Divine Binary Search Tree

Author: Mathias Adya Diwangkara Suganda

Time Limit	1 s
Memory Limit	256 MB

Sebuah *tree* dikatakan *Binary Search Tree* (BST) jika untuk setiap node *u*, nilai pada node *u* lebih besar dari semua nilai di subtree kiri *u* dan lebih kecil dari semua nilai di substree kanan *u*.

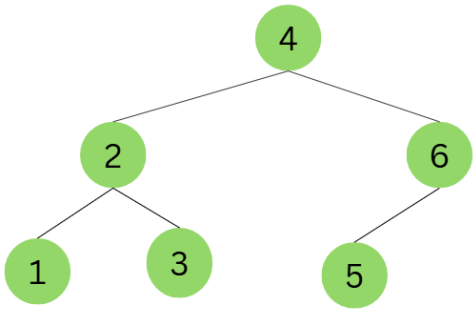
Sedikit modifikasi akan dibuat pada BST. *Tree* ini akan kita sebut sebagai *Divine Binary Search Tree* (DBST).

DBST adalah sebuah BST di mana untuk setiap level (kecuali level terakhir), semua cabang akan diisi secara penuh. Aturan ini tidak berlaku untuk semua node yang berada pada level terakhir (leaf nodes). Semua DBST adalah BST, namun tidak sebaliknya.

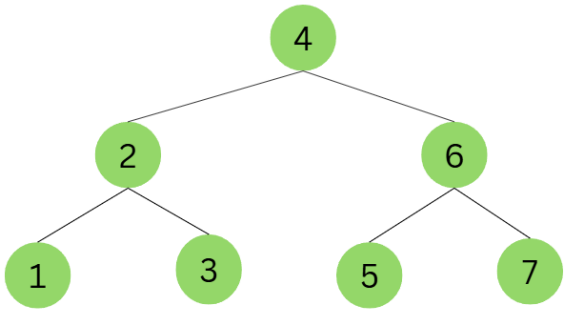
Untuk level terakhir, DBST memiliki 2 aturan:

1. Setiap node leaf harus terisi dari kiri dahulu.
2. Node leaf terakhir **bisa** tidak memiliki sibling.

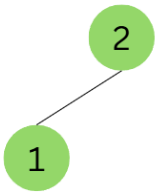
Berikut adalah beberapa contoh dari DBST yang **benar**.



Example 1

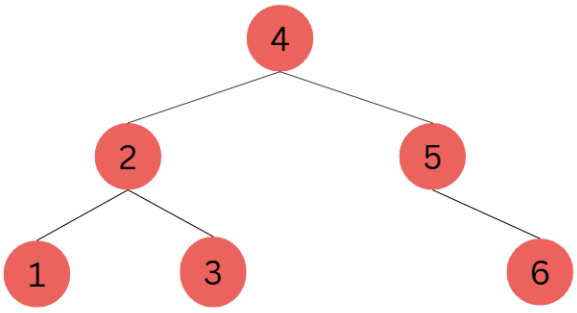


Example 2

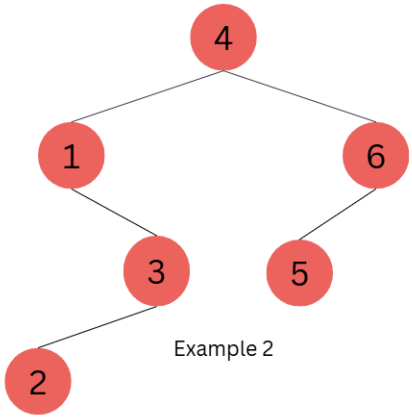


Example 3

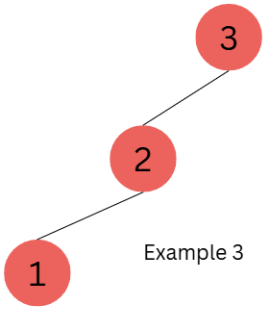
Berikut adalah beberapa contoh dari DBST yang **salah,** tapi masih merupakan BST.



