Xilinx Zynq FPGA,TI DSP,
MCU 기반의
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 정한별
hanbulkr@gmail.com

<11 일 차>

AVL 트리 (이진트리 →AVL 트리)

* 이진트리에서 검색을 빨리 하기 위해서 완전한 2 진트리 구성.
*이진트리의 경우는 만약 오른쪽 한방향으로 만 데이터가 쌓이게 되서 오른쪽 끝에 10 억번째쯤에 있는 데이터를 읽어와야 할 수도 있다. 그렇기 때문에 검색하는데 엄청난 시간이 소요될 수 있다. AVL 트리의 level 개념을 도입해서 순서를 균등하게 정렬화 함으로 검색의 효율을 높일 수 있다.

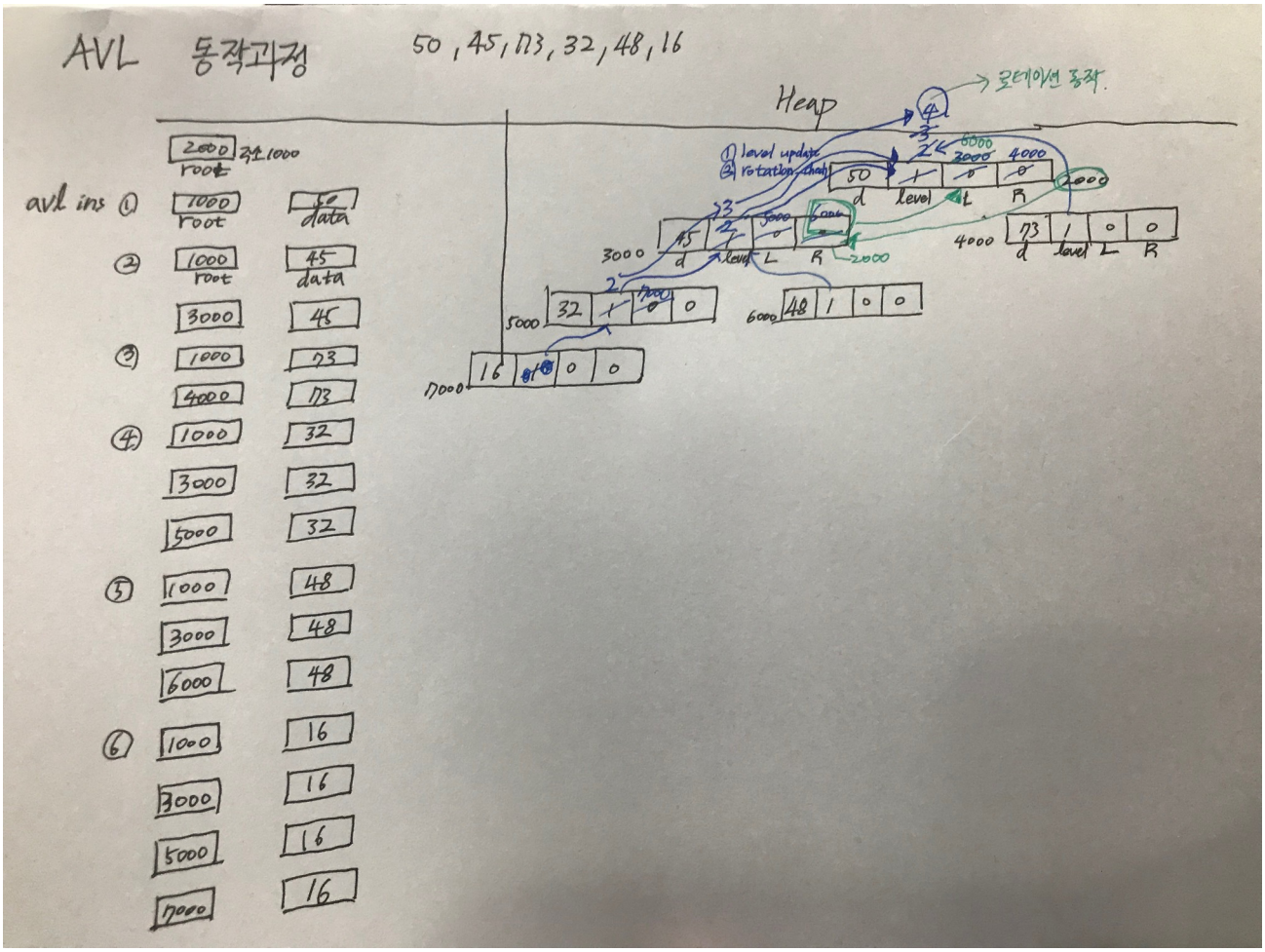
```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>

#define EMPTY 0

typedef struct __avl
{
	int data;
	int level;
	struct __avl *link_left;
	struct __avl *link_right;

}avl;

typedef enum __rot
{
	RR,
	RL,
	LL,
	LR
}rot;


avl *get_node()
{
	avl *tmp;
	tmp = (avl*)malloc(sizeof(avl));
	tmp->link_left = EMPTY;
	tmp->link_right = EMPTY;
	return tmp;
}

int update_level(avl *root)
```

```
{
		int left = root->link_left ? root ->link_left->level :0;
		int right = root->link_right ? root ->link_right->level :0;

		if(left>right)
				return left+1;
		return right +1;
}

int rotation_check(avl *root)
{
	int left = root->link_left ? root -> link_left->level : 0;
	int right = root->link_right ? root -> link_right -> level : 0;

	return right - left ;
}

avl *chg_node(avl *root)
{
		avl *tmp = root;

		if(!(root->link_left)){
			free(tmp);
			return root ->link_right;
		}
		else if(!(root->link_right)){
			free(tmp);
			return root ->link_left;
		}
}
avl *find_max(avl *root, int *data)
{
	if(root->link_right){
				root->link_right = find_max(root->link_right, data);
	}
	else{
				*data = root->data;
				root= chg_node(root);
	}
	return root;
}
int kinds_of_rot(avl *root, int data)
{
	printf("data = %d\n", data);

	//for RR and RL

	if(rotation_check(root) >1)
	{
```

```
                if(root->link_right->data >data)
                        return RL;
                return RR;
        }
        else if(rotation_check(root) <-1)
        {
                if(root->link_left->data < data)
                        return LR;
                return LL;
        }
}


avl *rotation(avl *root , int num )
{
                //RR:1
                //RL:2
