Red Black Tree

고재원

목차

- Tree
- Tree의 단점, 그래서 나온 AVL tree
- Rb tree 이거 고통을 주기 위한 용도 아냐? 쓰이긴 써?
- Rb tree 원리
- 실전

Tree?

그래프인데,

- 연결 되어 있으며
- 사이클이 없고
- 노드 - 링크 = 1인
- 방향이 없는

그래프이다.

Binary Search Tree?

1차원 배열을 생각해보자.

삽입 -> O(n)
검색-> O(n)

좀 빠르게 찾는 방법이 없을까?

왼쪽에는 나보다 작고 오른쪽에는 나보다 큰 애들

Binary Search Tree의 문제

이상적으로 데이터가 들어와서 Tree가 균형을 이룰 것 처럼 보인다.

그러나 1 2 3 4 5 순으로 들어온다면?
더 이상 tree가 아니라 그래프가 될 것이다.

즉 O(log n)이 항상 보장되지 않는 문제가 있다.

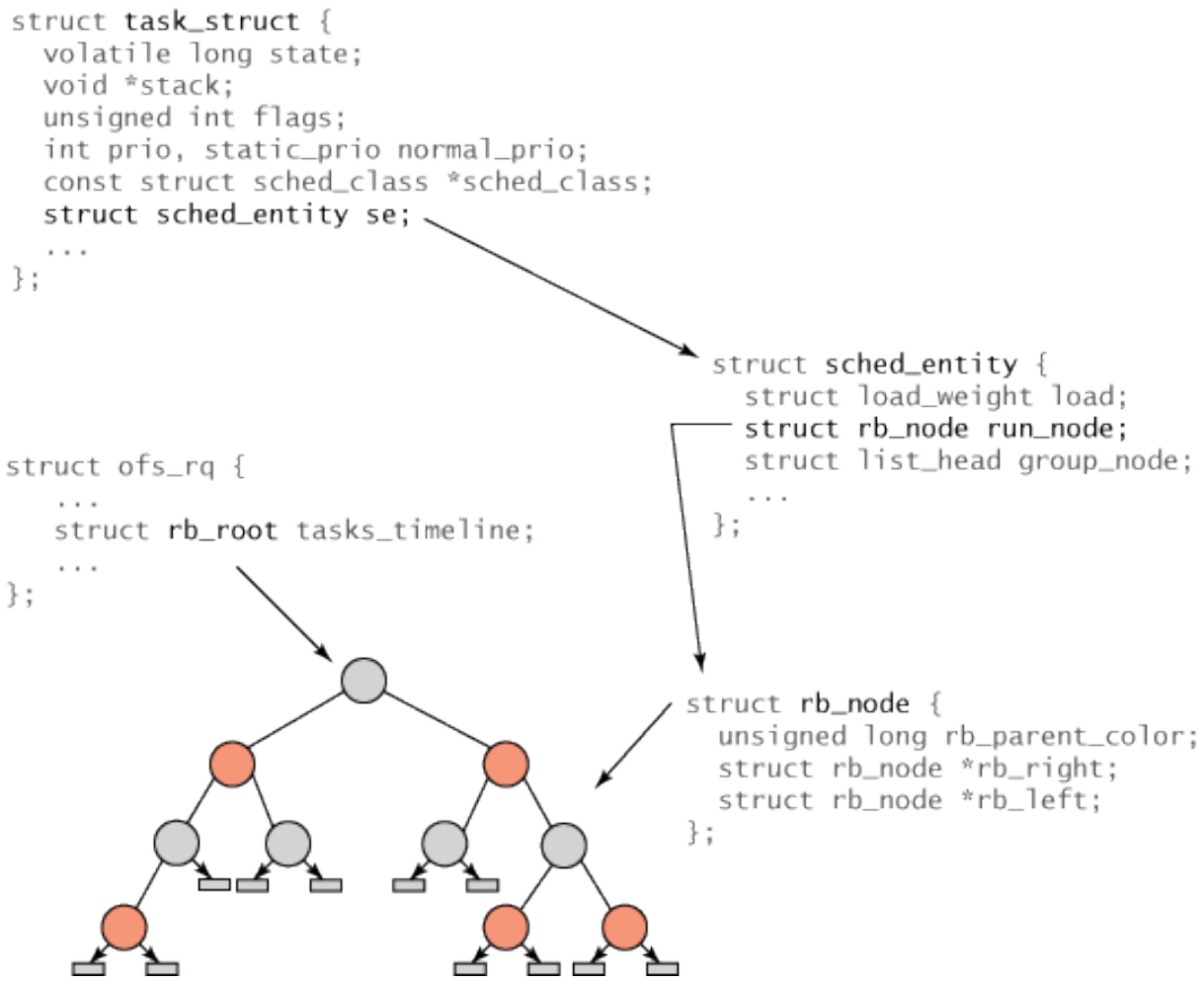
그래서..

- 검색을 항상 O(logN)으로 보장하는 방법이 없을까?

Avl tree

- BST의 단점을 보완하기 위해 만들어진 트리
- 스스로 균형을 잡는 기묘한 트리다.

아니 잠깐, 이거 진짜 쓰이는 거야?



In Linux kernel…

CFQ(completely fair queuing) :

프로세스들의 io작업을 균등하게 실행하기 위한 큐

아니 잠깐, 이거 진짜 쓰이는 거야?

```
/**
 *      struct sk_buff - socket buffer
 *      @next: Next buffer in list
 *      @prev: Previous buffer in list
 *      @tstamp: Time we arrived/left
 *      @skb_mstamp_ns: (aka @tstamp) earliest departure time; start point
 *              for retransmit timer
 *      @rbnode: RB tree node, alternative to next/prev for netem/tcp
```

In Linux kernel…

Packet Driver

아니 잠깐, 이거 진짜 쓰이는 거야?

이거 말고도 많은데 넣지는 않는다.

어쨌든, rb tree는 생각보다 많이 쓰이는 것을 알 수 있다.

들어가기에 앞서…

구현 -> 생각하지 말 것.

어떻게 동작하는지를 이해해야 구현을 생각 할 수 있다.
이해하지 못하고 구현을 하려고 하면 너무너무너무 어렵다

그냥 보면서 아하 그렇군! 끄덕끄덕 하면 된다.

Red Black tree, 알아보자.

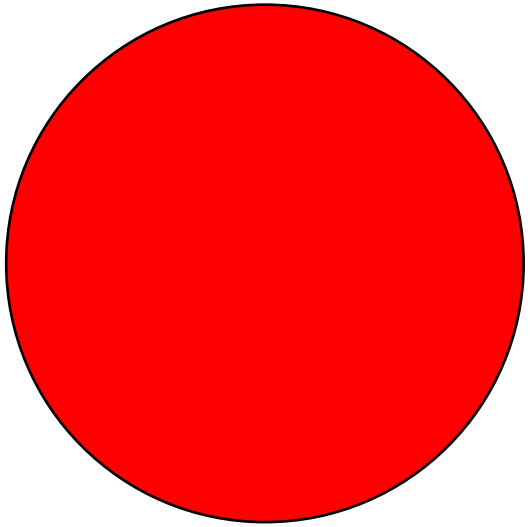
- NIL node?
  - 존재하지 않음을 의미하는 노드
  - 자녀가 없을 때 자녀를 이 노드로 설정
  - RB트리에서 leaf노드는 무조건 nil 노드

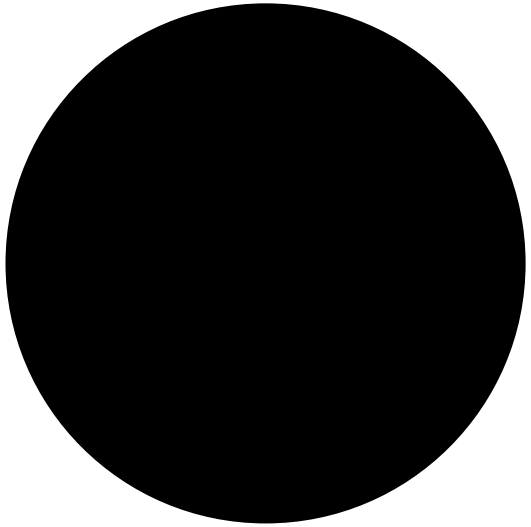
Red Black tree의 규칙

1. 모든 노드는 red or black
2. 루트 노드는 black
3. 모든 NIL, 즉 leaf노드는 black
4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.
5. 임의의 노드에서 자손 NIL노드까지 가는 경로들의 black수는 같다.(자기 자신 카운트 제외)
   즉 어떤 자손 NIL까지 상관없이 경로의 black 수는 같다.

