알고리즘_13~14일차 (2/20~2/21)

목차

트리

1. 트리
2. 이진트리
3. 이진 트리의 표현
4. 이진 트리의 저장
5. 연습문제
6. 이진탐색 트리
7. 힙

1. 트리

1-1. 트리의 개념

> 비선형 구조
> 원소들 간에 1:N 관계를 가지는 자료구조
> 원소들 간에 계층관계를 가지는 자료구조
> 상위 원소에서 하위 원소로 내려가면서 확장되는 트리(나무)모양의 구조

1-2. 트리의 정의

> 한 개 이상의 노드로 이루어진 유한 집합

> 다음의 조건을 만족한다.
> 노드 중 최상위 노드를 루트(root)라고 한다.
> 나머지 노드들은 n(>=0)개의 분리 집합으로 분리될 수 있다.
> 나머지 노드들은 각각 하나의 트리가 되며(재귀적 정의) 루트의 부트리(subtree)라고 한다.

1-3. 트리 용어 정리

> 노드 : 트리의 원소
> 간선(edge) : 노드를 연결하는 선 (부모 노드와 자식 노드를 연결)
> 루트 노드(root node) : 트리의 시작 노드
> 형제 노드(sibling node) : 같은 부모 노드의 자식 노드들
> 조상 노드 : 간선을 따라 루트 노드까지 이르는 경로에 있는 모든 노드들
> 서브 트리(subtree) : 부모 노드와 연결된 간선을 끊었을 때 생성되는 트리
> 자손 노드 : 서브 트리에 있는 하위 레벨의 노드들

> 차수
> 노드의 차수 : 노드에 연결된 자식 노드의 수
> 트리의 차수 : 트리에 있는 차수 중에서 가장 큰 값
> 단말 노드(리프 노드) : 차수가 0인 노드. 자식 노드가 없는 노드

> 높이
> 노드의 높이 : 루트에서 노드에 이르는 간선의 수. 노드의 레벨
> 트리의 높이 : 트리에 있는 노드의 높이 중에서 가장 큰 값. 최대 레벨

---

2. 이진 트리

> 모든 노드들이 2개의 서브트리를 갖는 특별한 형태의 트리
> 각 노드가 자식 노드를 최대 2개까지만 가질 수 있는 트리
> 왼쪽 자식 노드 (left child node)
> 오른쪽 자식 노드 (right child node)

2-1. 이진 트리 특성

> 레벨 i에서의 노드의 최대 개수는 2^i개
> 높이가 h인 이진 트리가 가질 수 있는 노드의 최소 개수는 (h + 1)개가 되며, 최대 개수는 (2^{h+1} -1) 개가 된다.

2-2. 이진 트리 종류

포화 이진 트리 (Full Binary Tree)

> 모든 레벨에 노드가 포화상태로 차 있는 이진 트리
> 높이가 h일 때, 최대의 노드 개수인 (2^{h+1} - 1)의 노드를 가진 이진 트리
> (높이 3일 때, 15개의 노드)
> 루트를 1번으로 하여 2^{h+1} - 1까지 정해진 위치에 대한 노드 번호를 가짐

완전 이진 트리(Complete Binary Tree)

> 높이가 h이고 노드 수가 n개일 때, 포화 이진 트리의 노드번호 1번부터 n번까지 빈 자리가 없는 이진 트리

편향 이진 트리(Skewed Binary Tree)

> 높이 h에 대한 최소 개수의 노드를 가지면서 한쪽 방향의 자식노드만을 가진 이진 트리
> (왼쪽 편향 이진 트리, 오른쪽 편향 이진 트리)

2-3. 이진 트리 순회

> 순회(traverasl) : 트리의 노드를 체계적으로 방문하는 것
> (트리의 각 노드를 중복되지 않게 전부 방문하는 것)

> 트리는 비선형구조이기 때문에 선형구조에서와 같이 선후 연결관계를 알 수 없다.
> 때문에 순회를 통해 알아내는 것
> 종류 : 전위순회, 중위순회, 후위순회

전위순회 (preorder traversal) : VLR

> 부모노드 방문 후, 자식노드를 좌우순서로 방문한다.

전위순회 수행방법

1. 현재 노드 n을 방문하여 처리한다 : V
2. 현재 노드 n의 왼쪽 서브트리로 이동한다 : L
3. 현재 노드 n의 오른쪽 서브트리로 이동한다 : R

```
def preorder_travers(T):
    if T:           # T is not None
        visit(T)    # print(T.item)
        preorder_traverse(T.left)
        preorder_traverse(T.right)
```

중위순회 (inorder traversal) : LVR

> 왼쪽 자식노드, 부모노드, 오른쪽 자식노드 순으로 방문한다.

중위순회 수행방법

1. 현재 노드 n의 왼쪽 서브트리로 이동한다 : L
2. 현재 노드 n을 방문하여 처리한다 : V
3. 현재 노드 n의 오른쪽 서브트리로 이동한다 : R

```
def inorder_travers(T):
    if T:           # T is not None
        preorder_traverse(T.left)
        visit(T)    # print(T.item)
        preorder_traverse(T.right)
```

후위순회 (pstorder traversal) : LRV

> 자식노드를 좌우순서로 방문한 후, 부모노드를 방문한다.

