EDDI
Electronic Design
Development Institute

에디로봇아카데미

임베디드 마스터 Lv2 과정

제 1기

2021. 12. 03

손표훈

CONTENTS

EDDI
Electronic Design
Development Institute

- AVL트리 비재귀 삭제 구현 문제점
- RedBlack트리 구현 전략
- 유압 바리케이드 : 부피 압력 방식

AVL 트리 비재귀 삭제 : 문제점

- 밸런싱 조정 함수에서 data를 파라메터로 받는 방법 – 2회 이상 연속회전시 문제 발생

```
void nr_insert_avl(avl **root, int data)
{
        avl **loop = root;
        avl *parent = NULL;

        while(*loop)
        {
                //이전 노드를 parent로 잡고 있어서 parent 연결이 끊긴 상황 발생
                parent = *loop;
                if((*loop)->data > data)
                        loop = &(*loop)->left;
                else if((*loop)->data < data)
                        loop = &(*loop)->right;
                else
                        break;
                //parent = *loop;
        }

        *loop = create_avl_node();
        (*loop)->data = data;
        (*loop)->parent = parent;
        (*loop)->level = 1;

        nr_update_level(root, data);
        adjust_balance(root, data);
}
```

```
void adjust_balance(avl **root, int data)
{
        avl **loop = NULL;
        int factor = 0;

        loop = find_tree_data(root, data);

        while(*loop)
        {
                printf("loop = %d\n", (*loop)->data);
                factor = calc_balance_factor(loop);
                if(ABS(factor) > 1)
                {
                        printf("unbalanced node = %d\n", (*loop)->data);
                        printf("root = %d\n", (*root)->data);
                        nr_run_rotation(factor, root, loop, data);
                        printf("after rotation root = %d\n", (*root)->data);
                        //print_avl(*root);
                        break;
                }
                loop = &(*loop)->parent;
        }
}
```

```
void RL_rotation(avl **root, avl **cursor)
{
        avl *top, *mid, *bot;

        top = *cursor;
        mid = top->right;
        bot = mid->left;

        bot->parent = top;
        mid->parent = bot;

        if(bot->right)
                bot->right->parent = mid;

        mid->left = bot->right;

        bot->right = mid;
        top->right = bot;

        update_level(&top);
        update_level(&mid);
        update_level(&bot);

        cursor = &bot->parent;

        RR_rotation(root, cursor);
}
```

→ Insert에서 밸런싱 함수의 파라메터로 데이터를 전달

→ 밸런싱 함수에서 데이터가 저장된 노드를 찾아 해당 노드부터 밸런싱이 깨진 노드를 찾는다

→ 밸런싱이 깨진 노드를 찾아 회전 함수의 파라메터로 전달

→ Loop의 값이 회전 시 계속 변경되면서 link가 끊김

AVL 트리 비재귀 삭제 : 문제점

- 해결

```
*loop = create_avl_node();
(*loop)->data = data;
(*loop)->parent = parent;

nr_update_level(loop);

//이전 코드에서 밸런싱 함수에 data를 파라메터로 넣었음
//data를 넣었을 때 문제점 :
//새로운 데이터가 삽입된 노드부터 시작하여 밸런스 깨진 노드를 찾음
//밸런싱 함수안에서 회전이 발생하고 회전 후 parent가 변하게 되어 link 깨짐 발생
//해결방법 :
//insert시 고정된 parent의 주소를 밸런싱 함수의 파라메터로 넘김
nr_balancing_tree(root, &(*loop)->parent);
```

```
void nr_balancing_tree(avl **root, avl **cursor)
{
        avl *loop = *cursor;
        int factor;

        while(loop)
        {
                factor = calc_balance_factor(&loop);
                //printf("loop %d's factor = %d\n", loop->data, factor);

                if(ABS(factor) > 1)
                {
                        nr_rotation_tree(root, &loop, factor);
                }

                loop = loop->parent;
        }
}
```

→ Insert시 새로 생성된 노드의 parent를 파라메터로 전달하면 고정된 parent로 인해 회전 후에도 link가 안깨짐

* 2중 포인터 사용시 전략 구성 단계에서 귀찮다고 함수 스택 그릴 때 main과 중간 과정들 빼먹지 말자...

