**추가과제**

2014104272 이준영

**1. test.c**

#include "ADT\_avl\_tree.h"

#include <stdlib.h>

#include <stdio.h>

#define sample\_NUM 10000000

int main(){

int\* M = (int\*)malloc(sizeof(int)\*sample\_NUM);

int search\_num = sample\_NUM;

AVL\_TREE\* new\_avl = create\_avl\_tree();

for(int i=sample\_NUM-1;i>=0;i--){ **//1부터 10000000의 숫자를 avl\_tree에 넣음**

\*M = sample\_NUM-i;

AVL\_insert(new\_avl, \*M);

M++;

}

int height = 1; **//root의 높이를 1로 설정**

int data;

**/\* 1부터 10000000까지 오름차순으로 데이터를 삽입하므로 AVL\_TREE에서 가장 오른쪽 끝에**

**는 가장 큰 숫자인 10000000이 들어가고 이 숫자가 들어간 위치의 높이는 항상 최대가 되기**

**때문에 while문을 이용해 이 AVL\_TREE의 최대 높이와 가장 오른쪽에 있는 data값(=가장 큰**

**값)을 출력하도록 구성했습니다. \*/**

while(new\_avl->root->right != NULL){

new\_avl->root = new\_avl->root->right;

data = new\_avl->root->data;

height++;

}

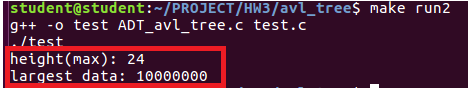
printf("height(max): %d\n",height);

printf("largest data: %d\n",data);

return 0;

}

**2 AVL\_TREE의 height 확인**

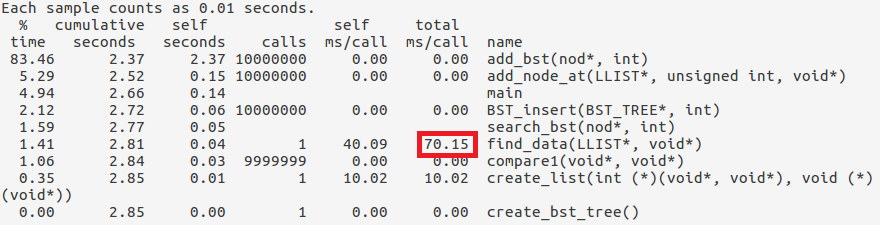


실행결과 높이가 24가 나왔습니다. 이 숫자가 avl\_tree에서 10000000개의 data를 넣었을 때 최소가 되는 값인지 확인하기 위해 수업시간에 배운 공식을 이용하면 h= +1 = 24 로 일치하는 것을 확인하였습니다. 만약 BST에 위와 같은 값을 넣었다면 높이가 10000000으로 AVL의 24와 매우 큰 차이가 나는 것을 알 수 있습니다.

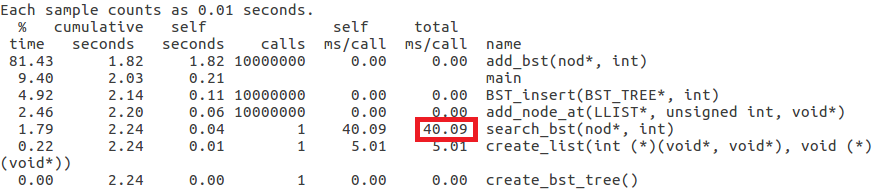
**3. 시간비교**

LLIST, BST\_TREE, AVL\_TREE 코드를 이용하여 특정 data를 찾는데 걸리는 시간을 측정했습니다.

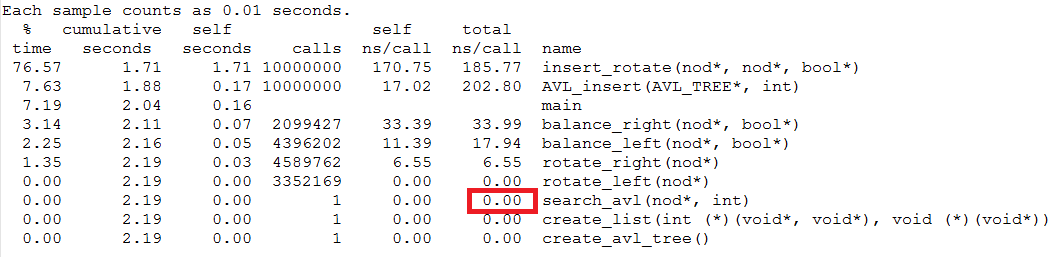
**profile\_llist.txt (sample 10000000)**



**profile\_tree.txt (sample 10000000)**



**profile\_avl\_tree.txt (sample 10000000)**



위의 자료들을 보면 AVL\_TREE의 경우에 data를 찾는데 소모되는 시간이 거의 없음(0ns/call)을 볼 수 있습니다.

(data를 찾는데 소모되는 시간: LLIST >= BST\_TREE >= AVL\_TREE) 확인할 수 있었습니다.