Red Black tree의 규칙 1

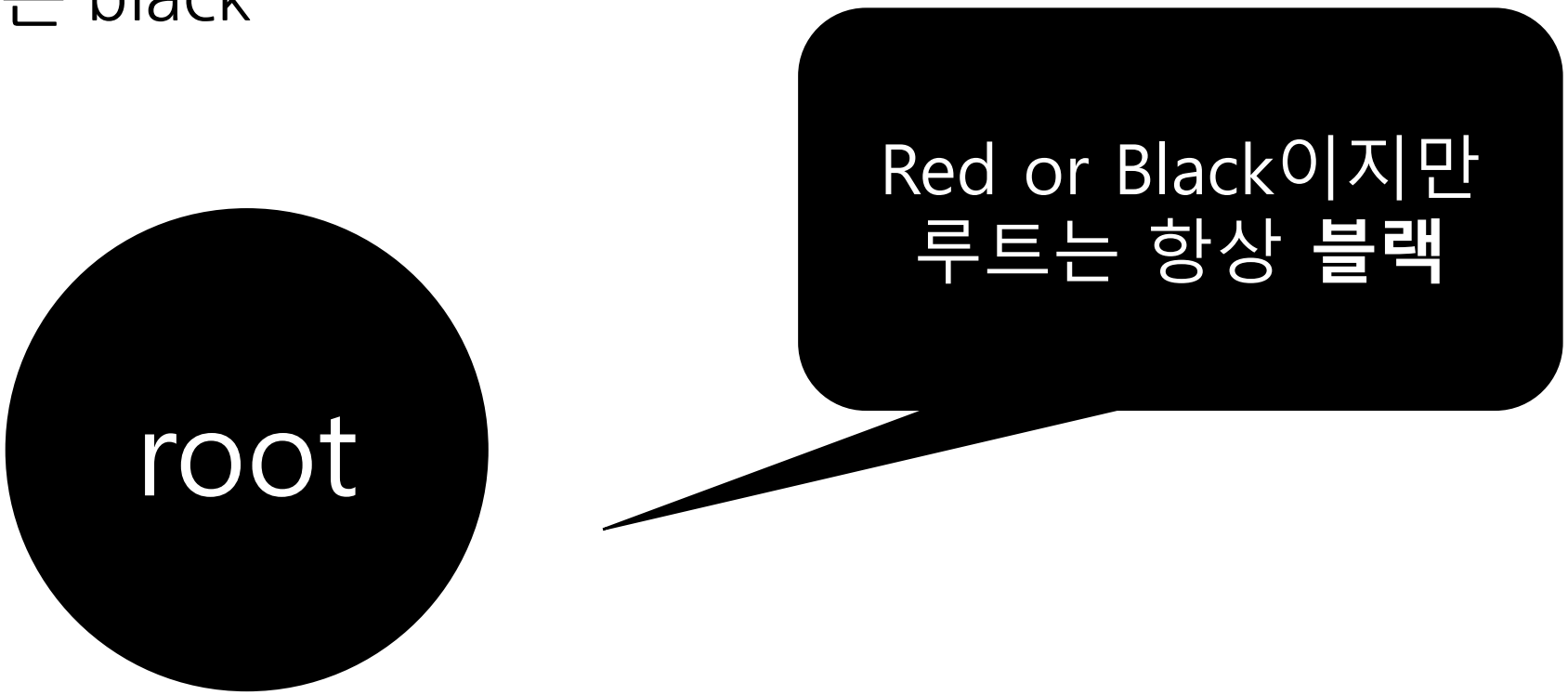
모든 노드는 red or black





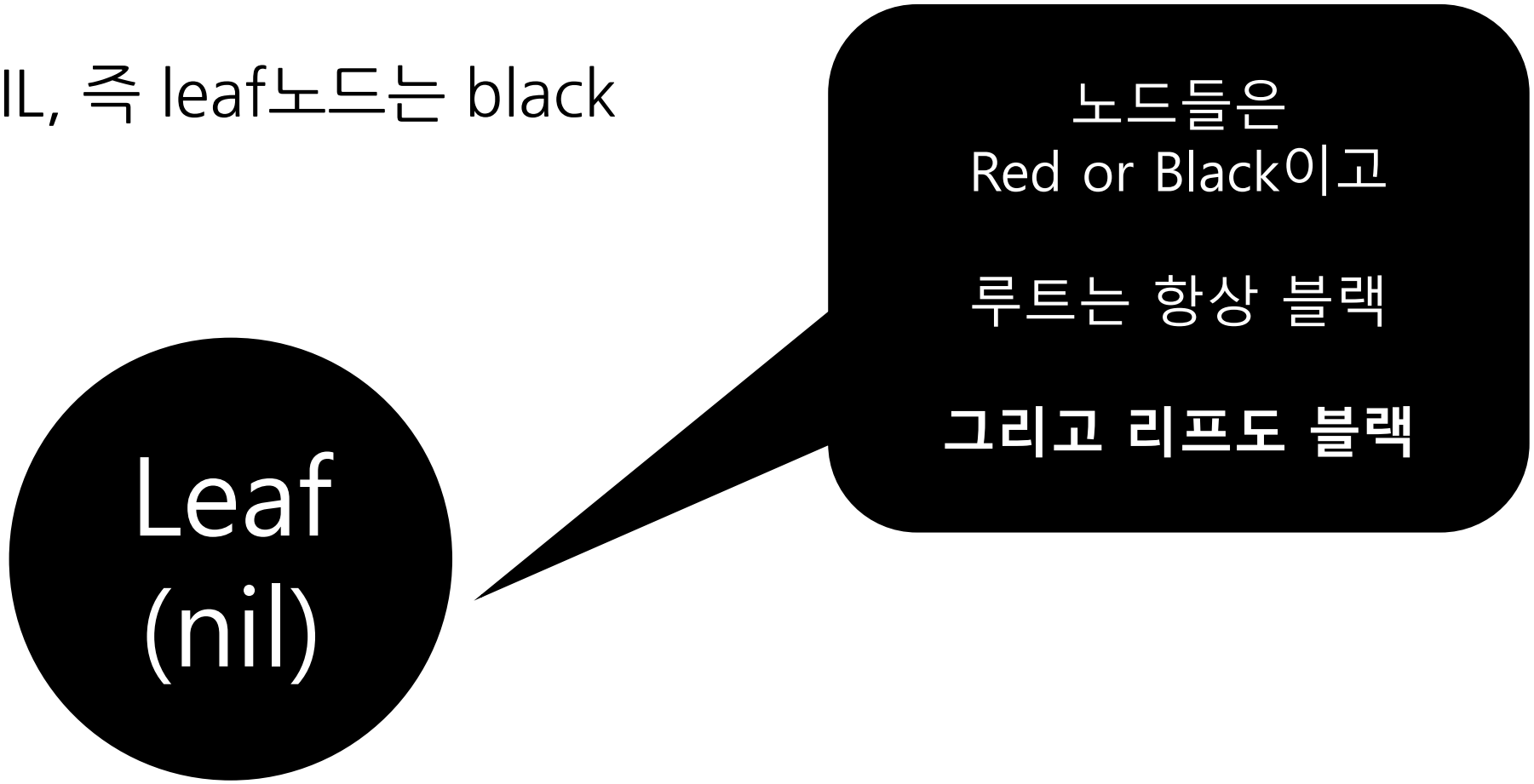
Red Black tree의 규칙 2

루트 노드는 black



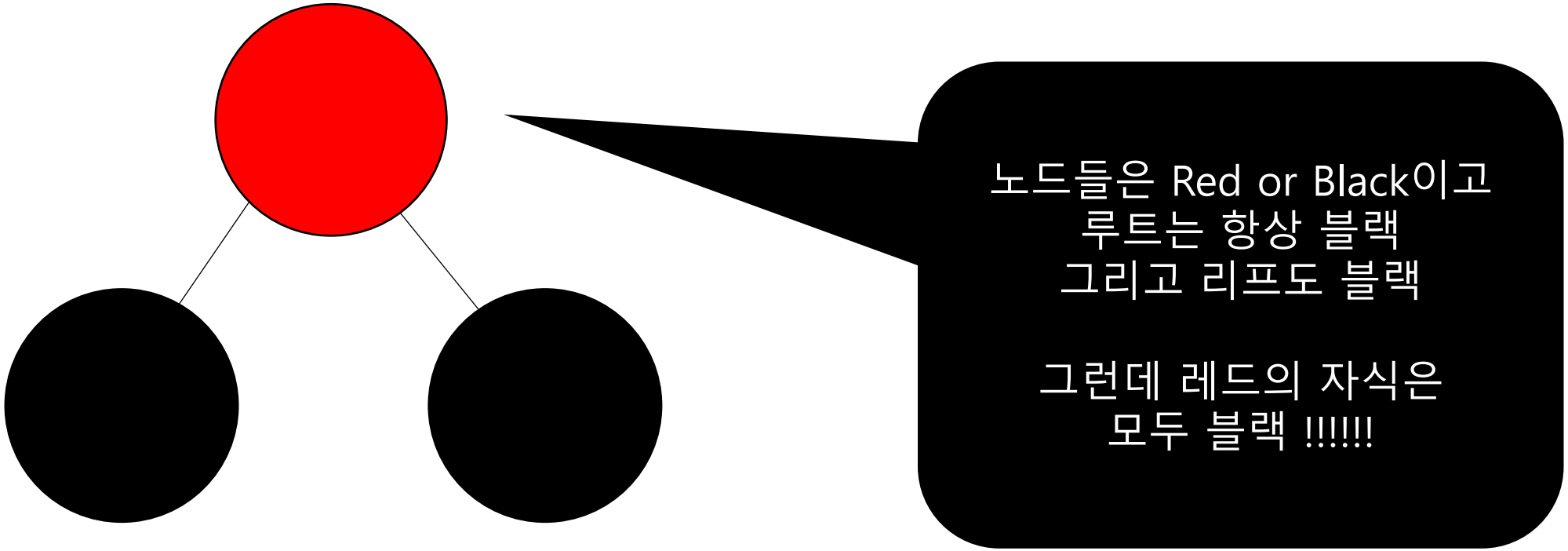
Red Black tree의 규칙 3

모든 NIL, 즉 leaf노드는 black



Red Black tree의 규칙 4

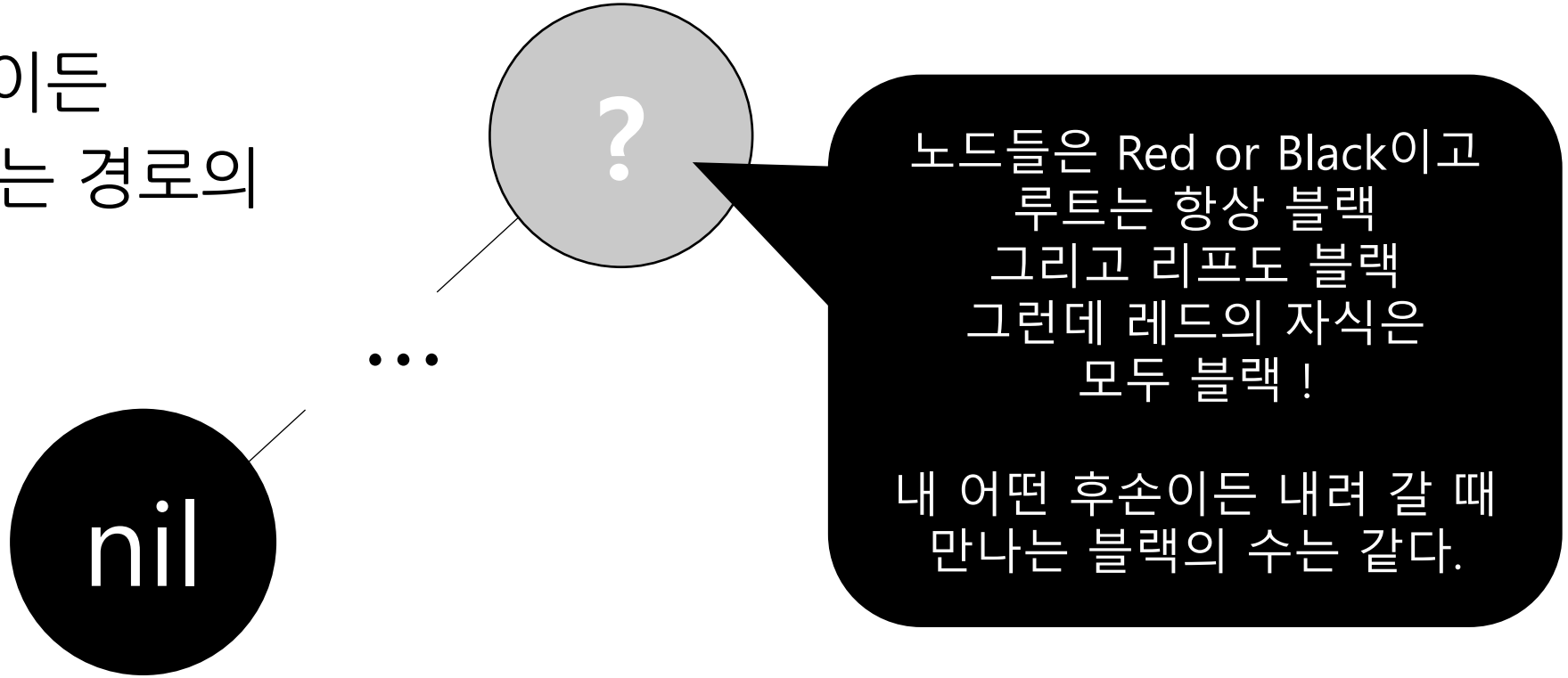
red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



Red Black tree의 규칙 5

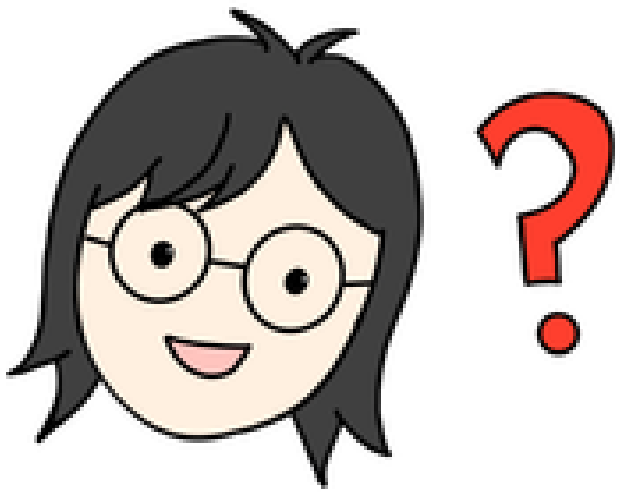
임의의 노드 -> **자손 리프 노드(nil)**까지 가는 경로에서 만나는 black 노드의 수는 같다. (자기 자신 카운트 제외)

즉 어떤 자손 NIL이든
해당 노드까지 가는 경로의
black 수는 같다.



Red Black tree의 규칙

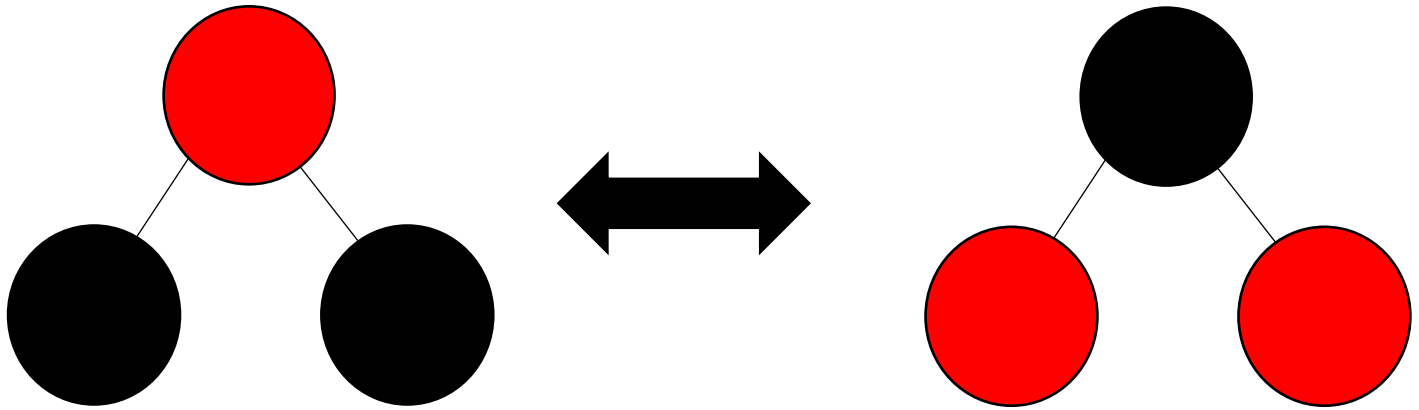
한 번 듣고 이해하기 어렵다. 다시 1번부터 반복해서 들어보자.



5번 특징 생각

5. 어떤 자손 NIL이든 해당 노드까지 가는 경로의 black 수는 같다.
 (자기 자신 카운트 제외)

그렇다면 만약, 어떤 노드의 두 자식이 같은 색을 가질 때,
부모와 두 자식의 색을 바꿔도 5번을 만족할까?



그렇다면…

그래그래 특징은 알았어.

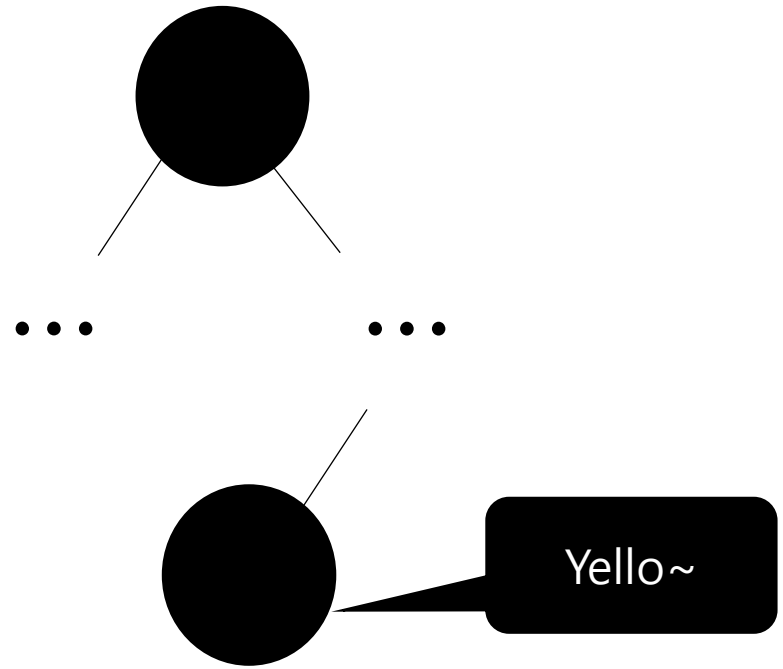
그런데, 삽입이 되었을 때 어떻게 해야 계속 위 조건들을 만족 할 수 있는거지?

삽입/삭제시 rbtree의 몇 번 특징을 위배하게 될까?