                //LL:3
                //LR:4
        switch (num){
                        case 1:

                                        break;
                        case 2:

                                        break;
                        case 3:

                                        break;
                        case 4:

                                        break;
                        default:

                                        break;
        }

        return root;
}


void print(avl *root)
{
                avl *tmp =root;
                if(root){
                }
                printf("data = %d ,", tmp->data);
                if(root->link_left){
                        printf("left = %d, " ,root->link_left->data);
```

```
                }
                else
                        printf("left = NUll, ");
                if(root->link_right){
                        printf("right = %d\n", root->link_right->data);
                }
                else
                        printf("right = NULL\n");
                print(root->link_left);
                print(root->link_right);
}

void avl_ins(avl **root, int data)
{
        if(!(*root))
        {
                *root = get_node();
                (*root)->data = EMPTY;
                return ;
        }

        if((*root)->data > data)
        {
                avl_ins(&(*root)->link_left , data);
        }
        else if((*root)->data <data)
        {
                avl_ins(&(*root)->link_right , data);
        }

        (*root)->level = update_level(*root);

        if(abs(rotation_check(*root)) >1)
        {
                printf("Rotation \n");
                *root = rotation(*root,kinds_of_rot(*root,data));
        }
}
avl *debinary(avl *root,int data)
{
        int num;
        avl *tmp;
        if(root == NULL)
        {
                printf("Not Found\n");
                return NULL;
        }
        else if(root->data >data)
                root->link_left = debinary(root->link_left,data);
```