RedBlack 트리 구현전략

EDDI
Electronic Design Development Institute

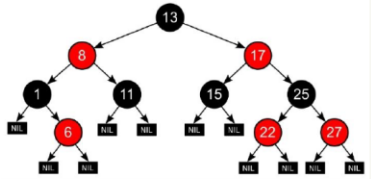
- RedBlack Tree란?

※ Red - Black Tree

1. Red-Black Tree ?

In computer science, a **red–black tree** is a kind of self-balancing binary search tree. Each node stores an extra bit representing "color" ("red" or "black"), used to ensure that the tree remains balanced during insertions and deletions.[3]

When the tree is modified, the new tree is rearranged and "repainted" to restore the coloring properties that constrain how unbalanced the tree can become in the worst case. The properties are designed such that this rearranging and recoloring can be performed efficiently.



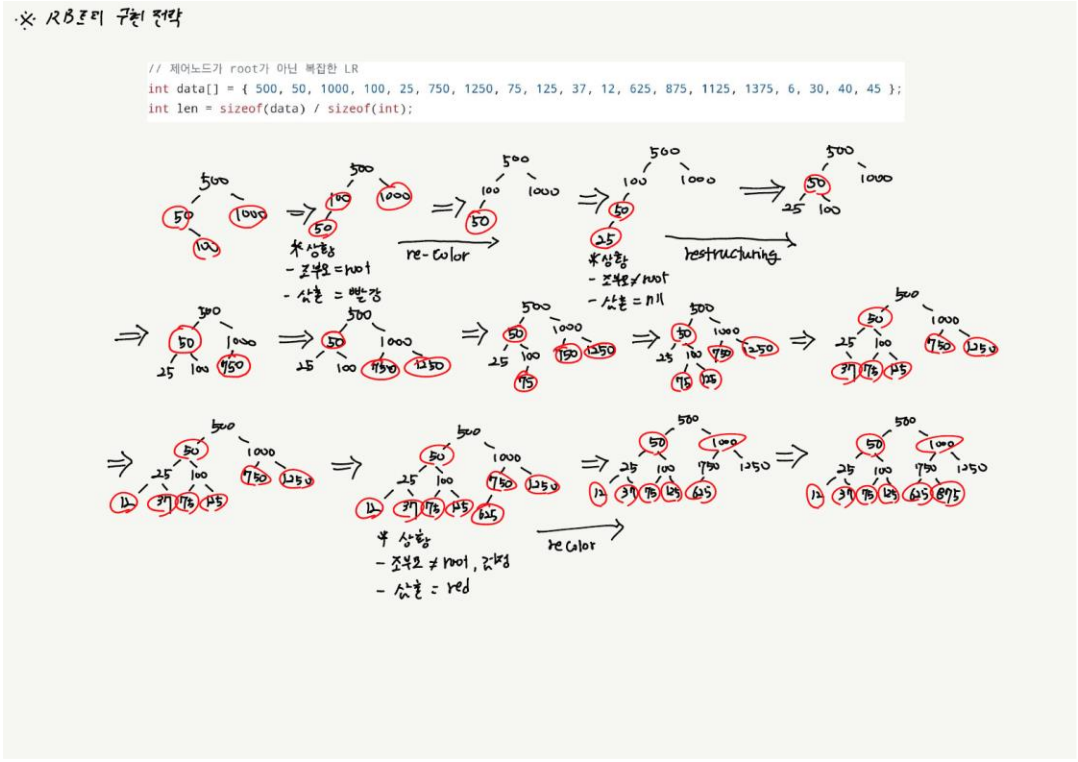
The re-balancing is not perfect, but guarantees searching in $\mathcal{O}(\log n)$ time, where n is the number of nodes of the tree. The insertion and deletion operations, along with the tree rearrangement and recoloring, are also performed in $\mathcal{O}(\log n)$ time.[4]

Tracking the color of each node requires only one bit of information per node because there are only two colors. The tree does not contain any other data specific to its being a red–black tree, so its memory footprint is almost identical to that of a classic (uncolored) binary search tree. In many cases, the additional bit of information can be stored at no additional memory cost.