새로운 노드 삽입 시

5. 어떤 자손 NIL이든 해당 노드까지 가는 경로의 black 수는 같다.
 (자기 자신 카운트 제외)

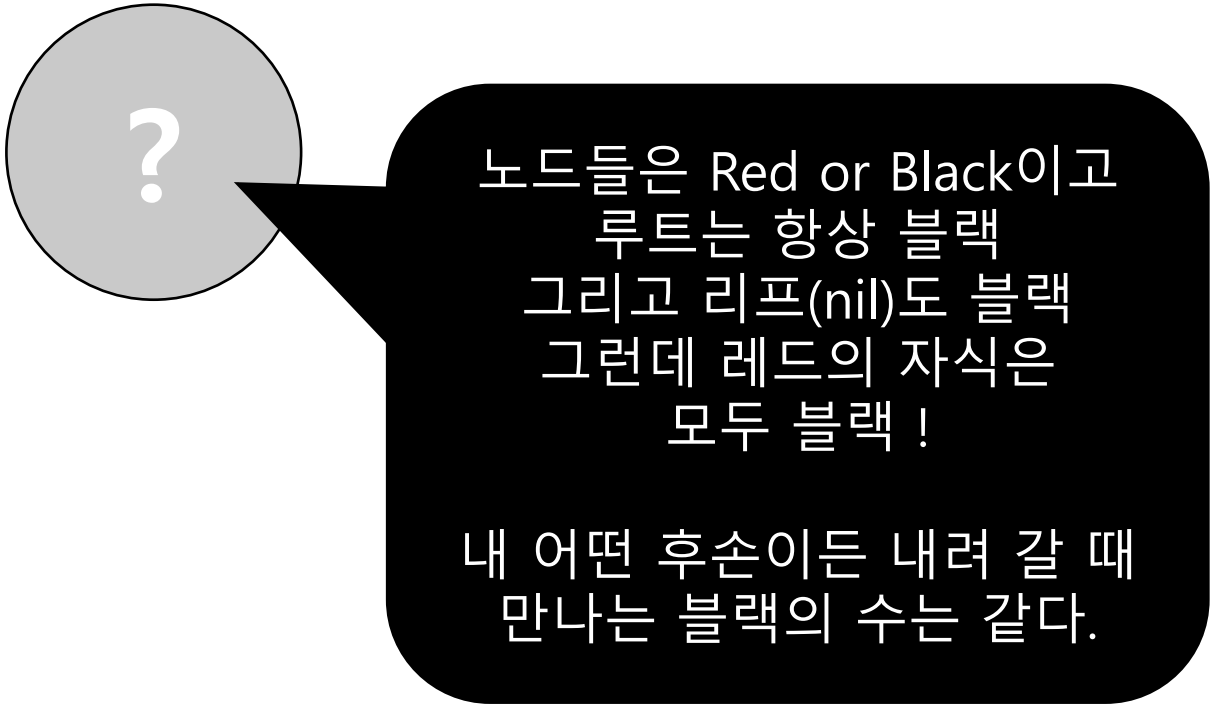
>> Nil 위에 black이 들어온다면?



그렇다면 삽입 시 어떤 조건을 넣어야 해?

- 무조건 삽입되는 노드는 Red로 하면 되는거 아닌가?

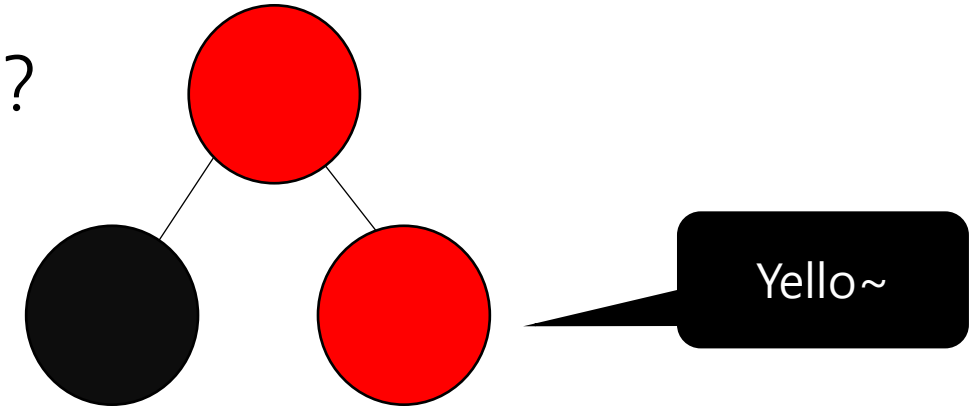
- 그렇다면 다른 문제는 없을까?



새로운 노드 삽입 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.

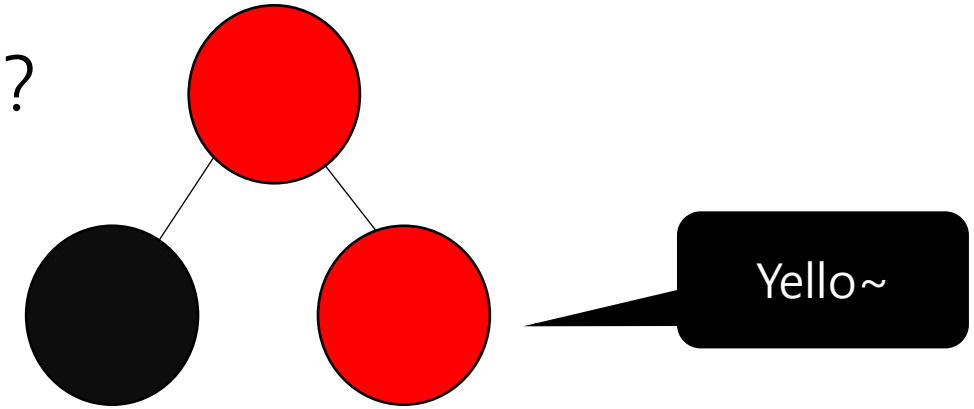
>> child에 red가 들어온다면?



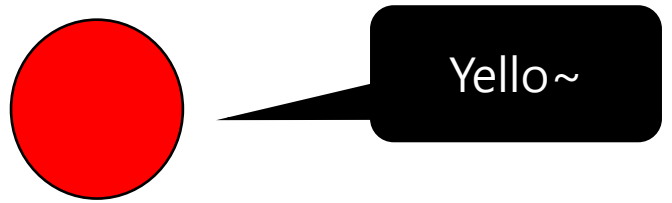
새로운 노드 삽입 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.

>> child에 red가 들어온다면?



>> 빈 트리에 처음 노드가 들어와서 루트가 red라면?

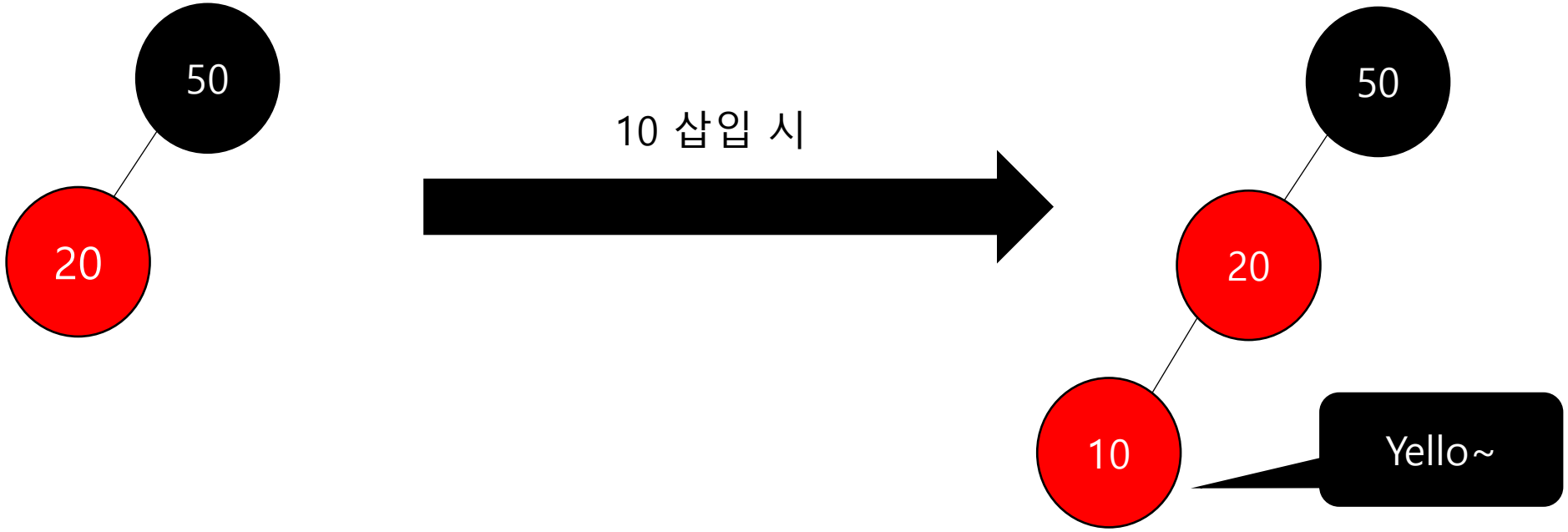


새로운 노드 삽입 시

결국 규칙을 따라가며 트리의 구성을 바꿔야 한다 ㅠㅠ
천천히 알아보자.

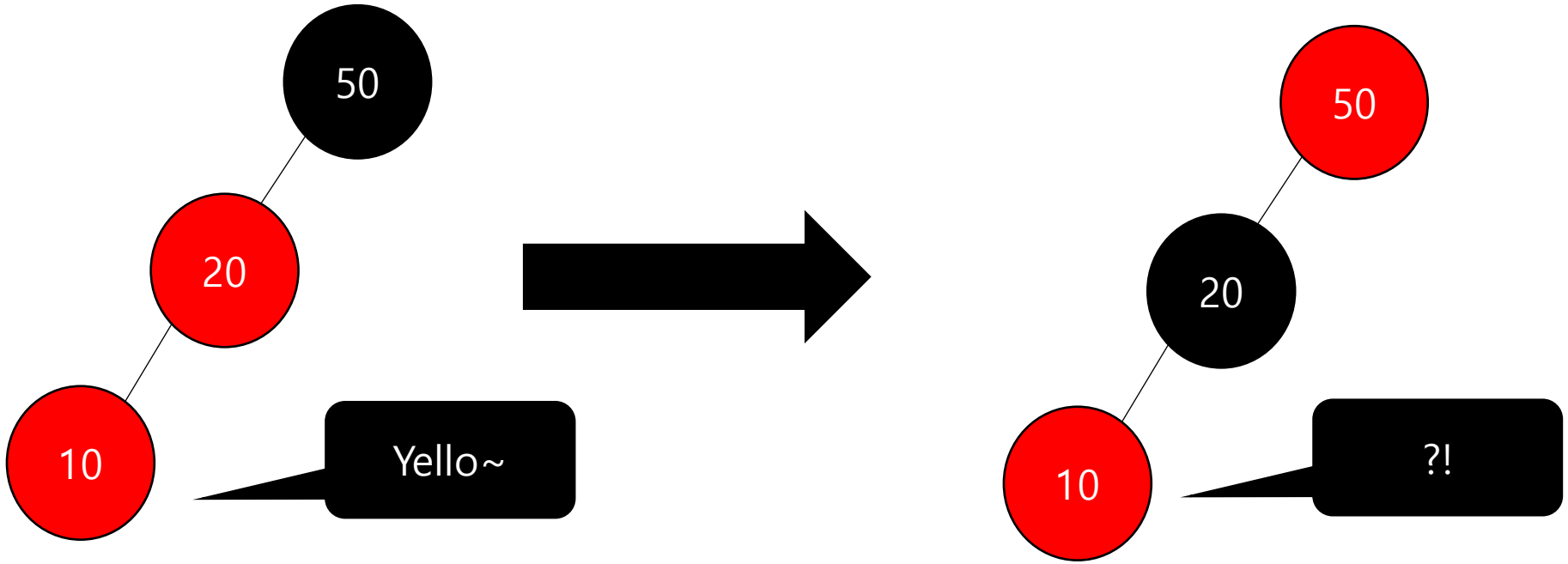
노드 삽입 시 4번을 위배한다면…

4. 레드의 자식은 모두 블랙 위배 시



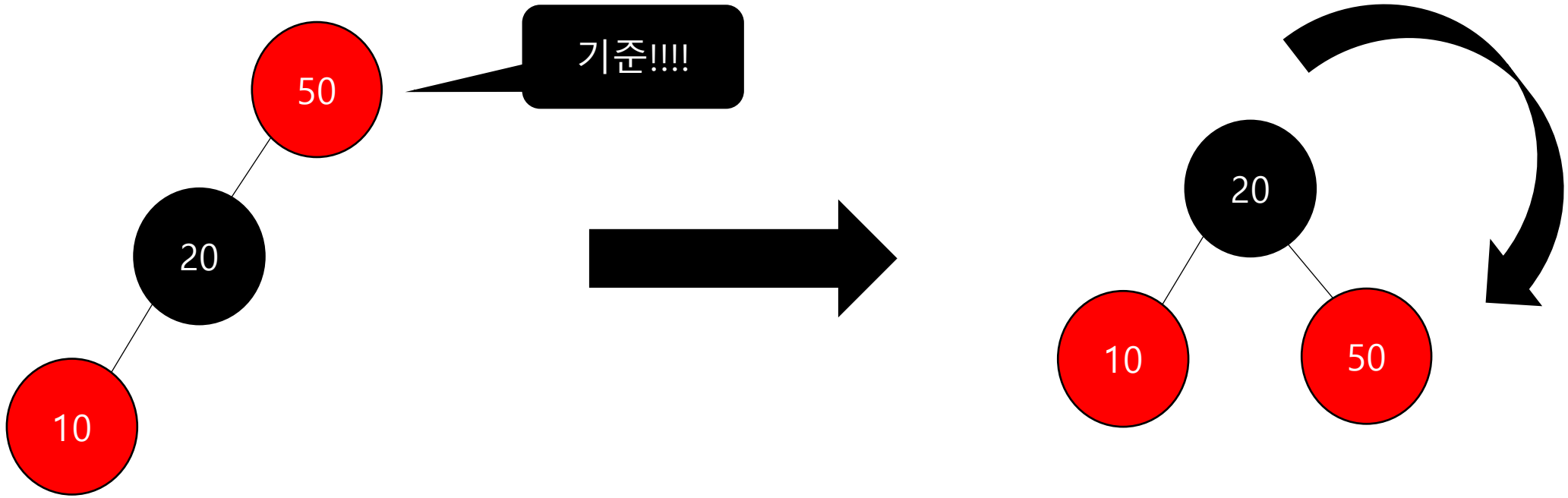
스킬 : 회전

1. 문제가 되는 노드의 부모와 할아버지의 색을 바꾼다.



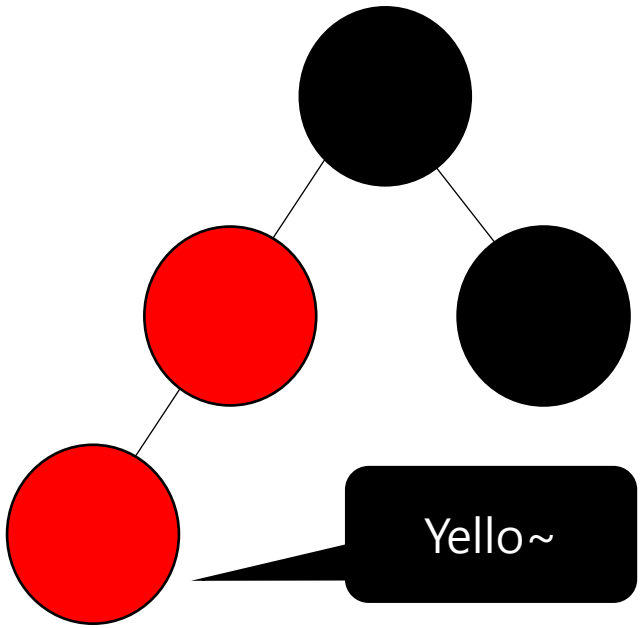
스킬 : 회전

2. 할아버지를 기준으로 왼쪽/오른쪽으로 회전한다.



