EDDI
Electronic Design
Development Institute

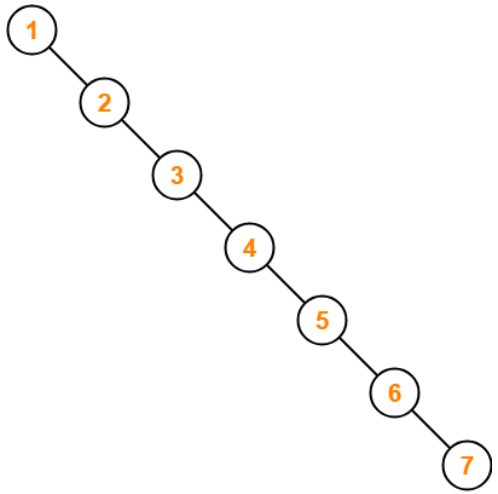
에디로봇아카데미
임베디드 마스터 Lv2 과정

제 1기

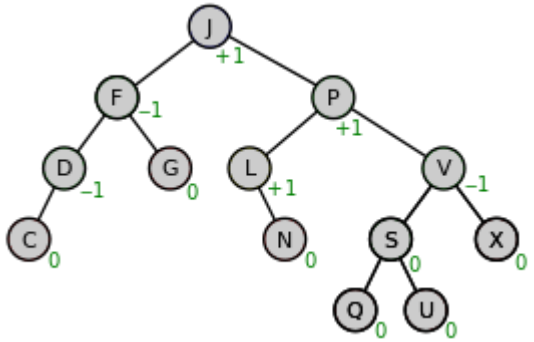
2021. 11. 06

김태훈

AVL TREE



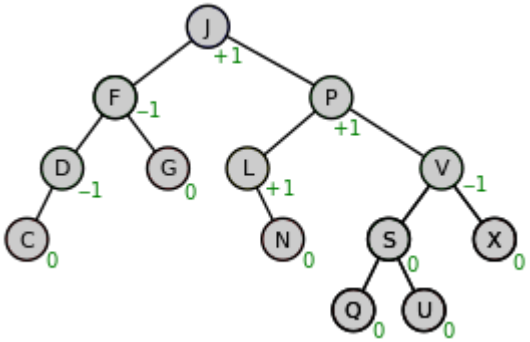
Skewed Binary Search Tree



Binary tree에 순차적인 데이터가 들어와서 Binary 속성을 활용하지 못해서 생기는 문제점 해결하기 위해서 나온 Tree

Adelson-Velsky and Landis Tree

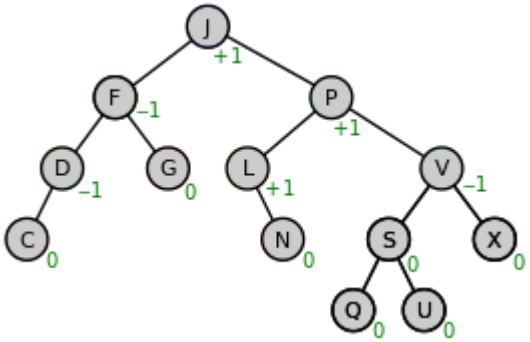
AVL TREE



특징

1. 이진 트리의 특징은 가지고 있음
2. 왼쪽 child의 level과 오른쪽 child 의 level 차이가 1을 초과해서는 안됨.

AVL TREE



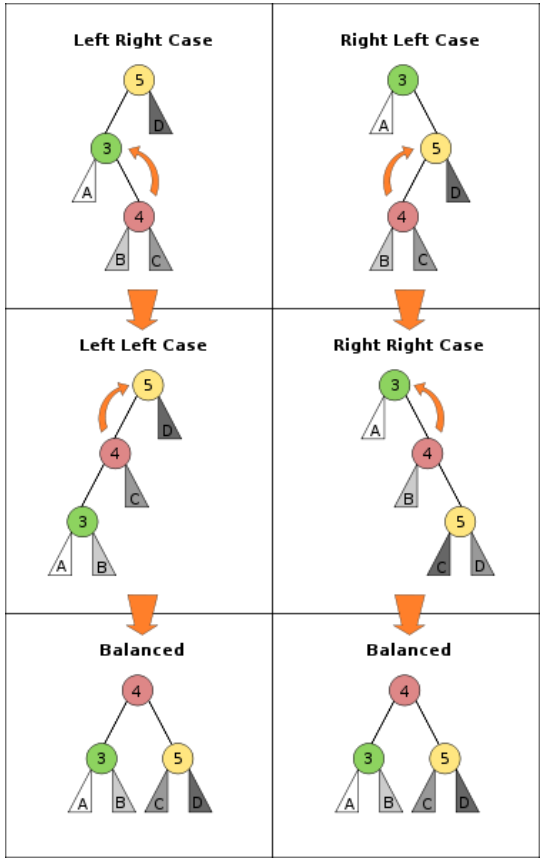
특징

1. 이진 트리의 특징은 가지고 있음
2. 왼쪽 child의 level과 오른쪽 child 의 level 차이가 1을 초과해서는 안됨.

2번 특징을 위반한다면?

Rotation 이라는 동작을 한다!

AVL TREE



간단하게 생각하면
왼쪽이 무거우면 오른쪽으로 살짝 돌려서 균형을 맞추고,
오른쪽이 무거우면 왼쪽으로 살짝 돌려서 균형을 맞추는 것.

이때 왼쪽 그림에서 left-left case와 right-right case가 된다면 바로 돌려서
균형을 맞출 수 있다.

