Data Structure

Week 12

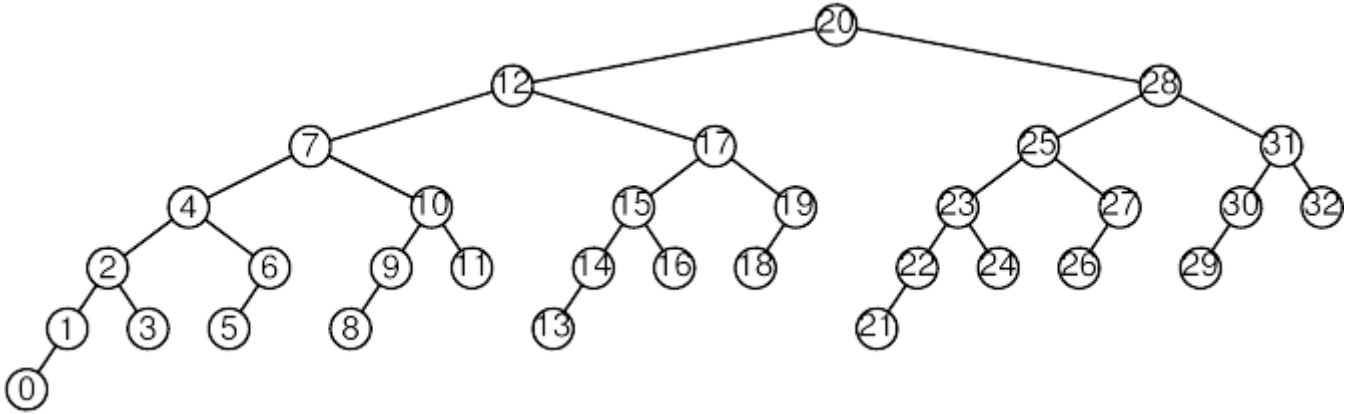
KyuDong SIM

1. 이번 주 실습 내용

-AVL Tree

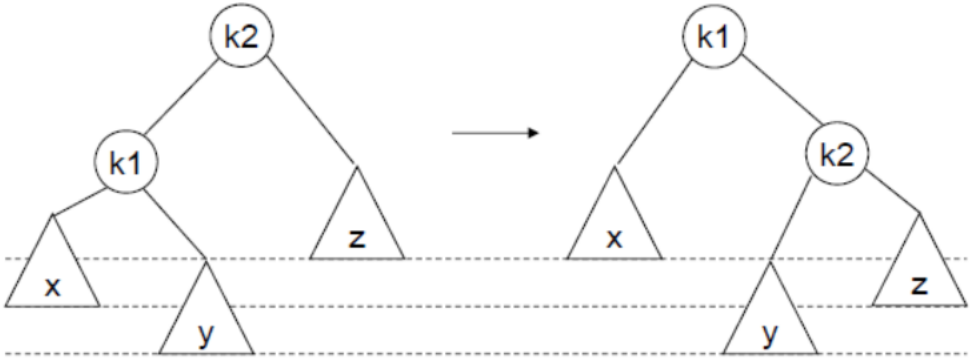
AVL Tree

- Balanced Binary search Tree (균형잡힌 이진 탐색 트리)
- 부분 트리의 높이 차이가 1보다 크지 않음



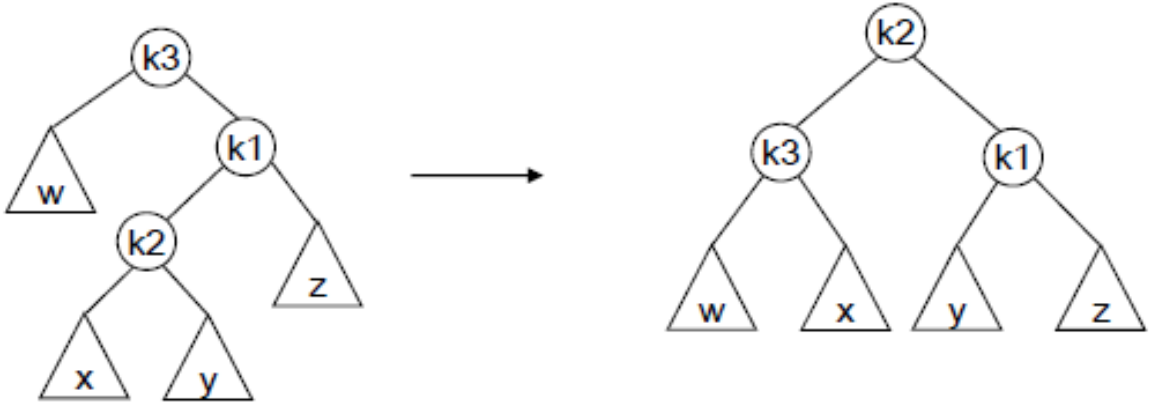
AVL Tree – Single Rotation (LL)

