- **11. 트리**
  - **개요**
    - **학습목표**
      - **그래프의 한 종류인 트리의 특성에 대해서 이해하고 그래프에서 트리를 구분할 수 있다.**
      - **이진 트리의 정의를 이해하고 이진 트리가 가질 수 있는 최대 노드의 개수와 n개의 노드를 갖는 이진 트리의 최소 높이를 계산할 수 있다.**
      - **이진 탐색 트리가 노드를 탐색하는 방법을 이해하고 효율적 탐색이 가능한 이진 탐색 트리를 구별할 수 있다.**
      - **주어진 그래프에 대한 최소 신장 트리를 크루스칼과 프림 알고리즘으로 각각 만들 수 있다.**
    - **주요용어**
      - **트리**
      - **노드**
      - **트리의 차수**
      - **이진 트리 최대 노드 수**
      - **이진 트리 최소 높이**
      - **완전 이진 트리**
      - **경사 이진 트리**
      - **포화 이진 트리**
      - **이진 탐색 트리**
      - **신장 트리**
      - **최소 신장 트리**
      - **크루스칼 알고리즘**
      - **프림 알고리즘**
  - **기본사항**
    - **트리(tree)**
      - **트리(tree)**
        - **사이클이 없는 단순 연결 그래프**
        - **부모와 자식들 사이의 관계와 같이 계층구조를 표현하기 위한 자료구조로서, 그래프 종류 중 하나이다.**
        - **종류**
          - **사소한 트리(trivial tree)**
            - **하나의 노드로 구성된 트리**
          - **공백 트리(empty tree)**
            - **어떠한 노드도 없는 트리**
          - **포레스트(forest)**
            - **한 개 이상의 트리로 구성된 그래프**
      - **노드(node)**
        - **꼭지점이라는 용어 대신 노드(node)를 사용**
        - **각 노드는 인접합 노드의 개수를 차수로 가지는 것이 아니라 자식 노드의 개수만을 차수로 가진다.**
      - **다양한 트리의 종류**
        - **노드의 차수**
          - **이진 트리**
            - **노드의 배열 방법**
              - **힙(heap)**
              - **이진 탐색 트리**
                - **노드의 깊이**
                  - **AVL 트리**
                  - **BB 트리**
                  - **스플레이 트리**
          - **m-원 트리**
            - **차수**
              - **트라이(trie)**
              - **m - 원 탐색 트리**

- B*-트리
- B+ 트리
- 2-3 트리
- 2-3-4 트리
- 레드-블랙(red-black) 트리

예제 11-1

루트 트리(rooted tree)

루트 트리

[트리의 순환적(recursive) 정의] 루트 트리 T는 다음 조건을 만족하는 1개 이상의 노드들 (v_1, v_2, ..., v_n)의 유한집합이다.

(1) 루트(root : v_1)라 부르는 특정한 노드가 1개 존재한다.

(2) 나머지 노드들(v_2, v_3, ..., v_n)은 m개의 서로 분리된 집합 T_1, T_2, ..., T_m으로 나뉘며 T_i는 다시 루트 트리가 된다. T_1, T_2, ..., T_m은 각각 v_1의 서브트리(subtree)라고 한다.

루트 트리 예시

트리 T에 대한 루트는 A이다.

A의 서브트리는 3개로 각각 B, C, D를 루트로 하는 트리이다.

재귀적으로 B 역시 E, F, G를 서브트리로 가진다.

서브 트리 C는 더 이상의 서브트리를 가지지 않는다.

방향 트리(directed tree)

트리의 경우 상하 방향성을 가지고 있기 때문에 이를 방향 그래프로 표현할 수 있다.

기본사항(2) [274]

트리의 주요 용어와 특징

루트(root)

트리는 진입차수가 0인 노드를 루트로 가진다.

루트는 자기 자신으로 들어오는 변을 가지지 않기 때문에 진입차수는 0이다.

트리는 연결 그래프이므로 이러한 성질을 만족하는 노드는 하나만 존재한다.

노드의 차수(degree of a node)

노드의 차수는 진입차수를 제외한, 진출차수를 의미한다.

트리에서 노드의 진입차수는 0(루트) 아니면 1이기 때문이다.

