



컴퓨터 알고리즘과 실습



학번 : 2016110056

학과 : 불교학부

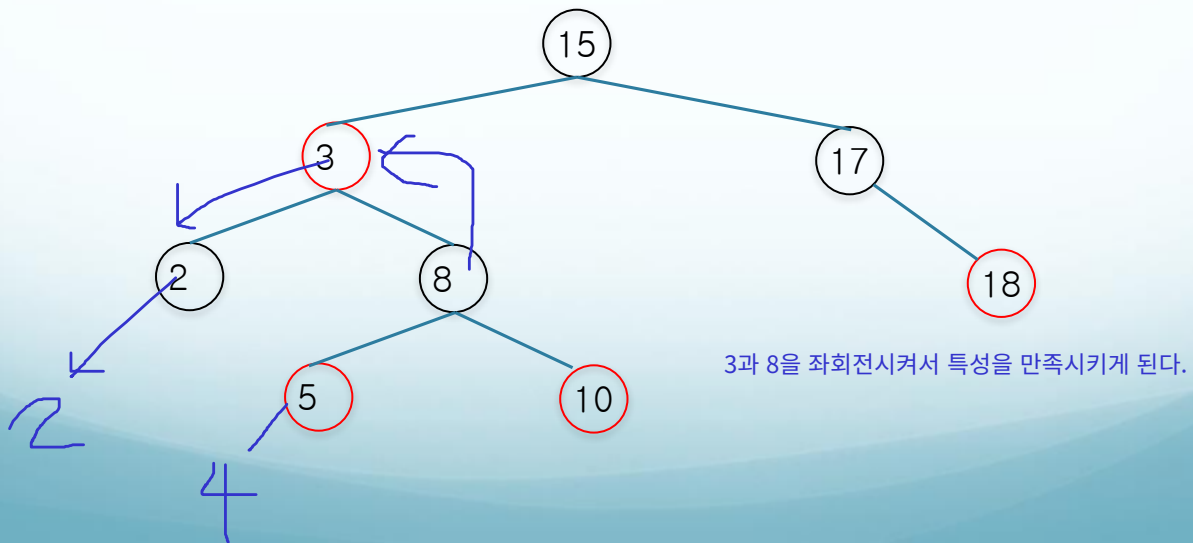
이름 : 박승원

날짜 : 2017년 4월 12일

문제 1

- 레드블랙트리

다음 레드블랙 트리에 4를 삽입하였을 때, 레드블랙의 특성을 유지하기 위하여 어떠한 단계들을 거치는지 트리 노드의 색이나 위치등이 변경할 때마다 이유를 설명하고 그림을 그리시오. (레드블랙트리의 어떤 특성을 위반하고 어떤 수업시간에 다룬 어떤 경우에 해당되는지, 그래서 어떤 방법을 적용했지는 등을 구체적으로 적으시오)



Listing 1: rotateLL과 rotateRR의 구현

```
#pragma once
#include<algorithm>
#include"tgraph.h"

template <typename T> class Tree : public Graph<T>
{
public:
    void insert (T n) {
        Vertex<T>* p = new Vertex<T>;
        p->data = n;
        p->edge = Graph<T>::insert(p->edge, nullptr, 0);
        p->edge = Graph<T>::insert(p->edge, nullptr, 0);
        Graph<T>::root = insert(Graph<T>::root, p);
    }

    Vertex<T>* find(T n) {
        return find(Graph<T>::root, n);
    }
}
```

```

int height() {
    return height(Graph<T>::root);
}

```

```

Vertex<T>* data() {
    connect();
    return Graph<T>::root;
}

```

```

void view() {
    connect();
    Graph<T>::view();
}

```

protected:

```

std :: vector<Vertex<T>*> vts;

```

```

void connect() {
    vts . clear ();
    connect(Graph<T>::root);
    Vertex<T>* p = Graph<T>::root;
    for(int i=1; i<vts.size () ; i++) {
        p->vertex = vts[i ];
        p = p->vertex;
    }
}

```

private:

```

void connect(Vertex<T>* p) {//connect vertexes for graph compatibility
    if (!p) return;
    vts . push . back (p);
    if (p->edge) {
        connect(p->edge->vertex);
        connect(p->edge->edge->vertex);
    }
}

```

```

int height(Vertex<T>* tree) {
    if (! tree ) return 0;
    if (! tree->edge) return 1;
    return 1 + std :: max(height(tree->edge->vertex), height(tree->edge->edge->vertex));
}

