6주차 3차시 균형나무와 외부 탐색법

【학습목표】

1. 234나무, 흑적 나무를 이해할 수 있다.
2. 외부 탐색법을 이해할 수 있다.

학습내용1 : 234나무

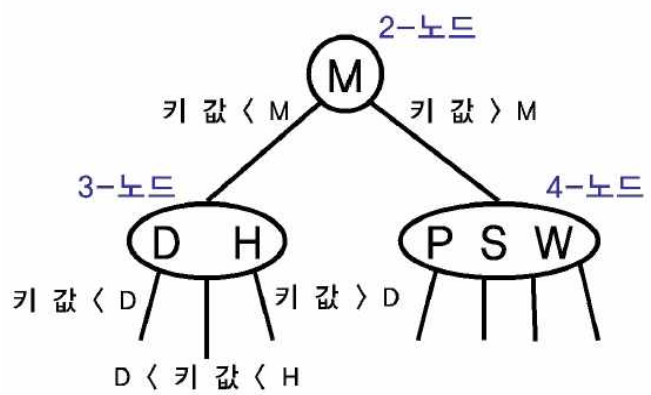
1. 균형나무

균형나무란 좌, 우 부분나무의 높이가 같은 나무를 말하며 균형인수가 ±1을 이하를 이루는 트리를 의미한다.
이진 탐색 나무의 최악의 탐색 시간 O(n) 이다

- ☞ A, B, C, D, E, F,.....
- ☞ Z, Y, X, W, V, U.....
- ☞ A, Z, B, Y, C, X.....

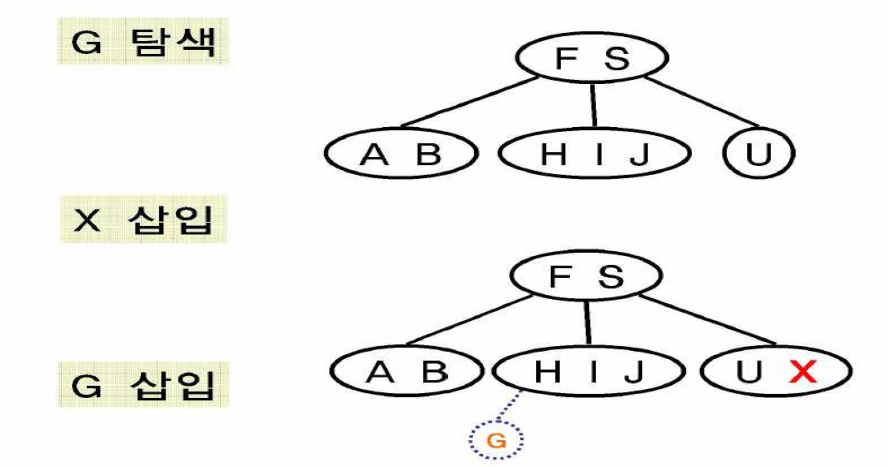
균형나무 라면 최악의 탐색 시간→O(logn)
경사나무가 만들어지지 않도록 하기 때문이다.

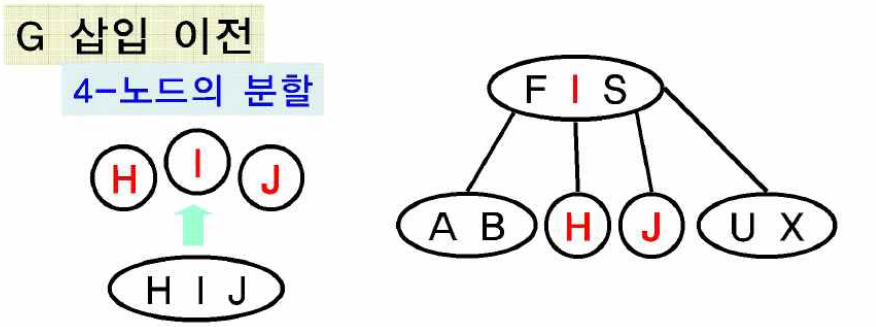
2-노드, 3노드, 4노드로 구성
K-노드 : 링크 K개, 키 K-1개

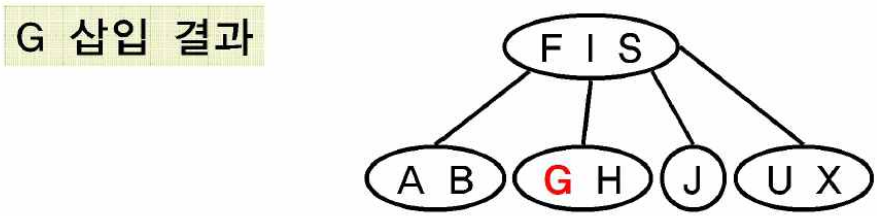


이진탐색 나무 = 2-3-4나무에서 2-노드로만 구성된 나무.

* 탐색과 삽입의 예







⦿ 만약 4-노드 분할 과정에서 부모노드도 역시 4-노드인 경우에는 어떤 문제가 발생?

☞ 4-노드 분할이 위쪽으로 연속적으로 전파.