Example 1

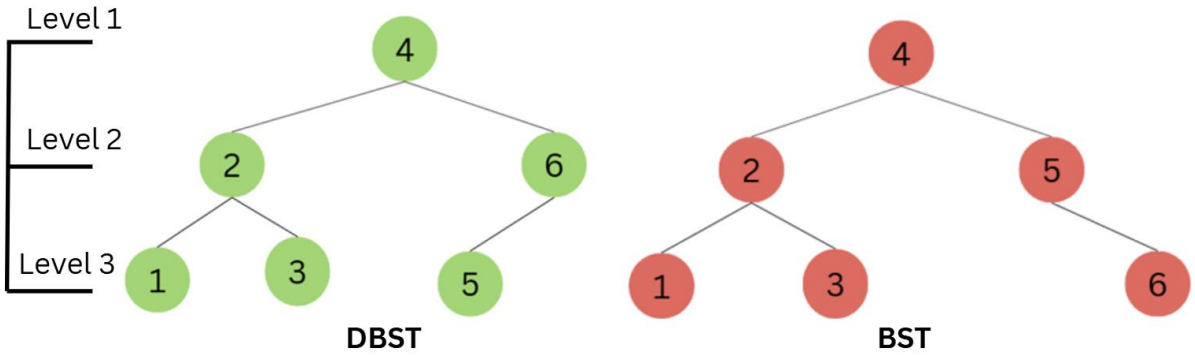


Example 2



Example 3

Untuk perbedaan yang lebih jelas, kita dapat mengamati 2 BST dibawah.



Di bagian kiri, node 5 terletak pada level 3 dan node 6 terletak pada level 2. Penempatan pada bagian kiri (DBST) membuat semua leaf node terisi dari kiri dahulu. Namun, di bagian kanan (BST) penempatan node 5 dan 6 terbalik dengan tree bagian kiri, sehingga subtree kiri dari node 5 tidak terisi, membuat semua leaf nodes tidak terisi dari kiri dahulu.

Diberikan sebuah array *a* berisi *n* banyak elemen, susunlah *Divine Binary Search Tree* yang diiisi semua node nya dengan elemen-elemen pada array *a*.

Format Masukan

Masukan terdiri dari dua baris.

Baris pertama berisi sebuah satu bilangan *n* ($1 \leq n \leq 2 \times 10^5$).

Baris kedua berisi elemen-elemen dari array *a* sebanyak *n* bilangan ($1 \leq a_i \leq 10^6$).

Dipastikan bahwa setiap elemen pada array *a* memiliki nilai yang unik/berbeda antar satu dengan elemen yang lain.

Format Keluaran

Keluarkanlah parent dari semua node *u*, di mana untuk setiap node *u* diurutkan secara *ascending*. Untuk node yang menjadi root, parent nya akan dianggap -1.

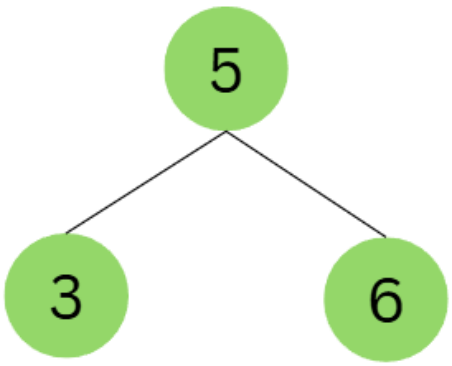
Contoh Masukan #1

```
3
6 5 3
```

Contoh Keluaran #1

```
5 -1 5
```

Penjelasan untuk contoh masukan/keluaran #1



Node	Parent
3	5
5	-1 (Root)
6	5

Keluaran diminta dengan urutan node secara *ascending*, sehingga keluarkan parent node 3, kemudian parent node 5, dan terakhir parent node 6.

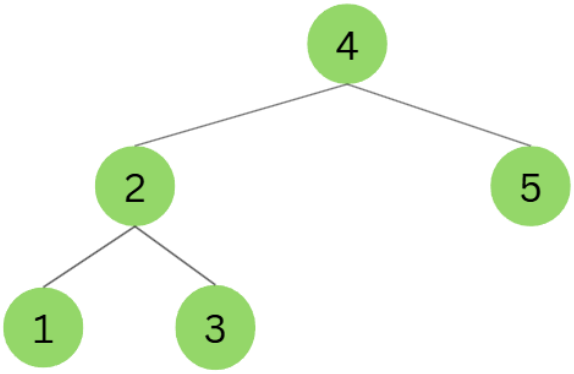
Contoh Masukan #2

```
5
1 2 3 4 5
```

Contoh Keluaran #2

```
2 4 2 -1 4
```

Penjelasan untuk contoh masukan/keluaran #2



Node	Parent
1	2
2	4
3	2
4	-1 (Root)
5	4

Sehingga, parent untuk node 1, 2, ..., 5 secara berturut-turut adalah 2, 4, 2, -1, 4.