```

```
}
```

```
Vertex<T>* find(Vertex<T>* p, T n) {  
    if (!p) return nullptr;  
    if (p->data == n) return p; // all below is <else if> due to return  
    if (n < p->data) {  
        p->edge->v = 1;  
        return find(p->edge->vertex, n);  
    } else {  
        p->edge->edge->v = 1;  
        return find(p->edge->edge->vertex, n);  
    }  
}
```

```
Vertex<T>* insert(Vertex<T>* p, Vertex<T>* n) {// insert by comparing value  
    if (!p) return n; // to make binary tree  
    if (n->data < p->data) p->edge->vertex = insert(p->edge->vertex, n);  
    else p->edge->edge->vertex = insert(p->edge->edge->vertex, n);  
  
    // avl tree rotate  
    int ll=0, rr=0, lr=0, rl=0, r=0, l=0;  
    if (p->edge->vertex) {  
        ll = height(p->edge->vertex->edge->vertex);  
        lr = height(p->edge->vertex->edge->edge->vertex);  
        l = 1 + std::max(ll, lr);  
    }  
    if (p->edge->edge->vertex) {  
        rr = height(p->edge->edge->vertex->edge->edge->vertex);  
        rl = height(p->edge->edge->vertex->edge->vertex);  
        r = 1 + std::max(rl, rr);  
    }  
    if (abs(l-r) > 1) {  
        int ar[] = { ll, lr, rr, rl };  
        int biggest = *std::max_element(ar, ar+4);  
        if (biggest == ll) return rotate_LL(p);  
        else if (biggest == rr) return rotate_RR(p);  
        else if (biggest == lr) {  
            p->edge->vertex = rotate_RR(p->edge->vertex);  
            return rotate_LL(p);  
        } else if (biggest == rl) {  
            p->edge->edge->vertex = rotate_LL(p->edge->edge->vertex);  
            return rotate_RR(p);  
        }  
    }  
}
```

```

    }
}
return p;
}

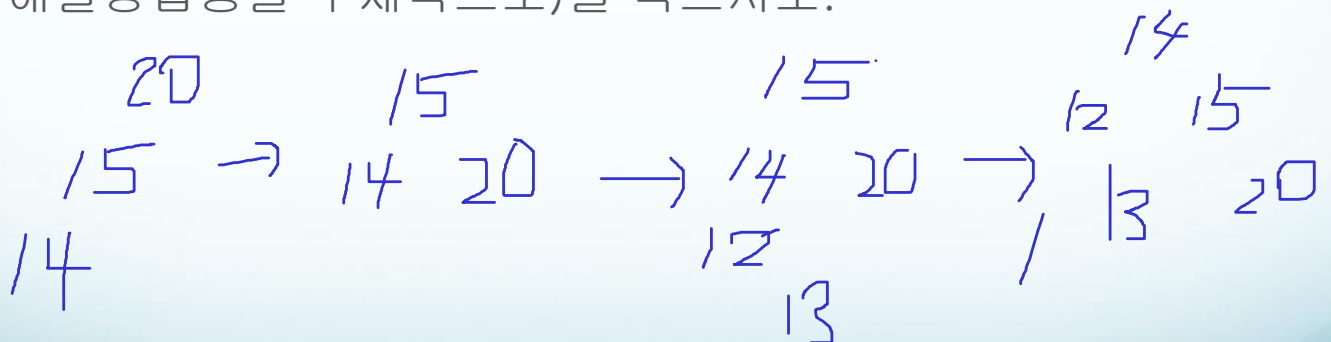
Vertex<T>* rotate_LL(Vertex<T>* A) {
    Vertex<T>* B = A->edge->vertex;
    A->edge->vertex = B->edge->edge->vertex;
    B->edge->edge->vertex = A;
    return B;
}

Vertex<T>* rotate_RR(Vertex<T>* A) {
    Vertex<T>* B = A->edge->edge->vertex;
    A->edge->edge->vertex = B->edge->vertex;
    B->edge->vertex = A;
    return B;
}
};

```

문제 2

빈 레드블랙 트리에 키 20,15,14,12,13,1을 차례로 삽입했을 때 만들어지는 레드블랙 트리의 모양을 보이시오. 삽입시마다 변경되는 트리 모양과 이유(위반되는 이유나 항목과 해당되는 위반 경우와 해결방법등을 구체적으로)를 적으시오.



처음에 14 삽입시에 우회전이 필요하고 13 삽입시 다시 14와 15를 회전시켜야 한다.