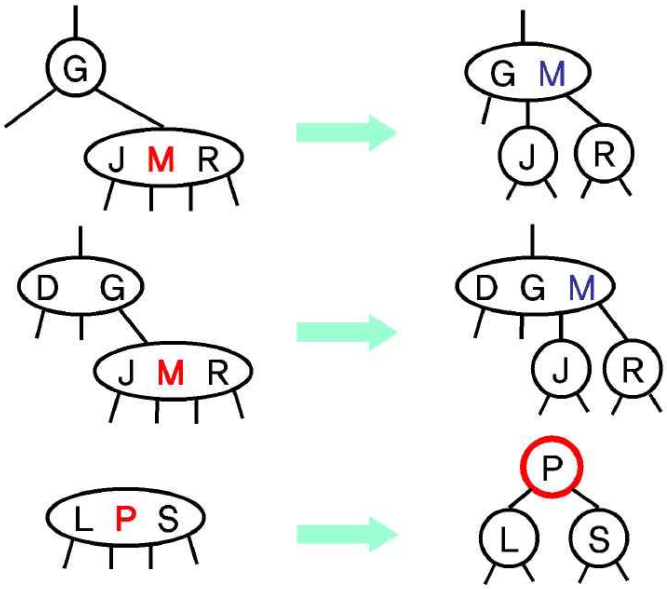
⦿ 자기 자신 및 부모 노드가 동시에 4-노드인 경우가 발생하지 않도록 함

☞ 삽입 지점을 탐색하는 과정에서 4-노드를 만나면 미리분할

⦿ 뿌리가 4-노드이면 세 개의 2-노드로 분할

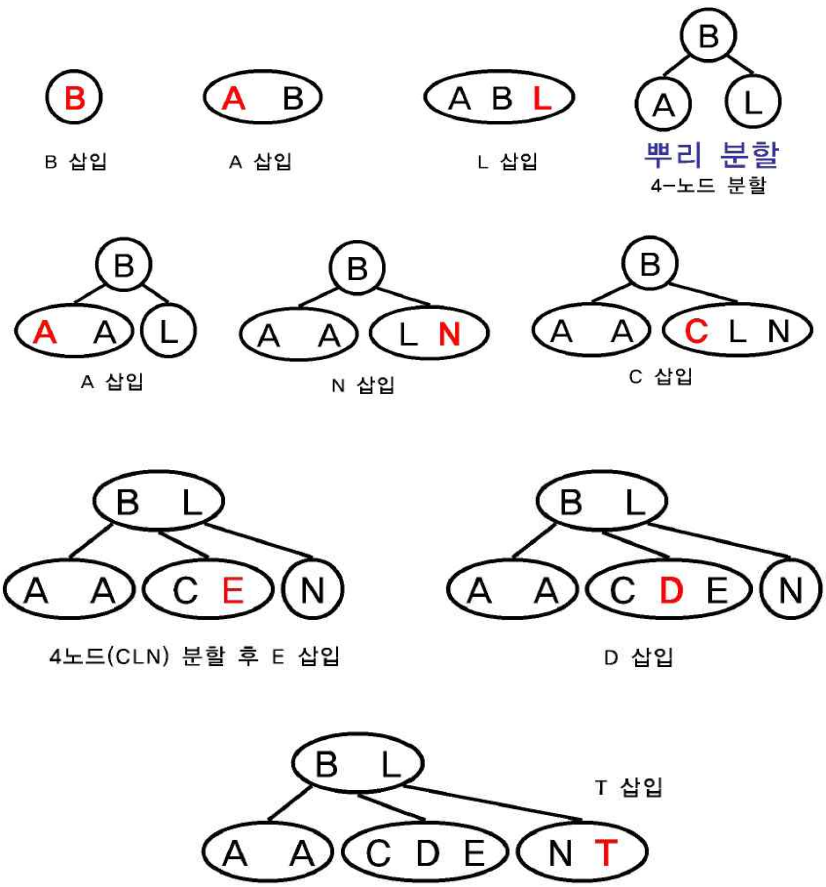
☞ 나무의 높이 1증가

* 노드의 분할



* 2-3-4 나무의 생성 예

키값 : B, A, L, A, N, C, E, D, T



나무의 높이가 완벽하게 균형을 이루었다!

* 2-3-4나무의 분석

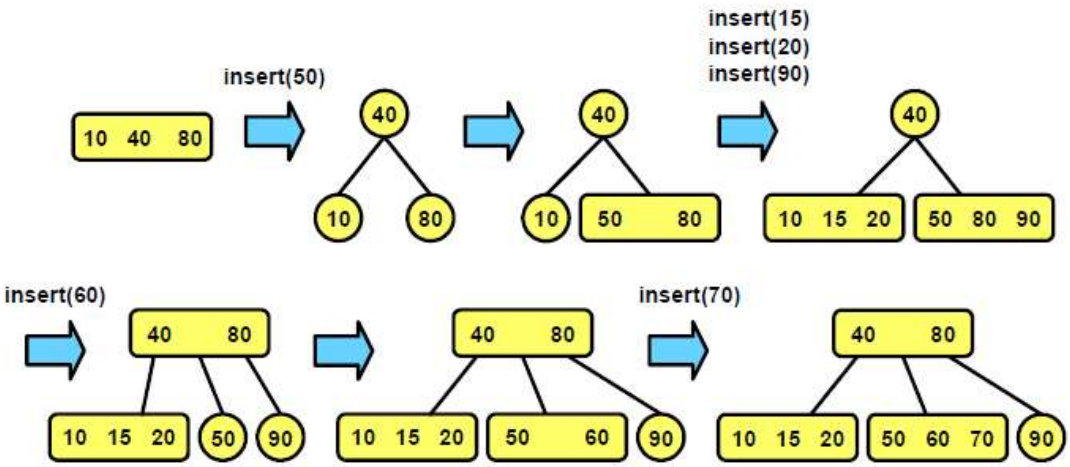
⦿ n개의 노드로 구성된 2-3-4 나무
☞ 나무의 최대 높이 ⌊log n⌋
☞ 탐색 시 방문하는 최대의 노드수 ⌊log n⌋+1
☞ 탐색 시간 → O(log n)

⦿ 최대 분할 노드의 수는 ⌊log n⌋+1 이지만 실험적으로는 평균 하나 이하로 발생.