- K2가 k1의 leftchild를 가짐
- K1이 k2의 부모가 됨

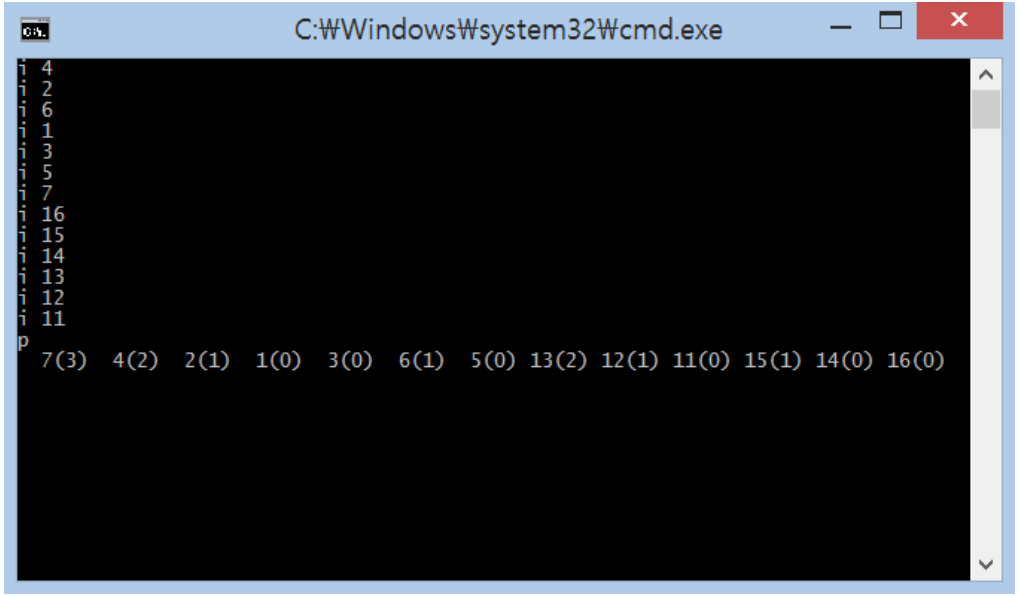


AVL Tree –Double Rotation (LR)

- K2와 K1의 Single rotation (L)
- K2와 K3의 Single rotation (R)



실습 결과 예



- 출력은 PreOrder 순을 사용
- 괄호는 각 Node의 Height

AVL Tree data type

```
struct AvlNode;
typedef struct AvlNode *Position;
typedef struct AvlNode *AvlTree;

struct AvlNode
{
    int Element;
    AvlTree Left;
    AvlTree Right;
    int Height;
};


static int Height(Position P)
{
    if (P == NULL)
        return -1;
    else
        return P->Height;
}
```

- 각 노드는 데이터, leftChild, rightChild, Height를 가짐
- Height는 각 node의 heiht를 출력, NULL일 경우 -1을 출력

Single Rotation

```
/*This function can be called only if K2 has a left child*
/*Perform a rotate between a node (K2) and its left child*/
/*Update heights, then return new root*/
static Position SingleRotateWithLeft(Position K2)
{
    Position K1;
    K1 = K2->Left;
    K2->Left = K1->Right;
    K1->Right = K2;

    K2->Height = MAX(Height(K2->Left), Height(K2->Right)) + 1;

    K1->Height = MAX(Height(K1->Left), K2->Height) + 1;

    return K1; /*new root*/
}
```

- LeftChild와 Single rotation
- MAX 함수 또는 정의가 필요

Double Rotation

```
/*This function can be called only if K3 has a left*/
/*child and K3's left child has a right child*/
/*Do the left-right double rotation*/
/*Update heights, then return new root*/
static Position DoubleRotateWithLeft(Position K3)
{
    /*Rotate between K1 and K2*/
    K3->Left = SingleRotateWithRight(K3->Left);

    /*Rotate between K3 and K2*/
    return SingleRotateWithLeft(K3);
}
```

- LeftChild와 double rotation
- SingleRotateWithRight() 함수 필요

Insert (1)

```c
AvlTree Insert(int X, AvlTree T)
{
    if (T == NULL)
    {
        /*Create and return a one-node tree*/
        T = (AvlTree) malloc(sizeof(struct AvlNode));
        if (T == NULL)
            printf("Out of space!!");
        else
        {
            T->Element = X;
            T->Height = 0;
            T->Left = NULL;
            T->Right = NULL;
        }
    }
    else if (X < T->Element)
    {
        T->Left = Insert(X, T->Left);
        if (Height(T->Left) - Height(T->Right) == 2)
            if (X < T->Left->Element)
                T = SingleRotateWithLeft(T);
            else
                T = DoubleRotateWithLeft(T);
    }
```

- Leaf Node일 경우 Node생성
- Leaf가 아닐경우 Height를 비교해서 rotation 실행

Insert (2)

```
    else
        if (X > T->Element)
        {
            T->Right = Insert(X, T->Right);
            if (Height(T->Right) - Height(T->Left) == 2)
                if (X > T->Right->Element)
                    T = SingleRotateWithRight(T);
                else
                    T = DoubleRotateWithRight(T);
        }
    /*Else X is in the tree already; we'll do nothing*/


    T->Height = MAX(Height(T->Left), Height(T->Right)) + 1;

    return T;
}
```

- LeftChild와 double rotation
- SingleRotateWithRight() 함수 필요

제출 및 알림

수업 중 확인 or 메일제출 (학번 기입)

메일 제출 :

주소 : (89kdsim@naver.com)

기한 : ~2016-05-25