Left Right나 Right Left는 left case와 right case를 만들어야 하기 때문에 child의 위치를 반대로 돌려주는 작업이 필요.

구현결과

EDDI
Electronic Design
Development Institute

```
embedded$ valgrind --leak-check=yes ./a.out
```

```
==13311==
==13311== HEAP SUMMARY:
==13311==     in use at exit: 0 bytes in 0 blocks
==13311==   total heap usage: 132,606 allocs, 132,606 frees, 2,282,704 bytes allocated
==13311==
==13311== All heap blocks were freed -- no leaks are possible
==13311==
==13311== For lists of detected and suppressed errors, rerun with: -s
==13311== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Valgrind 를 이용해서 delete시에 memory leak이 발생하나 검증.
메모리 누수 없이 insert와 delete를 구현.

구현결과

EDDI
Electronic Design Development Institute

```
void print_avl(void *tree)
{

    // TODO
    // 1. 순회하면서 level check
    // 2. 순회하면서 balance factor check
    // 두개 만족하면 avl tree가 깨지지 않은것.
```

정말로 AVL TREE의 특징을 만족하면서 insertion과 deletion을 구현했는지 확인하기위해서
Assert library를 이용해서 print함수에서 검증을 하였다.

```
assert(balance == 1 || balance == 0 || balance == -1);
```

```
assert(tmp->level - MAX(tmp->left ? tmp->left->level : 0, tmp->right ? tmp->right->level : 0) != 1);
```

구현결과

EDDI
Electronic Design
Development Institute

```
int main(void)
{
    avl *root = NULL;

    int i;
#if 1
    int data[10000] = { 0 };
    int len = sizeof(data) / sizeof(int);

    srand(time(NULL));

    init_data(data, len);
    print_arr(data, len);
```

Data수는 10000개로 진행했다.

구현결과

EDDI
Electronic Design Development Institute

```
void insert_avl(avl **root, int data)
{
    if (!(*root))
    {
        *root = create_avl_node();
        (*root)->data = data;
        (*root)->level = 1;
    }

    if ((*root)->data > data)
    {
        insert_avl(&(*root)->left, data);
    }
    else if ((*root)->data < data)
    {
        insert_avl(&(*root)->right, data);
    }

    update_level(root);

    adjust_balance(root, data);
}
```

Insertion의 초반부는
Binary tree와 동일하다.
중요한 것은 level을 갱신하는 것하고
Balance를 맞추는 동작이다.

구현결과

EDDI
Electronic Design
Development Institute

```
void update_level(avl **root)
{
```

```
(*root)->level = MAX((*root)->left ? (*root)->left->level:0, (*root)->right ? (*root)->right->level:0) + 1;
```

```
#define MIN(a,b) (((a)<(b))?(a):(b))
#define MAX(a,b) (((a)>(b))?(a):(b))
```

처음엔 조금 지저분하게 구현했었는데, 그냥 level의 특성에 맞게
MAX를 define하고 왼쪽 child와 오른쪽 child의 level의 최대값에 +1하면 자신의
Level이 된다.

구현결과

EDDI
Electronic Design
Development Institute

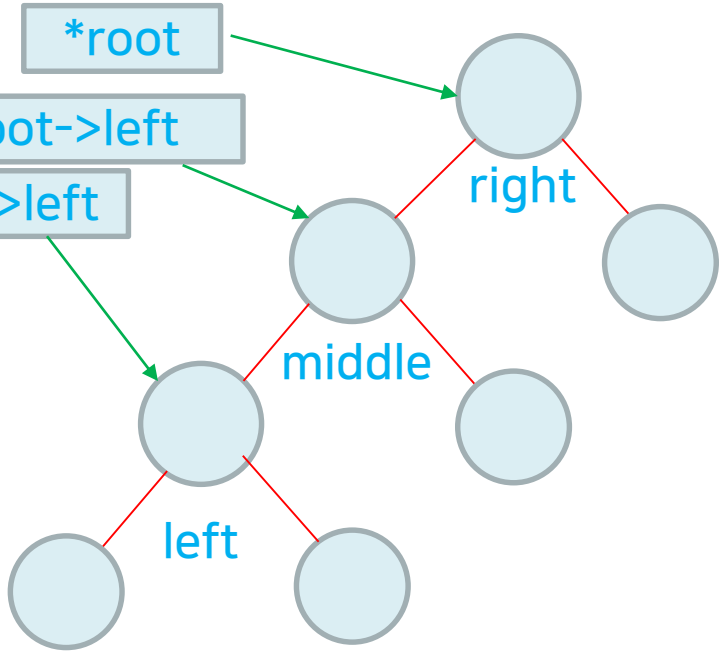
```
void adjust_balance(avl **root, int data)
{
    avl* rotate_middle;
    avl* rotate_left;
    avl* rotate_right;
    int balance = balance_factor(root);
    if(balance == 2)
    {
        balance = balance_factor(&(*root)->left);
        if(balance >= 0)
        {
```

Balance를 맞추는 것은 Left-left /Left-right / Right-left / Right-right
4가지 케이스를 나누어야한다.

구현결과

EDDI
Electronic Design Development Institute

Left-Left Case