Divine Binary Search Tree

Author: Mathias Adya Diwangkara Suganda

Time Limit	1 s
Memory Limit	256 MB

A tree is called a Binary Search Tree (BST) if for every node u, its value is greater than all values in its left subtree and less than all values in its right subtree.

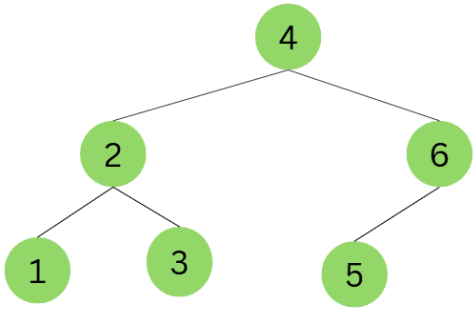
A slight modification is made on the BST. Where this tree we called as Divine Binary Search Tree (DBST).

DBST is a BST but for every level (except the lowest level), must be completely filled. This rule does not apply to the lowest level. All DBSTs are BSTs, but not all BSTs are DBSTs.

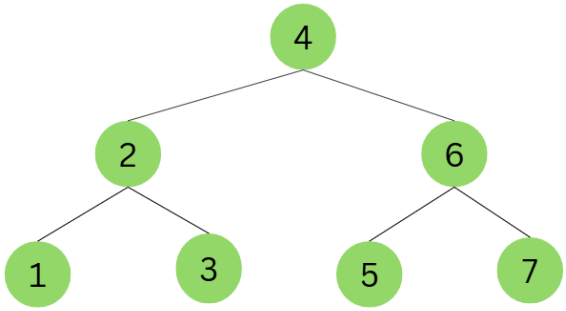
For the lowest level of the DBST, we have 2 set of rules:

1. Every leaf node is filled from as left as possible.
2. The last leaf node may not have a sibling.

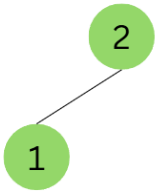
Below are some examples of valid DBSTs.



Example 1

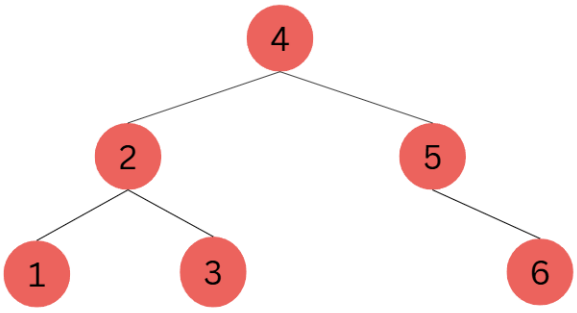


Example 2

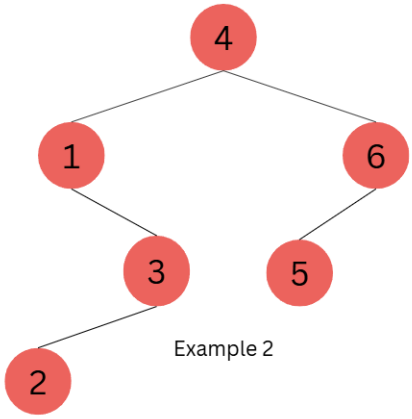


Example 3

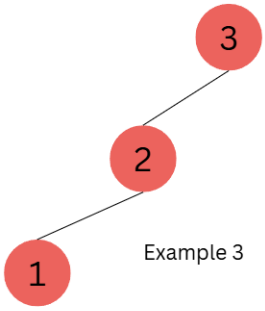
And here are some examples of the trees that are BSTs, but not DBSTs.