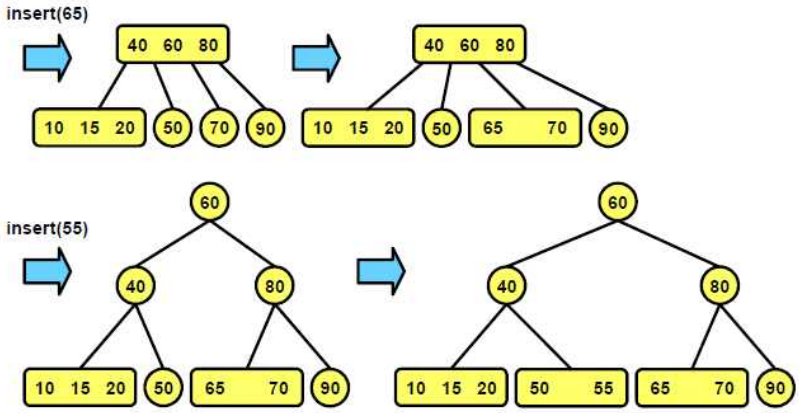
⦿ 2-3-4나무를 그대로 구현하면 노드 구조가 복잡
☞ 이진 탐색 나무보다 더 느려질 가능성이 많다.

2. 2-3-4 나무의 삽입

2-3-4 트리에서 삽입은 항상 단말 노드에서 이루어진다. 삽입할 노드를 찾아가는 동안 4-노드를 만나면 먼저 분할한다.



2-3-4 트리에서 삽입은 2-3 트리와 유사하다. 차이점은 2-3-4 트리에서는 삽입할 노드를찾아가는 동안 4-노드를 만나면 먼저 분할한다. 위 슬라이드에서 처음 3개의 값이 삽입되면 트리에 있는 유일노드가 4-노드가 된다. 그 다음에 또 다른 값을 삽입하고자 하면 이 노드가 먼저 분할된다. 4-노드가 분할될 때에는 중간 값이 부모노드로 올라가고, 왼쪽 값으로만 구성된 노드와 오른쪽 값으로만 구성된 노드가 새롭게 만들어진다.

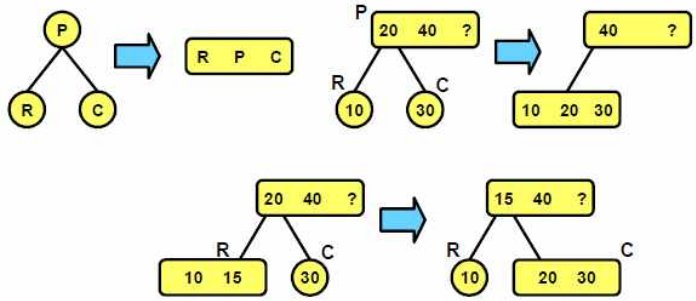


※ **Split**할 때에는 항상 중간 값이 부모노드로 올라간다.