```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

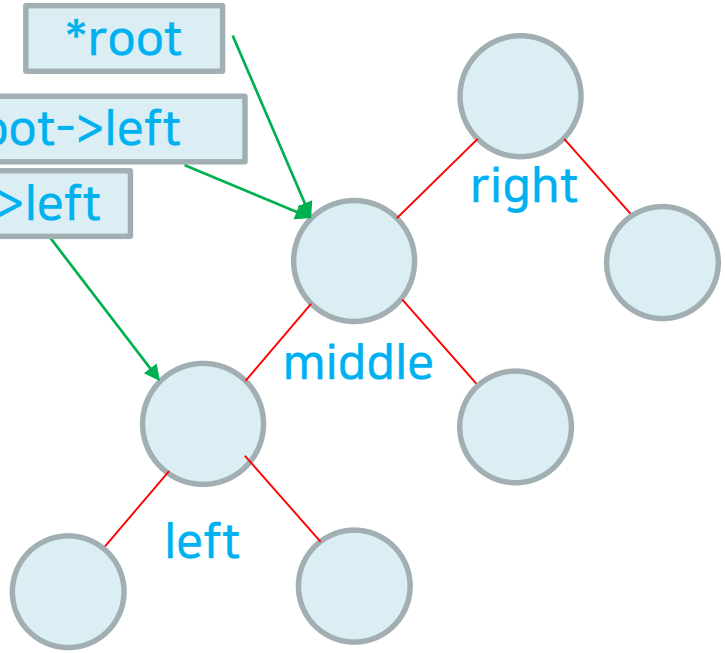
```
update_level(&rotate_right);
update_level(&rotate_middle);
```

구현결과

Left-Left Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

```
update_level(&rotate_right);
update_level(&rotate_middle);
```

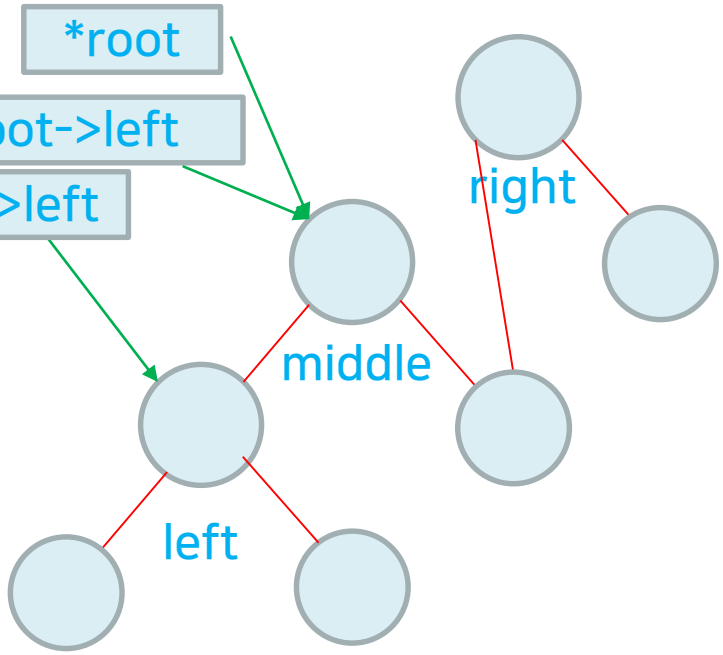


구현결과

Left-Left Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

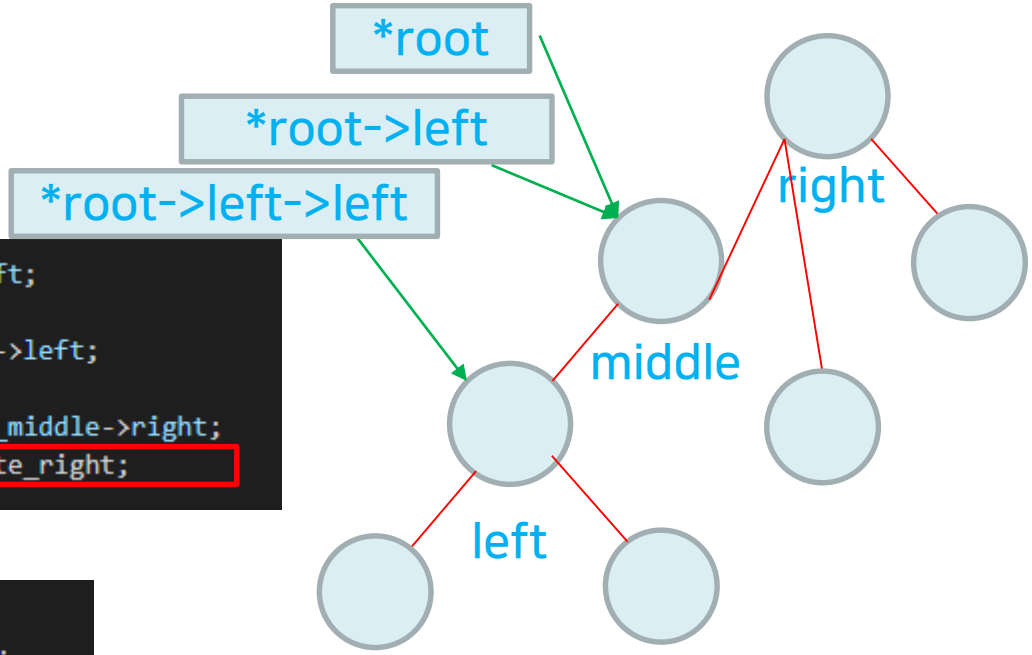
```
update_level(&rotate_right);
update_level(&rotate_middle);
```



