Long Take-Home Exam 3

(Deadline: 25th of May 2023, Thursday, 23:59)

1 Problem

You are given n solutions to a multi-objective optimization problem, trying to optimize three evaluation functions: α, β and γ. These solutions are labeled from 1 to n.

We say that a solution i *dominates* a solution j if

$$\alpha(i) \geq \alpha(j) \quad \wedge \quad \beta(i) \geq \beta(j) \quad \wedge \quad \gamma(i) \geq \gamma(j)$$

and at least one of α, β, or γ values is different for i and j. (Two solutions with the exact same α, β and γ values do not dominate each other.)

A solution j is *Pareto-optimal* if there is no solution i dominating it.

Write a program to find all Pareto-optimal solutions (i.e., the *Pareto front*) given a set of solutions with their $\alpha()$, $\beta()$ and $\gamma()$ values. Your program should print these solutions in increasing order of their labels.

2 Input Format

```
N
A1 B1 C1
...
AN BN CN
```

$\texttt{N}$ = The number of solutions.
$\texttt{A}_i$ = The $\alpha()$ value of solution with label i.
$\texttt{B}_i$ = The $\beta()$ value of solution with label i.
$\texttt{C}_i$ = The $\gamma()$ value of solution with label i.

3 Output Format

```
P1
P2
...
PM
```

$\texttt{P}_i$ = The label of the ith Pareto-optimal solution.
$\texttt{M}$ = The number of Pareto-optimal solutions.

4 Limits

$1 \leq \texttt{N} \leq 100\,000$
All $\texttt{A}_i$, $\texttt{B}_i$, $\texttt{C}_i$ are integers in the range $[1, 2\,000\,000\,000]$.
Time Limit: 1 second, Memory Limit: 256 MB, Stack Limit: 8 MB

5 Clarifications

- Your solution is expected as a C++ program source named `1the3.cpp` that reads from the standard input and writes to the standard output.

- It is OK to copy code from the sample codes we shared in our course website in ODTÜClass. You can also copy code from your previous THE submissions for this course. Copying from elsewhere will be considered cheating.
- You are supposed to submit your code via the VPL item in ODTÜClass. Use the "evaluate" feature to run the auto-grader on your submission.
- The grade from the auto-grader is not final. We can later do further evaluations of your code and adjust your grade. Solutions that do not attempt a "reasonable solution" to the given task may lose points.
- We will compile your code with `g++` using the options: `-std=c++2a -O3 -lm -Wall -Wextra -Wpedantic`
- Late submissions are not allowed.

6 Sample Input/Output Pairs

Input	Output
10 12 22 38 31 24 28 17 8 46 41 32 42 47 36 39 35 50 45 18 34 4 26 44 37 40 6 20 13 3 49	3 4 5 6 10

Input	Output
10 20 30 40 43 21 32 33 40 22 23 31 42 40 20 30 23 31 42 30 40 20 43 21 30 33 41 22 20 31 42	2 4 6 9

7 Hints

- In our last lecture, we described how to use a partial-max Fenwick tree to solve this problem. This $O(n \log n)$-time solution is expected to get full marks if well implemented.
- The domination condition in this THE is slightly more complicated than what we studied in the lecture. Be careful about solutions that have an equal α, β **or** γ value. You will need to apply some kind of tie-breaking procedure in individual dimensions. Some possibilities:
  - Instead of sorting by α, do a rank-space reduction on α to reveal the α-ties and perform Fenwick tree queries and updates in batches. You may have to do multiple queries/updates on the Fenwick tree for each batch, to cover different tie cases.
  - When you are doing the main sort on α, you can use β and γ as additional keys to ensure that solutions are inserted to the Fenwick tree in the dominance order in the case of an α tie; however, you still need a way to deal with the equivalent (non-dominating) solution pairs.
- 80% of the grading inputs will guarantee to have distinct α, β, γ values. You will not need tie-breaking to get those points.
- As usual, if you cannot code the aforementioned algorithm, please code a slower one to get partial points.