알고리즘과 자료구조 워크북

교과목명 : 알고리즘과 자료구조
차시명: 13차시 트리1

◈ 담당교수: 신일훈 **(서울과학기술대학교)**

◉ 세부목차
- 트리의 개념
- 이진트리 개념 및 트리 순회 방법
- 이진탐색트리 개념 및 주요 연산

학습에 앞서

■ 학습개요

트리는 노드와 노드를 연결하는 링크로 구성된 비선형 자료구조이다. 그래프와 다른 점은 사이클이 없다는 점이며 계층적인 구조를 갖는다. 트리 중에서 자식 노드가 최대 2개인 노드를 이진 트리라고 하며 이진 트리의 노드를 순회하는 방법은 크게 전위순회, 중위순회, 후위순회, 레벨순회 등이 있다. 이진 트리 중 모든 노드에 대해 노드의 키가 왼쪽 서브 트리의 노드들의 키보다 크고 오른쪽 서브 트리의 노드들의 키보다 작다는 조건을 만족하면 이진 탐색 트리라고 한다. 이진 탐색 트리는 선형 자료 구조에 비해 평균적으로 검색에 걸리는 시간을 줄일 수 있는 장점이 있다. 단 검색 연산의 최악의 시간 복잡도는 스택 또는 큐와 동일하다.

■ 학습목표

1	트리의 개념을 이해한다.
2	이진 트리의 개념 및 트리 순회 방법을 이해한다.
3	이진 탐색 트리의 개념 및 주요 연산을 이해한다.

■ 주요용어

용어	해설
트리	노드와 노드를 연결하는 링크로 구성된 계층 구조 형태의 비선형 자료구조로서 그래프와의 차이점은 사이클이 없다는 것이다.
이진 트리	트리이면서 각 노드의 자식 노드가 최대 2개인 트리를 이진 트리라고 한다.
이진 탐색 트리	이진 트리 중 모든 노드에 대해 노드의 키가 왼쪽 서브 트리의 노드들의 키보다 크고 오른쪽 서브 트리의 노드들의 키보다 작다는 조건을 만족하면 이진 탐색 트리라고 한다.

학습하기

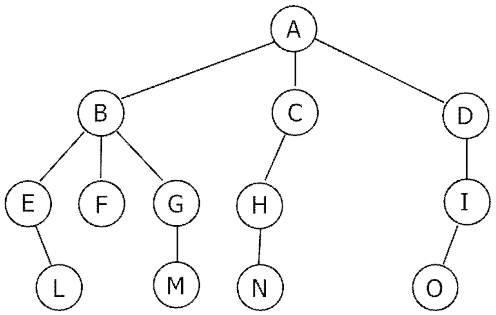
1. 트리 개념

가. 개념

- 노드와 노드를 연결하는 링크로 구성되는 자료구조로서, 계층 구조 형태를 띄는 비선형 자료구조임.
- 유사한 비선형 자료구조인 그래프와의 차이점은 사이클이 존재하지 않는다는 것이다.

나. 용어

- 루트(Root): 트리의 최상위 노드 (아래 그림에서는 A가 루트 노드임)
- 자식(Child): 노드의 하위에 연결된 노드 (아래 그림에서 B, C, D는 A의 자식임)
- 부모(Parent): 노드의 상위에 연결된 노드 (아래 그림에서 A는 B의 부모임)
- 차수(Degree): 자식 노드의 수 (아래 그림에서 A의 차수는 3이다.)
- 리프(Leaf): 자식이 없는 노드 (아래 그림에서 L, F, M, N, O는 자식이 없는 리프 노드이다.)
- 레벨(Level): 루트 노드의 레벨은 0 또는 1임. 자식 노드로 내려가며 레벨이 1씩 증가함. 깊이와 동일함.
- 높이(Height): 트리의 최대 레벨 (최상위 노드만 있을 때의 높이를 1이라고 한다면, 아래 그림에서 트리의 높이는 4이다.)



2. 이진트리 개념 및 트리 순회

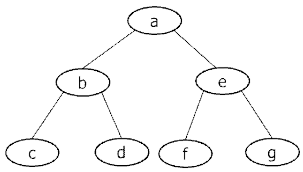
가. 이진트리 개념

- 각 노드의 자식 수가 2 이하인 트리를 이진트리라고 한다.