구현결과

EDDI
Electronic Design Development Institute

Left-Left Case




```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

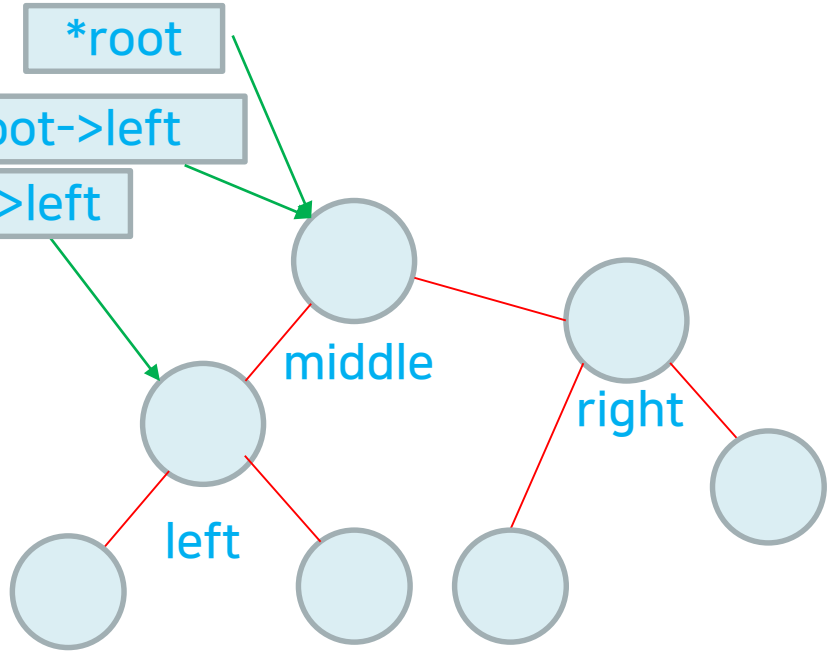
```
update_level(&rotate_right);
update_level(&rotate_middle);
```

구현결과

Left-Left Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

```
update_level(&rotate_right);
update_level(&rotate_middle);
```



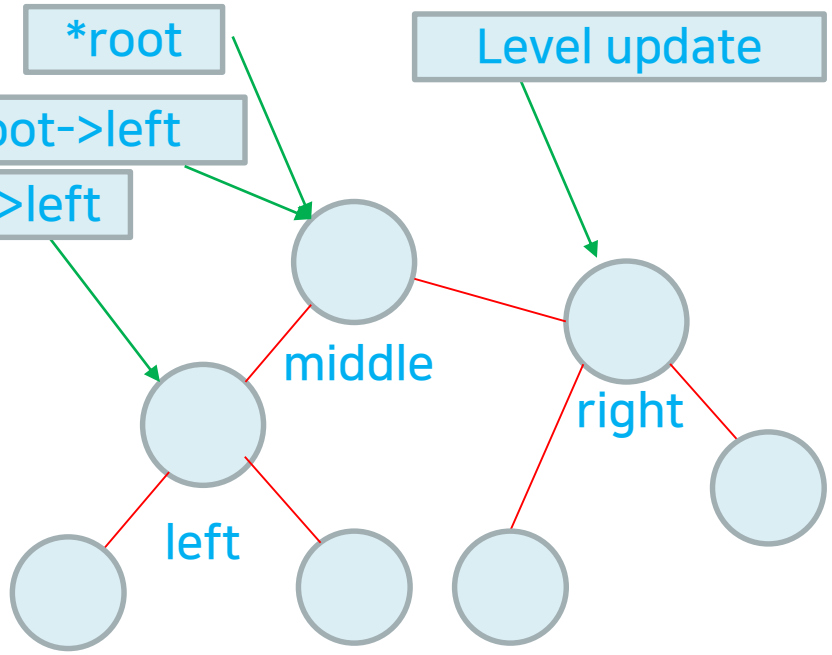
구현결과

EDDI
Electronic Design Development Institute

Left-Left Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

```
update_level(&rotate_right);
update_level(&rotate_middle);
```



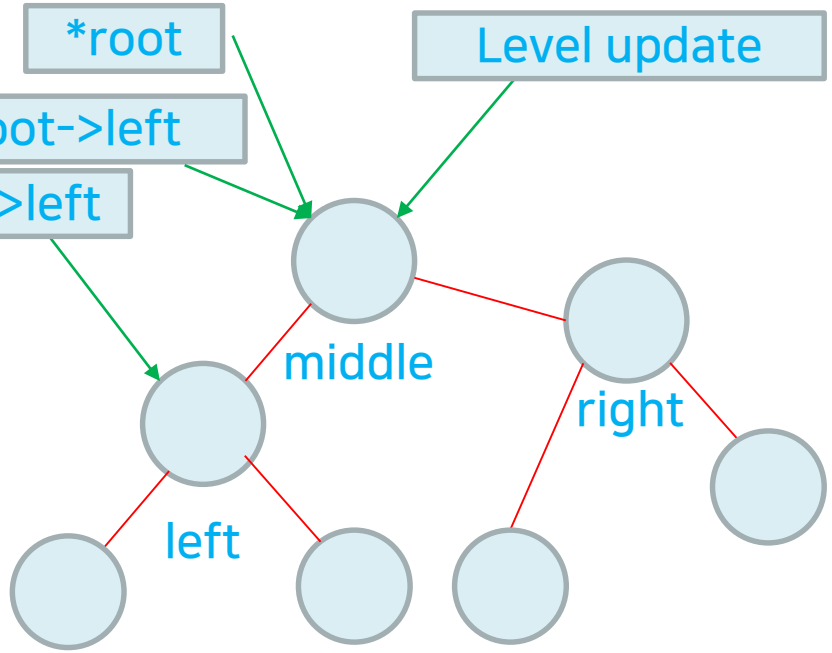
구현결과

EDDI
Electronic Design Development Institute

Left-Left Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

