**CSGE602040 - Struktur Data dan Algoritma**
**Semester Genap - 2020/2021**
**Lab 3**
**Deadline: Jumat, 30-Apr-2021, 23.55 WIB**

---

# Pak Ustadz Berbagi Takjil

## Deskripsi

Di bulan Ramadhan 1442 H ini, Pak Ustadz memiliki hobi baru yaitu membuat takjil untuk berbuka puasa. Pak Ustadz menekuni hobi tersebut di dapur rumahnya sendiri. Pak Ustadz dapat membuat segala jenis takjil hanya dengan menonton *tutorial* masak makanan tersebut di YouTube. Sebelum Pak Ustadz membuat takjilnya, beliau ingin mengumpulkan informasi jarak tetangga-tetangganya yang akan menerima takjil yang dibuat Pak Ustadz. Lalu Pak Ustadz ingin melihat bagaimana informasi jarak tersebut dalam bentuk *Binary Search Tree* (BST).

Setelah itu, Pak Ustadz membagi kloter pengantaran takjil ke **N** buah BST. Lalu, Pak Ustadz akan memasukkan jarak rumah penerima pertama sebagai *root tree,* yaitu jarak relatif penerima pertama dengan rumah Pak Ustadz. Penerima selanjutnya akan dihitung jarak relatifnya terhadap penerima sebelumnya. Setelah Pak Ustadz memasukkan jarak rumah mereka, Pak Ustadz akan menelpon tetangganya untuk mengkonfirmasi apakah mereka sedang di rumah atau tidak. Jika mereka sedang tidak berada dirumah, maka jarak rumah mereka dari rumah Pak Ustadz akan dihapus dari data kloter pengantaran takjil. Asumsikan kegiatan menghapus jarak dari data kloter pengantaran takjil akan dilakukan setelah BST telah berhasil dibuat.

Berikut *query-query* yang akan dijalankan:

- **REMOVE $a_i$**: Menghapus jarak relatif rumah tetangga dari rumah tetangga sebelumnya dari BST. Pada *query* ini, **$a_i$ dijamin *valid*** atau ada di dalam *tree*. Proses *remove node* menggunakan **kaidah *Succesor Inorder***.

Pak Ustadz meminta bantuan kepada kamu, yang merupakan *programmer* handal, untuk memberikan rute pengantaran takjil terbaik sehingga Pak Ustadz bisa membuat takjil dengan khidmat.

## Masukan

Baris pertama berisi *integer* **N** yang menyatakan banyaknya *Binary Search Tree* (BST) yang akan dieksekusi.

Untuk setiap BST, terdapat **2 + M** baris berisi:

- *Integer* **P** yang menyatakan banyaknya *node* jarak relatif pada *tree* tersebut dan diikuti oleh **P** buah *integer* **$a_1, a_2, \ldots, a_p$** yang merupakan jarak yang telah dicatat oleh Pak Ustadz dalam bentuk *sequence* **PostOrder Traversal** BST. PostOrder ini nantinya **dijamin** akan membentuk BST.
- *Integer* **M** yang menyatakan banyaknya *query* **REMOVE $a_i$**.

## Keluaran

Keluaran terdiri dari **N** buah *integer* yang menyatakan tinggi dari tree tersebut setelah dieksekusi. Keluarkan **-1** jika *tree* tersebut kosong (memiliki 0 buah *node*).

**Batasan**

$1 \leq N \leq 100000$

$1 \leq P \leq 100$

$0 \leq M \leq P$

$1 \leq a_i \leq 10^6$

**Contoh Masukan 1**

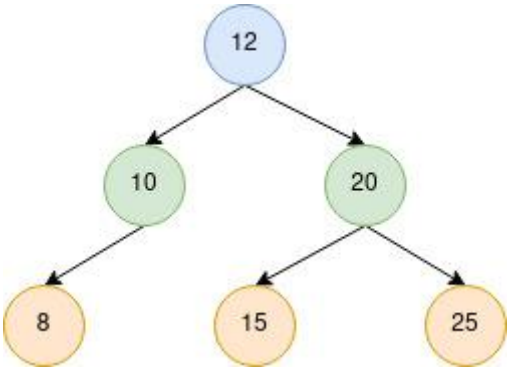
```
1
6 8 10 15 25 20 12
2
REMOVE 10
REMOVE 8
```

