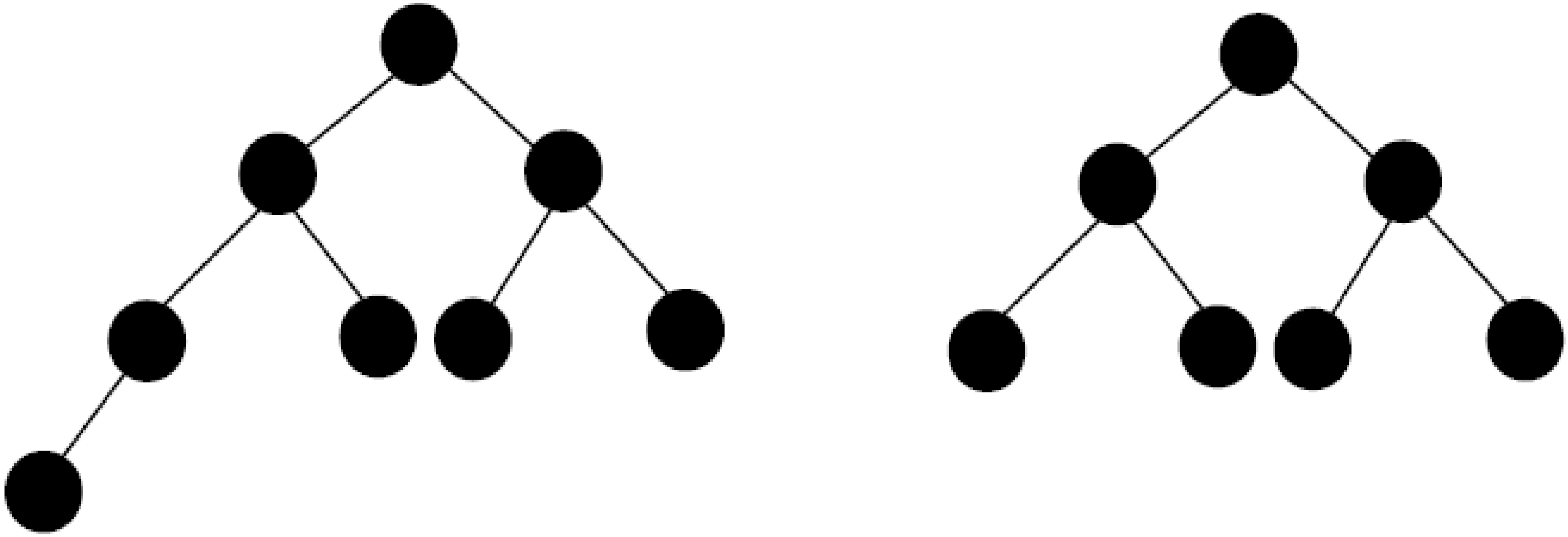
Tree Balancing Techniques

Dr. Manjusri Wickramasinghe

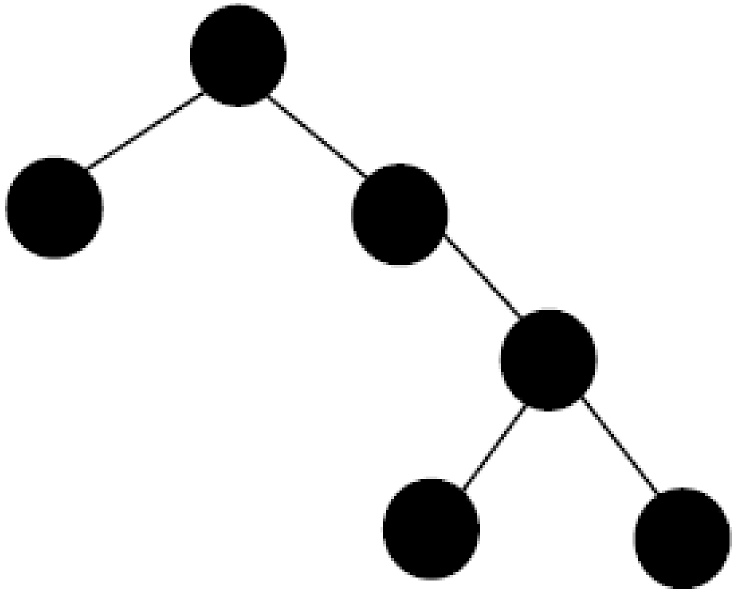
Preliminaries

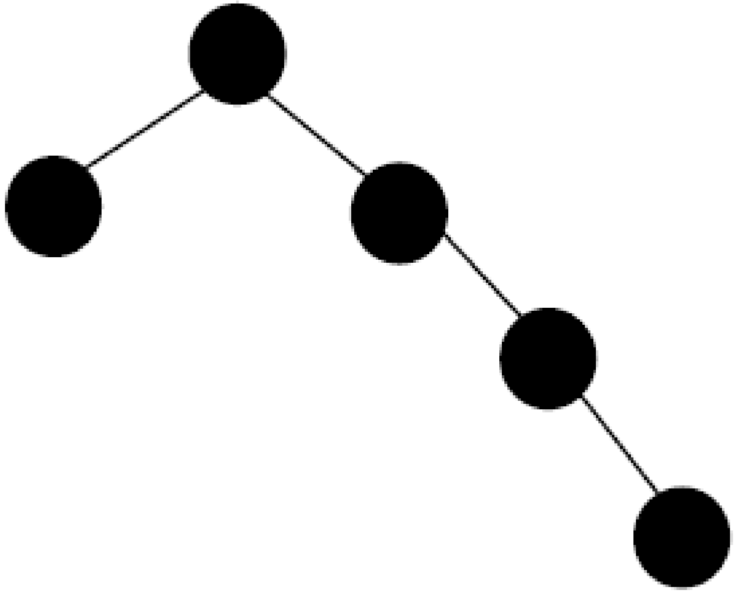
- What is a tree data structure?
- What is the purpose of using a tree data structure?
- What is tree balancing?

Balanced trees



Unbalanced trees





Are there any issues with having unbalanced trees?

UCSC

Tree Balancing…(1)