나. 이진트리 종류

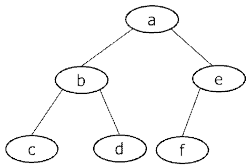
1) 포화이진트리

- 각 내부 노드(레벨이 높이보다 작은 노드)가 2개의 자식 노드를 가지는 트리



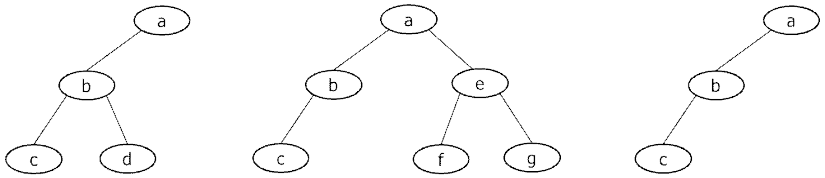
2) 완전이진트리

- 마지막 레벨을 제외한 트리가 포화이진트리이며, 마지막 레벨에는 노드들이 왼쪽부터 채워진 트리



3) 불완전이진트리

- 이진트리로서 완전이진트리가 아닌 트리를 불완전이진트리라고 한다.



다. 이진트리 순회

1) 전위순회(preorder traverse)

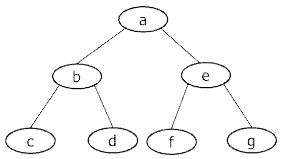
- 전위순회 알고리즘

1. 최상위 노드에서 순회 시작. 2. 노드 n에 도착하면 n을 먼저 탐색 3. 이후, n의 왼쪽 서브 트리의 노드들을 모두 순회 4. 왼쪽 서브 트리 순회 후에는 n의 오른쪽 서브 트리의 노드들을 모두 순회

- NLR 순서로 순회를 수행한다.
- 위의 알고리즘은 재귀적으로 기술되어 있음을 알 수 있다. 즉 왼쪽 서브 트리와 오른쪽 서브 트리의 노드들을 순회하는 작업이 각각 전체 트리를 순회하는 것과 동일한 알고리즘으로 수행된다.

- 전위순회 예시

a -> b -> c -> d -> e -> f -> g



- 최상위 노드인 a부터 시작하고 a를 먼저 방문한다.
- 이후, a의 왼쪽 서브 트리의 노드들을 모두 순회해야 한다.
- 즉, b, c, d가 왼쪽 서브 트리인데, 이 중 최상위 노드는 b이다.
- 따라서 b부터 먼저 순회하고, 이후 b의 왼쪽 서브 트리의 노드들을 탐색하므로 c를 탐색한다. b의 왼쪽 서브 트리 순회가 끝나면 오른쪽 서브 트리를 순회하며 따라서 d가 탐색된다.
- 이상으로 a의 왼쪽 서브 트리에 대한 순회가 종료된다.
- 이후 a의 오른쪽 서브 트리에 대해서도 동일한 순서로 순회가 수행된다.

2) 중위순회(inorder traverse)

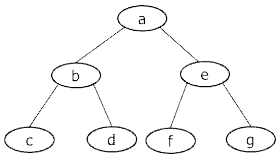
- 중위순회 알고리즘

1. 최상위 노드에서 순회 시작. 2. 노드 n에 도착하면 n의 탐색을 보류하고 먼저 n의 왼쪽 서브 트리의 노드들을 모두 순회. 3. 왼쪽 서브 트리의 모든 노드를 순회한 후에는 n을 탐색. 4. 이후 n의 오른쪽 서브 트리의 노드들을 모두 순회

- LNR 순서로 순회를 수행한다.
- 위의 알고리즘은 재귀적으로 기술되어 있음을 알 수 있다. 즉 왼쪽 서브 트리와 오른쪽 서브 트리의 노드들을 순회하는 작업이 각각 전체 트리를 순회하는 것과 동일한 알고리즘으로 수행된다.

