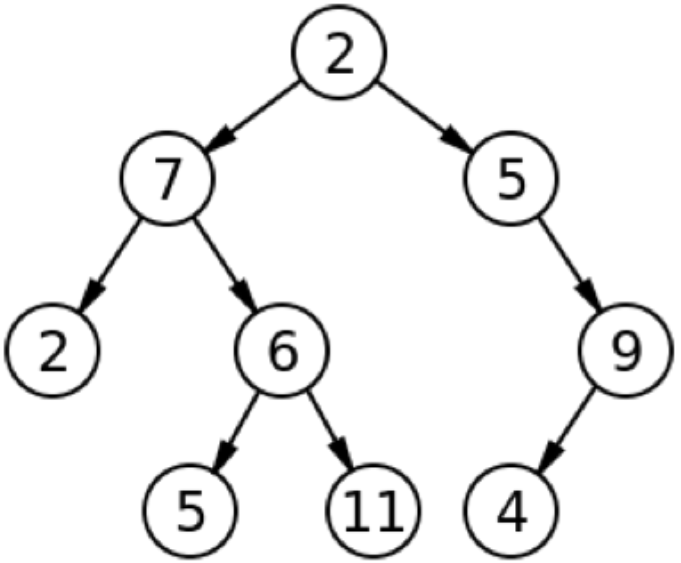
CS 280
Spring 2018

John Bowers, Professor
Mike Lam, Professor

Trees

Non-linear data structures

- Tree
  - Entities and hierarchical (parent/child) relationships
  - Single root node with no parent
  - At least one (and usually many) leaf nodes with no children

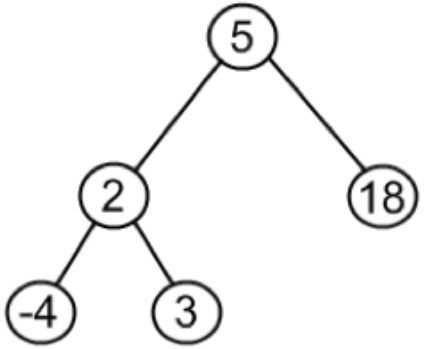


Tree implementations

- **Linked** objects

```
class Node {
   Node left, right;
   Node parent;
}
```

- **Array**-based storage
  - ParentID = floor((ChildID-1) / 2)
  - LeftChildID = ParentID*2 + 1
  - RightChildID = ParentID*2 + 2
- **Hash**-based storage
  - ChildID => ParentID
  - ParentID => List<ChildIDs>



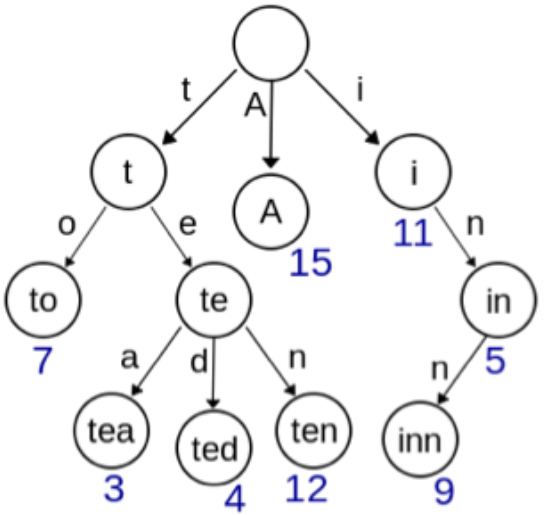

```
[ 5, 2, 18, -4, 3 ]
  0  1   2   3  4
```

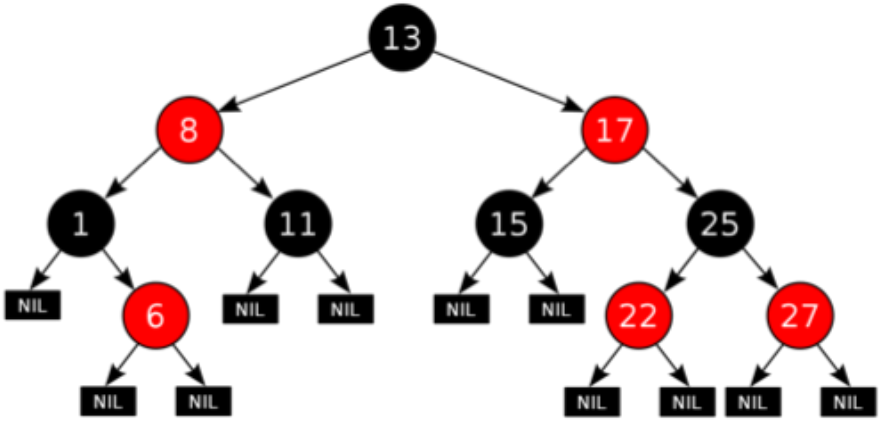
```
p[2]  = 5
p[18] = 5
p[-4] = 2
p[3]  = 2

c[5] = [2, 18]
c[2] = [-4, 3]
```