요약 : 삽입 후 4번 위배 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



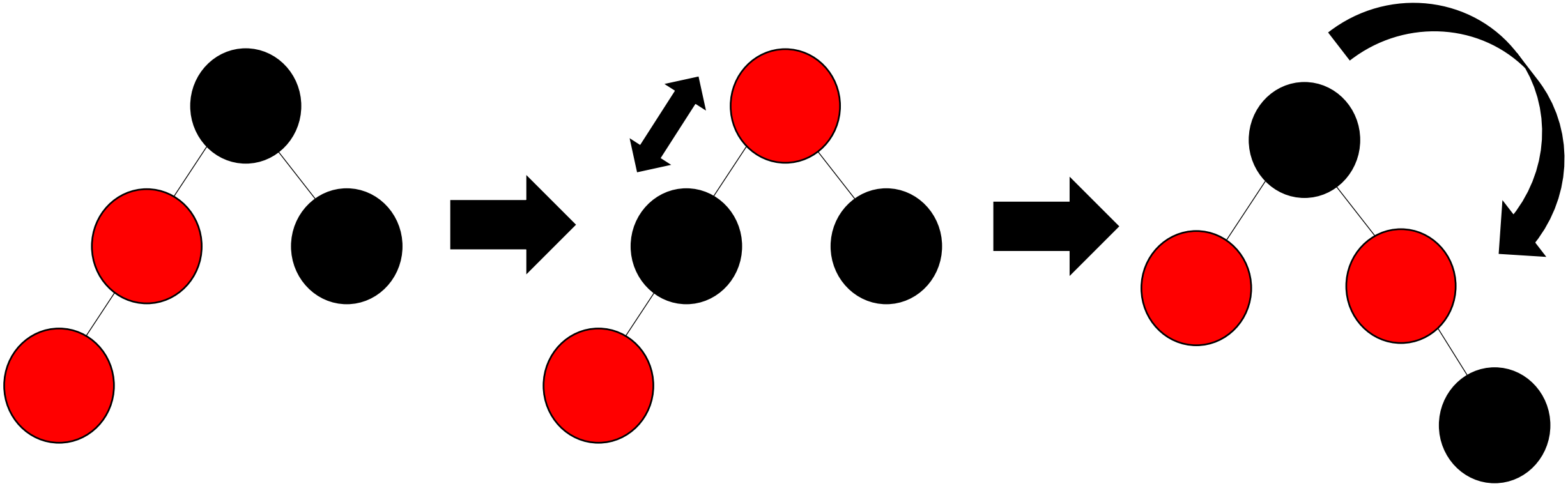
삽입된 red노드의 부모도 red일 때

- 부모의 왼쪽/오른쪽 자녀이고
- (부모가)할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

부모와 할아버지의 색을 바꾼 후
할아버지 기준으로 오른쪽/왼쪽으로 회전한다.

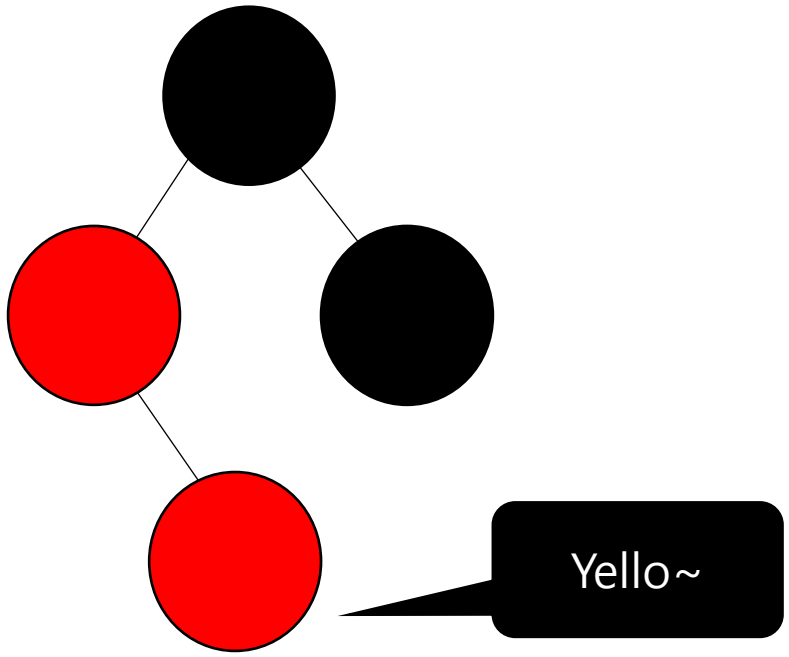
요약 : 삽입 후 4번 위배 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



그럼 이런 경우는?

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



삽입된 red노드의 부모도 red일 때

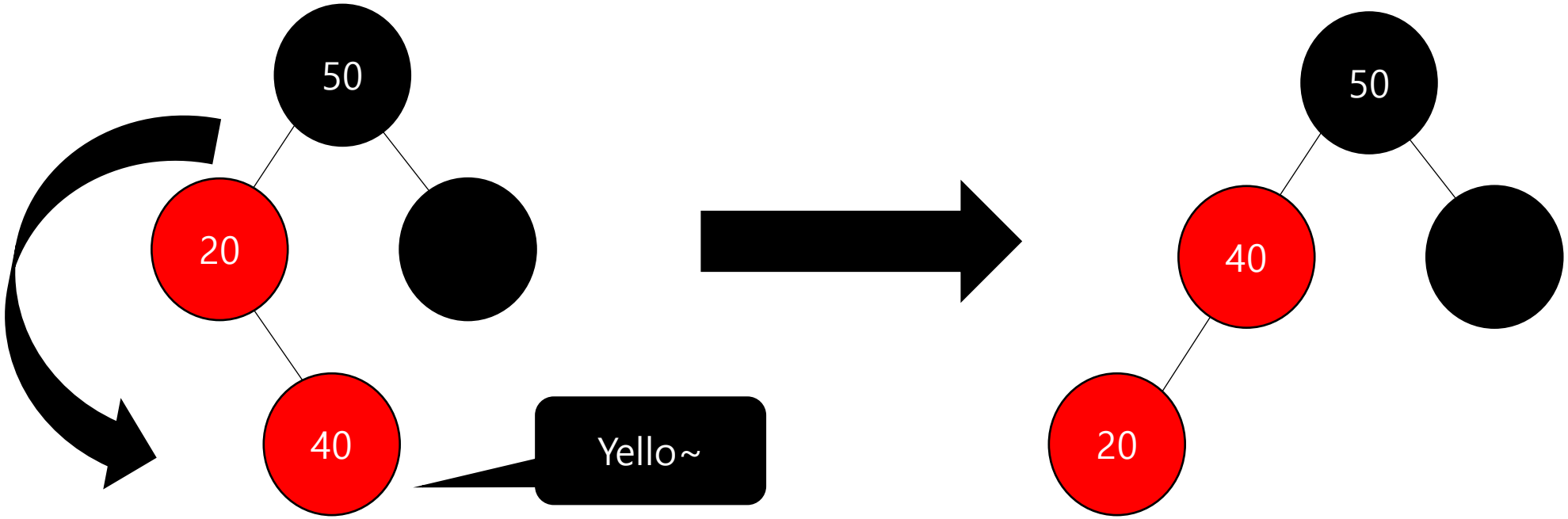
- 부모의 왼쪽/오른쪽 자녀이고
- (부모가)할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

과 다르게 이번에는

할아버지-부모 / 부모-나의 방향이 다르다면?!

스킬 : 회전2

꺾인 부분을 펴 주면, 즉 부모를 회전하면? -> 원래 문제와 똑같아진다.



스킬 : 회전 결론

삽입된 red노드의 부모도 red일 때

- 부모의 왼쪽/오른쪽 자녀이고
- (부모가)할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

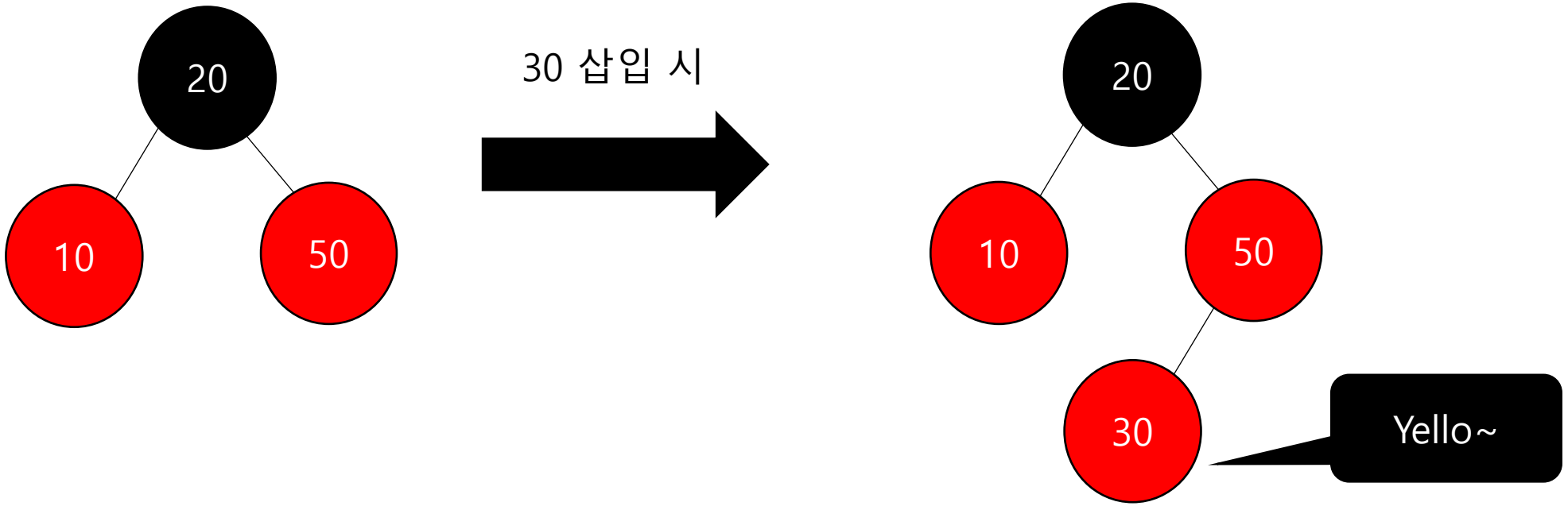
-> 할아버지 부모 색 바꾸고 회전

- 부모의 오른쪽/왼쪽 자녀이고
- (부모가)할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

-> 부모 회전 후 위의 방법으로 회전

다시 삽입 시 생기는 문제

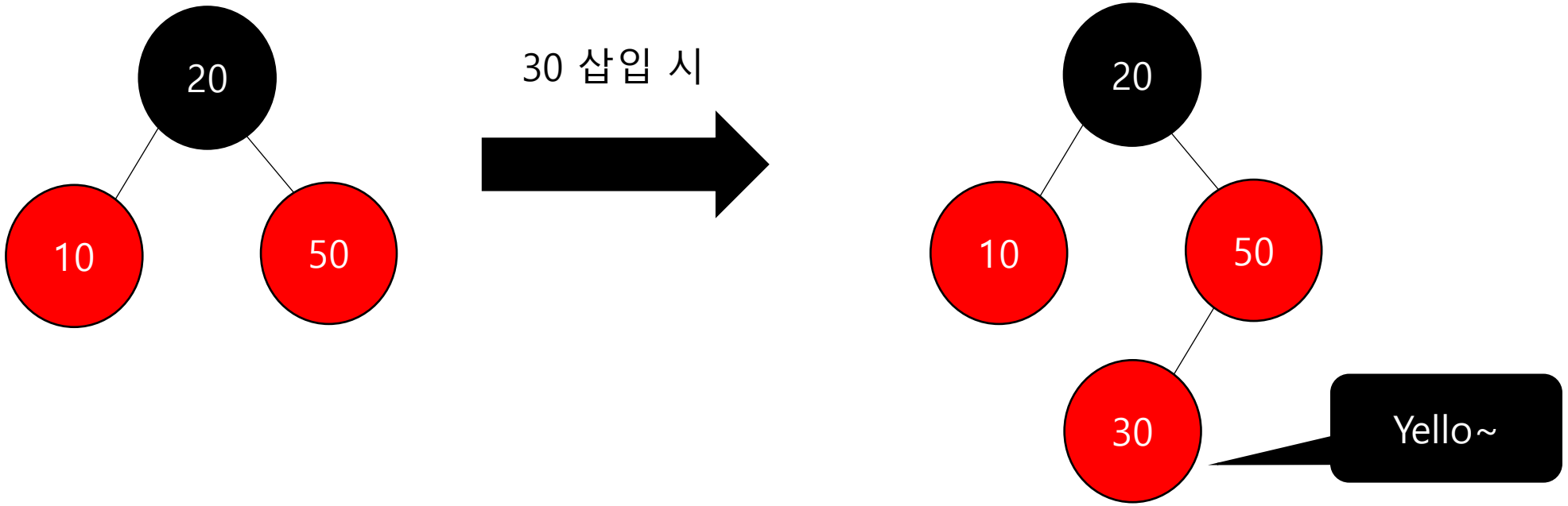
삽입된 red노드의 부모도 red일 때



아까 전 처럼 회전 해버려?

다시 삽입 시 생기는 문제

삽입된 red노드의 부모도 red일 때



이런! 이번에는 부모의 형제가 RED다!!!!!

