BTree Report

과목명 : 화일처리

담당교수 : 김혁만

학과 : 소프트웨어 융합

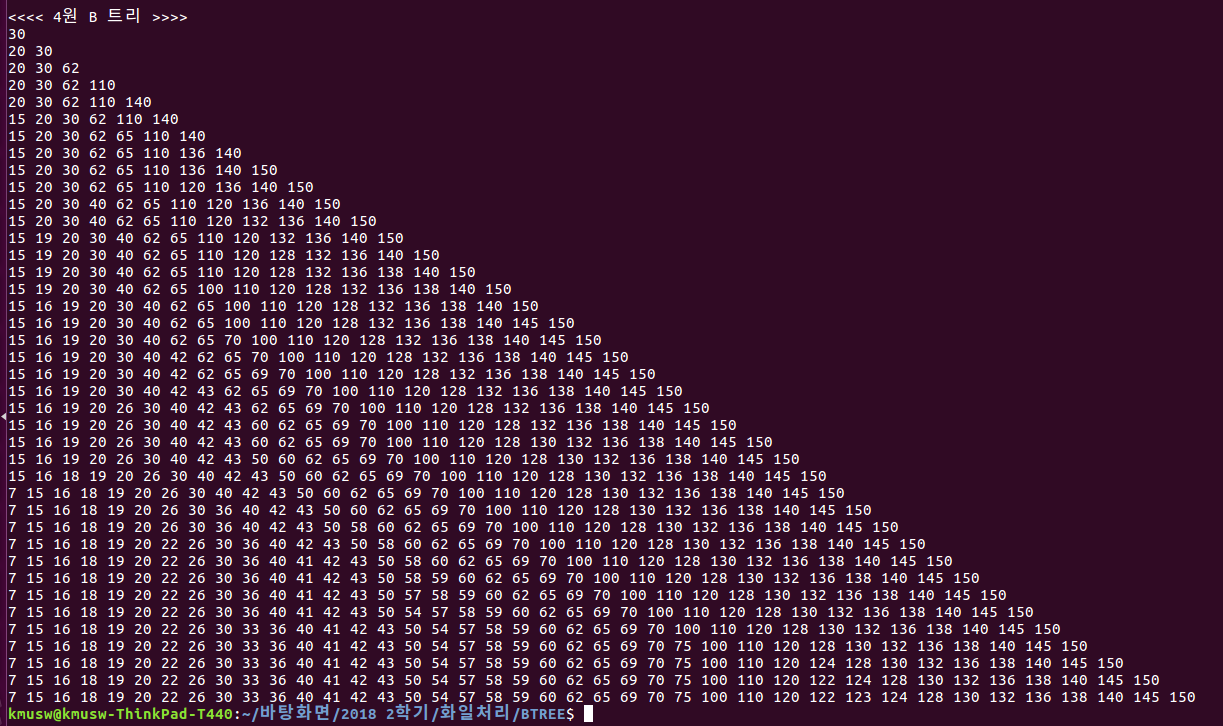
학번 : 20143038

이름 : 김선필

제출일 : 2018 .11 . 22

* 출력 결과





* CODE

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

국민대학교 화일처리 BTree Code

2018. 11. 22

작성자 : 김선필(20143038)

코드 요약 : Node class와 BT class를 이용한 BTree insert 알고리즘 구현

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <vector>

#include <stack>

#include <iostream>

#include <fstream>

using namespace std;

class Node{

public:

bool full; // fullflag

int nodeSize; // nodeSize

int n; // 현재 노드안의 key개수

bool overflow; // overflow flag

int idx; // subtree idx

vector<int> key; // key data

vector<Node\*> child; // child pointer

Node(int m)

{

nodeSize = m;

overflow = false;

n = 0;

idx = 1;

key.resize(m,0);

child.resize(m,0);

}

void insertKey(int);

Node\* backBuffer();

int getIdx();

int getMid()

{

return key[nodeSize/2];

}

friend class BT;

};

class BT{

private:

Node\* root; // 트리의 root

int M; // m원 트리

public:

BT(int size)

{

root = new Node(size);

M = size;

}

void insert(int);

void traverse(Node\*);

Node\* getRoot()

{

return root;

}

};

void Node::insertKey(int a)