3. 2-3-4 나무의 삭제

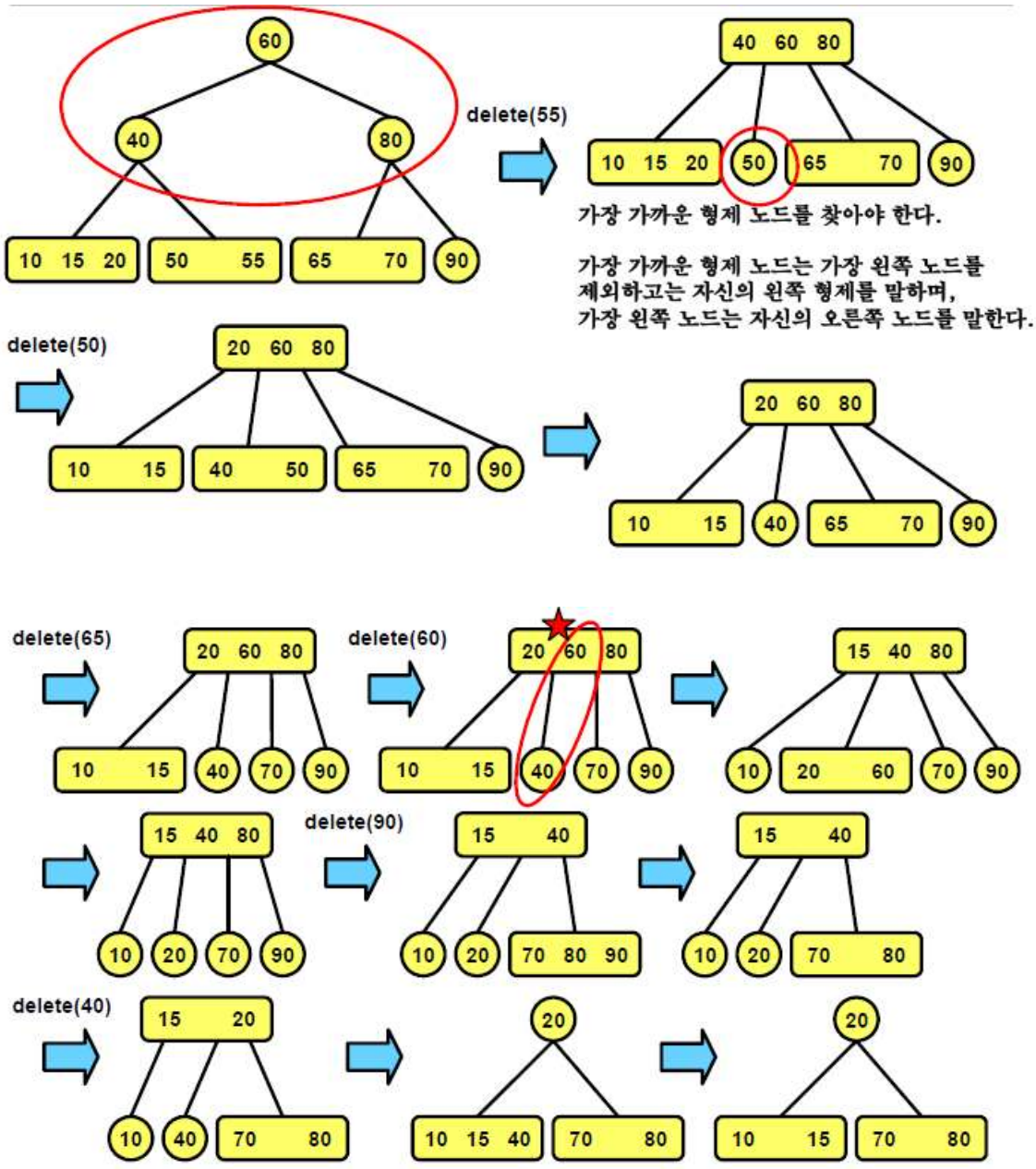
2-3-4 나무에서 삭제는 먼저 삭제할 값이 포함된 노드를 찾아야 한다. 그 노드가 단말노드가 아니면 그 값과 그것의 중위 순위 상에서 바로 앞 값을 교환한다. 이 값은 항상 단말노드에 있다. 즉, 제거는 항상 단말 노드에서 이루어지도록 한다. 이 때 이 노드가 3-노드 또는 4-노드이면 값을 제거한 다음에 종료한다. 반대로 이 노드가 2-노드이면 추가 작업이 필요하다. 이런 불편함을 해소하기 위해 삭제할 값이 포함된 노드를 찾을 때 만나게 되는 모든 2-노드를 3-노드 또는 4-노드로 바꾼다.(split 의 반대과정을 수행한다.) 이렇게 하면 모든 제거는 한 패스에 끝낼 수 있다.
2-3-4 트리에서 삭제도 2-3 트리와 마찬가지로 삭제할 값이 포함된 노드를 먼저 찾아야하며, 이 노드가 중간 노드이더라도 2-3 트리와 마찬가지로 결국에 삭제는 단말노드에서 이루어진다. 삭제가 이루어진 단말노드가 3-노드 또는 4-노드이면 값을 제거함으로써 삭제는 완료된다. 하지만 삭제가 이루어진 단말노드가 2-노드이면 빈 노드가 되므로 2-3 트리와 마찬가지로 추가 작업을 해야 한다. 보다 삭제를 수월하게 하기위해 2-3-4 트리에서는 삽입과 마찬가지로 삭제할 노드를 찾아가면서 만나게 되는 모든 2-노드를 3-노드 또는 4-노드를 미리 바꾼다. 이렇게 하면 삭제할 단말노드의 부모노드는 항상 3-노드 또는 4-노드가 되므로 삭제에 따라 연쇄적으로 루트까지 다시 올라가면서 추가 작업을 하지 않아도 된다.

- **2-**노드를 **3-**노드 또는 **4-**노드로 바꾸는 방법
  - 현재 노드가 **P**이고 삭제할 값이 있는 노드까지 경로 상에 있는 다음 자식 노드를 **C**라 하고, **C**의 가장 가까운 형제노드를 **R**이라 하자.



2-3-4 트리에서 삭제를 할 때 먼저 루트부터 탐색을 하면서 삭제할 값이 들어있는 노드를 찾는다. 그 경로 상에 있는 노드가 2-노드가 아니면 추가 작업 없이 다음 노드로 이동한다. 이때 현재노드를 P라 하고 찾아갈 경로 상에 있는 P의 자식노드를 C라 하자. C 노드가 2-노드이고 가장 가까운 형제노드가 2-노드인 경우는 다음과 같이 병합된다. 여기서 가장 가까운 형제노드란 2-3 트리에서 가장 가까운 형제노드 개념과 같다.

첫째, P가 2-노드인 경우는 P, C, 그리고 C의 형제노드가 병합되어 4-노드가 된다. 둘째, P가 3-노드인 경우는 C는 P 노드의 있는 하나의 값과 형제노드를 함께 병합하여 4-노드가 되고, P 노드는 2-노드가 된다.
C 노드가 2-노드이고 가장 가까운 형제노드가 3-노드 또는 4-노드인 경우는 다음과 같이 병합된다. 형제노드 값 중 하나가 P 노드로 올라가고 P 노드에 있는 값이 C 노드로 내려 C
노드가 3-노드가 된다.

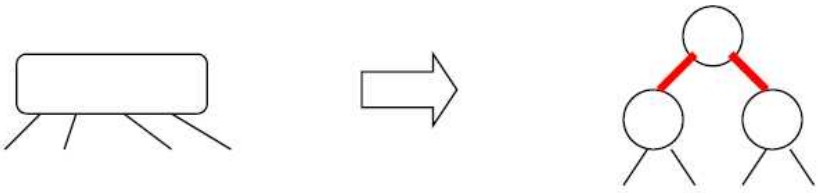


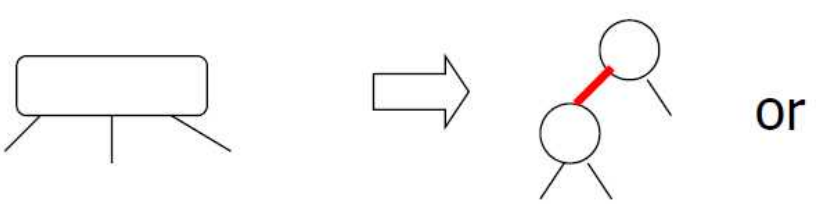
* n개의 노드로 구성된 2-3-4 나무가 있다고 할 때 나무의 높이는 최대 (log n) 이며 탐색 시 방문하는 노드 수는 최대 (log n) +1 이고 탐색 시간은 O(log n) 이다.
2-3-4 나무를 그대로 구현하면 노드 구조가 복잡하기 때문에 추가로 많은 노력이 요구되어 오히려 이진 탐색나무보다 더 느려질 가능성이 많다.