- The process of converting a tree structure into a one that is close to being full binary tree or is a full tree.

- Tree balancing methods fall into two categories as
  - Global balancing
  - Local balancing

Tree Balancing…(2)

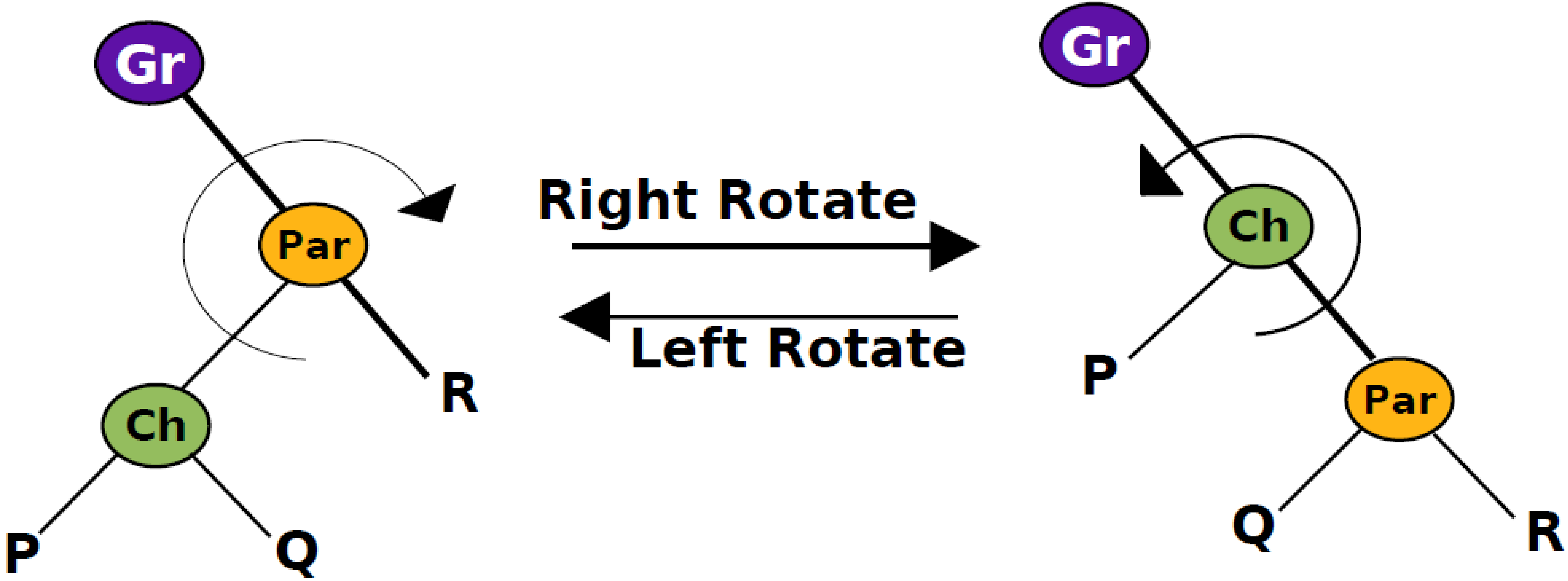
- Global balancing is the process of restructuring the tree once all the operations on it is complete. These type of techniques affect the entire tree.
  - E.g. DSW Algorithm

