Hotel Pak Asep

Worksheet 5

Deskripsi

Pak Asep baru saja memiliki sebuah hotel. Namun karena keterbatasan modal, hotel yang ia miliki belum memiliki kamar. Karena itu, ia berencana untuk membangun sebuah kamar dengan luas L_i setiap harinya.

Pak Asep yang memiliki modal pas-pasan tentu tidak akan menolak apabila seorang pelanggan ingin menginap di hotel miliknya. Berdasarkan pengamatan Pak Asep, hanya terdapat dua jenis pelanggan.

1. Pelanggan yang ingin menginap dengan luas kamar tidak kurang dari $L_{i.}$
2. Pelanggan yang ingin menginap di kamar terbesar yang ada.

Demi kenyamanan pelanggan, Pak Asep tidak akan membangun sebuah kamar apabila terdapat pelanggan yang menginap. Tetapi karena ia tidak ingin pembangunannya terganggu terlalu banyak, dalam satu hari ia hanya menerima seorang pelanggan yang menginap tepat satu hari.

Singkatnya, yang terjadi pada hotel Pak Asep dalam satu hari adalah salah satu dari hal-hal berikut.

1. Pak Asep membangun kamar dengan luas L_i
2. Terdapat pembeli yang ingin menginap dengan luas kamar tidak kurang dari L_i. Jika ada kamar dengan luas L_i, maka ia akan menginap di kamar tersebut. Jika tidak ada, ia akan menginap di kamar dengan luas lebih besar, yang paling dekat dengan L_i. Jika tidak ada kamar dengan luas lebih besar, maka ia tidak jadi menginap.
3. Terdapat pelanggan yang ingin menginap di kamar terbesar. Jika pada saat itu belum ada kamar yang dibangun, maka ia tidak jadi menginap.

Pak Asep yang bukan lulusan fasilkom tidak bisa membuat program. Untuk itu, ia meminta bantuan Anda membuatkan sebuah program yang dapat mencatat luas kamar yang ada dan mencarikan kamar yang sesuai dengan keinginan pelanggan.

Karena kamar dan pelanggan yang ada bisa sangat banyak, implementasi menggunakan struktur data linear seperti ArrayList dapat mengakibatkan Time Limit Exceeded karena memiliki kompleksitas O(N) per operasi.

Anda diharapkan untuk menyimpan luas kamar yang ada dengan Binary Search Tree agar operasi insert dan find dapat dilakukan dengan kompleksitas rata-rata O(log N) per operasi.

Catatan: Asisten akan mengecek kode yang Anda buat. Jika Anda tidak melakukan implementasi binary search tree, Anda akan langsung mendapatkan nilai 0 (nol).

Format Masukan

Baris pertama berisi sebuah bilangan bulat N, yaitu banyaknya hari yang dilewati.
N baris berikutnya berisi salah satu dari format berikut:

- 1 L_i, yang artinya Pak Asep membangun sebuah kamar dengan luas L_i.
- 2 L_i, yang artinya terdapat pelanggan yang ingin menginap dengan luas kamar tidak kurang dari L_i.
- 3, yang artinya terdapat pelanggan yang ingin menginap di kamar dengan luas terbesar.

Format Keluaran

Untuk setiap masukan jenis 2 dan 3, keluarkan sebuah baris berisi sebuah bilangan bulat yang menyatakan luas kamar yang ditempati. Jika pelanggan tidak mendapatkan kamar, keluarkan -1.

Batasan

Untuk semua testcase, berlaku:

- $1 \leq N \leq 50.000$
- $1 \leq L_i \leq 10^9$

Untuk 2/3 (66.666666666666666666…%) dari semua testcase, berlaku:

- Jenis event yang ada hanya berjumlah dua, yaitu:
  - 1 dan 2, atau
  - 1 dan 3.

  Perlu diperhatikan bahwa yang berjumlah dua adalah jenis event, bukan hari.

Contoh 1

Masukan

```
10
1 6
2 7
1 9
2 7
3
1 12
1 8
3
2 7
2 8
```

Keluaran

```
-1
9
9
12
8
8
```

Penjelasan

1. Kamar = { 6 }.
2. Tidak ada kamar dengan luas tidak kurang dari 7. Pelanggan tidak jadi menginap.
3. Kamar = { 6, 9 }.
4. Tidak ada kamar dengan luas 7. Luas kamar yang lebih besar dan paling dekat dengan 7 adalah 9. Pelanggan menginap di kamar dengan luas 9.
5. Luas kamar terbesar adalah 9. Pelanggan menginap di kamar dengan luas 9.
6. Kamar = { 6, 9, 12 }.
7. Kamar = { 6, 8, 9, 12 }.
8. Luas kamar terbesar adalah 12. Pelanggan menginap di kamar dengan luas 12.
9. Tidak ada kamar dengan luas 7. Luas kamar yang lebih besar dan paling dekat dengan 7 adalah 8. Pelanggan menginap di kamar dengan luas 8.
10. Ada kamar dengan luas 8. Pelanggan menginap di kamar tersebut.

Template

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Hotel {
    public static void main(String args[]) throws IOException {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

        int N = Integer.parseInt(reader.readLine());
        BinarySearchTree bst = new BinarySearchTree();

        for (int i = 0; i < N; i++) {
            String inputs[] = reader.readLine().split(" ");
            if (Integer.parseInt(inputs[0]) == 1) {
                bst.insert(Integer.parseInt(inputs[1]));
            } else if (Integer.parseInt(inputs[0]) == 2) {
                System.out.println(bst.find(Integer.parseInt(inputs[1])));
            } else if (Integer.parseInt(inputs[0]) == 3) {
                System.out.println(bst.findMax());
            }
        }
    }
}

class Node implements Comparable<Node> {
    public Integer value;
    public Node leftChild;
    public Node rightChild;

    public Node(Integer value) {
        this.value = value;
        this.leftChild = this.rightChild = null;
    }

    public int compareTo(Node other) {
        return value.compareTo(other.value);
    }
}

class BinarySearchTree {
    private Node root;

    BinarySearchTree() {
        this.root = null;
    }

    private Node insert(Node node, Node newNode) {
        // TODO: Implement insertion here
    }

    public void insert(int value) {
        this.root = insert(root, new Node(value));
    }

    private Node findMax(Node node) {
        // TODO: implement find Max here
    }
```

```
    public int findMax() {
        return findMax(root).value;
    }

    private Node find(Node node, Node searchNode) {
        // TODO: Implement find here
    }

    public int find(int value) {
        return find(root, new Node(value)).value;
    }
}
```