Specific kinds of trees

- Binary search trees
  - Total ordering: fast search
  - Balanced, complete, proper/full
  - AVL, red-black, 2-3-4 trees
- Heaps, Fibonacci heaps
  - Partial ordering: fast remove min/max
- B-trees, R-trees
- Tries, Huffman trees
- Binomial trees
- Fenwick trees





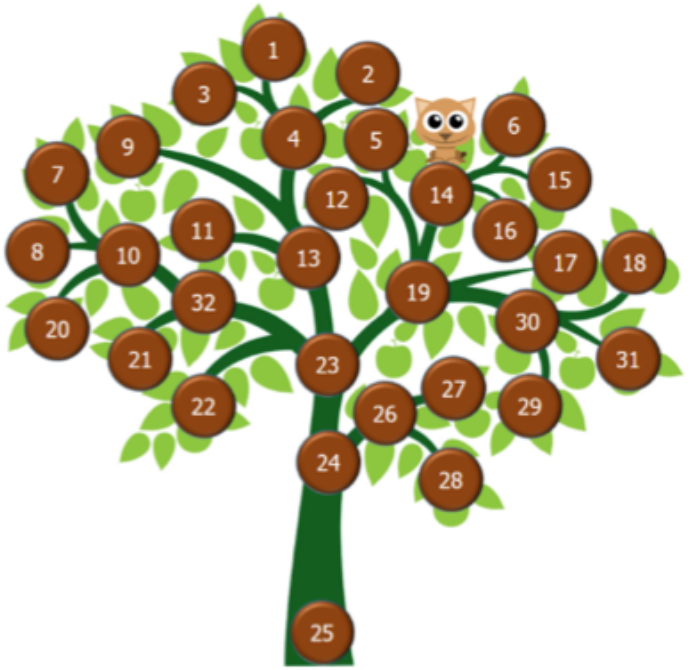
Takeaway

- No built-in structures
  - Be familiar with the different types of trees
- Some problems may require you to store a tree
  - And some will timeout if you do!
    - Some trees are infinitely large…

In-class problem

Kitten on a Tree

Ouch! A kitten got stuck on a tree. Fortunately, the tree's branches are numbered. Given a description of a tree and the position of the kitten, can you write a program to help the kitten down?



Input

The input is a description of a single tree. The first line contains an integer K, denoting the branch on which the kitten got stuck. The next lines each contain two or more integers $a, b_1, b_2, \ldots$. Each such line denotes a branching: the kitten can reach a from $b_1, b_2, \ldots$ on its way down. Thus, a will be closer to the root than any of the b_i. The description ends with a line containing -1. Each branch b_i will appear on exactly one line. All branch numbers are in the range $1..100$, though not necessarily contiguous. You are guaranteed that there is a path from every listed branch to the root. The kitten will sit on a branch that has a number that is different than the root.

The illustration above corresponds to the sample input.

Output

Output the path to the ground, starting with the branch on which the kitten sits.