Example 1

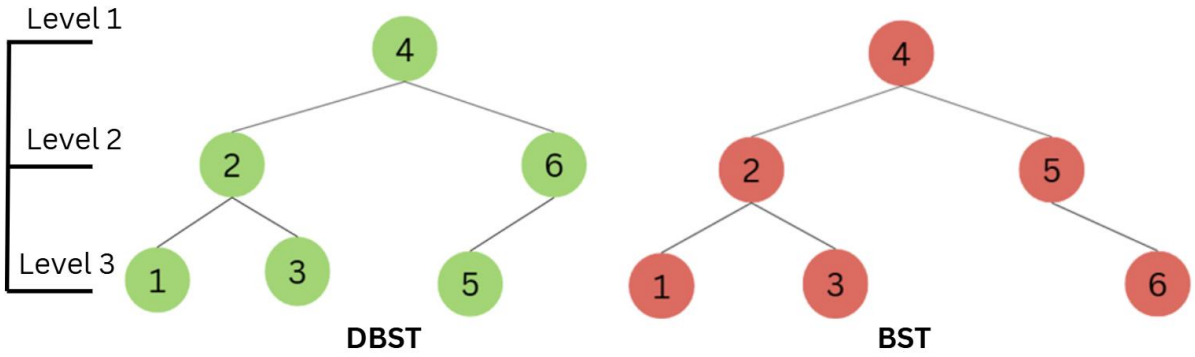


Example 2



Example 3

To clarify the difference, observe the two BSTs below:



On the left side (DBST), node 5 is at level 3 and node 6 is at level 2. All leaf nodes are filled from left to right. On the right side (BST), the placement of nodes 5 and nodes 6 is reversed. The left subtree of node 5 is empty, which violates the first rule.

You are given an array *a* of *n* elements. Construct a Divine Binary Search Tree such that all nodes of the tree are filled using the elements of a.

Input Format

The input consists of two lines:

The first line contains an integer *n* $(1 \leq n \leq 2 \times 10^5)$.

Second line contains *n* integers, the elements of array *a* $(1 \leq a_i \leq 10^6)$.

It is guaranteed that all elements in the array are unique.

Output Format

Output the parent of each node *u*, sorted by node value in ascending order.

For the node that becomes a root, its parent is considered -1.

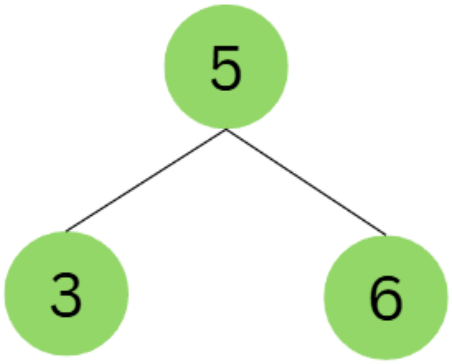
Sample Input #1

```
3
6 5 3
```

Sample Output #1

```
5 -1 5
```

Explanation for Sample Input/Output #1



Node	Parent
3	5
5	-1 (Root)
6	5

The output is ordered by ascending order value: so the parent of 3, then 5, then 6 is output.

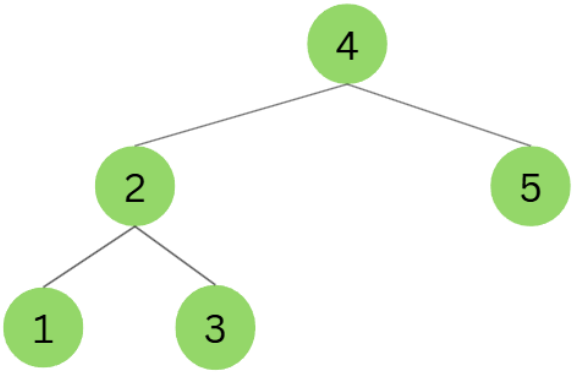
Sample Input #2

```
5
1 2 3 4 5
```

Sample Output #2

```
2 4 2 -1 4
```

Explanation for Sample Input/Output #2



Node	Parent
1	2
2	4
3	2
4	-1 (Root)
5	4

So, the parents of nodes 1, 2, ..., 5 in ascending order are: 2, 4, 2, -1, 4.