- Local balancing restructures the tree at each insert and delete operation if a height imbalance occurs. These type of techniques affect only a local part of the tree.
  - E.g. AVL Trees, Red Black Trees

Tree Balancing…(3)

- How are trees balanced?

Rotations…(1)

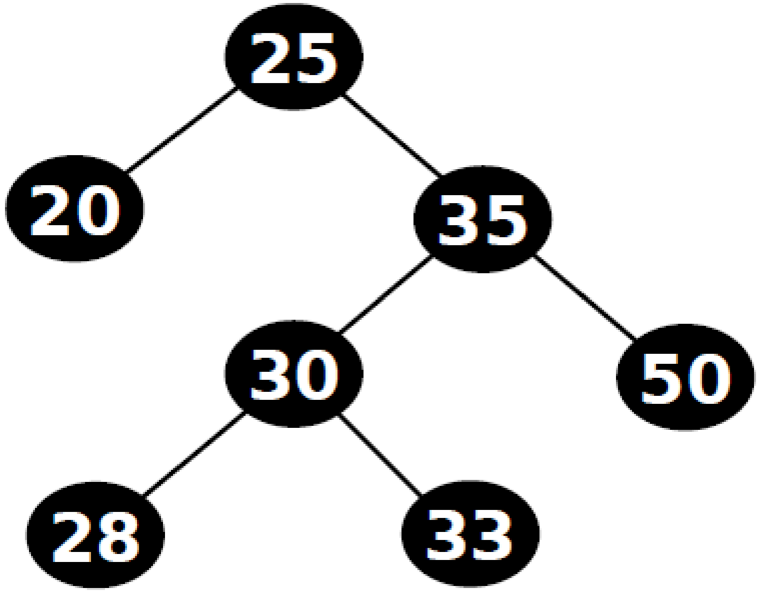


Right Rotate

Left Rotate

Rotations…(2)

- Exercise :- Rotation 30 about its parent.



Rotations…(3)

```
function rightRotate(Node child, Node GP, Node Par)
{
    if(Par != root)
        set GP as parent of child
        set right subtree of child as left child of Par
        set Par as right subtree of child
}
```

Rotations…(4)

- Rotations are symmetrical, so the right rotation can be converted to a left rotation and vice verse.
- A tree can be balanced using the single rotation once or multiple times. Sometimes the multiple case is also called with other names such as the double rotation.

DSW Algorithm

Overview…(1)

- DSW stands for Day, Stout and Warren which attribute to the inventors of this algorithm

- This is a global balancing algorithm

- Works on a tree which has no pending operations. The algorithms always creates trees that are perfectly balanced or close to being perfectly balance.