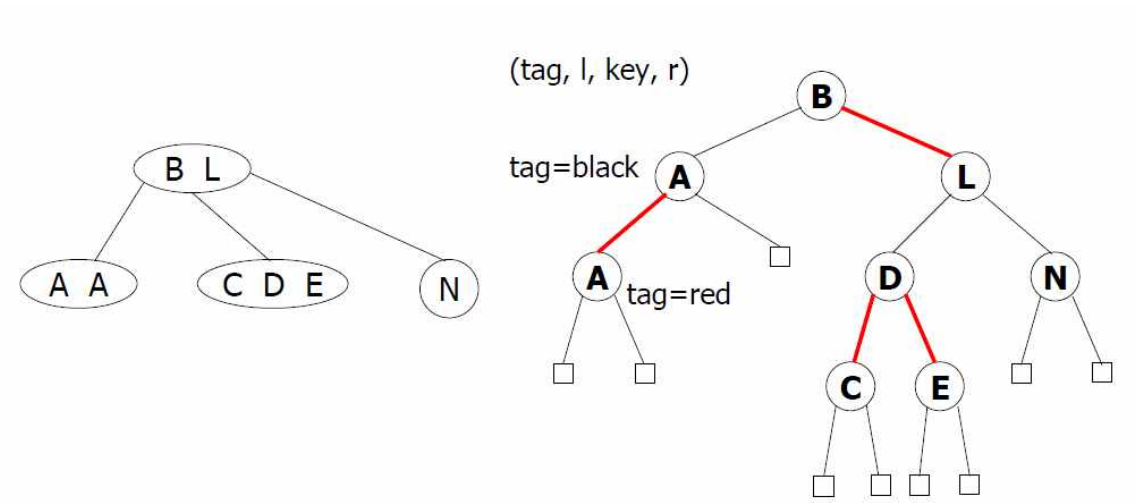
학습내용2 : 흑적나무

1. 흑적나무의 개요

흑적 나무(red black tree)는 위에서 공부한 2-3-4 나무를 이진 탐색나무 형태로 구현한 것을 말한다. 흑적나무의 링크에는 흑 링크와 적 링크의 두 종류가 있는데 흑 링크는 보통의 링크를 말하며 적 링크는 3-노드 또는 4-노드에 속하는 노드들을 연결할 때 사용한다.
2-3-4 나무와 흑적 나무를 그림으로 비교하면 다음과 같다.