2. RB Tree의 특성
(1) root 노드는 black
(2) 노드의 색은 black 또는 red
(3) 모든 leaf 노드는 black
(4) red의 자식노드는 black (red는 연속으로 올 수 없다) → black의 자식은 black이 될 수 있다.
(5) root 부터 leaf까지의 black 개수는 항상 동일하다. → 모든 leaf(nil) 부터 root까지의 경로를 말함
root의 왼쪽, 오른쪽의 black node의 개수도 같다?
(6) 삽입될 노드는 red이다.

RedBlack 트리 구현전략

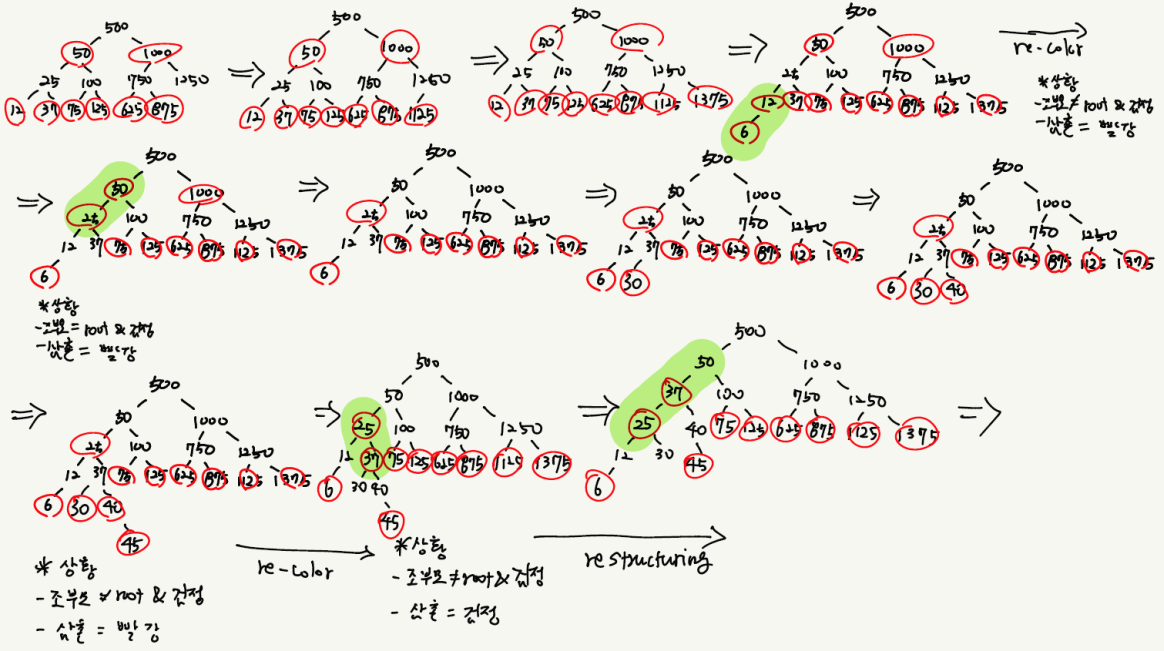
- RedBlack Tree 동작



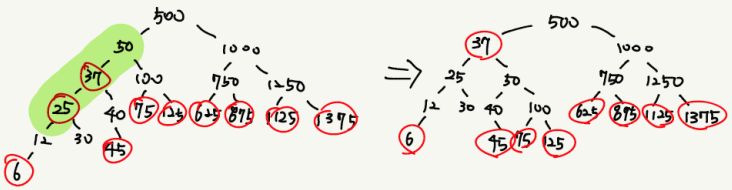
RedBlack 트리 구현전략

EDDI
Electronic Design Development Institute

```
// 제어노드가 root가 아닌 복잡한 LR
int data[] = { 500, 50, 1000, 100, 25, 750, 1250, 75, 125, 37, 12, 625, 875, 1125, 1375, 6, 30, 40, 45 };
int len = sizeof(data) / sizeof(int);
```



