CS29003 ALGORITHMS LABORATORY
ASSIGNMENT 10
Date: 07th Nov, 2021

Important Instructions

1. **Input:** To be taken from `stdin`
2. **Format of files to be submitted to Moodle and HackerRank:** <ROLLNO>_A10_scheduler.c(pp), <ROLLNO>_A10_mod_scheduler.c(pp)
3. You are to **stick to the input output formats strictly** as per the instructions in the respective problems.
4. Write your name, roll number and HackerRank ID at the beginning of each of your programs.
5. Do not use any global variable or STL unless you are explicitly instructed so.
6. Indent and comment your code properly.
7. Please follow all the guidelines. Failing to do so will result in penalty.
8. **There will be partial markings.**

Heaps

Problem: TAPASYA

You are Vaibhav, student of Jeetu Bhaiya who is starting his own institute after leaving PRODIGY and he needs your help organizing a workshop- **TAPASYA** for his students. Your friends agreed to help Jeetu Bhaiya to organize the workshop by teaching a topic of their choice.

- ▷ You have **N** friends and the workshop runs for **D** days.
- ▷ Unfortunately, there can be at most one lecture in a day since students need time to revise other subjects as well and you need to schedule the respective lectures accordingly.
- ▷ Your i**-th** friend arrives on day d_i and then stays till the end of the workshop and there was an agreement that he/she would take t_i lectures.
- ▷ Your friend gets paid additional c_i rupees for each lecture that he/she was promised in the agreement but could not be scheduled in the workshop due to the exhaustion of the **D** days.
- ▷ Since you are an algorithms expert, Jeetu Bhaiya asks you to to come up with a schedule that minimizes such additional costs.
- ▷ Formally, if your i**-th** friend gets k_i lectures then the total additional cost incurred would be :

$$\sum_{i=1}^{n} (t_i - k_i) * (c_i)$$

Your task is to minimize this total additional cost.

Hint : Greedy Algorithm Let S denotes the set of friends that have arrived till day-i, pick the friend that has the maximum additional cost requirement from the set S. If there are two friends with the same additional cost, the one with lower pID(unique identifier for each friend) is to be chosen. An efficient implementation of this greedy algorithm can be achieved by a scheduler maintaining a priority queue of individuals waiting for their lectures to be scheduled using the max-heap implementation with the key being the additional cost.

Part-(a) : Implement the scheduler

The scheduler is implemented as a function :

$$void\ scheduler(person\ personList[], int\ n)$$

The function takes as parameter a list of n persons in an array personList[], and schedules them according to the policy specified earlier. The function prints the pID for the lecture scheduled in workshop for every day until the workshop ends. You are free to define your $person$ struct with necessary parameters. Your function should run in $O(nlogn + D)$ time, where D is duration of the workshop.

Input Format

- ▷ The first line will have 2 space-separated integers N ($1 \leqslant N \leqslant 10^5$) and D ($1 \leqslant D \leqslant 10^5$) where N and D denote the number of friends and duration of workshop.
- ▷ Then N lines follow each having information about your friend. Each line has 4 space-separated integers pID, d_i, t_i and c_i. ($1 \leqslant d_i, t_i \leqslant D$) and $1 \leqslant c_i \leqslant 10^5$.
- ▷ Additionally $\sum_{i=1}^{n} t_i \geqslant D$ which means that it is always feasible to organize the workshop.

Output Format

- ▷ First line should have an integer denoting the minimum incurred additional cost.
- ▷ Then the second line should contain the schedule as D space separated integers denoting the pID who took the lecture on the day d_i of the workshop.

Sample Cases

FILE: *input-problem1.txt*

```
2 3
1 1 2 300
2 2 2 100
```

FILE: *output-problem1.txt*

```
100
1 1 2
```

Explanation

For the first day we have [person-1] available for taking a lecture.

For day-2, we have [pID=1,2] but the cost is maximum for [pID=1], hence he takes the lecture on that day.

For day-3 we only have [pID=2], since person-1 has already completed his lectures.

Hence the additional cost incurred would be : $300 * (2-2) + 100 * (2-1) = 100$ and the schedule for the workshop would be - $1, 1, 2$.

FILE: *input-problem2.txt*

```
3 3
1 1 2 5
2 2 1 10
3 3 2 3
```

FILE: *output-problem2.txt*

```
6
1 2 1
```

Explanation

For the first day we have [person-1] available for taking a lecture.