{

int i = 1;

vector<int>::iterator it = key.begin(); // key vector의 시작점

// 들어갈 key값 비교

while(i <= n && a > key[i])

{

i++;

}

for(int j = 0; j<i;j++)

{

it++;

}

key.insert(it,a);

for(int j = n+1;j>i;j--)

{

child[j] = child[j-1];

}

idx = i;

n++;

return;

}

void BT::insert(int key)

{

Node \*p = root; //root 포인터

Node \*x = new Node(M);

Node \*y = NULL;

stack<Node\*> childs;

bool isFull = false; // full check flag

bool finish = false; // finish flag

int i;

// root가 비어 있을 경우

if(p->n == 0)

{

p->insertKey(key);

return;

}

// PPT 상의 do while문으로 구현

do{

i = 1;

while(i <= p->n && key > p->key[i])

{

i++;

}

if(i<=p->n && key == p->key[i])

{

cout << "동일값 예외처리" << endl;

return;

}

else

{

childs.push(p);

}

}while((p=p->child[i-1]) != NULL);

// stack에서 하나씩 빼면서 key를 넣을 공간 탐색

p = childs.top();

childs.pop();

do{

if(p->n < M-1)

{

p->insertKey(key);

finish = true;

if(isFull){

int j = p->getIdx();

p->child[j] = y;

}

}

else

{

// overflow

Node \*temp = new Node(p->nodeSize+1);

temp->key = p->key;

for(int i = 0; i< p->nodeSize;i++)

{

temp->child[i] = p->child[i];

}

temp->n = p->n;

temp->insertKey(key);

key = temp->getMid();

int j = temp->getIdx();

if(isFull)

{

Node\* Y = y;

temp->child[j] = Y;

}

// 중요한 과정.. 안하면 잘 안됨..

p->key.clear();

p->key.resize(p->nodeSize);

p->child.clear();

p->child.resize(p->nodeSize);

p->n = 0;

int k;

for(k = 1; k<temp->nodeSize/2;k++)

{

p->key[k] = temp->key[k];

p->child[k-1] = temp->child[k-1];

p->n++;

}

p->child[k-1] = temp->child[k-1];

y = new Node(M);

y = temp->backBuffer();

isFull = true;

}

if(!childs.empty())

{

p=childs.top();

childs.pop();

}

// 높이가 증가할 경우

else if(!finish)

{

Node \*newHeight = new Node(M);

newHeight->key[1] = key;

newHeight->n = 1;

newHeight->child[0] = p;

newHeight->child[1] = y;

root = newHeight;

finish = true;

}

}while(!finish);

}

// MID값의 뒷부분을 리턴

Node\* Node::backBuffer(){

Node\* back = new Node(nodeSize-1);

int j = 1;

int i = 0;

for (i = nodeSize/2+1; i < nodeSize; i++) {

back->key[j] = key[i];

back->child[j-1] = child[i-1];

back->n++;

j++;

}

back->child[j - 1] = child[i - 1];

return back;

}

int Node::getIdx(){

int i = idx;

idx = 1; // idx 초기화

return i;

}

// 중위 순회 알고리즘으로 traverse 함수 구현

void BT::traverse(Node\* root){

if(root == NULL){

return;

}

else{

int i;

for ( i = 0; i < root->n; i++) {

traverse(root->child[i]);

if(i != M){

cout << root->key[i+1] << ' ';

}

}

if (i < M) {

traverse(root->child[i]);

}

}

}

int main()

{

int temp;

ifstream in("data.txt"); // 데이터량이 많기에 파일 데이터로 읽어들임

BT bt3(3); // 3원 B 트리

BT bt4(4); // 4원 B 트리

vector<int> data;

data.resize(40);

for(int i =0 ; i<40;i++)

{

in >> temp;

data[i] = temp;

}

cout << "<<<< 3원 B 트리 >>>>" << endl;

for (int i = 0; i < 40; i++) {

bt3.insert(data[i]);

bt3.traverse(bt3.getRoot());

cout << endl;

}

cout <<endl <<"<<<< 4원 B 트리 >>>>" << endl;

for (int i = 0; i < 40; i++) {

bt4.insert(data[i]);

bt4.traverse(bt4.getRoot());

cout << endl;

}

}