문제 3

카멜레온 문제

- 어느 섬에 a개의 빨강, b개의 노랑, c개의 네온 카멜레온이 있다. 어느 두 카멜레온이 만나면 제 3의 색으로 변한다고 하자. 예를 들어 빨강과 노랑 카멜레온이 만나면 각각 네온 카멜레온으로 변한다. 어느순간 이 섬에 모든 카멜레온이 한가지 색으로 통일될 수 있을까? 만약 가능하다면 그렇게 되는 전략을 서술하고 그렇게 되는 데 걸리는 시간(만남 횟수)을 적어 보아라. 단, a, b, c의 값은 서로 다르다.
- 만약 a=1, b=3, c=5 라면 섬의 카멜레온이 한가지 색으로 통일 되는 것이 가능할까?
- 만약 a=37, b=61, c=27 이라면 섬의 카멜레온이 한가지 색으로 통일 되는 것이 가능할까?

```
// 2016110056 박승원
#include<array>
#include<vector>
#include<iostream>
#include<algorithm>
using namespace std;

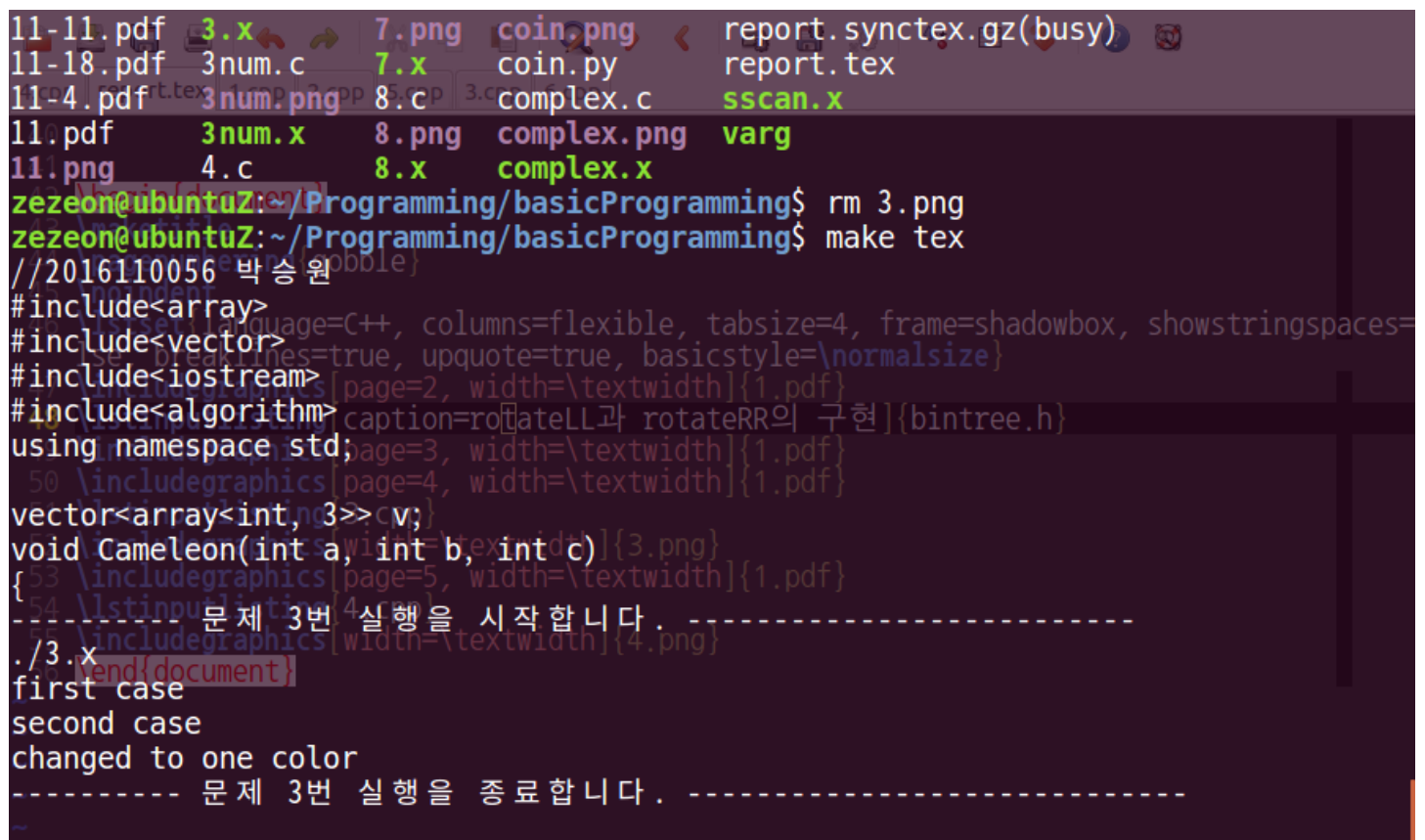
vector<array<int, 3>> v;
void Cameleon(int a, int b, int c)
{
    int k=0;
    if(!a) k++; if(!b) k++; if(!c) k++;
    if(k == 2) {
        cout << "changed to one color" << endl;
        return;
    }
    if(a<0 || b<0 || c<0) return;
    array<int, 3> ar;
    ar[0] = a; ar[1] = b; ar[2] = c;
    if(find(v.begin(), v.end(), ar) != v.end()) return;
```

```

    else v.push_back(ar);
    Cameleon(a-1, b-1, c+2);
    Cameleon(a+2, b-1, c-1);
    Cameleon(a-1, b+2, c-1);
}

int main()
{
    cout << "first case" << endl;
    Cameleon(1,3,5);
    cout << "second case" << endl;
    Cameleon(37,61,27);
}