**Contoh Keluaran 1**

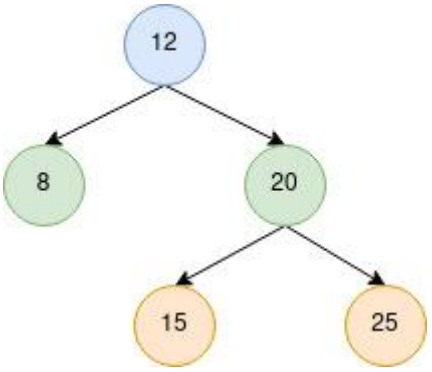
```
2
```

**Penjelasan 1**

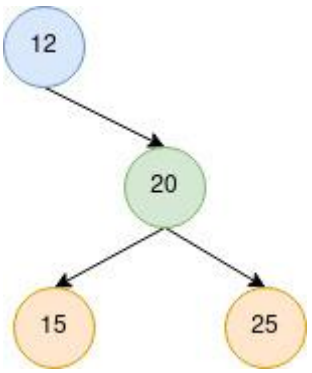
*Tree* yang memiliki postOrder Traversal tersebut adalah sebagai berikut:




Setelah itu, jika kita menjalankan *query* **REMOVE 10**, maka *tree* akan berubah menjadi seperti ini:




Setelah itu, jika kita menjalankan *query* **REMOVE 8**, maka *tree* akan berubah menjadi seperti ini:




Gambar di atas merupakan bentuk akhir dari *tree* tersebut. Dari gambar diatas, kita dapat mengetahui bahwa tinggi dari *tree* tersebut adalah **2**.

**Contoh Masukan 2**

```
2
5 2 8 17 10 3
1
REMOVE 10
3 20 50 30
3
REMOVE 30
REMOVE 20
REMOVE 50
```

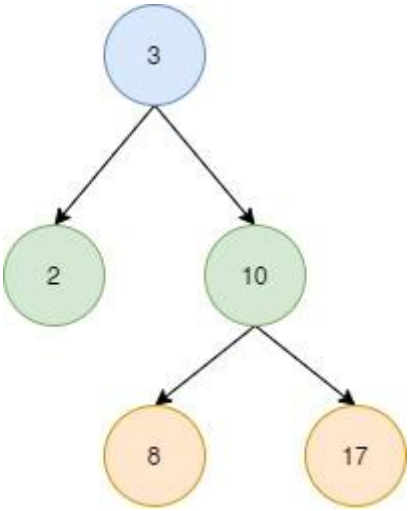
**Contoh Keluaran 2**

```
2
-1
```

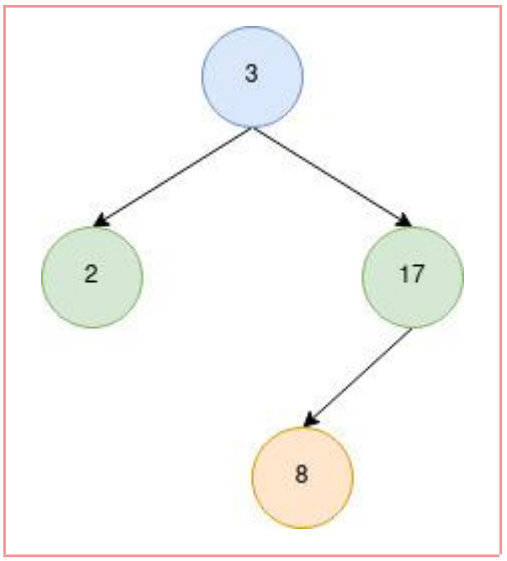
**Penjelasan 2**

**Tree 1**

Untuk *tree* pertama yang memiliki PostOrder Traversal tersebut adalah sebagai berikut:



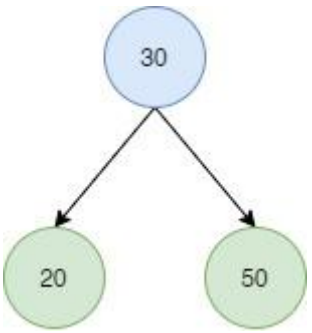

Setelah itu, jika kita menjalankan *query* **REMOVE 10**, maka *tree* akan berubah menjadi seperti ini:



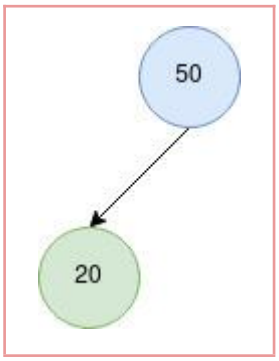

Dari bentuk akhir *tree* pertama, kita dapat mengetahui bahwa tinggi dari *tree* tersebut adalah **2**.

**Tree 2**

Untuk *tree* kedua yang memiliki PostOrder Traversal tersebut adalah sebagai berikut:




Setelah itu, jika kita menjalankan *query* **REMOVE 30**, maka *tree* akan berubah menjadi seperti ini:




Setelah itu, jika kita menjalankan *query* **REMOVE 20**, maka *tree* akan berubah menjadi seperti ini:


Setelah kita menjalankan *query* **REMOVE 50**, maka *tree* akan menjadi kosong. Karena isi akhir tree kedua adalah kosong, maka kita dapat mengetahui bahwa tinggi dari tree tersebut adalah **-1**.