자식 노드(child node) / 부모 노드(parent node)

트리의 노드 N이 서브트리를 갖는다면, 각 서브트리의 루트를 N의 자식 노드(child node)라 부른다.

N을 자식 노드의 부모노드(parent node)라 부른다.

형제 노드(sibling node)

같은 부모 노드를 갖는 노드들

리프(leaf) / 단말 노드(terminal node)

노드의 차수가 0인 노드들

내부 노드(internal node)

루트도 아니고 리프도 아닌 트리 내의 노드들

트리의 차수(degree of a tree)

한 트리 내의 각 노드 차수 가운데 최대 차수로 정의한다.

노드의 레벨(level)

트리의 루트로부터 그 노드까지의 경로길이

트리의 높이(height) = 깊이(depth)

노드의 레벨 중 가장 큰 레벨

트리의 무게(weight)

리프 노드들의 수

예제

노드의 레벨

레벨1 : B, C, D

레벨2 : E, F, G, H

레벨 3 : I, J, K

트리의 높이

- 3
- 트리의 무게
  - 7
- 주의사항
  - 호로위츠(Elis Horowiitz)와 샤니(Saratal Sahni)의 경우
    - 루트 레벨을 1로 정의하고 있어 각 노드의 레벨이 1만큼 더 큼
    - 트리의 높이도 트리 내의 노드가 가질 수 있는 최대 레벨로 정의 하고 있어 1만큼 더 큼
- 트리의 표현 방법
  - 벤 다이어그램
  - 중첩된 괄호
  - 결각
- 트리가 '닮았다'
  - 트리가 주어졌을 때, 구조는 동일하지만 노드의 데이터 내용이 다를 경우 이들 트리는 닮았다고 한다.
  - 즉, 모두 공집합이거나, 혹은 대응하는 서브트리들이 닮았다면 그 트리들은 같은 모양을 갖는다.
- 트리가 가지는 중요한 성질
  - n개의 노드를 가지는 트리는 n-1개의 변을 가진다.
  - (역도 성립한다.) n개의 노드를 가지는 연결 그래프가 n-1개의 변을 가진다면 해당 그래프는 트리이다.
  - 증명
    - 루트 노드를 제외한 모든 노드들이 단 하나의 부모를 가진다.
    - 세부 증명 [276]

2. 이진 트리 (277)
- 이진 트리(binary tree)
  - 이진 트리
    - 공집합이거나 모든 노드가 최대 2개의 서브트리를 가지는 루트 트리이다.
    - 왼쪽 서브트리(left subtree)와 오른쪽 서브트리(right subtree)로 구분되며, 각각 최대 하나씩만을 가진다.
      - 즉, 왼쪽 서브트리만 두 개를 가지거나 오른쪽 서브트리만 두 개를 가지는 노드가 존재하지 않는다.
    - 왼쪽 자식(left child)
      - 왼쪽 서브트리의 루트
    - 오른쪽 자식(right child)
      - 오른쪽 서브트리의 루트
  - 같은 이진 트리가 아닌 경우
    - (a), (c)는 같은 이진 트리가 아니다.
      - (a)는 노드 A가 왼쪽 서브트리를 가지는 경우이고, (c)는 노드 A가 오른쪽 서브트리를 가지는 경우이다.
    - (b)는 왼쪽 오른쪽이 아닌 아래로 향했으므로 이진 트리가 아니다.
  - 일반 트리와 이진 트리의 차이점
    - 일반 트리는 노드의 수가 0인 공집합이 허용되지 않지만, 이진트리는 공집합을 허용한다.
    - 이진 트리에서는 왼쪽과 오른쪽의 서브트리가 서로 다르지만 일반 트리에서는 그렇지 않다.
  - 이진 트리가 가질 수 있는 최대 노드 수
    - 이진 트리에서 레벨 i에 대해 그 레벨이 들어오는 변의 최대 수는 직전 레벨(i-1)에 있는 노드의 2배이다.
      - 레벨 0인 노드는 루트 로드로서 레벨 1인 노드는 레벨 0인 노드 수의 2배인 2개, 레벨 2인 노드는 4개 등으로 계산되므로 레벨 i인 노드의 최대 수는 2^i (i ≥ 0)이 된다.
    - 예를 들어, 높이가 2인 이진 트리는 최대 7개 노드를 가질 수 있으며, 높이가 9인 이진 트리는 최대 1,023개의 노드를 가질 수 있다.
- 완전 이진 트리(complete binary tree)
  - 완전 이진 트리
    - 높이가 h인 트리에서 레벨 0부터 h-1까지 모든 노드가 채워져 있고 레벨 h에서는 왼쪽 노드부터 차례로 채워진 이진 트리를 완전 이진 트리라 한다.
  - 완전 이진 트리는 같은 노드의 수를 갖는 트리 중 최소의 높이를 갖는다.
    - n개의 노드를 갖는 이진 트리의 최소 높이 H_min은 다음과 같다.
    - 예: n=1이면 0, n=7이면 2
  - 경사 이진 트리(skewed binary tree)