For day-2, we have [pID=1,2] but the cost is maximum for [pID=2], hence he takes the lecture on that day.

For day-3 we only have [pID=1,3], since person-2 has already completed his lectures. We allot the 3rd day lecture to person-1

Hence the additional cost incurred would be : $5 * (2-2) + 10 * (1-1) + 3 * (2-0) = 6$ and the schedule for the workshop would be - $1, 2, 1$.

Part-(b) : Implement the modified scheduler

Now suppose that all the friends are not independent, and there exist some pairs of friends (x, y) such that if x has delivered all his lectures before y even starts(note : y has to be present in the heap), it will increase the additional cost c_y (key of heap) denoted by a value $func(y)$ but if y has already delivered atleast one lecture, then nothing is done.

$$func(y) = \sum_{j \in heap} cost(j) \ni ancestor(j) = y$$

$$c_y = c_y + func(y)$$

$func(y)$ denotes the sum of additional costs present in the subtree of node in the heap where the current friend y is located. We call such a pair a dependency pair. You can assume that there can be $O(n)$ such pairs given. You will have to implement a new function $increasekey(pid)$ which takes the person id pid and updates the additional cost c_y using the equation mentioned above. Please refer to the example for better understanding.

Your job is to schedule the workshop using the same greedy policy mentioned in part-(a), subject to the change mentioned above. However, note that the additional cost for an individual already in the heap can change in the middle if another person that he depends on has completed all his lectures. Your function should still run in $O(nlogn + T)$ time. To do this, you may have to change the definition of the heap and the associated functions(e.g, $insertperson()$, $extractMax()$) and add a new function $increasekey(pid)$. You have to maintain the modified heap property(e.g, store the sum of corresponding subtree at each node as another parameter explained in the figure) after insertion and removal of individuals from the heap in order to achieve the $O(logn)$ complexity for heap operations.

Input Format

- ▷ The first line will have 3 space-separated integers N ($1 \leqslant N \leqslant 10^5$) and D ($1 \leqslant D \leqslant 10^5$) and P ($1 \leqslant P \leqslant N$) where N, D and P denotes the number of friends, total days of workshop and dependency pairs respectively.
- ▷ Then N lines follow each having information about a person. Each line has 4 space-separated integers pID, d_i, t_i and c_i. ($1 \leqslant d_i, t_i \leqslant D$) and $1 \leqslant c_i \leqslant 10^5$.
- ▷ Then P lines follow denoting the dependency pairs x_i, y_i where x_i, y_i can occur only once in this list of dependencies.
- ▷ Additionally $\sum_{i=1}^{n} t_i \geqslant D$ which means that it is always feasible to organize the workshop.

Output Format

- ▷ First line should print an integer denoting the incurred additional cost.
- ▷ Then the second line should contain D space separated integers denoting the pID who took the lecture on the day d_i of the workshop.

FILE: *input0-problem1.txt*

```
5 5 1
1 1 2 7
2 1 4 5
3 2 1 10
4 3 3 7
5 1 5 4
1 4
```

Let us denote the person-ids $\texttt{pids} = [1, 5]$ with alphabets $[a, e]$ for the given input.