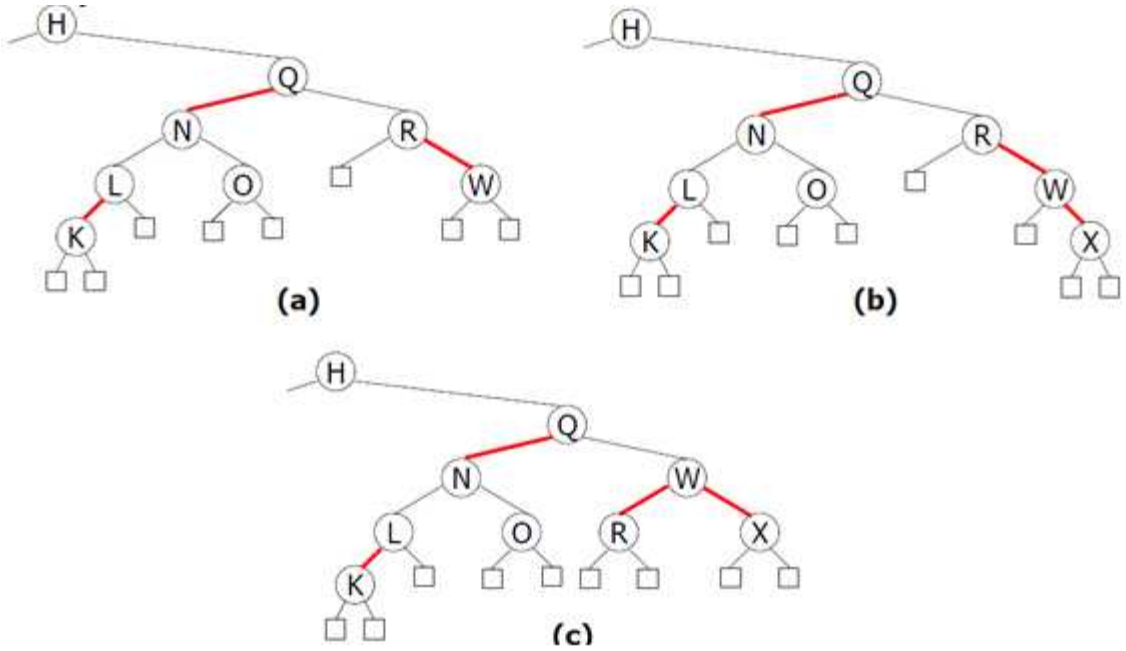






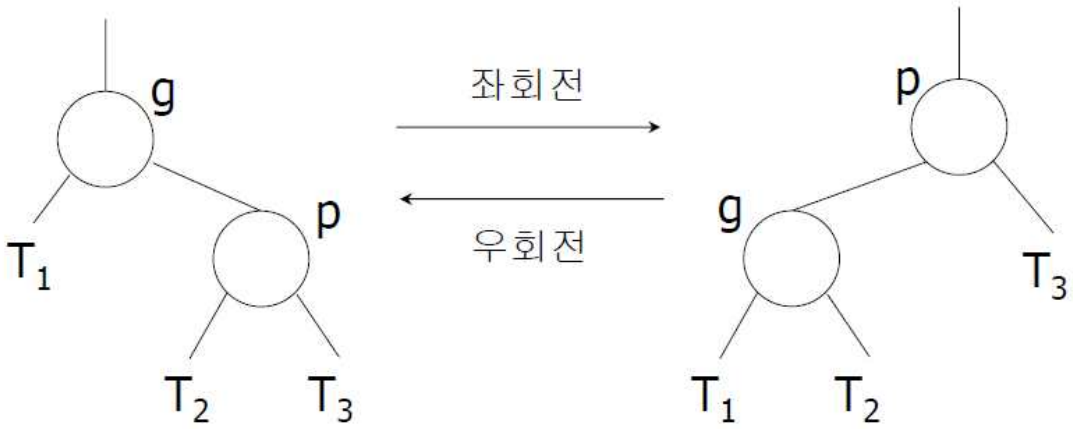
흑적 나무에서는 어떤 노드의 tag가 적색이면 이 노드의 두 자식 노드의 tag는 모두 흑색이다. 따라서 뿌리에서 잎까지의 경로 상에 두 개의 적 링크가 연달아 나타나지 않는다. 또한 어떤 노드에서 그 자손의 모든 잎 노드까지 단순 경로 상에 존재하는 흑 링크의 개수는 동일하다.
나무의 키의 총 수가 n인 흑적 나무의 가능한 최대 깊이는 O(log n)이다. 흑 또는 적의 색상을 고려하지 않을 때 가장 이상적으로 꽉 채워진 나무의 깊이는 ⌊log n⌋ g+1 이다. 그러므로 흑적나무가 아무리 잘 만들어져도 뿌리에서 임의의 잎에 이르는 경로에 존재하는 흑 노드의 개수는 ⌊log n⌋ g+1을 넘을 수 없다.
흑적 나무의 성질에 의해 적 노드는 두 개가 연속해서 존재할 수 없으므로, 뿌리에서 임의의 잎에 이르는 경로에서 적 노드의 개수가 흑 노드의 개수보다 많을 수는 없다. 그러므로 뿌리에서 임의의 잎에 이르는 경로의 길이는 2(⌊log

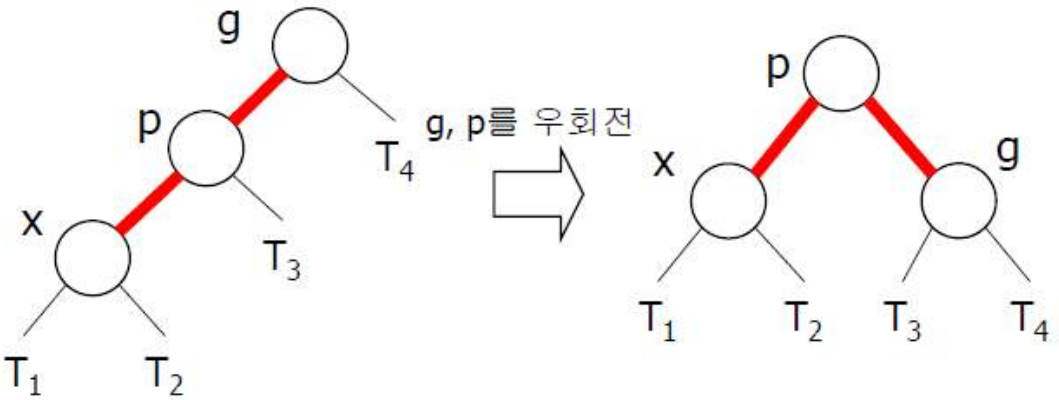
n ⌋ g+1)을 넘을 수 없다. 따라서 키의 총수가 n인 흑적 나무의 가능한 최대깊이는 O(log n)이다.
흑적나무는 사실상 이진 탐색 나무와 비슷하다고 할 수 있다. 탐색은 이진 나무의 탐색 연산과 동일하기 때문이다. 하지만 삽입 연산은 흑적 나무의 성질을 만족하기 위해서 추가적으로 회전과 분할이라는 것이 필요하다.
흑적 나무에 X 라는 노드를 삽입하는 과정을 그림으로 보면 다음과 같다.