후위순회 수행방법

1. 현재 노드 n의 왼쪽 서브트리로 이동한다 : L
2. 현재 노드 n의 오른쪽 서브트리로 이동한다 : R
3. 현재 노드 n을 방문하여 처리한다 : V

```
def posorder_travers(T):
    if T:           # T is not None
        preorder_traverse(T.left)
        preorder_traverse(T.right)
        visit(T)    # print(T.item)
```

---

3. 이진 트리의 표현

3-1. 배열을 이용한 이진 트리의 표현

> 이진 트리에 각 노드 번호를 다음과 같이 부여
> 루트의 번호를 1로 함
> 레벨 n에 있는 노드에 대하여 왼쪽부터 오른쪽으로 2^n부터 2^{n+1}-1까지 번호를 차례로 부여
> 노드 번호를 배열의 인덱스로 사용
> 높이가 h인 이진 트리를 위한 배열의 크기는? 2^{h+1}-1

노드 번호의 성질

> 노드 번호가 i인 노드의 부모 노드 번호 : i/2
> 노드 번호가 i인 노드의 왼쪽 자식 노드 번호 : 2 x i
> 노드 번호가 i인 노드의 오른쪽 자식 노드 번호 : 2 x i + 1 레벨 n의 노드 번호 시작 번호는? 2^n

단점

> 편향 이진 트리의 경우 사용하지 않는 배열원소에 대한 메모리 공간 낭비 발생
> 트리 중간에 새로운 노드를 삽입하거나 기존의 노드를 삭제할 경우 배열의 크기 변경이 어려워 비효율적

3-2. 연결 리스트를 이용한 이진 트리의 표현

> 배열을 이용한 이진 트리의 표현 단점을 보완하기 위해 사용

> 연결 자료구조를 이용한 이진트리의 표현
> 이진 트리의 모든 노드는 최대 2개의 자식 노드를 가지므로 일정한 구조의 단순 연결 리스트 노드를 사용하여 구현

3-3. 수식 트리(Expression Binary Tree)

> 수식을 표현하는 이진 트리 (수식 이진 트리)
> 연산자는 루트 노드 or 가지 노드
> 피연산자는 모두 잎 노드

수식 트리의 순회

> 중위 순회 : A/B * C * D + E (식의 중위 표기법)
> 후위 순회 : A B / C * D * E + (식의 후위 표기법)
> 전위 순회 : +**/A B C D E (식의 전위 표기법)

4. [참고] 이진 트리의 저장

> 교재 31쪽부터 33쪽 사진 넣기
> @@

---

5. 연습문제

6. 이진 탐색 트리

> 탐색작업을 효율적으로 하기 위한 자료구조
> 모든 원소는 서로 다른 유일한 키를 갖는다.
> key(왼쪽 서브트리) < key(루트 노드) < key(오른쪽 서브트리)
> 왼쪽 서브트리와 오른쪽 서브트리도 이진 탐색 트리다.
> 중위 순회하면 오름차순으로 정렬된 값을 얻을 수 있다.

6-1. 탐색연산

> 루트에서 시작

> 탐색할 키 값 x를 루트노드의 키 값과 비교
> (키 값 x = 루트노드의 키 값) : 원하는 원소를 찾았으므로 탐색연산 성공
> (키 값 x < 루트노드의 키 값) : 루트노드의 왼쪽 서브트리에 대해서 탐색연산 수행
> (키 값 x > 루트노드의 키 값) : 루트노드의 오른쪽 서브트리에 대해서 탐색연산 수행

> 서브트리에 대해서 순환적으로 탐색연산을 반복한다.

6-2. 삽입연산

> 1.먼저 탐색연산을 수행
> 삽입할 원소와 같은 원소가 트리에 있으면 삽입할 수 없으므로, 같은 원소가 트리에 있는지 탐색하여 확인
> 탐색에서 탐색 실패가 결정되는 위치가 삽입 위치

> 2.탐색 실패한 위치에 원소를 삽입

6-3. 이진탐색 트리의 성능

> 탐색(searching), 삽입(insertion), 삭제(deletion) 시간은 트리의 높이만큼 시간이 걸림
> O(h), h : BST의 깊이(height)

> 평균의 경우
> 이진 트리가 균형적으로 생성되어 있으면 O(log n)

> 최악의 경우
> 한쪽으로 치우진 경사 이진트리인 경우 : O(n)
> 순차탐색과 시간복잡도가 같다.

검색 알고리즘의 비교

> 배열에서의 순차 검색 : O(N)

> 정렬된 배열에서의 순차 검색 : O(N)

> 정렬된 배열에서의 이진탐색 : O(log N)
> (고정배열 크기와 삽입, 삭제 시 추가 연산 필요)

> 이진 탐색에서의 평균 : O(log N)
> 최악의 경우 : O(n)
> 새로운 원소를 삽입할 때 삽입시간을 줄인다.
> 평균과 최악의 시간이 같다. : O(log n)
> 완전 이진트리 또는 균형트리로 바꿀 수 있다면 최악의 경우를 없앨 수 있다.

> 해쉬 검색 : O(1)
> 추가 저장공간이 필요

---

7. 힙 (heap)

> 완전 이진 트리로 구현된 자료구조로소, 키 값이 가장 크거나 작은 노드를 찾기에 적합한 자료구조
> (완전 이진 트리에 있는 노드 중에서 키 값이 가장 크거나 작은 노드를 찾기 위해서 만든 자료구조)

> 최대 힙 (max heap)
> 키 값이 가장 큰 노드를 찾기 위한 완전 이진 트리
> 부모노드의 키 값 > 자식노드의 키 값
> 루트 노드 : 키 값이 가장 큰 노드

> 최소 힙 (min heap)
> 키 값이 가장 작은 노드를 찾기 위한 완전 이진 트리
> 부모노드의 키 값 < 자식노드의 키 값
> 루트 노드 : 키 값이 가장 작은 노드

7-1. 힙 삽입 연산

교재 53, 54 사진 넣기 @@

7-2. 힙 삭제 연산

교재 56쪽 사진 @@@@@@@@@@@@@@@@@@@@@@@@@