- 중위순회 예시

c -> b -> d -> a -> f -> e -> g



- 최상위 노드인 a부터 시작하고 a의 방문을 보류한다.
- 먼저 a의 왼쪽 서브 트리의 노드들을 모두 순회해야 한다.
- 즉, b, c, d가 왼쪽 서브 트리인데, 이 중 최상위 노드는 b이다.
- 따라서 b부터 시작하는데 b의 방문은 보류하고 b의 왼쪽 서브 트리의 노드들을 먼저 탐색한다. 따라서 c를 먼저 탐색한다. b의 왼쪽 서브 트리 순회가 끝나면 b를 순회하고, 이후 b의 오른쪽 서브 트리를 순회한다. 따라서 d가 탐색된다.
- 이상으로 a의 왼쪽 서브 트리에 대한 순회가 종료된다.
- 이후 a의 오른쪽 서브 트리에 대해서도 동일한 순서로 순회가 수행된다.

3) 후위순회(postorder traverse)

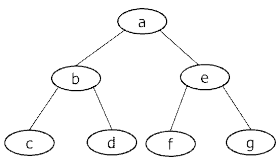
- 후위순회 알고리즘

1. 최상위 노드에서 순회 시작.
2. 노드 n에 도착하면 n의 탐색을 보류하고 먼저 n의 왼쪽 서브 트리의 노드들을 모두 순회.
3. 이후 n의 오른쪽 서브 트리의 노드들을 모두 순회
4. 마지막으로 노드 n을 탐색.

- LRN 순서로 순회를 수행한다.
- 위의 알고리즘은 재귀적으로 기술되어 있음을 알 수 있다. 즉 왼쪽 서브 트리와 오른쪽 서브 트리의 노드들을 순회하는 작업이 각각 전체 트리를 순회하는 것과 동일한 알고리즘으로 수행된다.

- 후위순회 예시

c -> d -> b -> f -> g -> e -> a



- 최상위 노드인 a부터 시작하고 a의 방문을 보류한다.
- 먼저 a의 왼쪽 서브 트리의 노드들을 모두 순회해야 한다.
- 즉, b, c, d가 왼쪽 서브 트리인데, 이 중 최상위 노드는 b이다.
- 따라서 b부터 시작하는데 b의 방문은 보류하고 b의 왼쪽 서브 트리의 노드들을 먼저 탐색한다. 따라서 c를 먼저 탐색한다. b의 왼쪽 서브 트리 순회가 끝나면 b의 오른쪽 서브 트리를 순회한다. 따라서 d가 탐색된다. 마지막으로 b를 순회한다.
- 이상으로 a의 왼쪽 서브 트리에 대한 순회가 종료된다.
- 이후 a의 오른쪽 서브 트리에 대해서도 동일한 순서로 순회가 수행된다.
- 마지막으로 a를 순회한다.

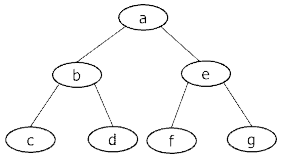
4) 레벨순회

- 레벨순회 알고리즘

1. 최상위 레벨의 노드부터 왼쪽에서 오른쪽 방향으로 순회한다.
2. 특정 레벨의 순회가 끝나면, 다음 하위 레벨의 노드들을 왼쪽부터 오른쪽 방향으로 순회한다.
3. 2번 작업을 마지막 레벨까지 반복한다.

- 후위순회 예시

a -> b -> e -> c -> d -> f -> g

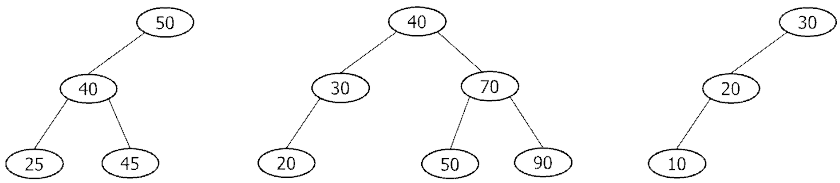


- 가장 최상위 레벨의 노드인 a를 순회한다.
- 다음 레벨은 b, e인데, 왼쪽부터 탐색하므로 b, e 순서로 탐색한다.
- 다음 레벨은 c, d, f, g인데 왼쪽부터 탐색하므로 c, d, f, g 순서로 탐색한다.