다시 삽입 시 생기는 문제

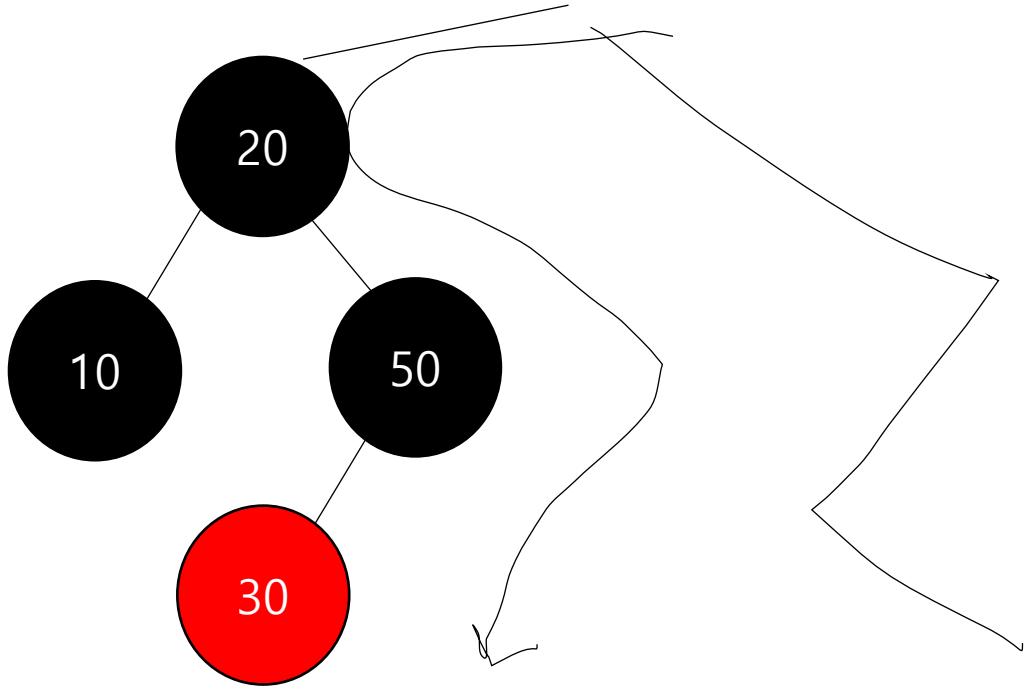
아니 그러면 지금 나도 red고 부모도 red고 부모의 형제도 red?!!

부모와 부모의 형제를 모두 black으로 바꾸면?

다시 삽입 시 생기는 문제

그런데… rbtree의 나머지 조건도 만족하는지 생각 해 봤어?

5. 어떤 자손 NIL이든 해당 노드까지 가는 경로의 black 수는 같다.
(자기 자신 카운트 제외)

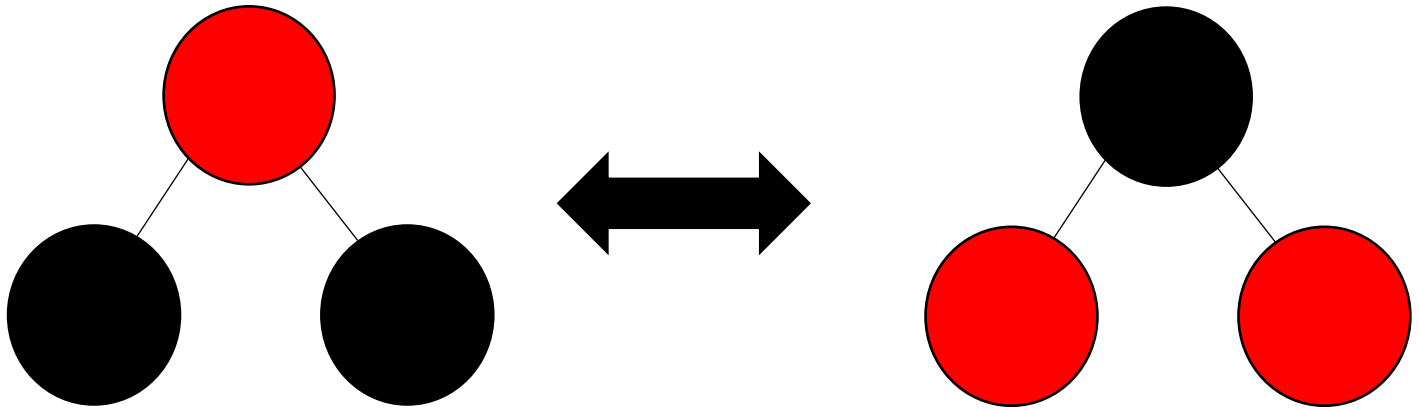


즉 좋은 방법은 아니다.

5번 특징 생각

5. 어떤 자손 NIL이든 해당 노드까지 가는 경로의 black 수는 같다.
 (자기 자신 카운트 제외)

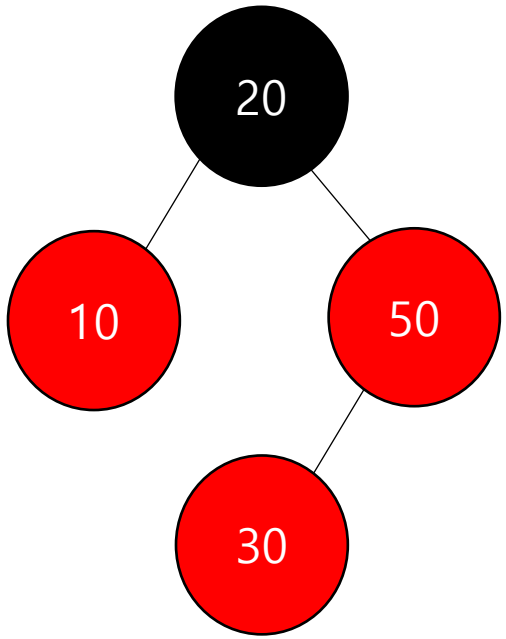
그렇다면 만약, 어떤 노드의 두 자식이 같은 색을 가질 때,
부모와 두 자식의 색을 바꿔도 5번을 만족할까?

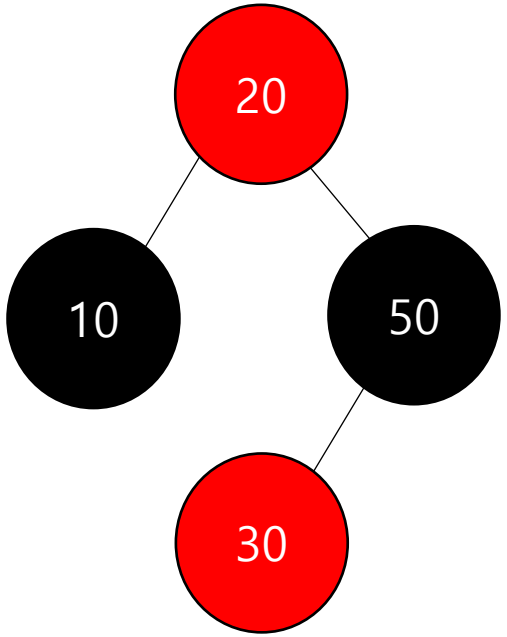


다시 삽입 시 생기는 문제

이걸 이용하면?

5. 어떤 자손 NIL이든 해당 노드까지 가는 경로의 black 수는 같다.
(자기 자신 카운트 제외)



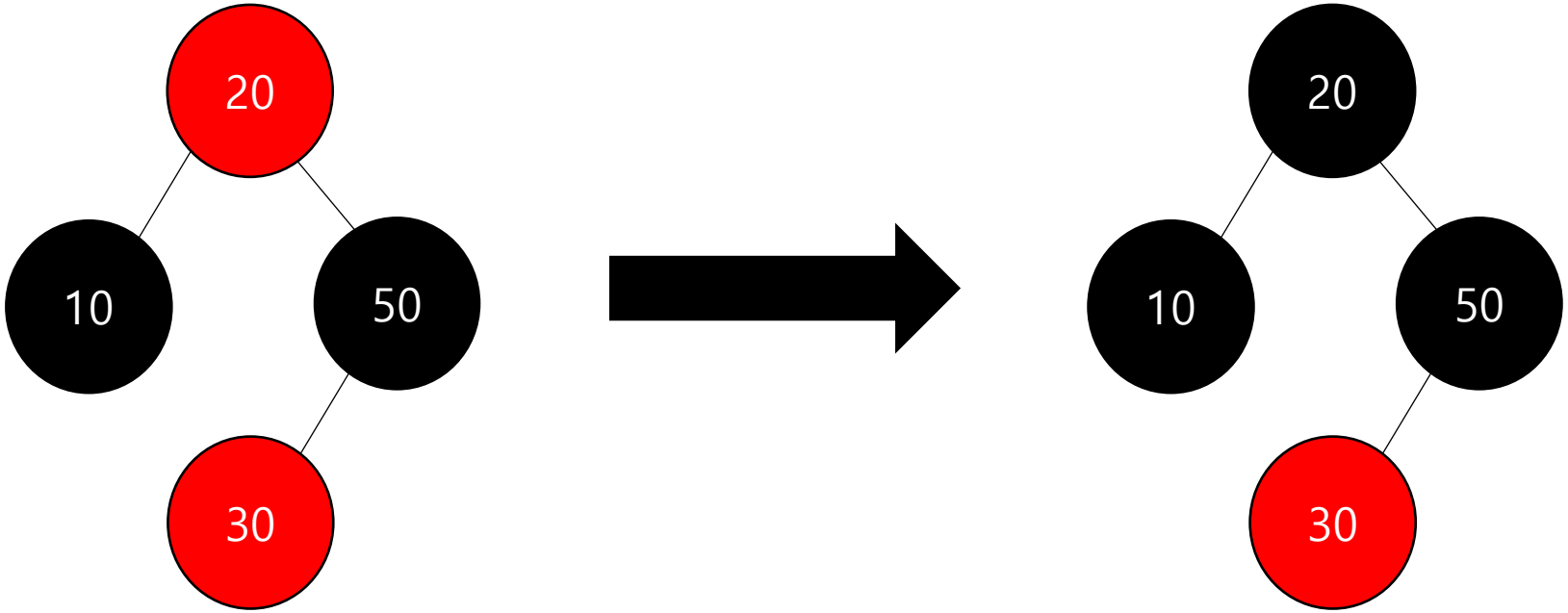


헉! 그런데 여기의 경우는 루트가 red야!!!!

2. 루트 노드는 black

다시 삽입 시 생기는 문제

이때는 그냥 red를 black으로 바꾼다. (root니깐 가능)

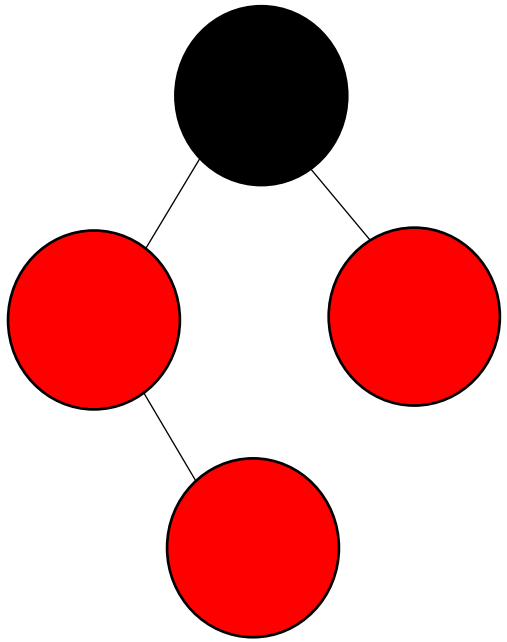


헉! 그런데 여기의 경우는 루트가 red야!!!!

2. 루트 노드는 black

요약 : 삽입 후 4번 위배 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



삽입된 red노드의

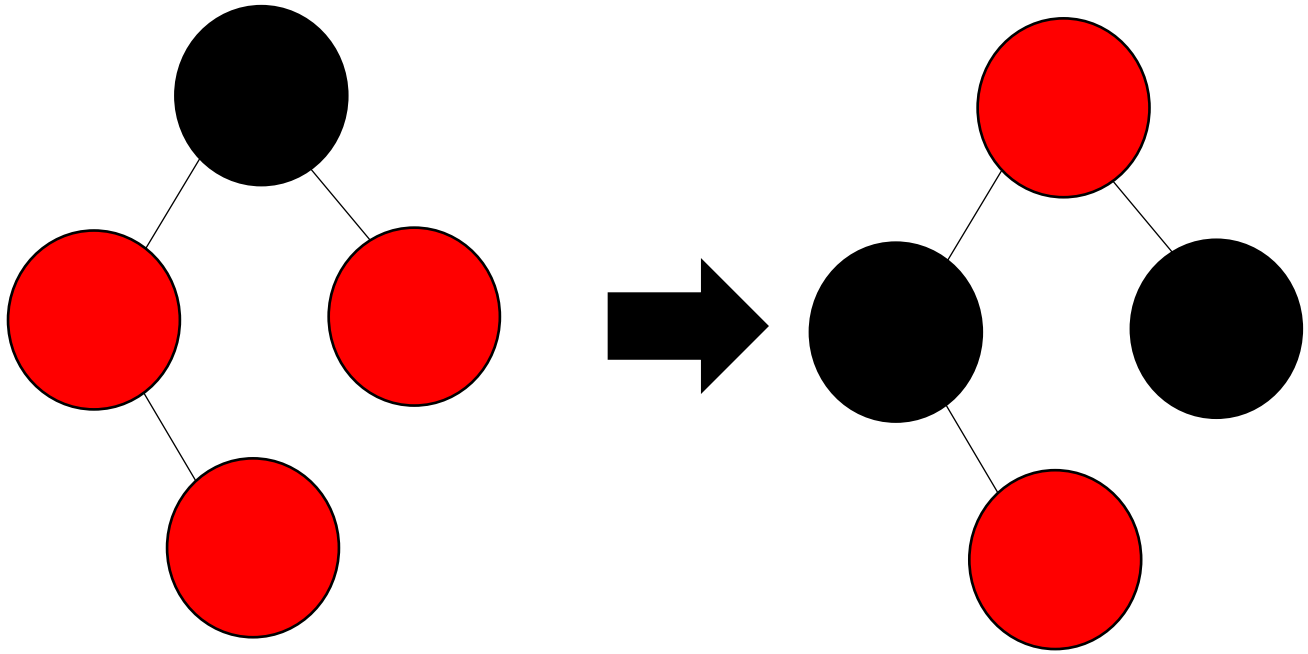
- 부모도 red고
- 부모의 형제도 red면

부모와 부모의 형제를 black으로 바꾸고
할아버지를 red로 바꾼 후

할아버지에서 다시 확인을 시작한다.

