Design and Analysis of Computer Algorithms

Homework #2: Due 11:59PM, March 27th, 2022 (Sunday).

Problem #1. (15 points) Programming

A Consistent Phonebook

We are given a phonebook with a list of phone numbers. This phonebook is called a consistent phonebook if there is no phone number which is the prefix of the others. Otherwise, it is an inconsistent phonebook. For example, the following phonebook has three numbers:

- 911
- 9760001
- 91141554

In this case, **911** is the prefix of **911**41554 so the phonebook is inconsistent.

Your task is checking if a phonebook is consistent or not.

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

- The first line is the number of phone numbers N
- The next N lines are the phone numbers (each line contains one phone number)

Output is written to the text file **output.txt** consisting of ONLY ONE NUMBER which is 0 if the phonebook is inconsistent and 1 otherwise.

Example:

input.txt	output.txt
5 113 12340 123440 12345 98346	1

Limitation:

- $1 \le N \le 10000$
- The length of phone numbers is less than 10 digits
- Processing time of the proposed algorithm ≤ 1 second

Notice:

- Using standard C, C++, Java, Python (equivalent)
- The name of the source code file is **phonebook.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**
- The **output file only has one value.**

Problem #2. (15 points) Programming

Sorting

We are given an array $A = [a_1, a_2, \dots, a_n]$ of distinct integer numbers from 1 to n (n is an odd number) that needs to be sorted in increasing order. Bui is a TA of Algorithms course. He proposes an algorithm which needs at least k iterations to finish. In the algorithm, Bui defines an operation $f(i)$ to compare a_i with a_{i+1}, if $a_i > a_{i+1}$, values of them are exchanged. At m^{th} iteration, if m is odd, execute $f(1), f(3), \dots, f(n-2)$, otherwise, execute $f(2), f(4), \dots, f(n-1)$. Bui is busy with his homework, so he wants someone to help him implement the algorithm and find the value of k.

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

- The first line includes value of n
- The second line includes elements of the array $a_1\ a_2\ \dots\ a_n$

Output is written to the text file **output.txt** consisting of ONLY ONE NUMBER which is **the value of k.**

Example:

input.txt	output.txt
7 4 5 7 1 3 2 6	5

Explanation:

- At 1st iteration: call $f(1), f(3), f(5)$. The array after 1st iteration is: [4,5,1,7,2,3,6]
- At 2nd iteration: call $f(2), f(4), f(6)$. The array after 2nd iteration is: [4,1,5,2,7,3,6]
- At 3rd iteration: call $f(1), f(3), f(5)$. The array after 3rd iteration is: [1,4,2,5,3,7,6]
- At 4th iteration: call $f(2), f(4), f(6)$. The array after 4th iteration is: [1,2,4,3,5,6,7]
- At 5th iteration: call $f(1), f(3), f(5)$. The array after 5th iteration is: [1,2,3,4,5,6,7]

Note: In the example, index of the array starts from 1 but in your code, the index starts with 0

Limitation:

- $n \leq 999$
- Running time of the algorithm ≤ 2 seconds

Notice:

- Using standard C, C++, Java, Python (equivalent)
- 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.(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 #4.(15 points)

Using Figure 6.4 of textbook (slide #63~70 in lecture note) as a model, illustrate the operation of HEAPSORT on the array A = {5; 13; 2; 25; 7; 17; 20; 8; 4}.

Problem #5. (10 points) The following algorithm uses a divide-and-conquer strategy to search an unsorted list of numbers. Given a list of numbers A and a target number t, the algorithm returns 1 if t is in the list, and 0 otherwise.

```
UNSORTED-SEARCH(A, t, p, q)
if q − p < 1
    if A[p] = t return 1 else return 0
if UNSORTED-SEARCH(A, t, p, ⌊(p+q)/2⌋) = 1 return 1
if UNSORTED-SEARCH(A, t, ⌊(p+q)/2⌋ + 1, q) = 1 return 1
return 0
```

a) Determine recurrence relation of the algorithm and solve it.

b) How is the running time of this algorithm in comparison with a naive algorithm that simply iterates through the list to look for the target?

Problem #6. (15 points)

Solve the following recurrence:

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

Problem #7. (15 points). We have a disk with 1000 pages, and cache memory with space 4. When a memory request is made, if the page isn't in the cache, a page fault occurs. We then need to bring the page into the cache and throw something else out if the cache is full. Our goal is to minimize the number of page faults. There are several online algorithms to evict a page when a page fault occurs. In this problem, we consider Least-Recently-Used (LRU) and First-In-First-Out (FIFO) algorithms. LRU algorithm evicts the page that has been unused for the longest time. An offline algorithm is given the whole input sequence from the beginning and output the solution with the least misses.
We are given the input sequence as following:

A = 1, 2, 3, 1, 2, 4, 5, 1, 2, 3, 4, 5

We use a competitive ratio to analyze an online algorithm where the competitive ratio is computed based on number of page faults. A better online algorithm must have a lower competitive ratio.

$$\text{Competitive ratio} = \frac{\#\text{ of page faults of online algorithm}(\sigma)}{\#\ of\ \text{page faults of offline algorithm}(\sigma)}$$

Task:
Determine which online algorithm should be used in this problem.

What you have to submit:

1) Your source programs, executable files, input and output files
   (each programming problem is in one folder, put one of your test cases in the input and output files).
2) Documentation file (**HW2.DOCX**)
   - Solution of the assigned problems.
   - Write the explanation about your implementation.

◆ Submit your compressed file named as HW2_ID_NAME.zip (ex. HW2_2013711123_홍길동.zip) 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!