3. 이진탐색트리 개념

가. 이진탐색트리 개념

- 이진트리의 한 종류로서 다음의 조건을 만족하는 트리를 이진탐색트리라고 한다.
- 모든 노드에 대해, 노드 N의 키가 N의 왼쪽 서브 트리의 키들보다 크고, N의 오른쪽 서브 트리의 키들보다 작아야 함.
- 다음의 트리들은 이진탐색트리의 예시이다. 노드의 숫자가 키를 의미한다. 모든 노드들이 위 조건을 만족하고 있음을 알 수 있다.



나. 이진탐색트리 장점

- 선형자료구조 대비 탐색을 보다 효율적으로 수행할 수 있다.

1) 최대값 탐색

- 오른쪽 자식을 계속 따라가는 것을 반복하고, 더 이상 오른쪽 자식이 없는 노드를 만나면, 해당 노드가 최대값을 가진 노드이다.
- 선형 탐색과 달리, 모든 노드를 비교하지 않고, 일부 노드만을 따라가므로, 일반적으로 최대값을 더 빨리 찾을 수 있다.

2) 최소값 탐색

- 왼쪽 자식을 계속 따라가는 것을 반복하고, 더 이상 왼쪽 자식이 없는 노드를 만나면, 해당 노드가 최소값을 가진 노드이다.
- 선형 탐색과 달리, 모든 노드를 비교하지 않고, 일부 노드만을 따라가므로, 일반적으로 최소값을 더 빨리 찾을 수 있다.

3) 특정값 탐색

- 알고리즘은 다음과 같다.

1. 최상위 노드를 타겟 노드로 설정하고 탐색 시작
2. 타겟 노드의 키와 탐색 키를 비교
3. 탐색 키가 작으면 왼쪽 자식 노드를 새로운 타겟 노드로 설정하고 2를 반복. 이 때 만약 왼쪽 자식 노드가 없으면 찾는 노드가 없다는 의미이므로 실패를 반환하고 종료
4. 탐색 키가 크면 오른쪽 자식 노드를 새로운 타겟 노드로 설정하고 2를 반복. 이 때 만약 오른쪽 자식 노드가 없으면 찾는 노드가 없다는 의미이므로 실패를 반환하고 종료
5. 두 키가 동일하면, 노드를 발견했으므로 해당 노드를 반환하고 탐색을 종료함

- 선형 탐색과 달리, 모든 노드를 비교하지 않고, 일부 노드만을 비교하며 특정 키를 발견하므로, 일반적으로 탐색을 더 빠른 시간에 수행할 수 있다.

다. 이진탐색트리 탐색의 최악 시간복잡도

- 최소값, 최대값, 특정값을 찾는 탐색 모두 트리의 높이 만큼의 비교 연산 또는 순회 연산을 수행해야 한다. 따라서 트리의 높이를 h라고 하면 시간복잡도는 O(h)이다.
- 최악의 경우에는 리프 노드를 제외한 모든 노드가 자식을 하나만 갖는 선형 구조의 트리가 될 수 있으므로 탐색 시간의 최악 시간복잡도는 O(n)이다. n은 노드의 개수.
- 하지만 이는 특수 경우이며 발생 확률이 낮다. 일반적으로는 자식을 두 개 갖는 노드가 존재할 것이므로 평균적인 탐색 시간은 O(n)보다 낮다.

