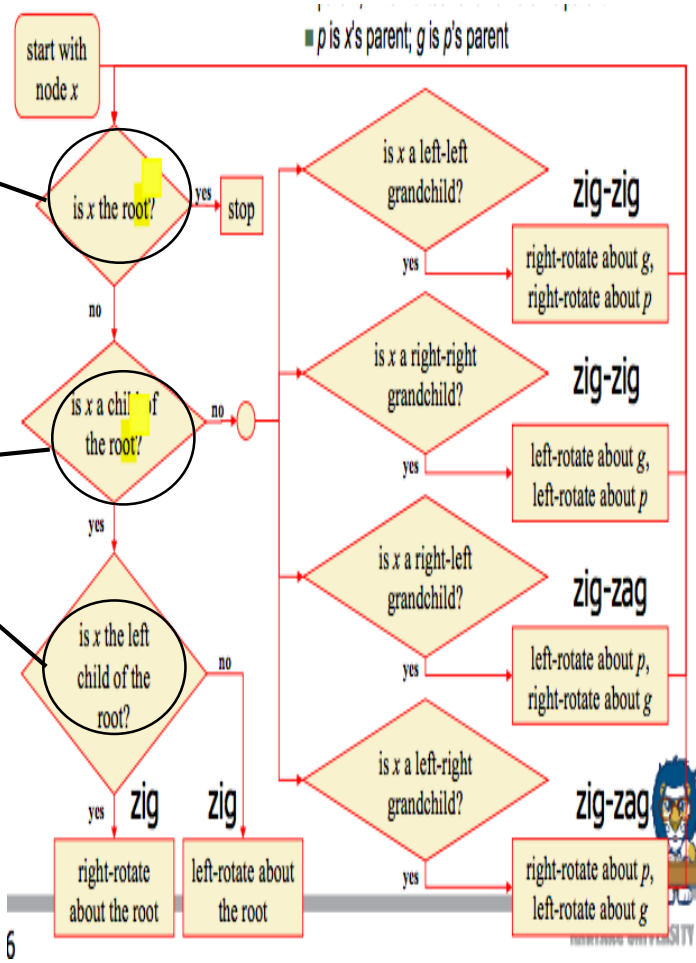
```
void ZigZag(Tree* tree,Node* node){

  Node* Root = tree->root;
  while(1){
    if(Root == node)
      break;
    else if(Root->left == node){
      tree->root = Zig_R(node->par);
      break;
    }
    else if(Root->right == node){
      tree->root= Zig_L(node->par);
      break;
    }



    else{
      Node* Gnode = node->par->par;
      if(Gnode == tree->root){
        if(Gnode->left == node->par){
          if(node->par->left == node)
            Gnode = Zig_Zig_R(Gnode);
          else
            Gnode = Zig_Zag_LR(Gnode);
        }
        else{
          if(node->par->left == node)
            Gnode = Zig_Zag_RL(Gnode);
          else
            Gnode = Zig_Zig_L(Gnode);
        }
        tree->root = Gnode;
        break;
      }
```



```
Node* Zig_Zig_L(Node* z){
  printf("Rotate LL\n");
  Node* y = z->right;
  Node* x = y->right;
  z->right = y->left;
  y->left = z;
  y->right = x->left;
  x->left = y;
  z->par = y;
  y->par = x;

  return x;
}
```