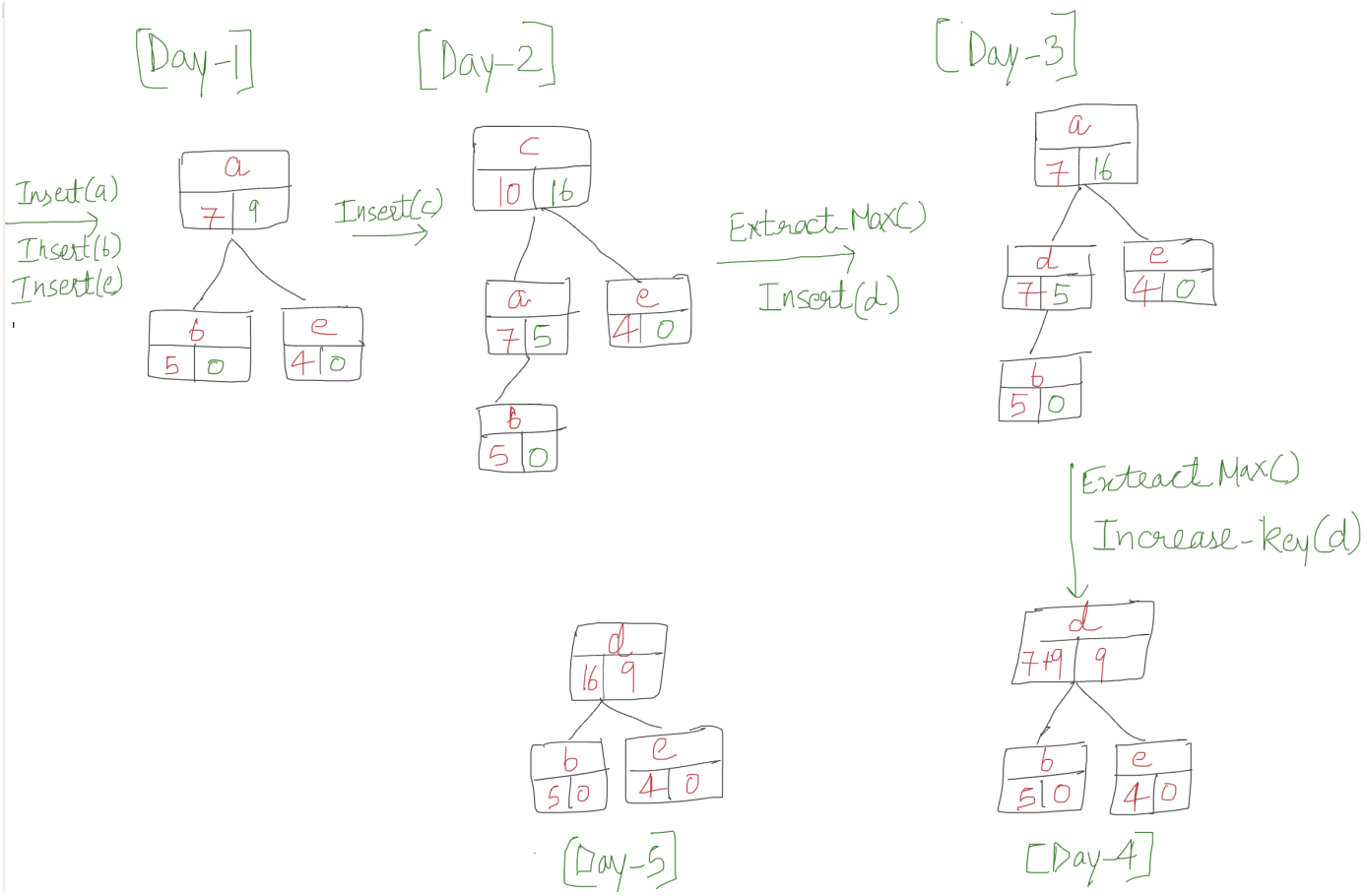

Figure 1: Each node in the heap maintains a (k,s) pair where k and s denotes the key and sum of the keys in the sub-tree of that node respectively.

FILE: *output0-problem1.txt*

```
56
1 3 1 4 4
```

Explanation

Note that the individuals entering the Heap must be first sorted by d_i (arrival day) and then by pID. **The order of insertion is important**.

Note : The operations to be performed must follow this order : extractMax()$->$increaseKey()$->$insert(), if any of that exist for that particular day.

For day-1 after inserting pIDs[a,b,e], we have [pID-a] at the top of the queue for taking a lecture.

For day-2, we first insert [pID=c] and since person-c is at the top of the queue, he takes the lecture on that day.

For day-3, we first extract the friend with pID=c and then insert the friend with pID=d. Since pID=a is on the top, we allocate the 3rd day slot to him.

For day-4, we first extract the individual with pID=a from the heap and then perform the increase-key operation on individual with pID=d. The key for person($pid = d$) becomes now $c_y(d) = 7 + 9 = 16$. Finally pID=d takes the lecture on day-4.

For the day-5, we have the individual with pID=d at the top of the queue for taking a lecture.

Hence the additional cost incurred would be : $7*(2-2)+5*(4-0)+10*(0-0)+16*(3-2)+4*(5-0) = 56$ and the schedule for the workshop would be - $1, 3, 1, 4, 4$.