요약 : 삽입 후 4번 위배 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



삽입된 red노드의

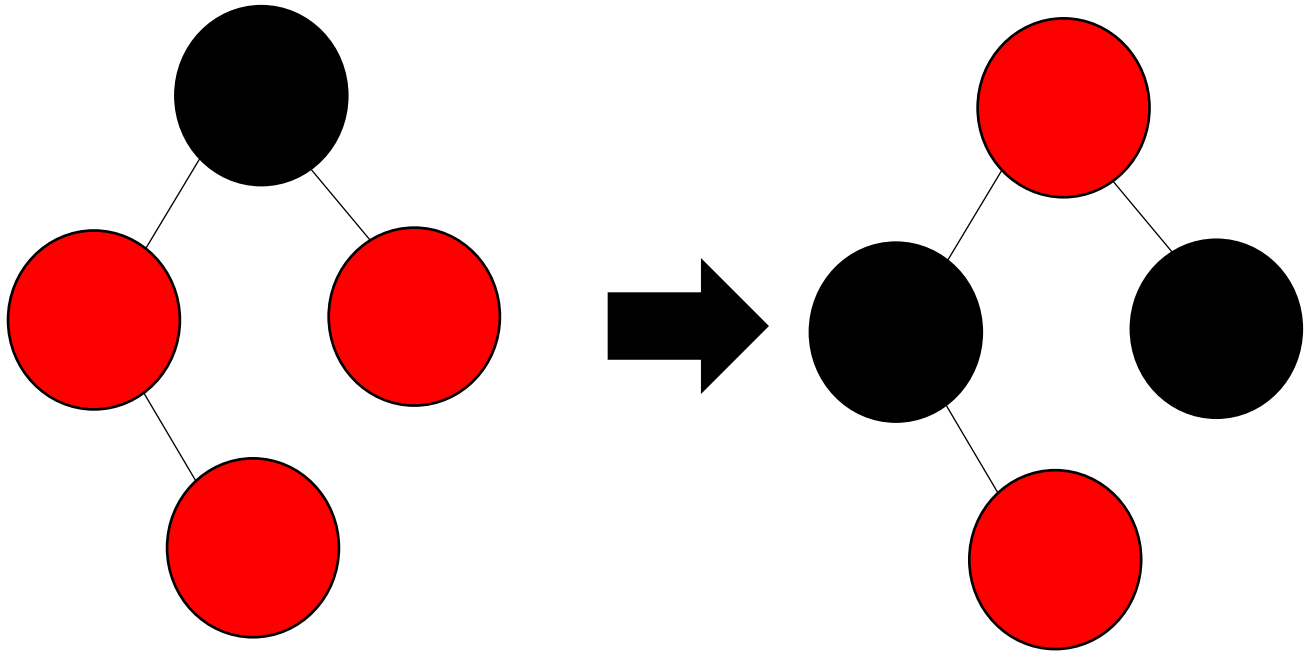
- 부모도 red고
- 부모의 형제도 red면

부모와 부모의 형제를 black으로 바꾸고
할아버지를 red로 바꾼 후

할아버지에서 다시 확인을 시작한다.

요약 : 삽입 후 4번 위배 시

4. red의 자식들은 black이다. 즉 red는 연속적으로 존재 할 수 없다.



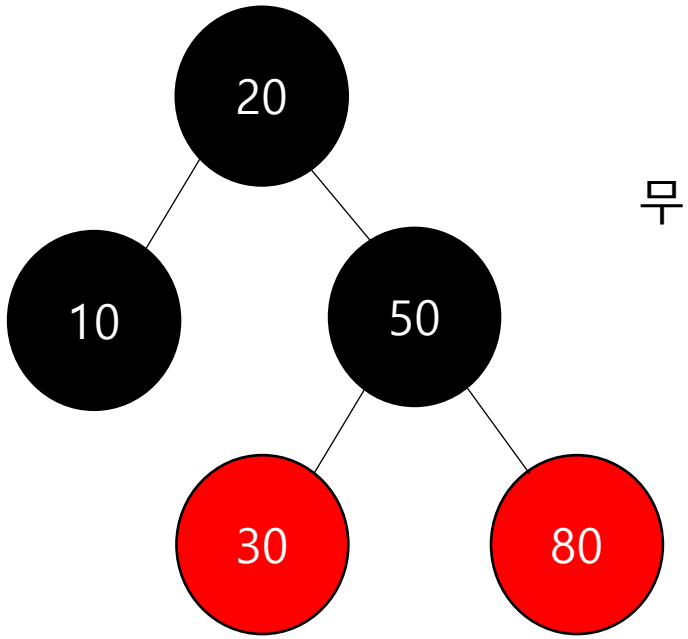
삽입된 red노드의

- 부모도 red고
- 부모의 형제도 red면

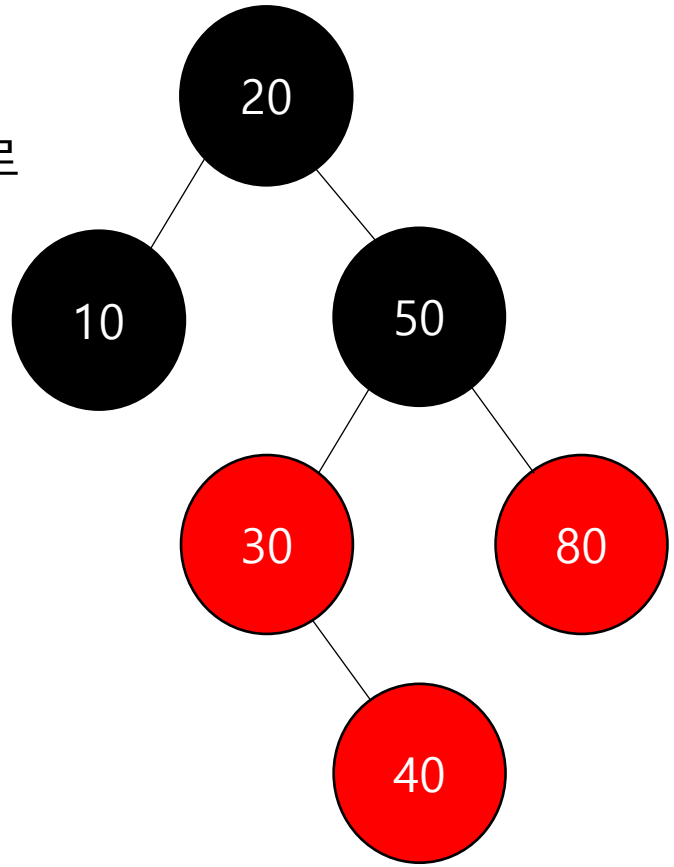
부모와 부모의 형제를 black으로 바꾸고
할아버지를 red로 바꾼 후

할아버지에서 다시 확인을 시작한다.

실전

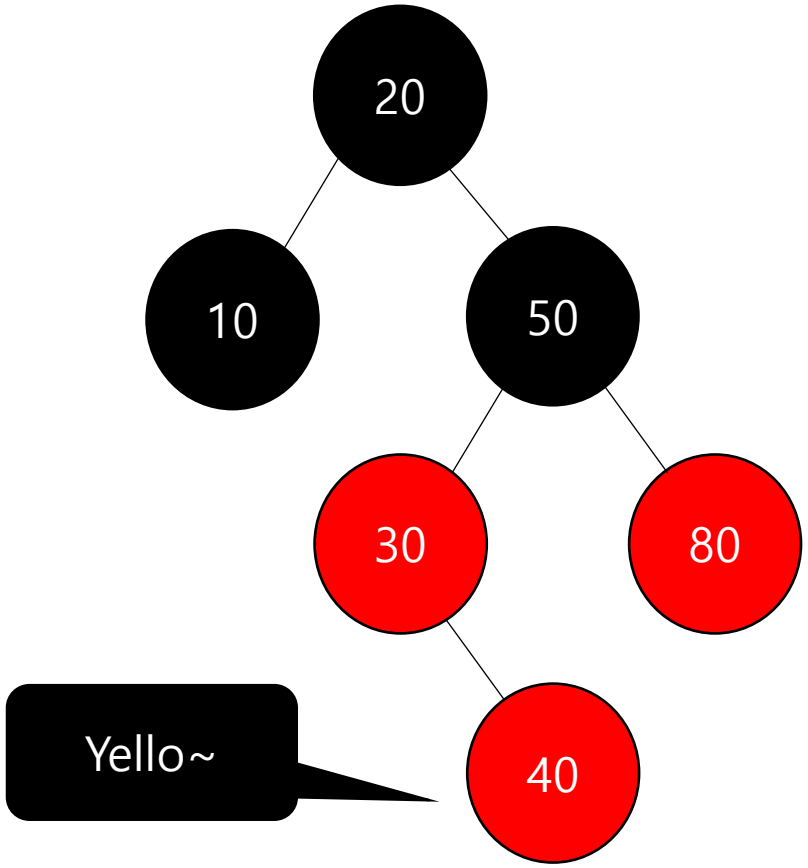


무조건 삽입되는 노드는 Red로



여기에 40을 넣어보자!!!!

실전



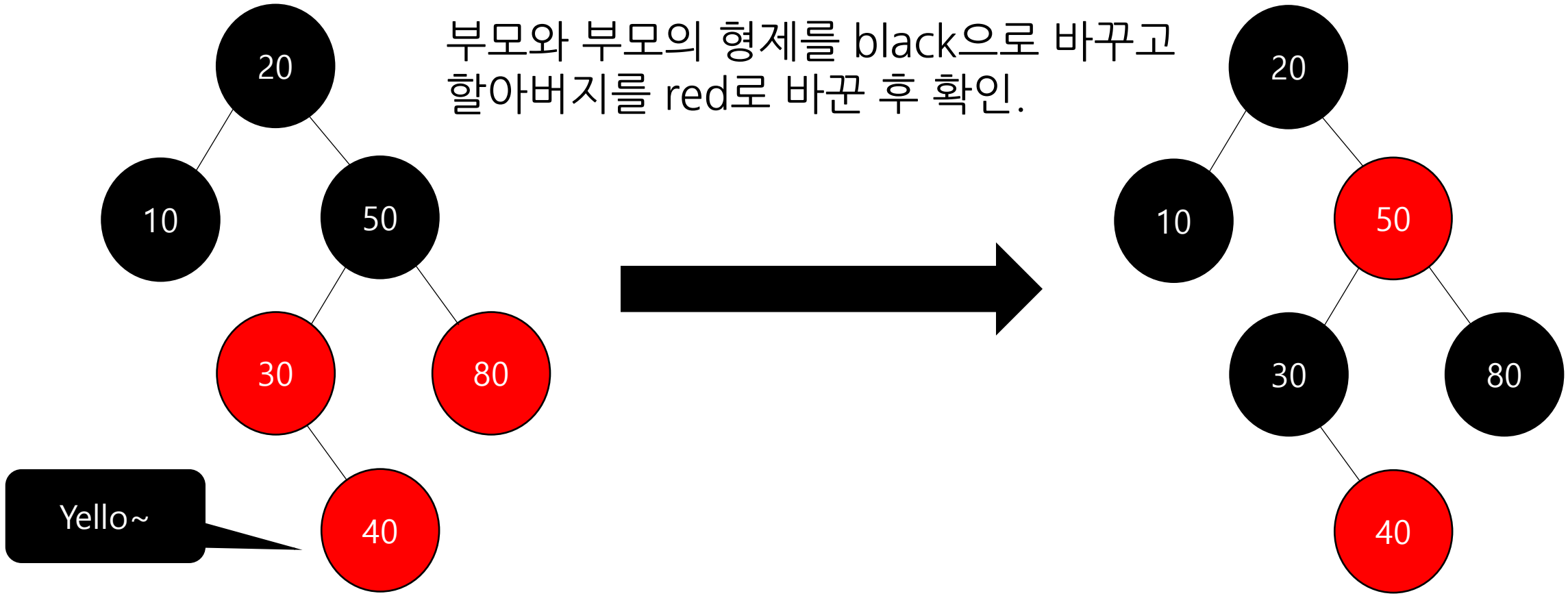
4. red의 자식들은 black이다.
즉 red는 연속적으로 존재 할 수 없다. **를 위배 할 때**

삽입된 red노드의
- 부모도 red고
- 부모의 형제도 red면

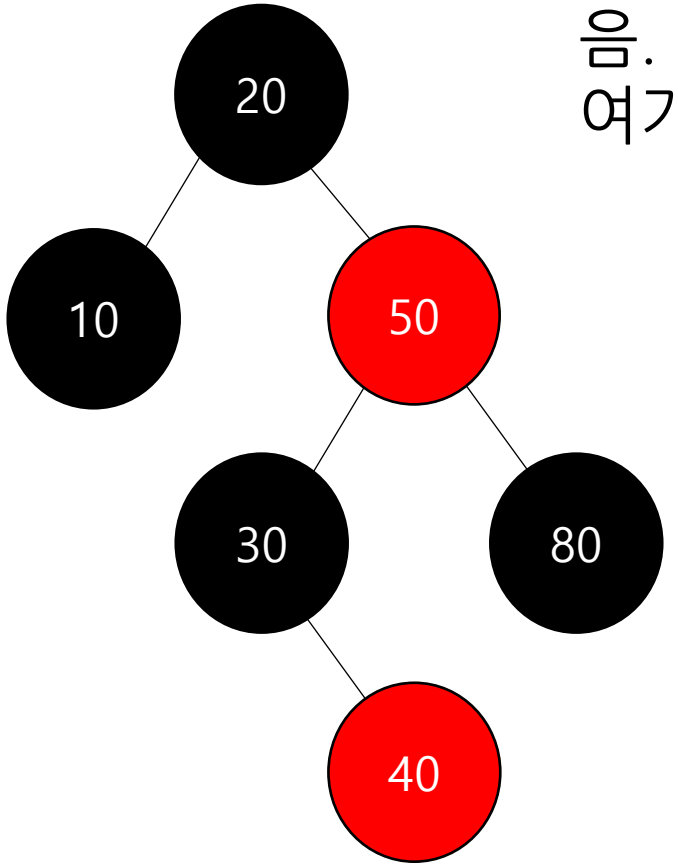
부모와 부모의 형제를 black으로 바꾸고
할아버지를 red로 바꾼 후

할아버지에서 다시 확인을 시작한다.

