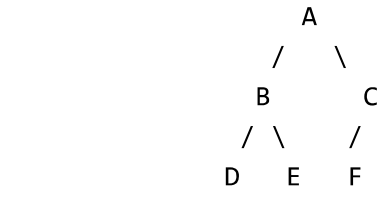
Heap Data Structure

- Treelike strucutre
- Represented in an array

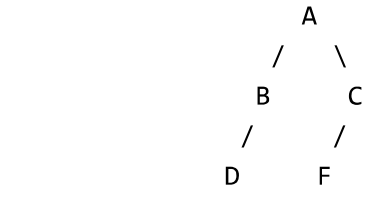
In [ ]:
```
1
```

In [ ]:
```
1
```

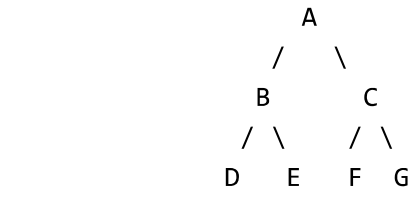
Heap is a Complete Binary tree
Filled L->R



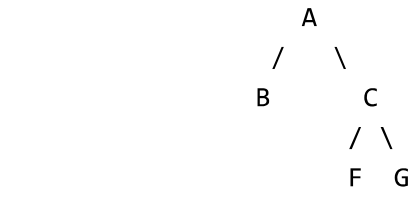
Yes



NO



Yes

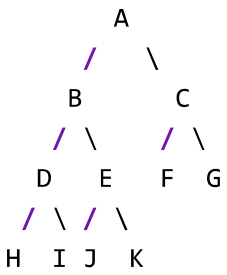


No

In [ ]:
```
1
```

Array representation of complete binary tree

Child and parent



Data	A	B	C	D	E	F	G	H	I	J	K
Index	0	1	2	3	4	5	6	7	8	9	10
ParentIndex		0	0	1	1	2	2	3	3	4	4

```
leftChild  = parentIndex*2 + 1
rightChild = parentIndex*2 + 2

parentIndex = floor( (childIndex - 1) / 2 )

floor(1.5) => 1
floor(1.9) => 1
floor(1.1) => 1
floor(1)   => 1
floor(0.9) => 0
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Heap property

Binary tree in which the root node is always less than(min heap) or equal to the child nodes. Above property is true for all sub trees of the complete binary tree.

```
          2
        /   \
       4     3
      / \
    10   8
Is it a min heap: YES
```

```
          2

Is it a min heap: YES
```

Building heap

```
          2
        /   \
       4     1
      / \
    10   8

  Data   2 4 1 10 8
  Index  0 1 2 3  4
```

Operations:

- Heapify (siftUp/Down repeatedly)
- GetMax/Min: O(1)
- Del max/min: log N
- Add new key: log N

If you need remove a random key by value, it's an O(N) operation since finding index of an element in a heap is O(n). This can be reduced by keeping additional mapping of value to indexes

Internal Ops:

- SiftUp
- Sift Down

To add a new key: Append and then do siftUp
To delete max/min: Swap with last element and then do siftDown

Sorted Array:

- min, max: O(1)
- search: log N
- Udates, insert/delete an element keeping the sorted property ? O(N)

HashMap

- min, max: O(N)
- search for key: O(1)
- Update: O(1)

Balanced BST:

- min, max: log N
- search: log N
- Update: log N

Heap:

- min, max: O(1)
- search: O(N), keep an additional map O(1)
- Update for min and max: log N
- Update any random key: O(N) keep an additional map O(1): log N

In [ ]:
```
1
```

In [ ]:
```
1
```

When to use what
Static Data

- ordering: sorting
- min/max: heap/sorting

Dynamic Data

- ordering: bst
- min/max: heap/bst

In [ ]:
```
1
```

Heap sort

- buildMaxHeap(data)
- repeat N times
  - removeMax(): max element is at the beginning of array -> move it to the end

Why heap sort is not good ?comparisons and swaps

In [ ]:
```
1
```