CodeForces

Contest 1374

Problem E2

@mrkhosravian

Problem

Summer vacation has started so Alice and Bob want to play and joy, but... Their mom doesn't think so. She says that they have to read **exactly** m books before all entertainments. Alice and Bob will read each book **together** to end this exercise faster.

There are n books in the family library. The i-th book is described by three integers: t_i — the amount of time Alice and Bob need to spend to read it, a_i (equals 1 if Alice likes the i-th book and 0 if not), and b_i (equals 1 if Bob likes the i-th book and 0 if not).

So they need to choose **exactly** m books from the given n books in such a way that:

- Alice likes **at least** k books from the chosen set and Bob likes **at least** k books from the chosen set;
- the total reading time of these mm books is **minimized** (they are children and want to play and joy as soon a possible).

The set they choose is **the same** for both Alice an Bob (it's shared between them) and they read all books **together**, so the total reading time is the sum of t_i over all books that are in the chosen set.

Your task is to help them and find any suitable set of books or determine that it is impossible to find such a set.

Input

The first line of the input contains three integers n, m and k ($1 \le k \le m \le n \le 2 * 10^5$).
The next n lines contain descriptions of books, one description per line: the i-th line contains three integers t_i, a_i and b_i ($1 \le t_i \le 10^4$, $0 \le a_i, b_i \le 1$), where:

- t_i — the amount of time required for reading the i-th book;
- a_i equals 1 if Alice likes the i-th book and 0 otherwise;
- b_i equals 1 if Bob likes the i-th book and 0 otherwise.

Output

If there is no solution, print only one integer -1.

If the solution exists, print T in the first line — the minimum total reading time of the suitable set of books. In the second line print m distinct integers from 1 to nn in any order — indices of books which are in the set you found.

If there are several answers, print any of them.

Example 1

Input:

```
6 3 1
6 0 0
11 1 0
9 0 1
21 1 1
10 1 0
8 0 1
```

Output:

```
24
6 5 1
```

Solve example 1


06:00



11:00



09:00




21:00



10:00



08:00

Alice  Bob 

There are 6 books
Alice and Bob both likes 1 books

There are four types of book:

1. 0 0 - both Alice and Bob doesn't like these books — books { 1 }
2. 1 0 - only Alice likes these books — books { 5, 2 }
3. 0 1 - only Bob likes these books — books { 6, 3 }
4. 1 1 - both Alice and Bob like these books — books { 4 }

Now sort these groups by time:

1. { 06:00 }
2. { 10:00, 11:00 }
3. { 08:00, 09:00 }
4. { 21:00 }

Solve example 1


06:00



11:00



09:00




21:00



10:00



08:00

Alice  Bob 

1. First we create **BOOKS** set.
2. Then for total k we add books to our BOOKS set.

Here k is 1. Then we just add one book for each of them or add a mutual book for both.

Now we check if we should add a book that is mutual with both of them or add separate books for each one (alice and bob).

Check which one is smaller.

- Mutual book = 21:00
- Alice = 10:00 , Bob 08:00

So 10:00 + 08:00 < 21:00 , then we add separate books for each one. # 10 + 8 = 18

Final BOOKS set will be:
BOOKS = { 10:00, 08:00 } # not important to be sorted.

3. Now we fixed books that alice and bob likes, but BOOKS set has 2 books and we need 3 books to read.

Solve example 1


06:00



11:00



09:00




21:00



10:00



08:00

Alice  Bob 

4. So we must add 1 other book to make BOOKS size 3.
5. There 4 possible ways to add a book to our BOOKS set.

- Add a book that none of them like it.
- Add a book that just Alice likes it.
- Add a book that just Bob likes it.
- Remove A book that both likes it from BOOKS set and add 2 book for alice and bob separately.

We must choose the one that is smallest. If There wasn't any, there is no answer then return -1.

Here book 1 has the smallest time and we add it to BOOKS set.

BOOKS set is now = { 10:00, 08:00, 06:00 }

The Answer is the sum of BOOKS set time and their index.
10 + 8 + 6 = 24
indexes = 5 , 6 , 1

Now go to code

Initializing

* fr (fast reader) is a custom scanner that uses Java BufferedReader

```
private void solve() {
    final int n = fr.nextInt(), m = fr.nextInt(), k = fr.nextInt();

    List<Book> groupA = new ArrayList<>(),
            groupB = new ArrayList<>(),
            groupAB = new ArrayList<>(),
            groupNotAB = new ArrayList<>();

    for (int index = 0; index < n; index++) {
        int time = fr.nextInt(), a = fr.nextInt(), b = fr.nextInt();
        Book book = new Book(index + 1, time);
        if (a == 1 && b == 1) groupAB.add(book);
        else if (a == 1) groupA.add(book);
        else if (b == 1) groupB.add(book);
        else groupNotAB.add(book);
    }

    Collections.sort(groupA);
    Collections.sort(groupB);
    Collections.sort(groupAB);
    Collections.sort(groupNotAB);

    Set<Book> books = readingBooks(groupNotAB, groupA, groupB, groupAB, m, k);
    printAns(books);
}
```