```
update_level(&rotate_right);
update_level(&rotate_middle);
```

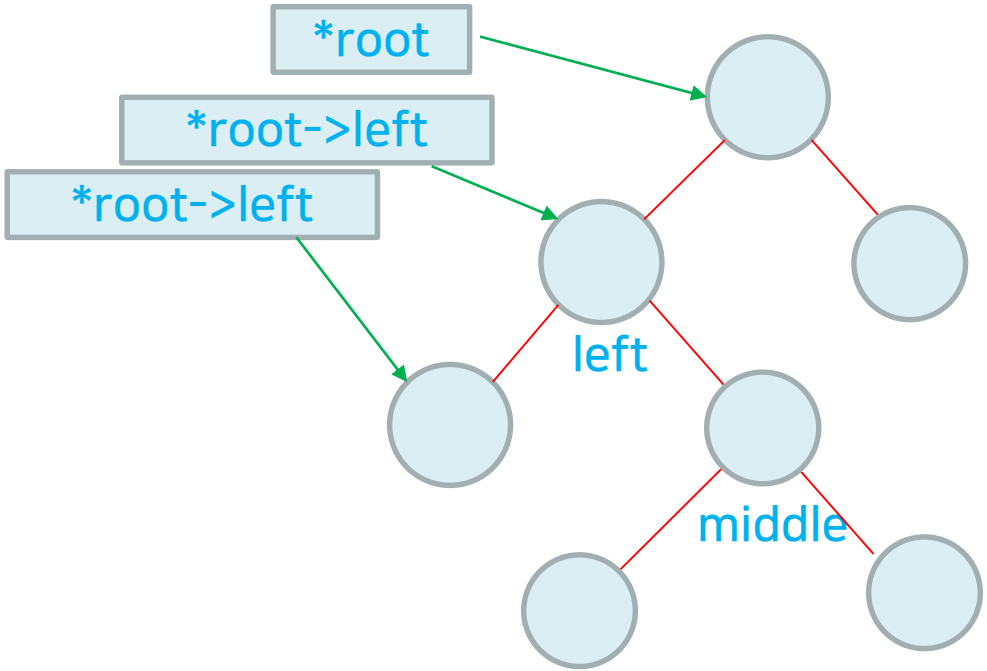


구현결과

Left-Right Case

```
rotate_middle = (*root)->left->right;
rotate_left = (*root)->left;

(*root)->left = rotate_middle;
rotate_left->right = rotate_middle->left;
rotate_middle->left = rotate_left;
```

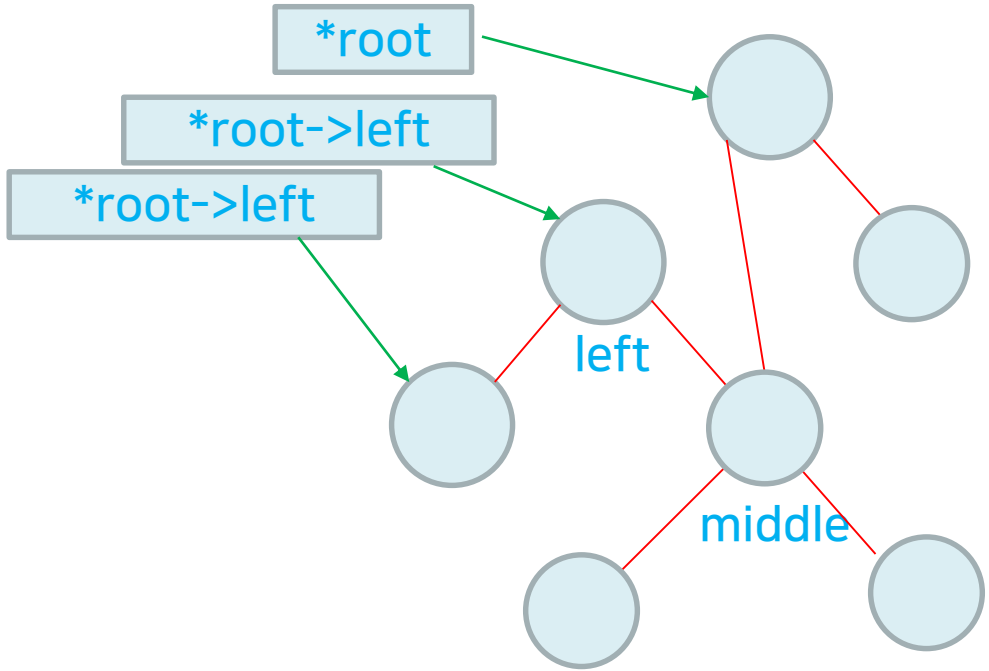


구현결과

Left-Right Case

```
rotate_middle = (*root)->left->right;
rotate_left = (*root)->left;

