2801ICT Computing Algorithms – Assignment 2 (10%)

(Due Week 11)

This assignment must be done individually. The submission date is as specified in the Course Profile (**May13, 2019**) and the submission method will be communicated during trimester.

Problem Description

Scott and Rusty are playing a game named "Maximize The Score". In this game, there are *n* balls placed on the table. Each ball has a value written on it.

The game starts with a toss. If head comes, then Scott starts the game, else Rusty starts the game. The game is played in the form of rounds. Hence, if Scott wins the toss, the first round will be played by Scott, second round will be played by Rusty, third round will be played by Scott and so on. In each round, the player is allowed to take at max *k* turns. In each turn, the player picks one ball from the table.

Please note that each round is entirely played by one player only so if it is Scott's round, then he will play *x* turns ($x <= k$) and the next round will be played by Rusty.

Rusty is little bit crazy about sum of digits and so he will only pick the ball, whose sum of digits of the value is maximum (for e.g. if 4 and 11 are present on the table, he will pick 4 as the sum of digits of 4 is greater than that of 11). If more than one ball have maximum sum of digits, then he can pick any one of them. Scott doesn't care about sum of digits so he can pick any ball from table. They both want to maximize their score and so both of them will play optimally. Score of a player is the sum of values of all the balls taken by the player.

Your task is to print the score Scott and Rusty will achieve if they both play optimally.

See the sample case for better understanding

Input format:

First line contains **an integer *T***, denoting the number of test-cases.

Next **3*T* lines** contain the test-cases as shown below:

First line contains **two integers *n* and *k***, representing the number of balls present on the table and the maximum number of turns allowed per round.

Second line contains ***n* space separated integers** ($a_1, a_2, \ldots a_n$). Here, $\mathbf{a_i}$ indicates the score written on *i*th ball.

Third line contains **result of the toss**. Here, "HEADS" indicates Scott starts the game, while "TAILS" indicates Rusty starts the game.

Output format:

For each testcase, print scores achieved by Scott and Rusty, if they both play optimally.

Constraints:

$1 <= T <= 10$
$1 <= n, k <= 100000$
$1 <= a_i <= 1000000000(1e9)$

Sample Input

2
3 2
1000 99 98
TAILS
2 1
5 6
HEADS

Sample Output

1000 197
6 5

Explanation

In **first testcase**, the winner of toss is Rusty and so Rusty plays first round. In this round, he can have 2 turns and so he will take 2 turns. In first turn, he has no choice but to pick 99 as it has largest sum of digits. Similarly, in second turn he has to pick 98. Now, Scott plays his round. As there is only one ball left, Scott can only take one turn and pick that ball. So, final score of Scott = 1000 and final score of Rusty = 99 + 98 = 187.

In **second testcase**, the winner of toss is Scott and so Scott plays the first round. In this round, he will of course pick 6 as he wants to maximize the score. Then Rusty plays second round and picks 5. So, final score of Scott = 6 and final score of Rusty = 5.

Results for C++ Implementation

Input	Output	CPU Time (Secs)	Input	Output	CPU Time (Secs)
1	1000 197	0.000005	6	4726793900 3941702128	0.000010
2	240 150	0.000005	7	13793 12543	0.000015
3	2100000000 98888899	0.000004	8	**2173 1665**	0.000040
4	9538 2256	0.000011	9	3923529875 3049188235	0.000008
5	30031 17796	0.000057	10	0 284401	0.000027

Results for Python Implementation

Input	Output	CPU Time (Secs)	Input	Output	CPU Time (Secs)
1	1000 197	0.000029	6	4726793900 3941702128	0.000167
2	240 150	0.000051	7	13793 12543	0.000266
3	2100000000 98888899	0.000033	8	**2173 1665**	0.000827
4	9538 2256	0.000160	9	3923529875 3049188235	0.000119
5	30031 17796	0.001435	10	0 284401	0.000573

Notes

1. When solving this problem, you should demonstrate your understanding of implementing a **priority queue using heap**. Instead of using sorted arrays, you must use priority queues to allow Scott and Rusty to pick balls according to their preferences. **You should not use any library/package for priority queue and heap**.
2. You must design your own algorithm and write the program code submitted. The program must be able to be run from the command line passing the path to the Input.txt file as an argument. The naming convention for your program file is: <student_number>_ Maximise_The_Score.py (or <student_number>_ Maximise_The_Score.cpp or <student_number>_ Maximise_The_Score.java)(without the <>) and try and keep the program within a single file.
3. You must produce a Power-point presentation describing your algorithm (including the pseudocode), correctness of your algorithm, the results of the problem (sample Input.txt is provided) and performance analysis.

Marking

This assignment is worth 10% of your final grade. The assignment will be marked out of 100 and marks will be allocated as follows:

1. (**Algorithmic Design**) Overview, Algorithm Description, Algorithm Pseudo-Code: **30 marks**
2. (**Implementation**) Quality of code (e.g. appropriate naming conventions, code comments, class structure, method structure, appropriate use of data structures etc.): **15 marks**
3. (**Result and Algorithm Analysis**) The results of the problem (sample Input.txt is provided), *correctness of the algorithm* and performance analysis: 10+15+15=**40 marks**
4. (**Presentation**) Power-point presentation: **15 marks**

Submission

Student should submit a zip file (<student_number>_Algorithm_Assignment_2.zip) consisting of the following:

1. A document covering 1 and 3: <student_number>_Algorithm_Assignment_2.pdf,

2. Source code file: <student_number>_ Maximiser_Le_Score.py (cpp/java) and

3. A power point presentation file: <student_number>_Algorithm_Assignment_2.ppt (/pptx).

Please note that all submitted assignments will be analysed by a plagiarism detector including the submitted source code of the program. If any form of plagiarism is found, it will be dealt with in accordance with university policy: https://www.griffith.edu.au/academic-integrity/academic-misconduct.