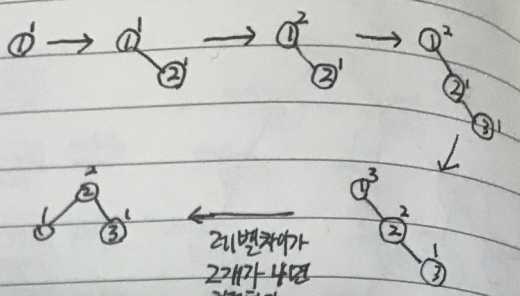
```
        else if(root->data <data)
                        root ->link_right = debinary(root->link_right, data);
        else if(root->link_left && root->link_right)
        {
                        root->link_left =find_max(root->link_left, &num);
                        root->data = num;
        }
        else
                        root=chg_node(root);
        return root;
}



int main(void)
{
        avl *root=EMPTY;
        int a[]={50,45,73,32,48,46,16,37,120,127,124};
        int i, num=0;
        int len = sizeof(a)/sizeof(int);

        for(i=0 ; i <len;i++)
                        avl_ins(&root, a[i]);
        print(root);
        root=debinary(root,50);
        print(root);

        return 0;
}
```


AVL 동작과정
50, 45, 73, 32, 48, 16
Heap
로테이션 동작.
① level update
② rotation check
avl ins
root
data
level
L
R
d

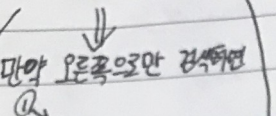
11일차

이진트리 → AVL 트리

검색을 빠르게하기 위해서 → 완전한 2진트리 구성



만약 오른쪽으로만 검색하면



10억

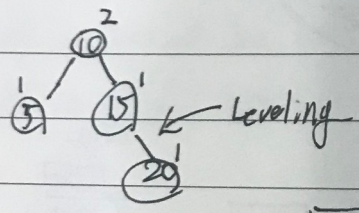
엉망이게 생겨먹는다.

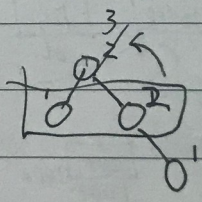
↳ 나를 기준으로 오른쪽과 왼쪽에 2차이가 나면 안된다.

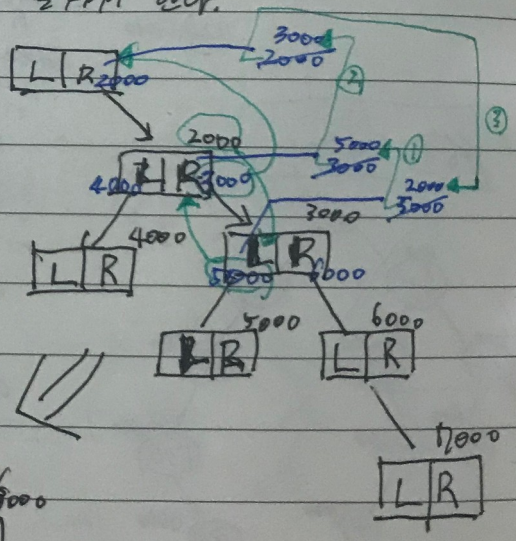
* ~~왼쪽 1개 오른쪽 2개면~~
왼쪽과 오른쪽의 높이 차가 1이하 여야 한다.

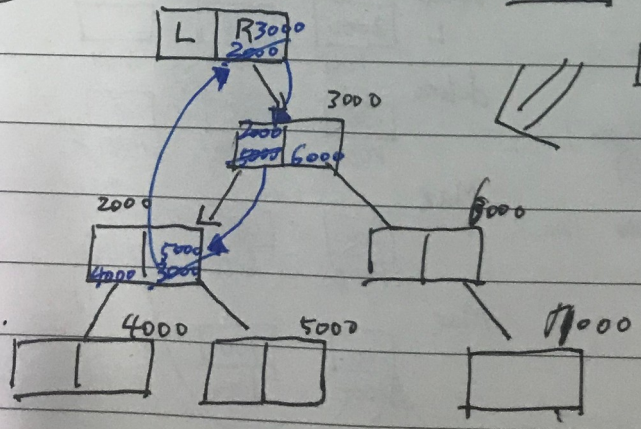
① 저장시 Level은 매긴다.