```



```

11-11.pdf 3.x 7.png coin.png report.synctex.gz(busy)
11-18.pdf 3num.c 7.x coin.py report.tex
11-4.pdf 3num.png 8.c complex.c sscan.x
11.pdf 3num.x 8.png complex.png varg
11.png 4.c 8.x complex.x
zezeon@ubuntuZ:~/Programming/basicProgramming$ rm 3.png
zezeon@ubuntuZ:~/Programming/basicProgramming$ make tex
//2016110056 박승원
#include<array>
#include<vector>
#include<iostream>
#include<algorithm>
using namespace std;
50 \includegraphics[page=2, width=\textwidth]{1.pdf}
vector<array<int,3>> v;
void Cameleon(int a, int b, int c)
{
53 \includegraphics[page=3, width=\textwidth]{1.pdf}
54 \lstinputlisting[page=4, width=\textwidth]{1.pdf}
----- 문제 3번 실행을 시작합니다. -----
./3.x
first case
second case
changed to one color
----- 문제 3번 실행을 종료합니다. -----

```

첫번째 케이스는 불가능하고 두번째 케이스는 가능하다.

문제 4

정수 X에 사용할 수 있는 연산은 다음과 같이 세 가지 이다.

1. X가 3으로 나누어 떨어지면, 3으로 나눈다.
2. X가 2로 나누어 떨어지면, 2로 나눈다.
3. 1을 뺀다.

정수 N이 주어졌을 때, 위와 같은 연산 세 개를 적절히 사용해서 1을 만들려고 한다. 연산을 사용하는 횟수의 최소값을 출력하시오.

입력 : 1보다 크거나 같고, 10^6 보다 작거나 같은 자연수 N

출력 : 연산을 하는 횟수의 최소값

결과 : 10234 -> 10, 82813 -> 13

조건 : 프로그램 실행시간 2초이내

```
//2016110056 박승원
```

```
#include<iostream>
```

```
#include<chrono>
```

```
using namespace std;
```

```
int main(int ac, char** av)
```

```
{
```

```
    auto from = chrono::system_clock::now();
```

```
    int n = atoi(av[1]);
```

```
    int ar[n+1];
```

```
    ar[n] = 0;
```

```
    for(int i=n-1; i>0; i--) {
```

```
        ar[i] = ar[i+1] + 1;
```

```
        if(i*2 <= n) ar[i] = std::min(ar[i], ar[i*2]+1);
```

```
        if(i*3 <= n) ar[i] = std::min(ar[i], ar[i*3]+1);
```

```
    }
```

```
    cout << ar[1] << endl;
```

```
    auto to = chrono::system_clock::now();
```



```
    cout << (to - from).count()/1000 << " milliseconds" << endl;
}
```

```
zezeon@ubuntuZ: ~/Programming/basicProgramming
People beset by desire run here and there, like a snared rabbit, and
those trapped in the bonds of attachments keep returning for a long time
to suffering. 342
4.cpp 5.cpp 3.cpp 6.cpp
1 //2016110056 박승원
zezeon@ubuntuZ:~/Programming/basicProgramming$ cd basicProgramming/
/home/zezeon/Programming/basicProgramming
zezeon@ubuntuZ:~/Programming/basicProgramming$ ./4.x 10234
12
165 milliseconds
zezeon@ubuntuZ:~/Programming/basicProgramming$ ./4.x 80323
17
1883 milliseconds
zezeon@ubuntuZ:~/Programming/basicProgramming$ head 4.cpp
//2016110056 박승원
#include<iostream>
#include<chrono>
using namespace std;
16
17
18
19
20
21
22
int main(int ac, char** av)
{
    auto from = chrono::system_clock::now();
    auto to = chrono::system_clock::now();
    int n = atoi(av[1]);
    cout << (to - from).count()/1000 << " milliseconds" << endl;
}
```