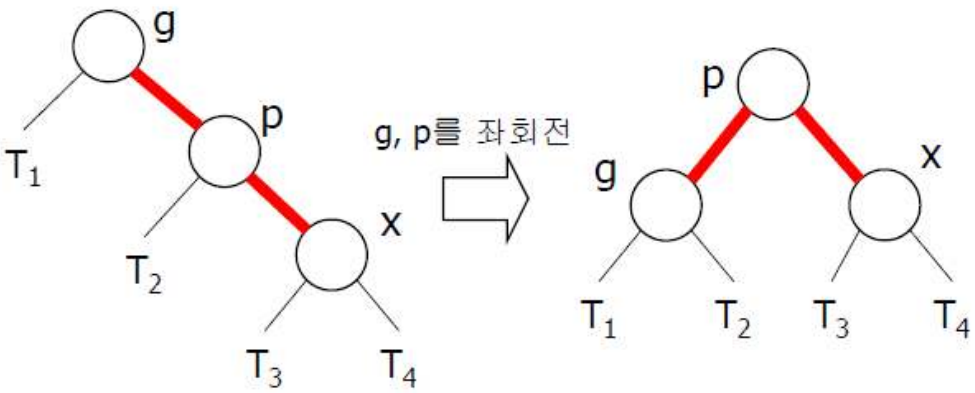


2. 회전

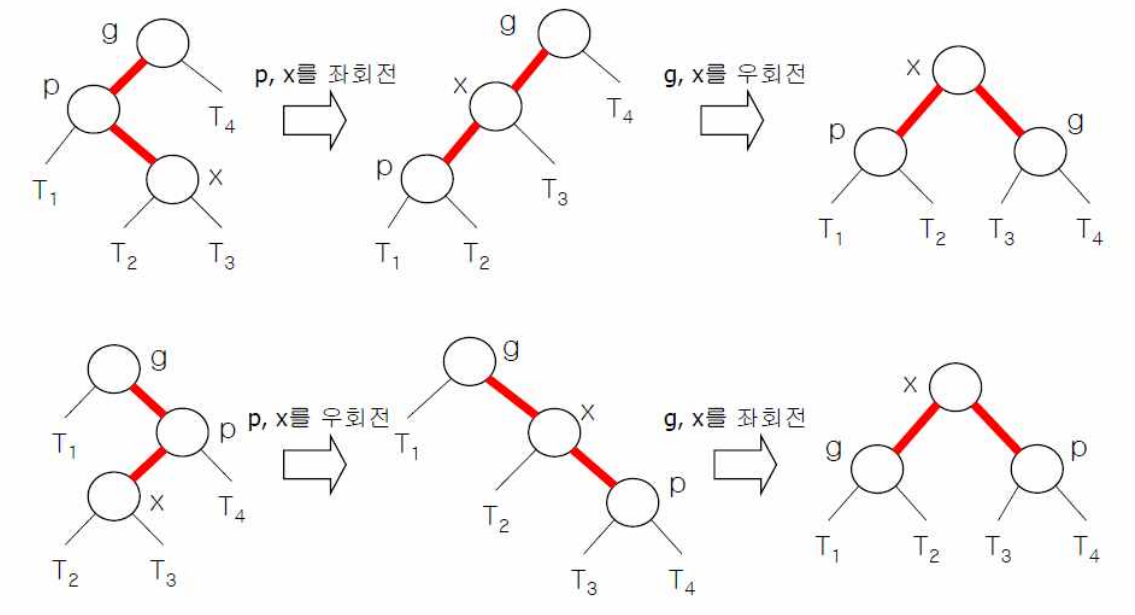
좌회전과 우회전을 그림으로 나타내면 다음과 같다. 아래와 같이 하나만 회전하는 경우를 단일회전이라고 한다.




g
p
x
T1
T2
T3
T4
g, p를 우회전
p
x
g
T1
T2
T3
T4

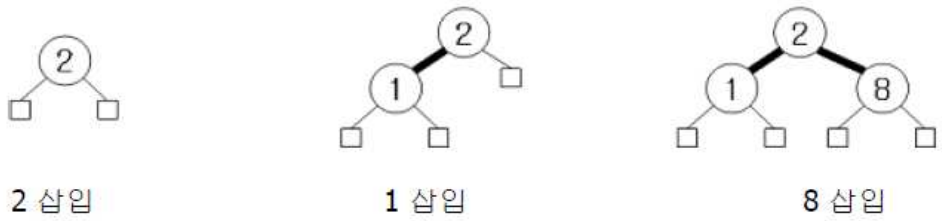

g
T1
p
T2
x
T3
T4
g, p를 좌회전
p
g
x
T1
T2
T3
T4

이중회전을 그림으로 보면 다음과 같다.



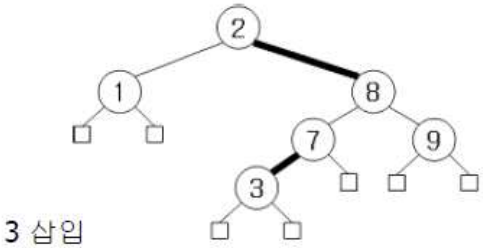
3. 흑적나무 생성 과정

위에서 배운 흑적 나무의 특수한 성질을 이용해서 아래의 데이터를 흑적 나무로 생성하는 과정을 그림으로 살펴보겠다.
초기 데이터 : {2, 1, 8, 9, 7, 3, 6, 4, 5}