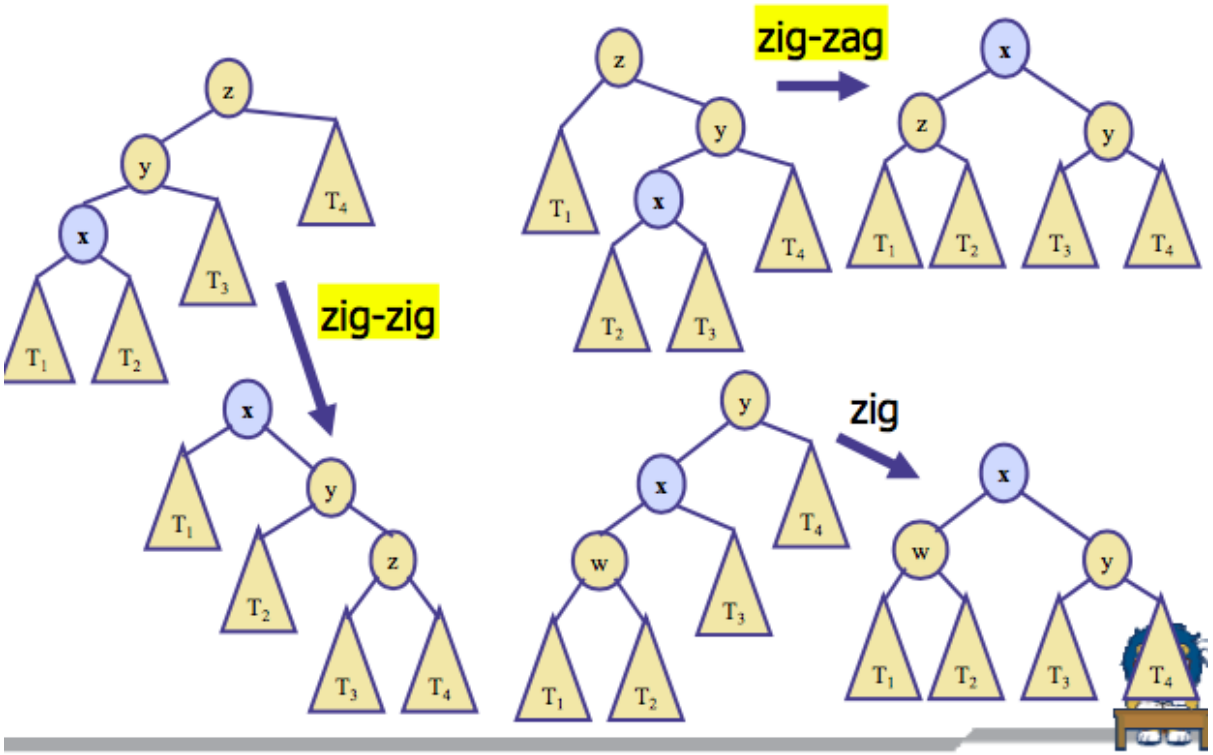
```
Node* Zig_Zag_RL(Node* z){
  printf("Rotate RL\n");
  Node* y = z->right;
  Node* x = y->left;
  z->right = x->left;
  y->left = x->right;
  x->left = z;
  x->right = y;
  z->par = x;
  y->par = x;
  return x;
}
```

x,y,z에 대해

노드 조정작업을 하고

최상위 노드를 (z->x)

변경한다.



```
void InsertTree(Tree* tree,Node *node){
  printf("Insert : %d\n",node->data);
  InsertNode(tree->root,node);
  ZigZag(tree,node);
}
```

삽입 후 해당 노드에 대한 splay

```
Node* S_SearchNode(Tree* tree,Data data){
  Node* comNode = tree->root;
  Node* nearNode = comNode;
  while(comNode != NULL){
    nearNode = comNode;
    if(comNode->data == data)
      return comNode;
    else if(comNode->data>data)
      comNode = GetLeft(comNode);
    else
      comNode = GetRight(comNode);
  }
  return nearNode;

}

Node* SearchTree(Tree* tree,Data data){
  printf("Search : %d\n",data);
  Node* target = S_SearchNode(tree, data);
  if(target->data != data)
    printf("near Node :%d\n",target->data);
  ZigZag(tree,target);
  return target;
}
```

Data에 해당하는 노드를 찾고 없는 경우 가까운 노드를 리턴

찾은 노드나 근처 노드에 대해 splay를 진행한다.

```
Data DeleteTree(Tree* tree,Data data){//error
  printf("Delete : %d\n",data);
  Data delData = DeleteNode(tree->root,data);
  SearchTree(tree,data);
  if(delData == NOTRETURN){
    printf("no data\n");
    return NOTRETURN;
  }
  return delData;
}
```

삭제 후 대체 노드를 찾아 그 노드에 대한 splay를 진행한다.

(Search 에서 splay)

-zig와 zigzig를 별개로 구현.

-조상노드가 root가 아닌 경우를 고려.

-delete에서 splay 대상 노드를 찾는 함수

실행 화면

```
jeongjiseong-ui-MacBook-Air:week4 jisung$ ./splay
Insert : 4
Rotate R
[4][6]
Insert : 2
Rotate R
[2][4][6]
Insert : 9
Rotate LL
Rotate L
[9][2][6][4]
Insert : 3
Rotate RR
Rotate L
[3][9][2][6][4][6]
Search : 0
near Node :2
Rotate RR
[2][9][6][3][4][6]
Search : 9
Rotate L
[9][2][6][3][4][6]
Insert : 10
Rotate LL
Rotate LL
[10][3][9][2][6][4][6][4]
Delete : 0
Search : 0
near Node :2
Rotate RR
Rotate R
no data
[2][10][9][6][4][3][6][4]
```