Overview…(2)

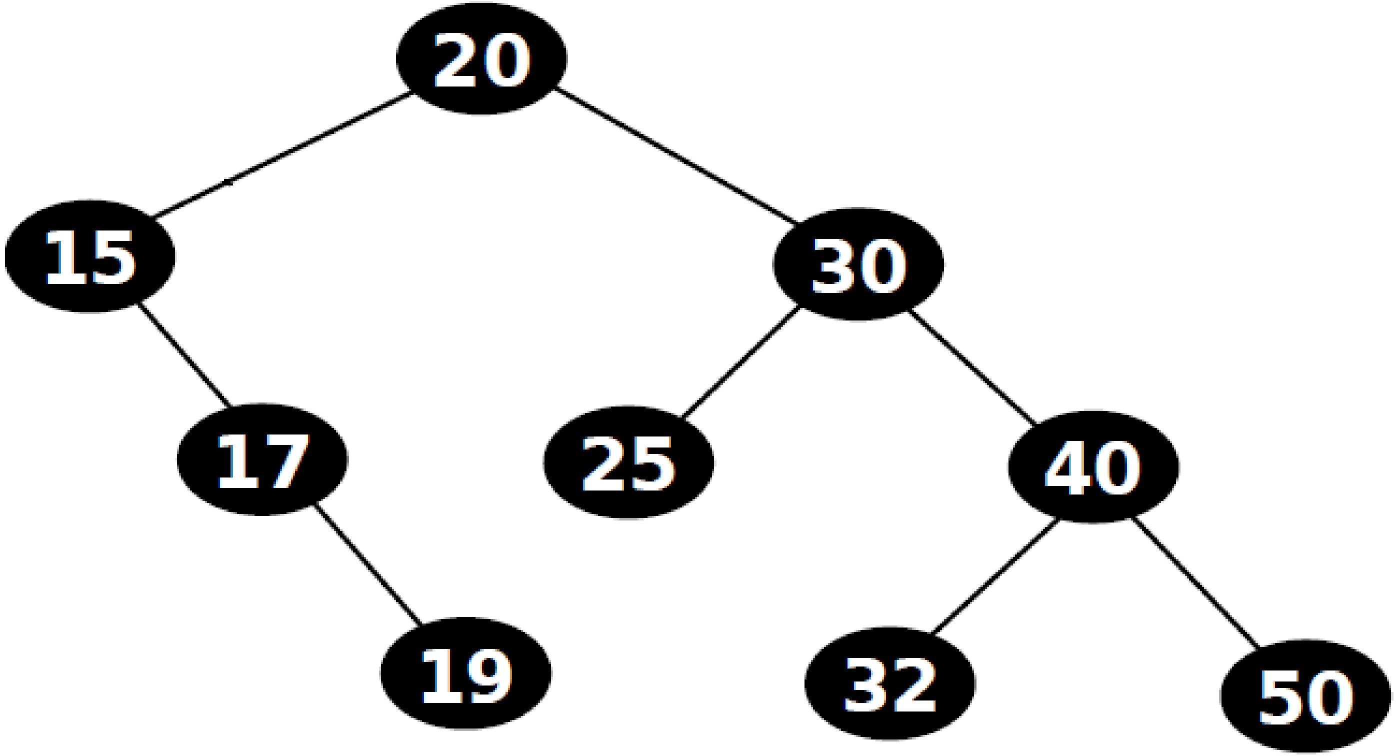
- The DSW algorithm consists of two phases as given below
  1. Create vine
  2. Create a balanced tree

Creating a Vine (Backbone)…(1)

```
function createVine(Node root){
    tmp = root;
    while(tmp != null){
        if(tmp has left child)
            right rotate left child around tmp
            update tmp to the child which just became parent
        else
            tmp = tmp.right
    }
}
```

Creating a Vine (Backbone)…(2)

- Exercise :- Convert to a backbone



Create a Balanced Tree

```
function CreateBalancedTree(int numOfNodes){
    int m = 2^(lg2(numOfNodes + 1)) - 1;
    make n - m right rotations
    while (m > 1)
        m = m/2
        make m left rotations starting top of vine
}
```

Questions ?