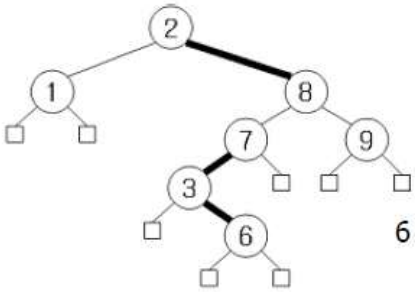


2 삽입 1 삽입 8 삽입

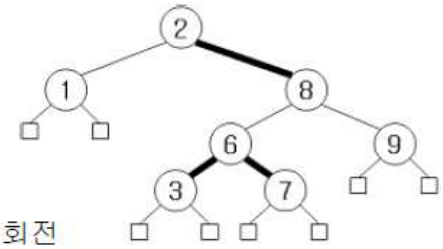
9 삽입 7 삽입



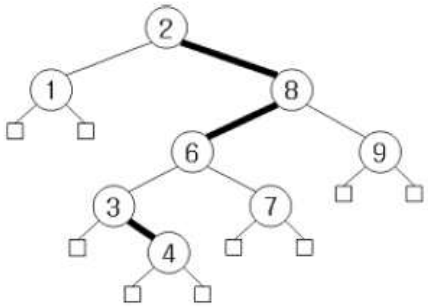
3 삽입



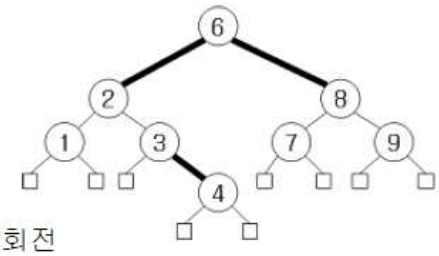
6 삽입



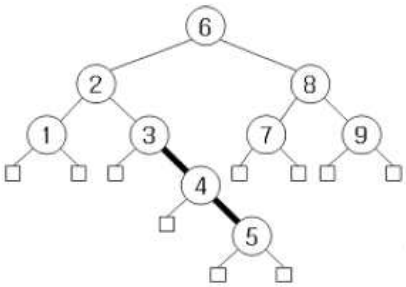
이중 회전



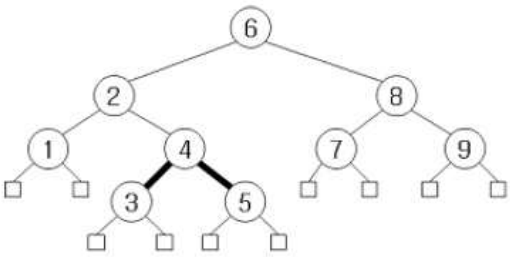
4 삽입



이중 회전



5 삽입



단일 회전

학습내용3 : 외부 탐색법

1. 외부 검색(external search)

검색하고자 하는 레코드의 수가 많고 크기가 매우 커서 디스크와 같은 외부 기억 장치에 저장된 파일로부터 필요한 레코드를 찾아내는 검색 방법.

2. B-나무(Balanced m-way search tree)

B-트리(Balanced Tree)는 R. Bayer와 E. McCreight가 1972년에 발표한 논문에서 제안한 방법으로 해싱이 아닌 거의 모든 큰 규모의 파일에서 독점적으로 사용되는 방법으로 삽입, 삭제와 키 범위 탐색을 필요로 하는 응용 프로그램을 위한 표준 파일 조직이다. B-트리는 모든 리프 노드들을 같은 레벨에 있게 함으로써 항상 높이 균형을 이루는 특징을 갖고 있어 업데이트와 탐색 연산은 단지 몇 개의 디스크 페이지에만 영향을 미친다. 또한 관련된 레코드들을 같은 디스크 페이지에 저장하여 지역 참조성을 이용한다. 그리고 트리에 있는 모든 노드가 적어도 어떤 최소비율로 소유하게 되어 공간/탐색/업데이트 연산 시 필요한 디스크 읽기 횟수를 줄일 수 있다.
즉, 차수가 m인 B-트리를 간단히 정의하면

- 차수 m인 경우 각각의 내부 노드가 m 개까지의 가지를 포함하는 트리로
- 내부노드들은 m 개의 자식을 가지고 m-1개의 키 값을 저장한다.
- 루트는 리프이거나 적어도 두 개의 자식을 가져야 하며,
- 루트와 리프를 제외한 각 노드는 ⌈m/2⌉와 m 사이의 자식들을 가진다.
- 모든 리프는 트리에서 같은 레벨에 있으며, 트리는 항상 높이 균형을 이루고,
- 리프가 아닌 노드의 키 값의 수는 그 노드의 서브트리수보다 하나 적으며, 각 리프노드는 최소 ⌈m/2⌉-1 개와 최대 m-1개의 키 값을 갖는다.
- 또한 한 노드 안에 있는 키 값들은 오름차순을 유지한다.

【학습정리】

1. 균형 나무
- 좌측 부분나무와 우측 부분나무의 높이가 같은 나무
- 이진 탐색 나무에서 경사 나무가 만들어지지 않도록 나무의 균형을 유지한다면 최악의 경우에도 O(logn)을 유지할 수 있을 것이다.

2. 2-3-4 나무
- 2-노드, 3-노드, 4-노드로 구성 → k-노드는 k개의 링크와 k-1개의키로 구성된다.

- 4-노드의 분할
  → 4-노드에서 삽입이 이루어지는 경우 트리의 높이가 높아지는 것을 피하기 위해 4-노드를 둘로 분할하여 가운데 키를 부모로 보내고 좌측과 우측 노드는 각각 2-노드로 구성한다.
  → 한편, 이러한 분할이 위쪽으로 연속적으로 전파되는 것을 방지하기 위해 삽입 지점의 탐색 과정에서 4-노드를 만나면 미리 분할한다. (이런 과정을 통해 자기 자신 및 부모 노드가 동시에 4-노드인 경우를 피할 수 있다.)
- 탐색 시간 O(logn) : 2-3-4 나무를 그대로 구현하면 노드의 구조가 복잡해서 이진 탐색 나무보다 더 느려질 가능성이 많다.

3. 흑적 나무
- 2-3-4 나무를 이진 탐색 나무 형태로 구현한 나무.
- 링크의 종류
  → 흑 링크: 보통의 링크와 동일
  → 적 링크: 2-3-4 나무에서 3-노드와 4-노드에 속하는 노드들을 연결할 때 사용하는 링크.
- 흑적 나무의 노드는 (tag, leftchild, key, rightchild)로 구성되며, tag 필드는 해당 노드로 들어오는 링크의 색상을 표현한다.
- 흑적 나무의 성질
  ① 어떤 노드의 태그가 적색이면 이 노드의 두 자식 노드의 태그는 모두 흑색이다. 따라서 뿌리에서 잎까지의 경로 상에 두 적 링크가 연달아 나타나지 않는다.
  ② 어떤 노드에서 그 자손의 모든 잎 노드까지의 단순 경로 상에 존재하는 흑링크의 개수는 동일하다.
- 회전 연산이란?
  삽입 또는 분할에 의해 적 링크가 잇달아 나타나는 것을 바로잡기 위하여 이들 노드들을 왼쪽 또는 오른쪽으로 회전하는 것.
  분할 연산의 수행 내용은? 4-노드를 분할하기 위하여 색을 바꾸고 필요하면 노드들을 회전시킨다.