실전


20
10
50
30
80
40
Yello~
부모와 부모의 형제를 black으로 바꾸고
할아버지를 red로 바꾼 후 확인.
20
10
50
30
80
40

실전



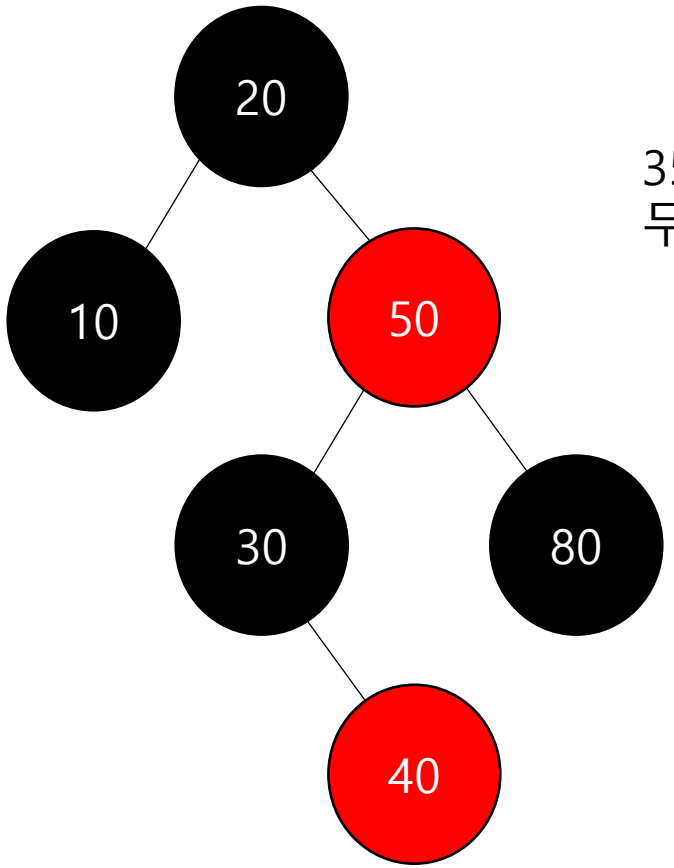
음. 잘 만족 하는 군.
여기에 35를 넣어 볼까?

?

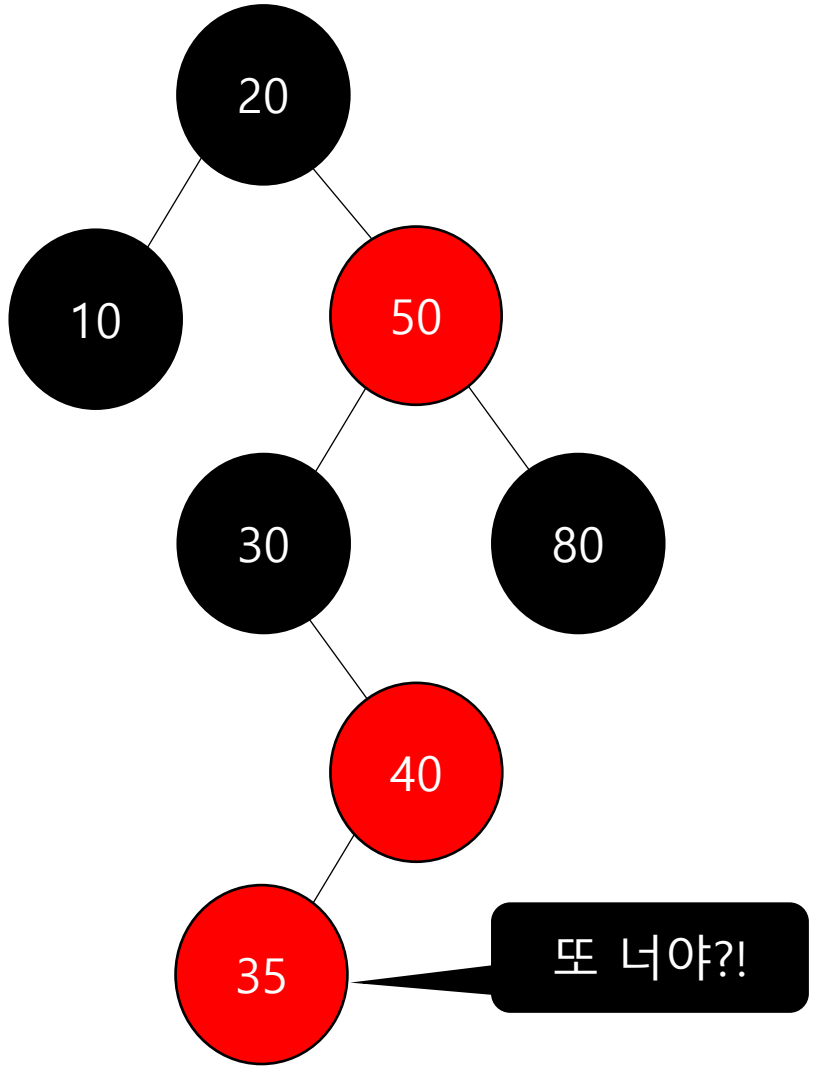
노드들은 Red or Black이고
루트는 항상 블랙
그리고 리프(NIL)도 블랙
그런데 레드의 자식은
모두 블랙 !

내 어떤 후손이든 내려 갈 때
만나는 블랙의 수는 같다.

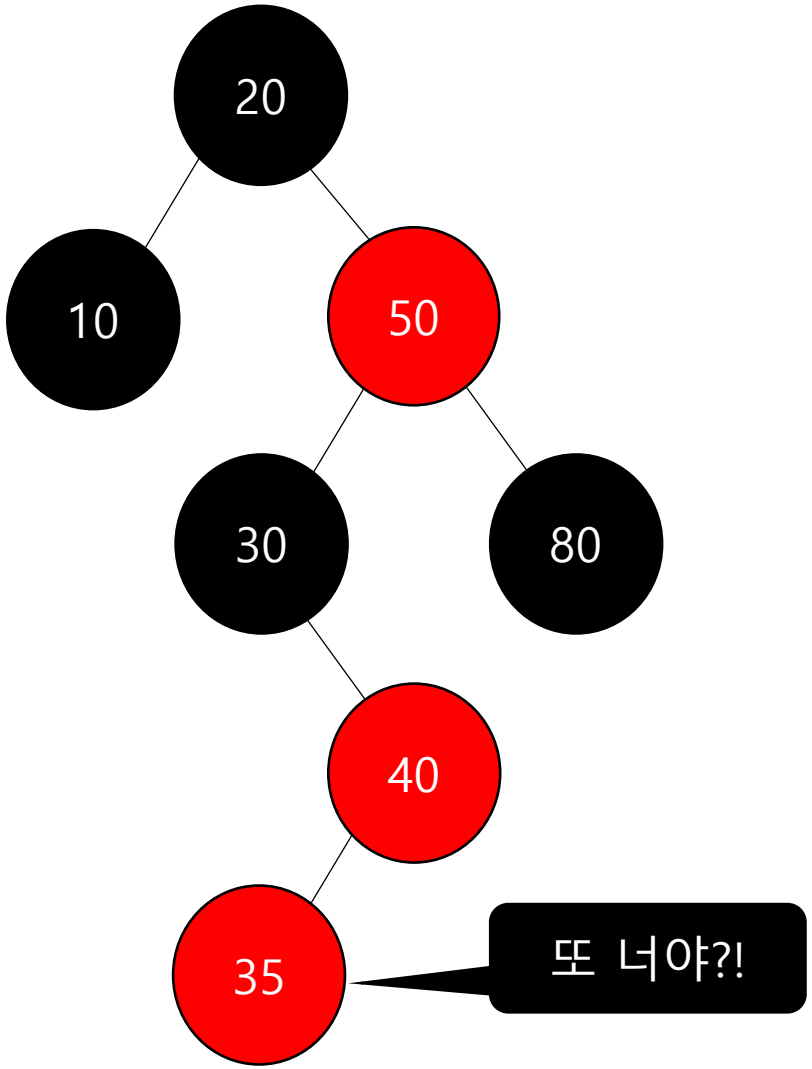
실전



35삽입
무조건 삽입되는 노드는 Red로



실전

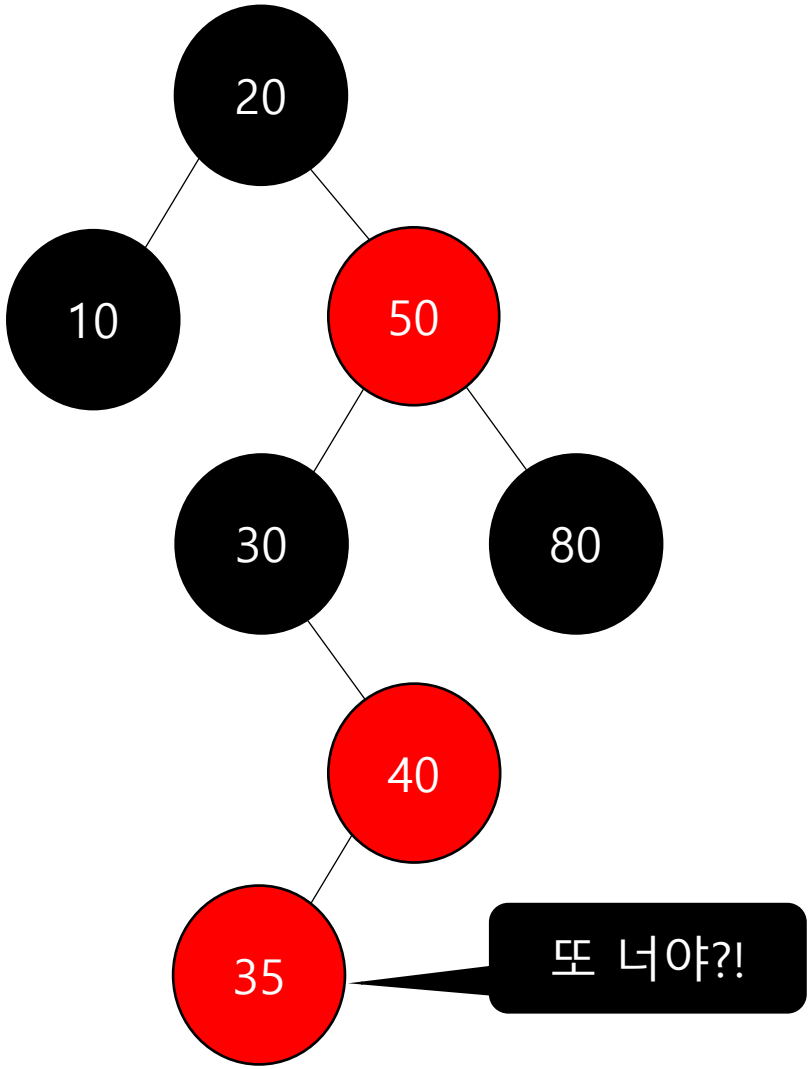


4. red의 자식들은 black이다.
즉 red는 연속적으로 존재 할 수 없다. **를 위배**

- 부모의 오른쪽/왼쪽 자녀이고
- (부모가) 할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

-> 부모 회전 후 위의 방법으로 회전

실전



4. red의 자식들은 black이다.
즉 red는 연속적으로 존재 할 수 없다. **를 위배**

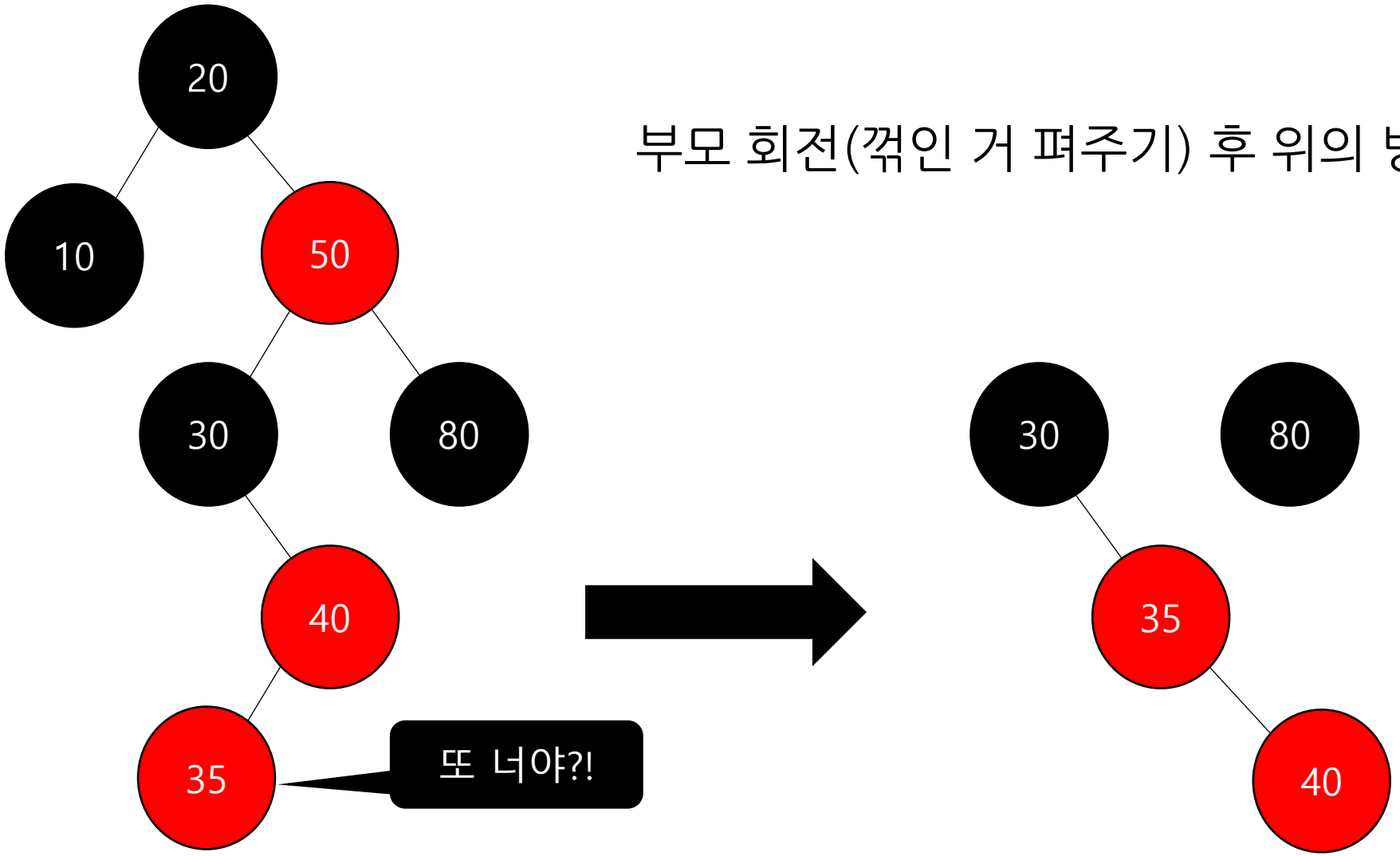
- 부모의 오른쪽/왼쪽 자녀이고
- (부모가) 할아버지의 왼쪽/오른쪽 자녀이고
- 부모의 형제는 black이라면