Create Books class

```
static class Book implements Comparable<Book> {

    private final int index;
    private final int time;

    public Book(int index, int time) {
        this.index = index;
        this.time = time;
    }

    @Override
    public int compareTo(Book o) {
        return this.time - o.time;
    }

}
```

Create Action class

```
static class Action {

    private final String name;
    private final Integer value;

    public Action(String name, Integer value) {
        this.name = name;
        this.value = value;
    }

}
```

Find the best action and beats method

```
private String findTheBestAction(ArrayList<Action> integers) {

    Optional<Action> min = integers
      .stream()
      .filter(item -> item.value != null)
      .min(Comparator.comparingInt(a -> a.value));

    return min.map(action -> action.name).orElse("null");
}


private boolean beats(Integer a, Integer b) {
    return b == null || (a != null && a < b);
}
```

Reading Books Algorithm 1/5

```
private Set<Book> readingBooks(List<Book> groupNotAB, List<Book> groupA,
                               List<Book> groupB, List<Book> groupAB,
                               int m, int k)
{

    int ai = 0; // group A index
    int bi = 0; // group B index
    int abi = 0; // group AB index
    int noni = 0; // group not AB index

    Set<Book> books = new HashSet<>();


    /* ********************************************************************** */

    // EveryThing will be done here

    /* ********************************************************************** */


    return books;
}
```

Reading Books Algorithm 2/5

```
for (int i = 1; i <= k; i++) {
    Integer both = (groupAB.size( ) > abi) ? groupAB.get(abi).time : null;
    Integer separate = (groupA.size( ) > ai && groupB.size( ) > bi)
      ? groupA.get(ai).time + groupB.get(bi).time
      : null;

    if (both == null && separate == null) return null;

    if (beats(both, separate)) {
        books.add(groupAB.get(abi++));
    } else {
        books.add(groupA.get(ai++));
        books.add(groupB.get(bi++));
    }
}
```

Reading Books Algorithm 3/5

```
int booksDiff = Math.abs(m - books.size()); // books difference

if (m < books.size()) {
    // remove books
    for (int i = 1; i <= booksDiff; i++) {
        if (ai == 0 || bi == 0 || groupAB.size() <= abi) return null;
        books.remove(groupA.get(--ai));
        books.remove(groupB.get(--bi));
        books.add(groupAB.get(abi++));
    }
} else {
    // add more books

    /* ******************************** */

    // here we will add more books to our list

    /* ******************************** */
}
```

Reading Books Algorithm 4/5

```
Action swap = new Action( "swap", (groupA.size() > ai && groupB.size() > bi && abi > 0)
                ? groupA.get(ai).time + groupB.get(bi).time - groupAB.get(abi - 1).time
                : null
);
Action addNone = new Action( "addNone", (groupNotAB.size() > noni)
                ? groupNotAB.get(noni).time
                : null
);
Action addA = new Action( "addA", (groupA.size() > ai)
                ? groupA.get(ai).time
                : null
);
Action addB = new Action( "addB", (groupB.size() > bi)
                ? groupB.get(bi).time
                : null
);
Action addBoth = new Action( "addBoth", (groupAB.size() > abi)
                ? groupAB.get(abi).time
                : null
);
```

Reading Books Algorithm 5/5

```
    String action = findTheBestAction(new ArrayList<Action>() {{
        add(swap);
        add(addNone);
        add(addA);
        add(addB);
        add(addBoth);
    }});

    switch (action) {
        case "null":  return null;
        case "swap":
            books.remove(groupAB.get(--abi));
            books.add(groupA.get(ai++));
            books.add(groupB.get(bi++));
            break;
        case "addNone": books.add(groupNotAB.get(noni++));  break;
        case "addA":    books.add(groupA.get(ai++));        break;
        case "addB":    books.add(groupB.get(bi++));        break;
        case "addBoth": books.add(groupAB.get(abi++));      break;
    }

}
```

Finally Print the answer

```
private void printAns(Set<Book> books) {
    if (books == null)
        System.out.println(-1);
    else {
        int time = 0;
        StringBuilder s = new StringBuilder();
        for (Book b : books) {
            time += b.time;
            s.append(b.index).append(" ");
        }
        System.out.printf("%d\n%s\n", time, s);
    }
}
```

Fast Reader – (skip if you just use java.util.Scanner)

```
class FastReader {
    private final BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    private StringTokenizer st;

    public String nextLine() {
        try {
            return br.readLine();
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
    }

    public String next() {
        while (st == null || !st.hasMoreTokens()) {
            st = new StringTokenizer(nextLine());
        }
        return st.nextToken();
    }

    public int nextInt() {
        return Integer.parseInt(next());
    }
}
```

Now You can solve the problem

FINISHED

I hope u enjoyed.

@mrkhosravian

Code Images: carbon.now.sh
Slides: slidesgo.com