- 이진 트리가 한쪽으로 경사진 경우
- 포화 이진 트리(full binary tree)
  - 포화 이진 트리
    - 높이가 h인 트리에서 레벨 0부터 h까지 모든 노드가 채워진 이진트리
  - 예제 11-2
- 3. 이진 탐색 트리[280]
  - 이진 탐색 트리 (binary search tree)
    - 이진 탐색 트리
      - 이진 트리의 노드가 키값을 가지고 있으며 키들이 다음의 속성을 만족할 때, 이진 탐색 트리라 한다.
        - (1) 임의의 노드 N_i에 대해, N_i의 왼쪽 서브트리의 키 값들은 N_i의 키값 k_i보다 작아야 한다.
        - (2) 임의의 노드 N_i에 대해, N_i의 오른쪽 서브트리의 키값들은 N_i의 키값 k_i보다 커야 한다.
    - 키값 $k_1, k_2, ..., k_n$을 가지는 n개의 레코드들의 집합에 대한 이진 탐색 트리는 《정의 11.7》의 조건에 따라 노드 N_i가 키값 중의 하나인 k_i를 가지는 이진 트리를 말한다.
    - 키(key)
      - 키(key)는 각 노드의 식별자로서, 특정 노드에 대한 탐색은 그 키값을 이진 탐색 트리에서 찾는 것을 의미한다. 따라서 키값들은 서로 구별되는 값이어야 한다.
    - 레코드들의 집합에 대해 키를 배열하는 방법에 따라 이진 탐색 트리가 여러 개 생성될 수 있다
      - (a) A, B, C, D 순 키 삽입 (b) A, C, B, D 순 키 삽입
    - 키 탐색 알고리즘
    - 이진 탐색 트리에서 각 노드가 어떻게 서로 연결되어 있는지에 따라 키를 찾기 위해 필요한 비교 횟수가 결정되므로, 가능하면 모든 키를 빠르게 찾을 수 있도록 이진 탐색 트리를 구성하는 것이 좋다.
      - [그림 11-11] 키가 D인 레코드 찾기 (a) 트리 : 4번의 비교 (b) 트리 : 2번의 비교
    - 이진 탐색 트리 탐색 효율성 구하기[283]
      - (1) 모든 노드가 탐색되는 데 걸리는 비교 횟수의 합을 이용해 이진 탐색 트리를 서로 비교하는 것
        - 모든 노드 탐색에 소요되는 횟수비교
          - (a) 1+2+3+4 = 10(번) (b) 1+2+3+3 = 9 (번)
      - (2) (각각의 노드가 탐색될 확률이 서로 다를 수 있으므로) 확률정보까지 보정한 비교 방법
        - 그러므로 각각의 노드가 탐색될 확률이 서로 다르다면, 더 좋은 이진 탐색 트리를 만들기 위해 이러한 확률 정보를 반영해야 한다. 키 k_i가 탐색 대상이 될 확률을 p_i라 하자. 여기서 k_i(i=1, 2, ..., n)는 트리 안에 있는 키들 중의 하나이며 탐색이 성공적인 것이라 가정하면 다음과 같이 된다.
        - 따라서 이진 탐색 트리에서 모든 노드가 탐색되는 데 걸리는 비교 횟수의 기대값은 다음과 같이 볼 수 있다.
          - 여기서 c_i는 키 k_i에 도달하기 위해 요구되는 비교 횟수이다. 루트의 레벨이 0이므로 c_i는 노드 N_i의 레벨값 +1이 된다.
        - 예제[284]
          - 특정 키가 탐색의 대상이 될 확률을 아래와 같이 정의하면, A : 0.2 B : 0.4 C : 0.3 D : 0.1 [그림 11-11]의 이진 탐색 트리 (a), (b), (c), (d)에서 각각의 기대값은 다음과 같이 계산된다. (a) : 0.2 × 1 + 0.4 × 2 + 0.3 × 3 + 0.1 × 4 = 2.3 (b) : 0.2 × 1 + 0.4 × 3 + 0.3 × 2 + 0.1 × 3 = 2.3 (c) : 0.2 × 2 + 0.4 × 1 + 0.3 × 2 + 0.1 × 3 = 1.7 (d) : 0.2 × 3 + 0.4 × 2 + 0.3 × 1 + 0.1 × 2 = 1.9 계산 결과 (c)의 기대값이 가장 작으므로 (c)의 이진 탐색 트리를 가지는 것이 탐색 성능을 향상시키는 데 도움이 된다.
- 4. 트리의 활용 [284]
  - 신장 트리 (spanning tree)
    - 신장 트리
      - 그래프 G가 주어졌을 때 G의 모든 노드를 연결하고 사이클이 존재하지 않는 G의 부분 그래프를 G의 신장 트리라 한다.
      - 즉, 신장 트리는 그래프 G의 모든 노드를 포함하는 트리이다.
      - 예시
  - 최소 신장 트리(minimum spanning tree)
    - 최소 신장 트리
      - 그래프 G의 모든 변의 가중치 합을 총 가중치(total weight)라고 했을 때, G의 신장 트리 중에서 가장 작은 총 가중치를 가지는 트리를 최소 신장 트리라 한다.
    - 완전 그래프에 대한 신장 트리의 개수
      - 완전 그래프 Kn의 신장 트리의 개수는 n^(n-2)
    - 최소 신장 트리를 구하는 알고리즘
      - 크루스칼 알고리즘