RedBlack 트리 구현전략



* RB트리의 double-red 해결 방안 → 부모=자식=red

1. re-Coloring : 삼촌 = red

(1) 조부모 = root : 부모, 삼촌의 색을 변경

(2) 조부모 ≠ root : 조부모, 부모, 삼촌의 색을 변경

2. restructuring : 삼촌 = black or nil

(1) 조부모 = root

① 조부모 > 부모 > 자식 : LL

② 조부모 > 부모 < 자식 : LR

③ 조부모 < 부모 < 자식 : RR

④ 조부모 < 부모 > 자식 : RL

⑤ 회전 후 부모와 조부모의 색을 변경

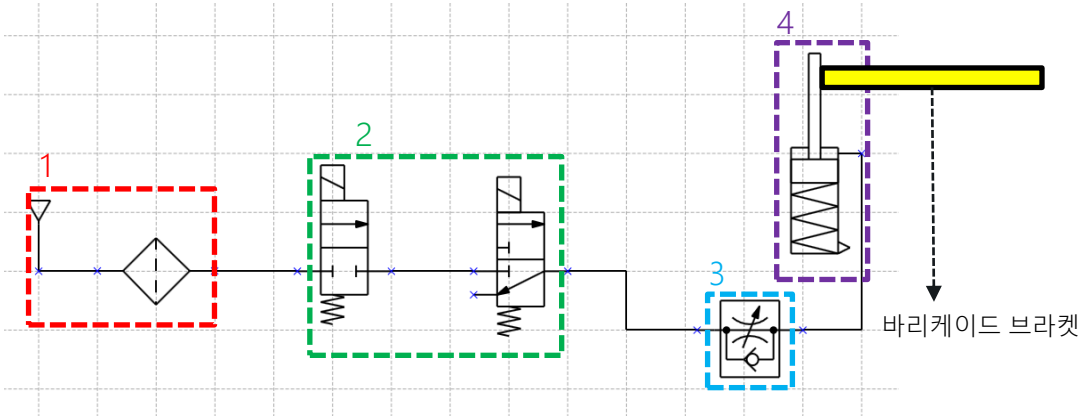
(2) 조부모 ≠ root

* 회전은 조부모가 root인 때와 동일(①~④)

⑤ 회전 후 자식노드의 색을 변경

유압 바리케이드 : 부피 압력 방식

- 유압 회로도



➢ 1 : 공압원으로 에어컴프레셔(?) 사용

✓ 고려사항 :
(1) 실린더 동작 필요 압력에 따라 공급압 계산
(2) 동작전압 형태 : AC or DC
(3) 정격전압
(4) 정격전류

➢ 2 : 방향제어 밸브(솔레노이드 밸브)

✓ 고려사항 :
(1) 2포트 밸브 : 압력 공급/차단
(2) 3포트 밸브 : 방향 제어
(3) 정격전압
(4) 정격전류
(5) 정격압력

➢ 3 : 속도 제어밸브(유량제어) – 기본 구성이 완료된 후 고려

✓ 고려사항 :
(1) 밸브 스위치 회전 각도당 제어량 파악
(2) 자동조작 방안 구성

➢ 4 : 단동 공압 실린더

✓ 고려사항 :
(1) 동작 시 필요 압력계산
(2) 로드 쪽과 바리케이드용 브라켓 연결방안 고려
(3) 젯봇 크기를 고려한 로드 총 길이 선정

* 바리케이드 브라켓 : 적당한 부품 없을 시 캐드로 간단하게 설계하여 3D 프린팅으로 구현