-> 부모 회전 후 위의 방법으로 회전

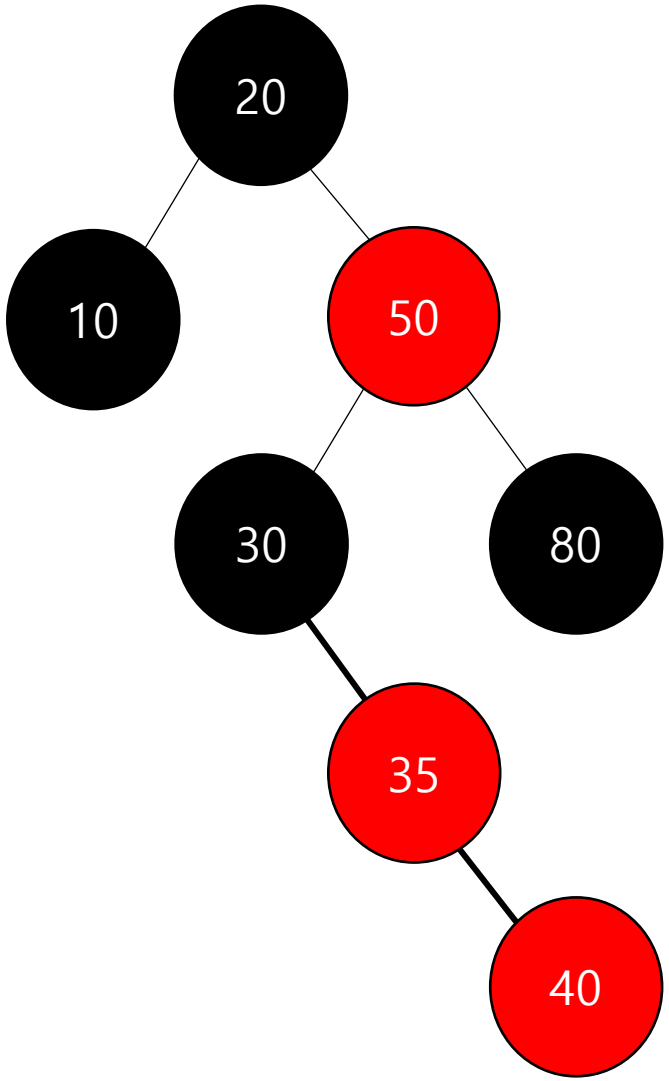
실전

4. red의 자식들은 black이다.
즉 red는 연속적으로 존재 할 수 없다. **를 위배**

부모 회전(꺾인 거 펴주기) 후 위의 방법으로 회전



실전

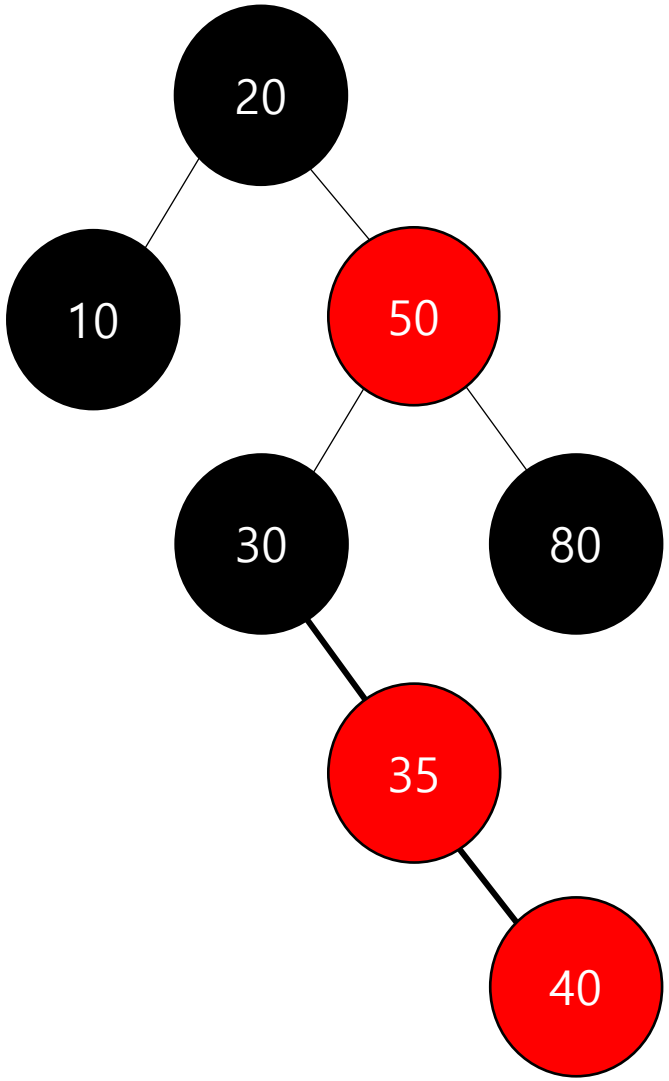


4. red의 자식들은 black이다.
즉 red는 연속적으로 존재 할 수 없다. **를 위배**

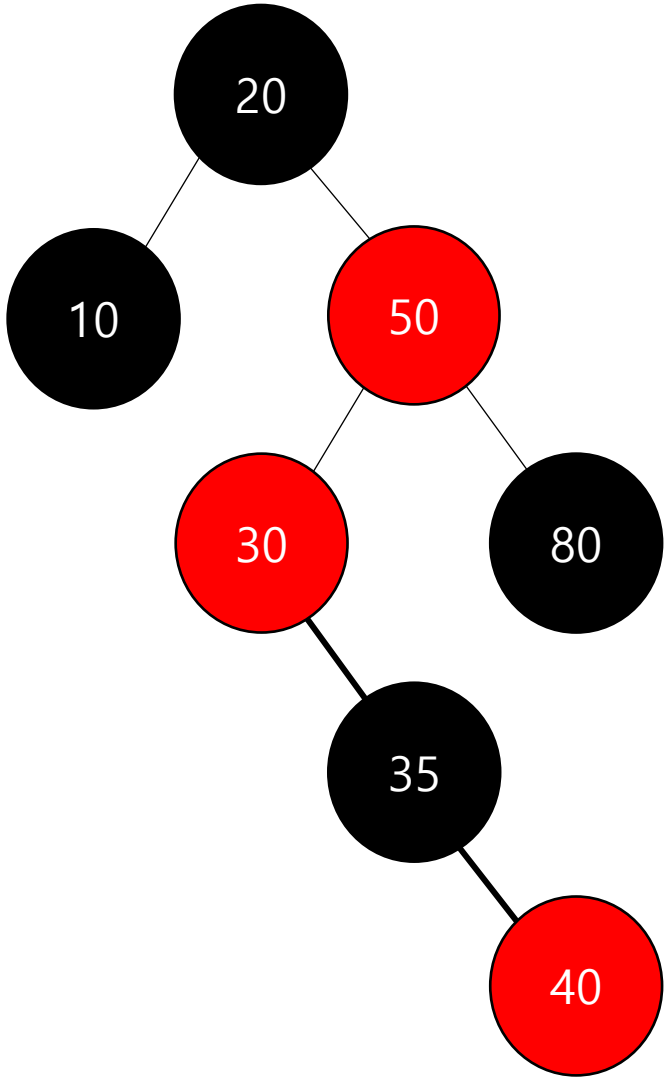
삽입된 red노드의 부모도 red일 때
•부모의 왼쪽/오른쪽 자녀이고
•(부모가)할아버지의 왼쪽/오른쪽 자녀이고
•부모의 형제는 black이라면

부모와 할아버지의 색을 바꾼 후
할아버지 기준으로 오른쪽/왼쪽으로 회전한다.

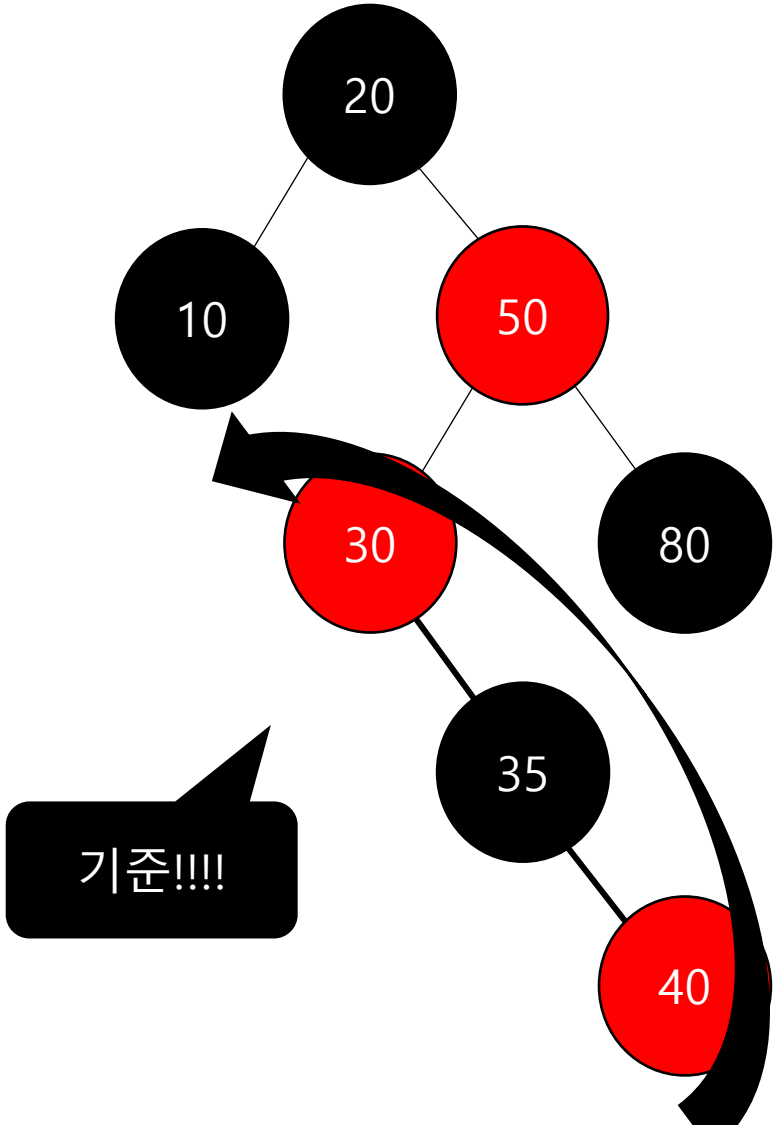
실전



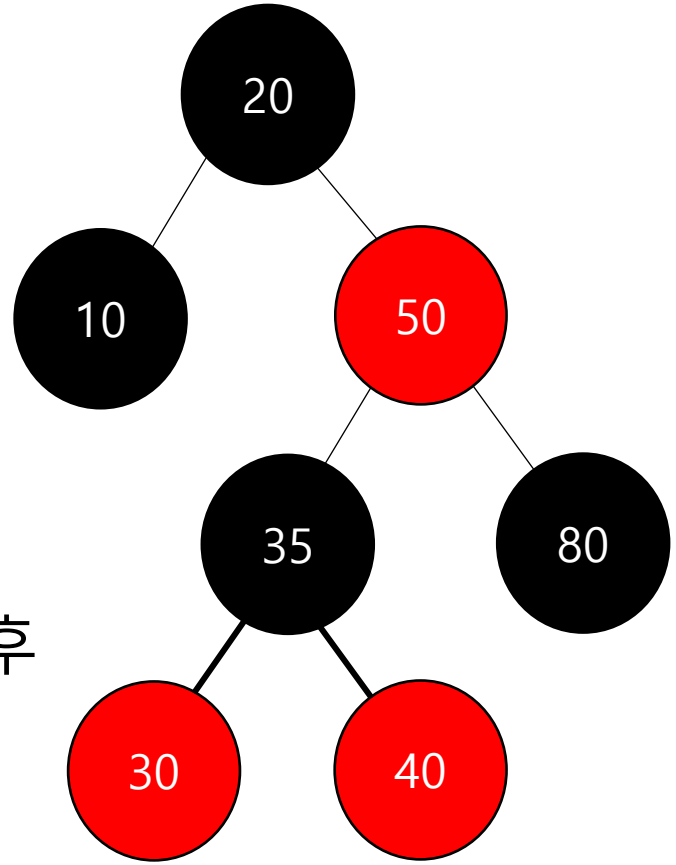
부모와 할아버지의 색을 바꾼 후
할아버지 기준으로
오른쪽/왼쪽으로 회전한다.



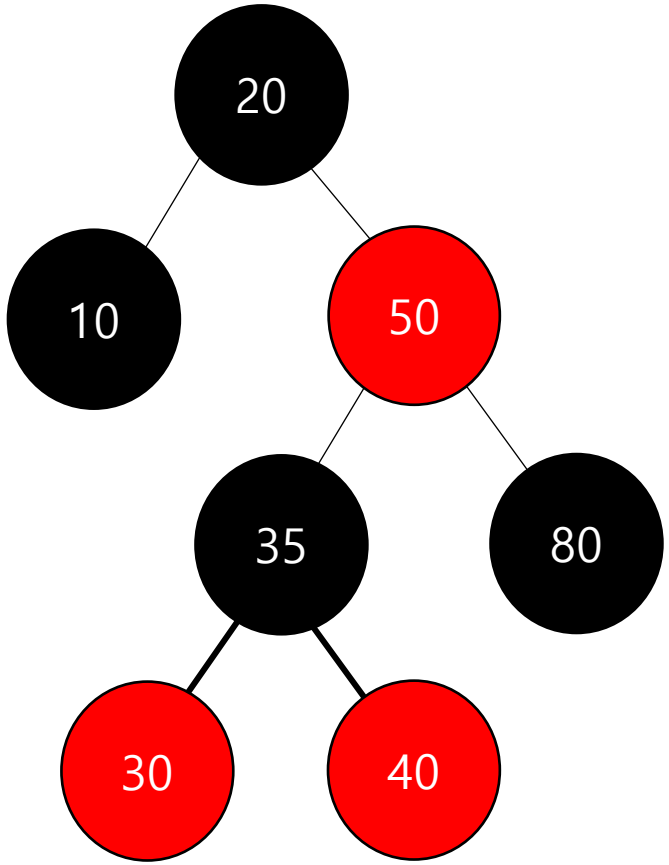
실전



부모와 할아버지의 색을 바꾼 후
할아버지 기준으로
오른쪽/왼쪽으로 회전한다.



실전 : 해치웠나..?



25를 넣어 봅시다.

나아아아중에 고민해 보는 걸로!

끝

그러면 삭제는요?!
다음 시간에~
도망가~~~~~