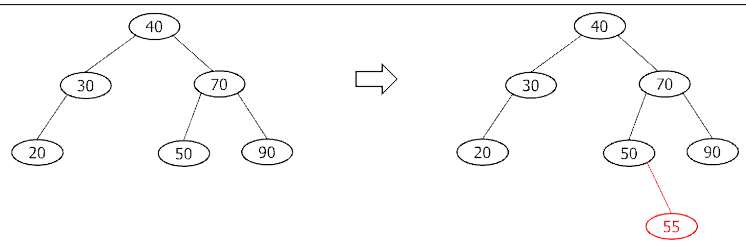
4. 이진탐색트리 주요 연산

가. 삽입

- 삽입 알고리즘

1. 최상위 노드가 None이면 (트리가 비어 있는 상태), 최상위 노드로 삽입하고 종료
2. 최상위 노드부터 시작하여 노드의 키와 탐색 키를 비교
    - 탐색 키가 더 작고 왼쪽 자식 노드가 존재하면 왼쪽 자식 노드에 대해 2번 작업 반복
    - 탐색 키가 더 크고 오른쪽 자식 노드가 존재하면 오른쪽 자식 노드에 대해 2번 작업 반복
    - 키가 동일하면 충돌 또는 update 케이스. 실패로 처리 또는 값 수정 후 종료
    - 키가 다른데 반복할 자식 노드가 존재하지 않으면, 키에 따라 해당 노드의 왼쪽 자식 또는 오른쪽 자식 노드로 삽입

- 삽입 예시



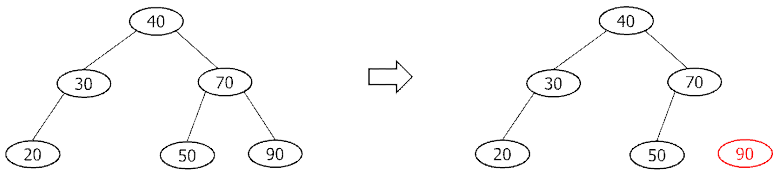
- 삽입할 키는 55이다.
- 최상위 노드부터 비교를 시작하며, 비교 결과에 따라 왼쪽 자식 또는 오른쪽 자식을 따라간다.
- 결국 50이 타겟 노드가 되고 이와 비교를 하면 55가 크므로 오른쪽 자식을 따라가야 한다.
- 하지만 오른쪽 자식이 존재하지 않으므로 50의 자식 노드로 55 노드를 추가한다. 이때 55가 더 크므로 오른쪽 자식 노드로 삽입된다.

나. 삭제

- 삭제 알고리즘

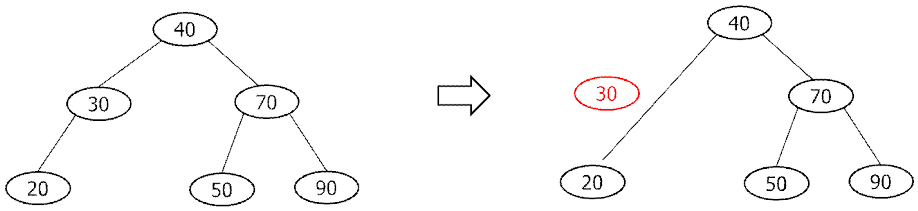
1. 탐색을 수행하여 삭제할 노드를 발견한다. 만약 삭제할 노드가 존재하지 않으면 삭제는 실패로 처리된다.
2. 삭제할 노드와 그 부모 노드를 발견한다.
3. 삭제할 노드가 리프노드인 경우
    - 부모 노드와 삭제할 노드를 연결하는 링크 제거
4. 삭제할 노드가 하나의 자식 노드를 갖는 경우
    - 부모와 삭제할 노드의 자식 노드를 직접 연결 (손자가 자식으로 바뀌는 격)
5. 삭제할 노드가 두 개의 자식 노드를 갖는 경우
    - 삭제할 노드의 왼쪽 서브 트리 중 키가 가장 큰 노드 또는 오른쪽 서브 트리 중 키가 가장 작은 노드를 후계자 노드로 설정한다.
    - 후계자 노드의 키를 포함한 모든 데이터를 삭제할 노드로 복사한다.
    - 후계자 노드를 삭제한다. 이때 후계자 노드는 리프 노드이던지, 아니면 자식을 하나만 갖는 노드인 경우 밖에 없으므로 3번 또는 4번 삭제 작업과 동일한 알고리즘으로 삭제를 수행한다.

