**Ví dụ**

**Input**

5

3 1 2 3

2 1 1

4 10 5 5 1

0

1 2

**Output**

19

**Box:** 1 2 3

**Day 1:** 3 - 1 = 2

**Box:** 1 1 2

**Day 2:** 2 - 1 = 1

**Box:** 1 1 5 5 10

**Day 3:** 10 - 1 = 9

**Box:** 1 5 5

**Day 4:** 5 - 1 = 4

**Box:** 2 5

**Day 5:** 5 - 2 = 3

⇒ 2 + 1 + 9 + 4 + 3 = 19

Với mỗi ngày: Lấy ra max và min

Muốn lấy max: **maxHeap**

Muốn lấy min: **minHeap**

**\*\*\***Tuy nhiên, không được lấy trùng.

Để kiểm soát được một cái bill đã được dùng hay chưa → đánh dấu lại

Mỗi cái bill sẽ được đánh dấu bằng 1 chỉ số, số thứ tự của bill đó.

→ Đánh dấu cái bill mang số thứ tự id đã được dùng, thì khi mà gặp lại cái bill đó ở cái heap còn lại → Bỏ nó đi

**Giải thuật**

**Bước 1:** Nhập dữ liệu

**Bước 2:** Khai báo một maxHeap và một minHeap để lưu trữ các hóa đơn, một cái mảng đánh dấu

**Bước 3:** Mỗi ngày, đưa các hóa đơn trong ngày vào maxHeap và minHeap

**Bước 4:** Lấy ra một hóa đơn chưa được sử dụng trong maxHeap và một hóa đơn chưa được sử dụng trong minHeap, tính chênh lệch và đánh dấu lại

**Bước 5:** Đưa ra kết quả

**Mã giả:**

**class bill:**

**value, id;**

**taken = []; // Mảng đánh dấu, ban đầu tất cả bằng false, 10^6 + 1 phần tử**

**maxHeap = []; // maxHeap**

**minHeap = []; // minHeap**

**money = 0; // Biến lưu kết quả**

**nbills = 0; // Đánh dấu index của bill**

**read(n);**

**for day = 1 to n: {**

**read(k);**

**for i = 1 to k: {**

**read(x);**

**nbills += 1;**

**maxHeap.push(bill(x, nbills));**

**minHeap.push(bill(x, nbills));**

**}**

**while taken[maxHeap.top.id] == true:**

**maxHeap.pop;**

**while taken[minHeap.top.id] == true:**

**minHeap.pop;**

**taken[minHeap.top.id] = true;**

**taken[maxHeap.top.id] = true;**

**money += maxHeap.top.value - minHeap.top.value;**

**minHeap.pop;**

**maxHeap.pop;**

**}**

**print(money);**

**Độ phức tạp: O(M\*logM)**

**M là tổng số lượng bill**