(*root)->left = rotate_middle;
rotate_left->right = rotate_middle->left;
rotate_middle->left = rotate_left;
```

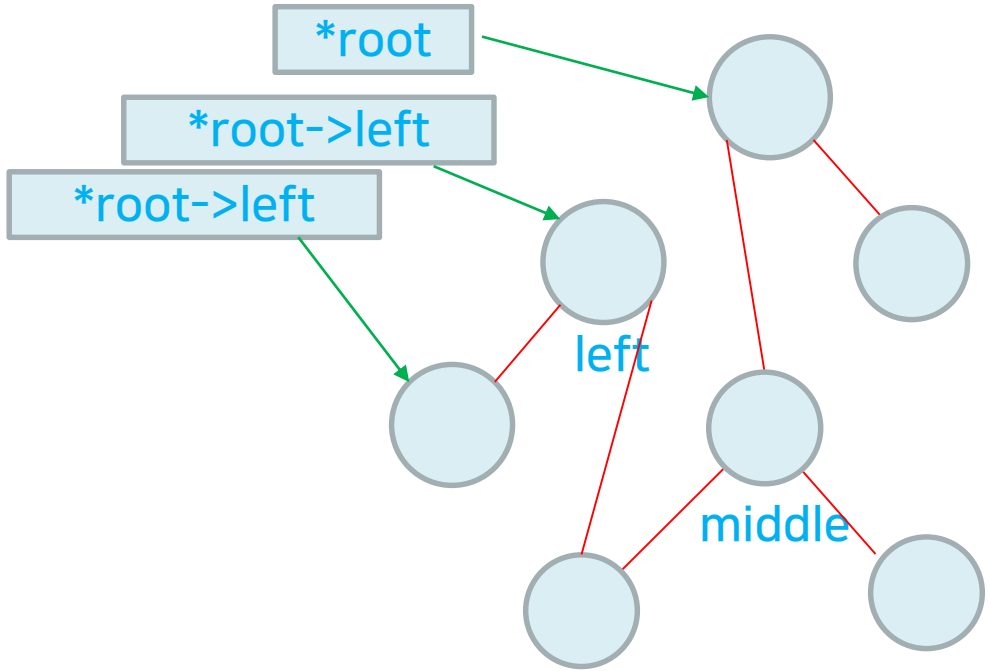


구현결과

Left-Right Case

```
rotate_middle = (*root)->left->right;
rotate_left = (*root)->left;

(*root)->left = rotate_middle;
rotate_left->right = rotate_middle->left;
rotate_middle->left = rotate_left;
```

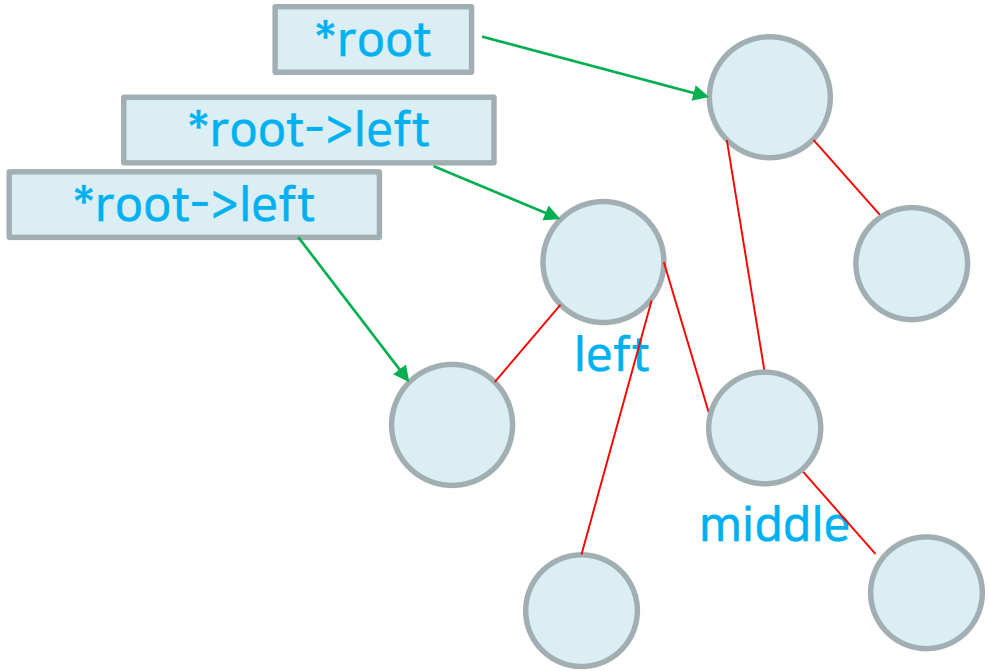


구현결과

Left-Right Case

```
rotate_middle = (*root)->left->right;
rotate_left = (*root)->left;

(*root)->left = rotate_middle;
rotate_left->right = rotate_middle->left;
rotate_middle->left = rotate_left;
```

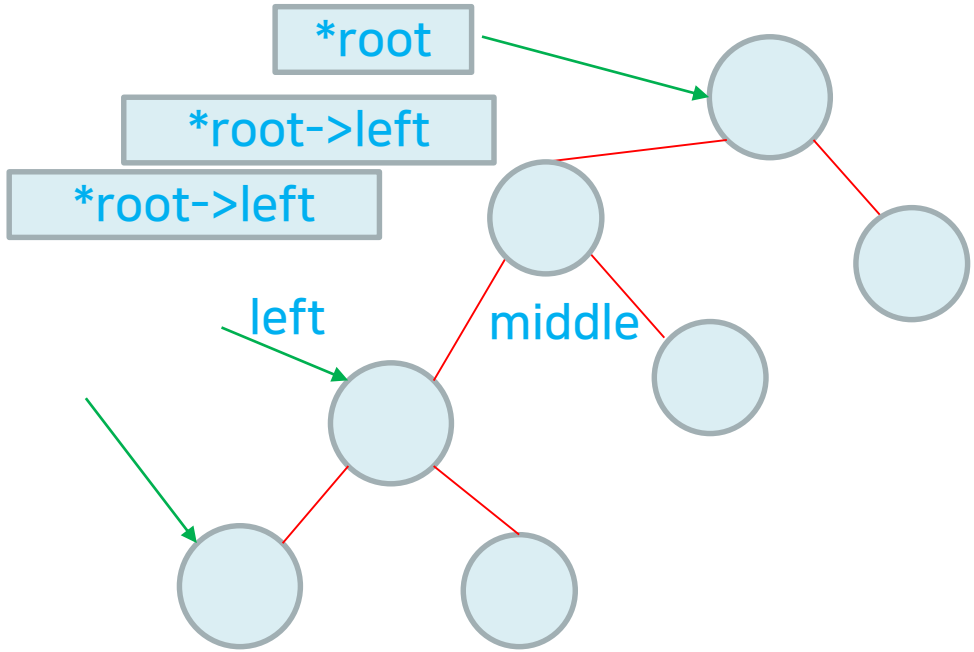


구현결과

Left-Right Case