- 삭제 예시1



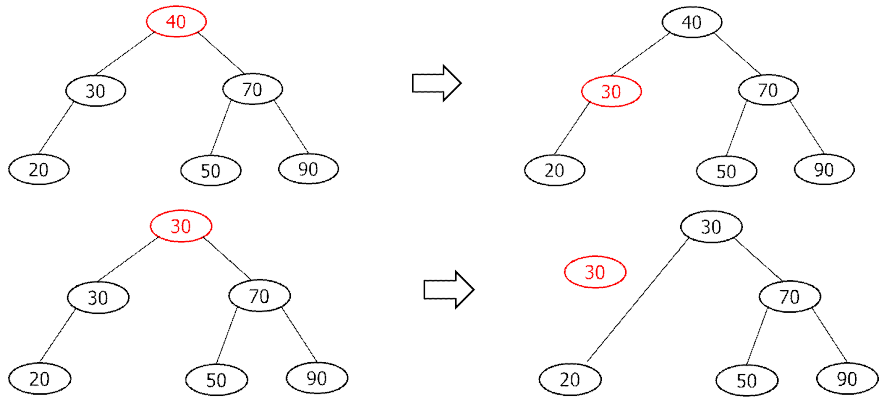
- 삭제할 키는 90이다.
- 삭제할 노드와 그 부모(70)를 발견한다. (탐색 알고리즘과 동일함)
- 삭제할 노드가 리프노드이므로 부모와 삭제할 노드를 연결하는 링크를 제거함으로써 삭제가 완료된다.
- 필요하면 삭제할 노드의 메모리를 반납한다.

- 삭제 예시2



- 삭제할 키는 30이다.
- 삭제할 노드와 그 부모(40)를 발견한다. (탐색 알고리즘과 동일함)
- 삭제할 노드가 자식을 하나만 갖는 노드이므로, 부모와 삭제할 노드의 자식(20)을 직접 연결한다. 이 때 삭제할 노드가 왼쪽 자식이므로 20도 40의 왼쪽 자식이 된다.
- 필요하면 삭제할 노드의 메모리를 반납한다.

- 삭제 예시3



- 삭제할 키는 40이다.
- 삭제할 노드와 그 부모를 발견한다. (최상위 노드이므로 부모 없음)
- 삭제할 노드가 자식을 두 개 갖는 노드이므로, 후계자 노드를 찾는다.
- 후계자는 왼쪽 서브트리에서 가장 키가 큰 노드이다. 이 경우는 30이 후계자로 설정된다.
- 후계자 노드의 키와 모든 데이터를 삭제할 노드로 복사한다.
- 이후 후계자 노드를 삭제한다. (이 예에서 후계자 노드는 자식을 하나만 갖는 노드이다. 따라서 후계자 노드의 부모와 후계자 노드의 자식 노드가 직접 연결된다.)
- 필요하면 후계자 노드의 메모리를 반납한다.

연습문제

1. 트리와 그래프의 차이점을 설명하시오..

정답 : 트리와 그래프는 모두 노드와 링크로 구성되는 비선형 자료구조이지만 그래프와 달리 트리에는 사이클이 존재하지 않는다.

해설 : 정답 참조

2. 이진 탐색 트리의 검색 연산의 최악의 시간 복잡도는?.

정답 : O(N)

해설 : 이진 탐색 트리는 트리의 균형을 유지하기 위한 회전 연산을 수행하지 않으므로 최악의 경우에 일반적인 연결 리스트와 같은 skewed tree가 될 수 있다. 이 때의 검색 연산의 시간 복잡도는 O(N)이다.

3. 중위 순회에 대해 설명하시오.

정답 : 중위 순회 방식은 최상위 노드부터 순회를 시작한다. 노드 n에 도착하면 n의 탐색을 보류하고 먼저 n의 왼쪽 서브 트리들을 전부 순회하고, 이후, 노드 n을 탐색하고, 마지막으로 노드 n의 오른쪽 서브 트리들을 전부 순회한다. 각 서브 트리들을 순회할 때도 동일한 규칙을 따른다. 즉 LNR 순서로 순회한다.

해설 : 정답 참조

정리하기

1. 트리의 개념 및 주요 용어.
2. 이진 트리의 순회 방식
3. 이진 탐색 트리의 개념 및 주요 연산

참고자료

- 파이썬과 함께하는 자료구조의 이해, 양성봉, 2021, 생능출판

다음 차시 예고

- 이진탐색트리의 구현에 대해 학습한다.