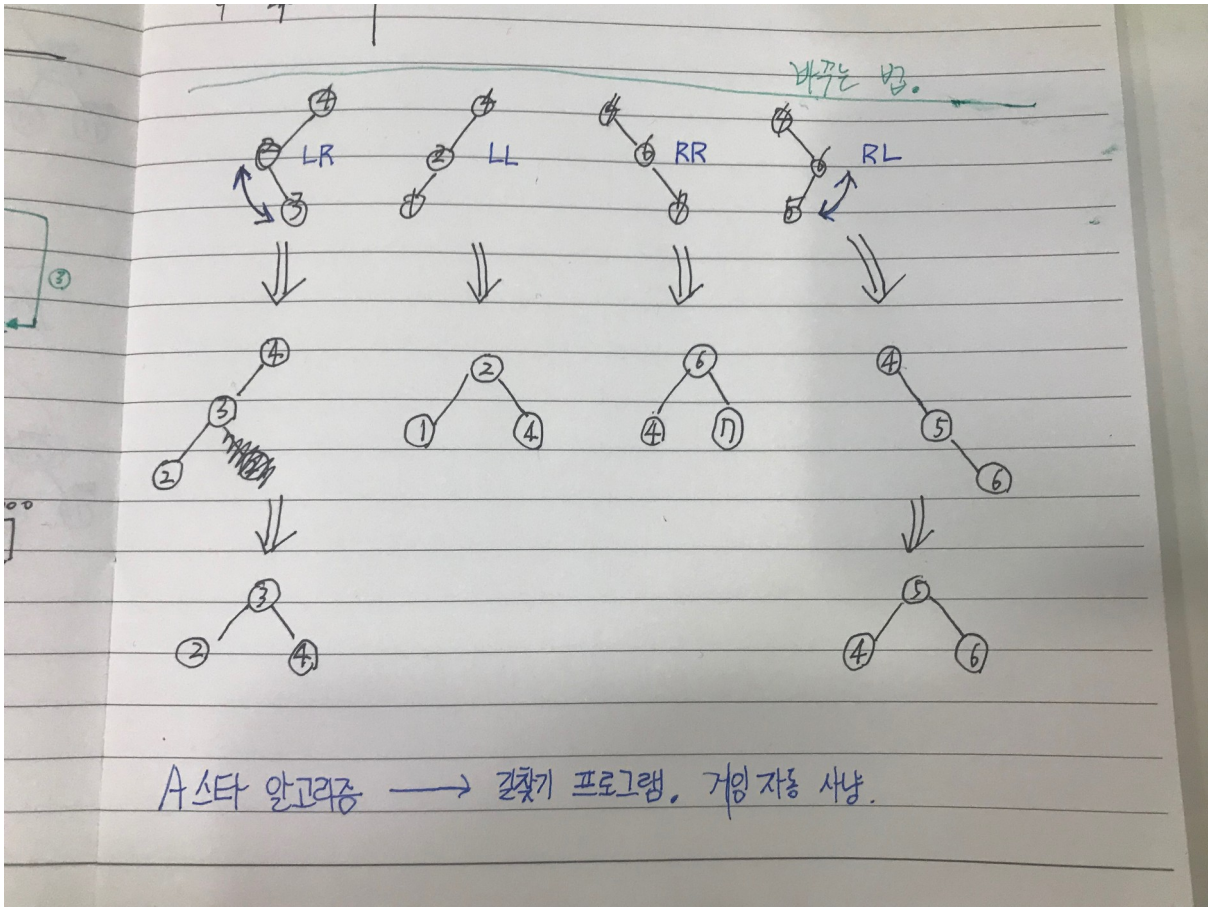
② Level을 매긴 값들이 맞게 저장되게 하나씩 맞추어야 한다.











A스타 알고리즘 ⟶ 길찾기 프로그램, 게임 자동 사냥.