그리디 알고리즘

➔ 트리 구조에서 사용되는 알고리즘으로 매 절차에서 최선의 선택지를 선택하는 알고리즘

➔ 알고리즘 절차 전체에서의 최선이 아닌 매 절차 순간마다의 최선의 선택지를 고르기 때문에 효율적이지 못하다.

밑의 코드 부분은 트리 구조에 대한 그리디 알고리즘으로 리프 노드에서 가장 큰 값을 찾는 코드다.

코드

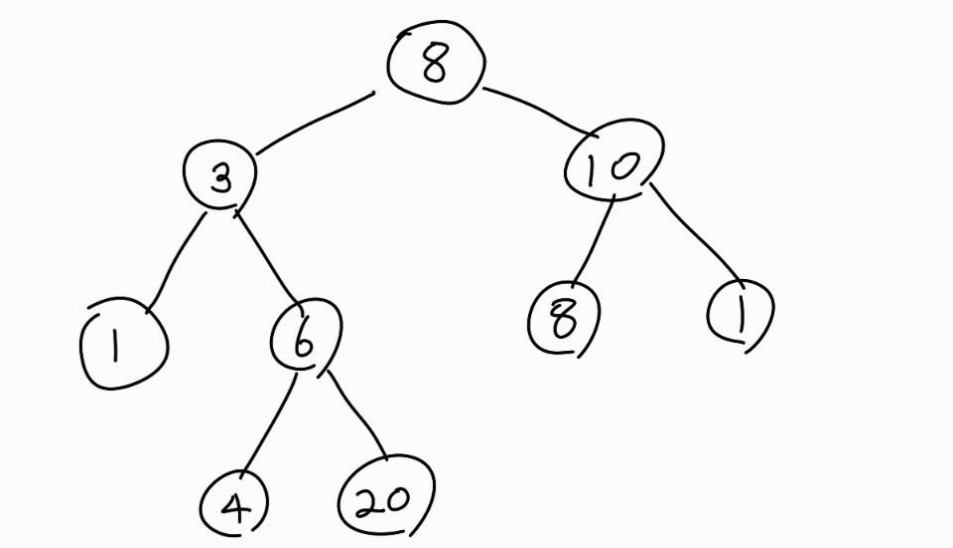
- **트리 부분**

```
public class Greedy {
    public static void main(String[] args) {
        //Binary Tree Create
        Tree treeNode = new Tree(8);
        Greedy greed = new Greedy();

        treeNode.addLeft(3);
        treeNode.left.addLeft(1);
        treeNode.left.addRight(6);
        treeNode.left.right.addLeft(4);
        treeNode.left.right.addRight(20);

        treeNode.addRight(10);
        treeNode.right.addLeft(8);
        treeNode.right.addRight(1);

        //Greedy(가장 큰 값 찾기)
        int result = greed.greedSearch(treeNode);
        System.out.println("결과 : " + result);


    }
```

- 그리디 구현 부분

```
24
25      public int greedSearch(Tree treeNode) {
26          int depth = 0;
27          while(true) {
28              if(treeNode.right == null && treeNode.left == null) {
29                  System.out.println("깊이 : " + depth);
30                  return treeNode.data;
31              }
32
33              if(treeNode.left.data > treeNode.right.data) {
34                  treeNode = treeNode.left;
35                  depth++;
36              }
37              else if(treeNode.left.data < treeNode.right.data) {
38                  treeNode = treeNode.right;
39                  depth++;
40              }
41              else if(treeNode.left.data == treeNode.right.data){
42                  treeNode = treeNode.left;
43                  depth++;
44              }
45
46          }
47      }
48
49  }
50
```

트리 구조



루트 노드에서 시작하여 3과 10 중에 더 큰 노드로 이동, 10노드로 이동하여 8과 1중에 더 큰 노드로 이동.

8은 자식 노드가 없는 리프 노드이기 때문에 결과는 8이 된다.