```
rotate_middle = (*root)->left->right;
rotate_left = (*root)->left;

(*root)->left = rotate_middle;
rotate_left->right = rotate_middle->left;
rotate_middle->left = rotate_left;
```



구현결과

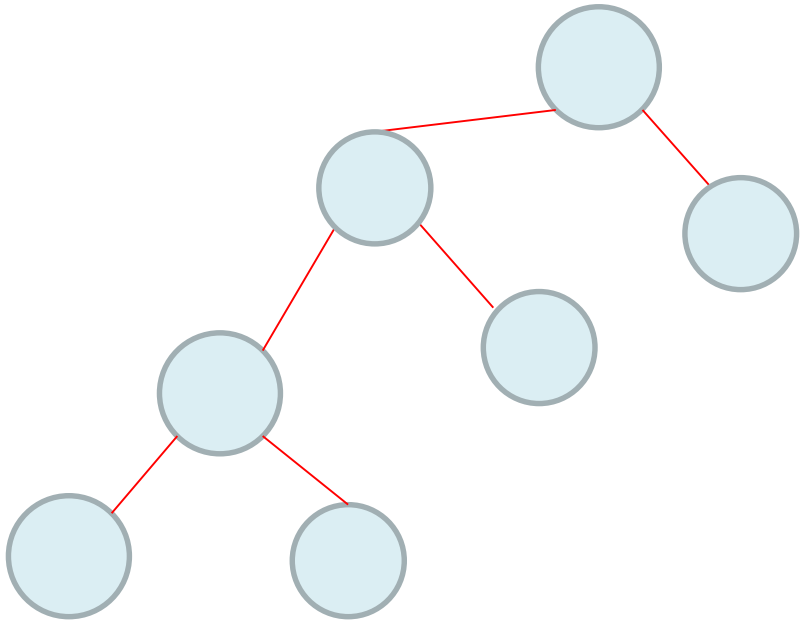
EDDI
Electronic Design Development Institute

Left-Right Case

```
rotate_middle = (*root)->left;
rotate_right = (*root);
rotate_left = (*root)->left->left;
*root = (*root)->left;
rotate_right->left = rotate_middle->right;
rotate_middle->right = rotate_right;
```

Left-Left랑 같아서 재사용 가능!!

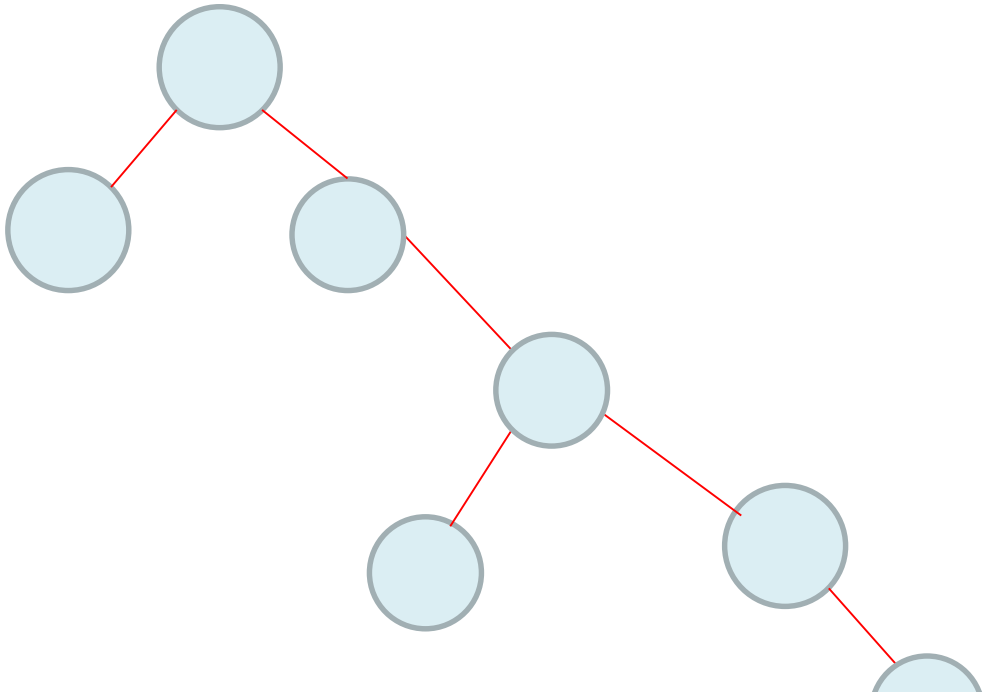
```
update_level(&rotate_right);
update_level(&rotate_middle);
```



구현결과

EDDI
Electronic Design
Development Institute

Right-Left case와 Right-Right case는
대칭성에 의해서 동일한 방법으로 하면 된다.



구현결과

EDDI
Electronic Design
Development Institute

DELETE