In-class problem

Output

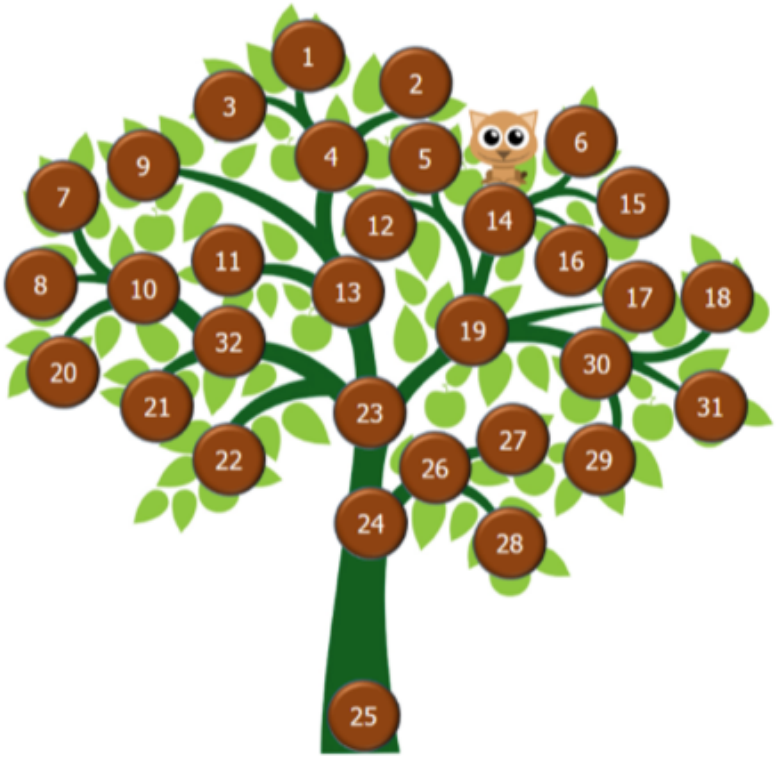
Output the path to the ground, starting with the branch on which the kitten sits.

Sample Input 1

```
14
25 24
4 3 1 2
13 9 4 11
10 20 8 7
32 10 21
23 13 19 32 22
19 12 5 14 17 30
14 6 15 16
30 18 31 29
24 23 26
26 27 28
-1
```

Sample Output 1

```
14 19 23 24 25
```



In-class problem

Output

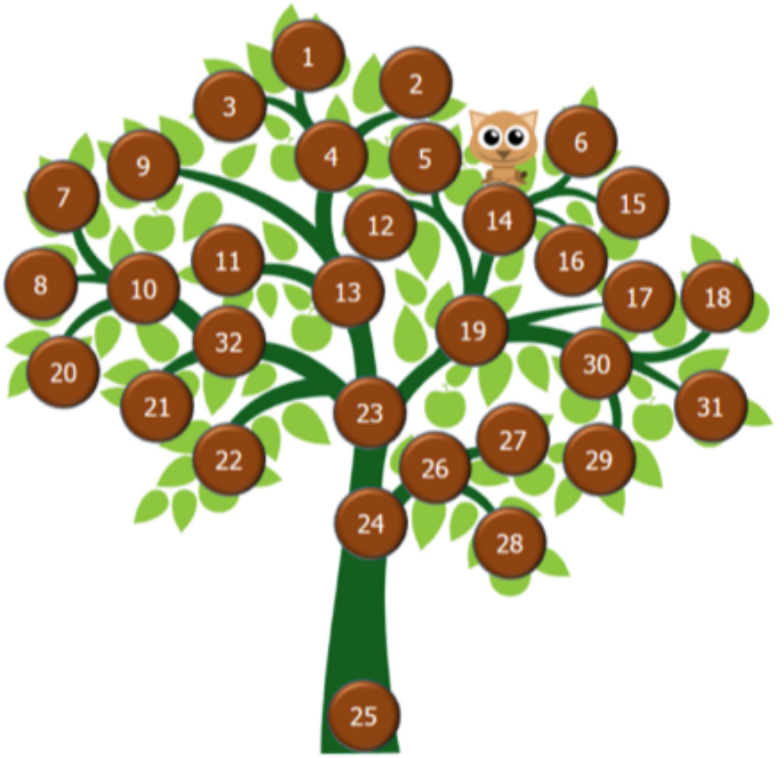
Output the path to the ground, starting with the branch on which the kitten sits.

Sample Input 1

```
14
25 24
4 3 1 2
13 9 4 11
10 20 8 7
32 10 21
23 13 19 32 22
19 12 5 14 17 30
14 6 15 16
30 18 31 29
24 23 26
26 27 28
-1
```

Sample Output 1

```
14 19 23 24 25
```



Motion is always toward the root!

Pseudocode

```
// child id => parent id
Map<String,String> parent = new HashMap<>()

read K

until input is "-1":
    read a
    for each b in remainder of line:
        parent[b] = a

while K is not null:
    print K
    K = parents[K]
```