorithms

Homework #2: Due 11:59PM, April 08th, 2024 (Monday).

Problem #1. (20 points) Programming

Uniform Universes

In 2030, humans have a marvelous progress in exploring the universe. They get shock when they realize that there is not only one universe in which the earth is. The fact is that there are many universes outside and our universe is not alone!!!The interesting thing is all universes contain the same number of planets with different sizes and **planets of a universe is fixedly numbered from 1 to n**. Two universes A and B are uniform if their arrays of planet-sizes $(a_1, a_2, a_3, \dots, a_n)$ and $(b_1, b_2, b_3, \dots, b_n)$, respectively, satisfy the following rules:

- $If\ a_i < a_j\ then\ b_i < b_j\ \forall i, j$
- $If\ a_i = a_j\ then\ b_i = b_j\ \forall i, j$
- $If\ a_i > a_j\ then\ b_i > b_j\ \forall i, j$

Example:

- Given two universes A and B with two arrays of planet-sizes $(1, 3, 2)$ and $(12, 50, 31)$, respectively. Universe A and B are uniform because
  - $a_1 = 1 < a_2 = 3\ and\ b_1 = 12 < b_2 = 50$
  - $a_1 = 1 < a_3 = 2\ and\ b_1 = 12 < b_3 = 31$
  - $a_2 = 3 > a_3 = 2\ and\ b_2 = 50 > b_3 = 31$
- Given two universes A and B with two arrays of planet-sizes $(1, 3, 2)$ and $(12, 50, 10)$, respectively. Universe A and B are not uniform because:
  - $a_1 = 1 <\ a_3 = 2\ but\ b_1 = 12 > b_3 = 10$

Given N pairs of universes, you help scientists count how many pairs are uniform.

Input is read from the text file **input.txt** consisting of:

- The first line has 2 numbers which are the number of pairs of universes **m** and the number of planets **n** in a universe**. Noting that all universes have the same number of planets**.
- The next **m** couple lines are the sizes of planets $(a_1, a_2, a_3, \dots, a_n)$ and $(b_1, b_2, b_3, \dots, b_n)$ of **m** pairs of universes**. Noting that the sizes $(\boldsymbol{a_1, a_2, a_3, \dots, a_n})$**

and ($b_1, b_2, b_3, \dots, b_n$) are corresponding to the planet 1st, 2nd, 3rd … n-th of universes A and B, respectively.

- Output is written to the text file **output.txt** consisting of ONLY ONE NUMBER which is **the number of uniform pairs.**

Example:

input.txt	output.txt
2 3 3 5 1 5 24 4 2 8 1 20 24 22	1

Limitation:

- $1 \leq m \leq 2 \cdot 10^5$
- $2 \leq n \leq 3000$
- $1 \leq the\ size\ of\ a\ planet \leq 10^6$
- Running time of the algorithm ≤ 1 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 **universe.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

Competition

- A competition containing m subjects is coming. The school needs to choose some subjects to participate in and form a team for each chosen subject with the condition that **the number of students in each team should be the same**. There are n students, the i-th student specializes in subject s_i with skill level of l_i. For each subject, the school only selects students specializing in that subject. The goal is to **maximize the total sum of skill levels of all teams**. The school will skip the competition if every non-empty team has negative sum of skill levels.

- Input is read from the text file **input.txt** consisting of:
  - The first line contains n and m - the number of students and number of subjects.
  - Each of the next n lines contains two integers s_i and l_i - subject of specialization and the skill level of the i-th student.

- Output is written to the text file **output.txt** consisting of ONLY ONE NUMBER which is the maximum total sum of skill levels of all teams. If the school skips the competition, the output value is -1.

Example:

input.txt	output.txt
6 3 2 6 3 6 2 5 3 5 1 9 3 1	22

Explanation:

- Choose students 1, 2, 3, 4
  + The team for the 2-nd subject includes students 1&3 with the skill levels of 6 and 5
  + The team for the 3-rd subject includes students 2&4 with the skill levels of 6 and 5
  + The total sum of skill levels is: 6 + 5 + 6 + 5 = 22

Limitation:

- $1 \leq n, m \leq 10^5$
- $1 \leq s_i \leq m$
- $-10^4 \leq l_i \leq 10^4$
- 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 **sorting.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)

Illustrate the operation of HEAPSORT on the array A = {D, B, G, E, A, H, C, F} (sort by alphabetical order).

Problem #4. (15 points)

What value is returned by the following function? Express your answer as a function of *n*. Give the worst-case running time using the O-notation.

```
function mystery(n)
{
    r = 0;
    for i = 1 to n – 1 do
        for j = i + 1 to n do
            for k = 1 to j do
                r = r + 1;
    return r;
}
```

Problem #5. (15 points)

Write pseudo code of an algorithm to sort an n-element array a in non-descending order using at most n swaps. Your algorithm should output all swap operations in the order they are performed. For each swap operation, output two values i and j denoting the indices of the two swapped elements. What is the time complexity of your algorithm?

Problem #6. (10 points)

Solve the following recurrence:

$$T(n) = 4T(n/2) + n^2 lgn$$

Problem #7. (10 points) You are given a set of n numbers, and you wish to find i largest elements in sorted order using a comparison-based algorithm. Describe the algorithm that sorts the numbers, and lists the i largest elements with the best asymptotic worst-case running time. Analyze the running times of the algorithms in terms of n and i.

What you have to submit:

- For programming problems: source codes, input data, and output data.
- For other problems: submit a **Word** file (HW2.docx) including your answers.
- Compress all files into one zip file named HW2_StudenID_NAME.zip (ex. HW2_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.
- ✓ 10% OF PENALTY POINTS FOR EACH DAY LATE, ONLY TWO DAYS LATE ARE ALLOWED.
- ✓ ALL THE HOMEWORK MATERIALS (INCLUDING EMAIL CONTENTS AND DOCUMENTATION) SHOULD BE MADE IN **ENGLISH**.

Good luck!