- 프릴 알고리즘
- 특징
  - n개의 노드와 m개의 변을 가진 그래프에 대해서
    - 크루스칼 알고리즘은 m $\log_2$m의 복잡도로 최소 신장 트리를 구할 수 있고
    - 프림 알고리즘은 n^2의 복잡도로 최소 신장 트리를 구할 수 있다.

- (1) 크루스칼(Kruskal) 알고리즘 [286]
  - 알고리즘
    - 단계 1. 가중치의 오름차순으로 변을 정렬한다.
    - 단계 2. 가장 작은 가중치의 변부터 차례대로 트리에 추가한다. (단, 추가될 변이 사이클을 생성한다면 추가하지 않는다.)
    - 단계 3. 모든 노드가 연결될 때까지 단계 2를 반복한다.
  - 예제
    - 알고리즘 적용
    - 의사코드 [288]

```
function Kruskal(G) {
    foreach vertex v in V[G] {
        Define an elementary cluster C(v) ← {v};
    }
    Initialize a priority queue Q using the weights of edges as keys;
    tree T ← ∅;
    while T has fewer than n - 1 edges do {
        (u, v) ← Q.removeMin();
        Let C(v) be the cluster containing v;
        Let C(u) be the cluster containing u;
        if (C(v) ≠ C(u)) {
            Add edge (u, v) to T;
            Merge C(v) and C(u) into one cluster;
        }
    }
    return T;
}
```

- (2) 프림(Prim) 알고리즘 [289]
  - 알고리즘
    - 단계 1. 임의의 노드 하나를 트리에 추가한다.
    - 단계 2. 트리와 연결된 노드 중에서 트리의 노드가 아니면서 가장 작은 가중치로 연결된 노드를 추가한다.
    - 단계 3. 모든 노드가 연결될 때까지 단계 2를 반복한다.
  - 알고리즘 적용
- 예제 11-3 [290]
  - 동일한 가중치를 가지는 변에 의해서 서로 다른 최소 신장 트리가 만들어질 수 있음
  - 서로 다른 트리라고 하더라도 이들 트리의 가중치의 합은 모두 최소의 값(즉, 13으로 동일한 값)을 가지므로 이들이 최소 신장 트리임은 분명하다.