```
void delete_avl(avl **root, int data)
{
    stack *top = NULL;
    int num;
        while (*root)
        {
        push(&top, root);
                if ((*root)->data > data)
        {
                        root = &(*root)->left;
        }
                else if ((*root)->data < data)
        {
                        root = &(*root)->right;
        }
                else
        {
                        break;
        }
        }


    if((*root)->left && (*root)->right)
    {
        find_max(&(*root)->left, &num);
        (*root)->data = num;
    }
    else
    {
        (*root) = chg_node(*root);
        pop(&top);
        push(&top, root);
    }
```

```
while(stack_is_not_empty(top))
{

    avl **t = (avl **)pop(&top);
0
    printf("stack : %d\n",(*t)->data);
dif
    update_level(t);

    assert((*t));

    adjust_balance(t,(*t)->data);
}
```

구현결과

EDDI
Electronic Design Development Institute

DELETE

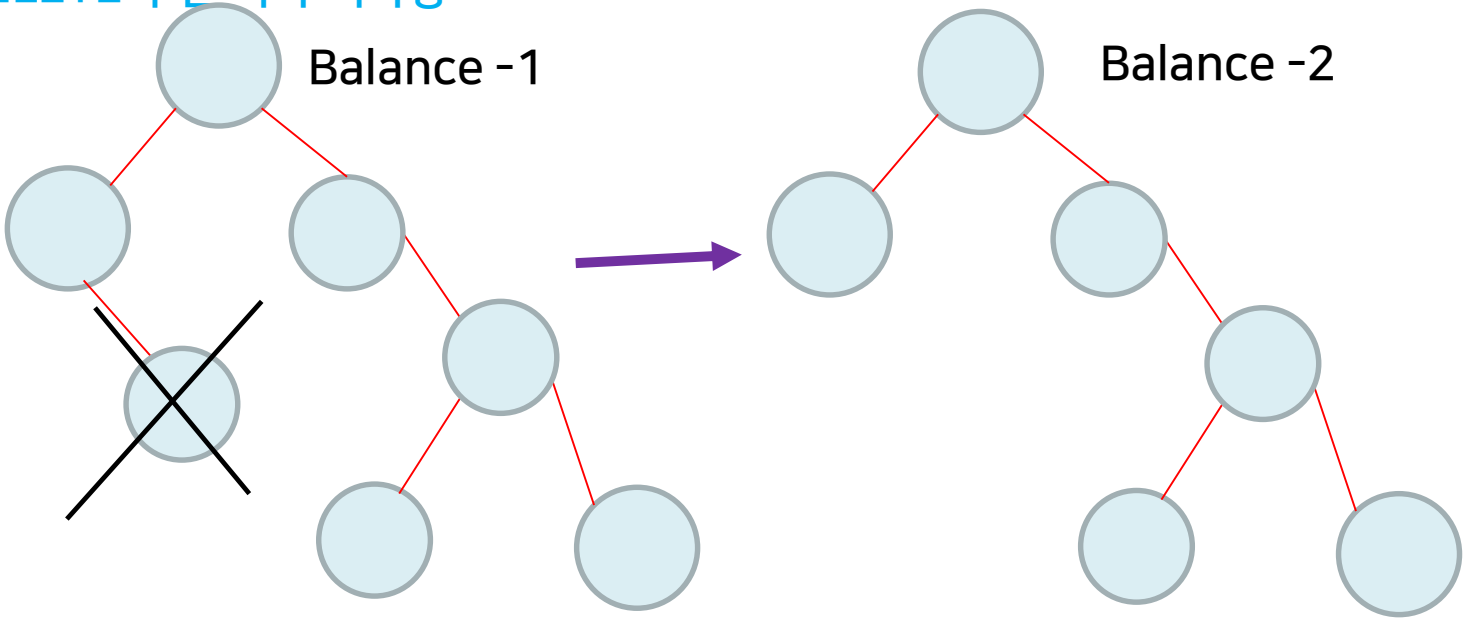
```
void find_max(avl **root, int *data)
{
    stack *top = NULL;
    while(*root)
    {
        push(&top, root);
        if((*root)->right)
        {
            root = &(*root)->right;
        }
        else
        {
            *data = (*root)->data;
            *root = chg_node(*root);
            pop(&top);
            push(&top, root);
            break;
        }
    }
    while(stack_is_not_empty(top))
    {
        avl **t = (avl **)pop(&top);
        update_level(t);
        adjust_balance(t,(*t)->data);
    }
}
```

Delete는 삭제는 binary와 같은 방식이다.
근데 어느 node에서 balance가 깨질지 모르니까 Stack에 쌓아놓고 back trackin해서 찾아야 한다..

구현결과

EDDI
Electronic Design
Development Institute

DELETE 구현 시 주의사항



Right right? Right left? -> right right로 간주하고 rotate

구현결과

EDDI
Electronic Design Development Institute

Header를 이용한 print함수 보편적으로 사용가능하게 만들기

```
void print_avl(void *tree)
{
```

```
if(!tree)
    return;
int header = *((int*)tree); // NULL POINTER EXCEPTION
avl* tmp;
```

```
typedef struct _avl avl;
struct _avl
{
    int header;
    int data;
    struct _avl *left;
    struct _avl *right;
    int level;
};
```

Void 포인터로 parameter를 선언하고
Int 포인터로 void 포인터를 type casting하면
Struct의 앞 4byte 값만 읽게 된다.
그 다음 int 값을 읽어서 switch로 구분하면 된다.