💯 자료구조와실습 기말고사 대비

컴퓨터공학전공 2020112736 안성현

7. Trees

- Binary tree : 노드의 유한집합, 루트, 왼쪽 서브트리, 오른쪽 서브트리 두 개로 분리된 이진 트리

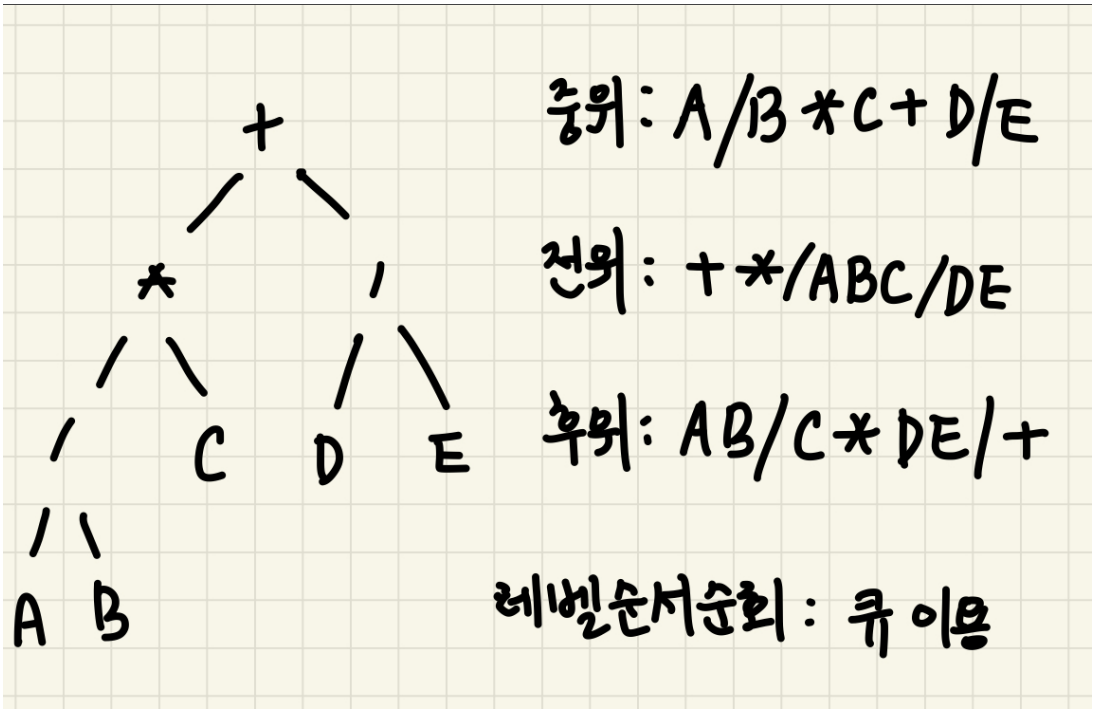
▼ Properties of Binary tree

  - 간선의 수 : n-1 (노드 : n)
  - 최소 노드 수 : n = h
  - 최대 노드 수 : n = 2^h - 1
  - 최대 높이 : h = n
  - 최소 높이 : h = log2(n+1)

- 포화이진트리(full BT) : n = 2^h - 1
- 완전이진트리(complete BT) : full BT에다가 다음 레벨에는 왼쪽부터 채운거
- Array Representations of BT
  - 부모의 위치 : i / 2
  - 왼쪽 자식 : 2i
  - 오른쪽 자식 : 2i + 1

  but, 비효율적인 메모리 사용, 삽입, 삭제 → complete BT에만 사용
- Linked Representation of BT → Doubly Linked List 생각
- BT Traversal (순회) → tree 모양으로 생각
  - 중위 : LVR
  - 전위 : VLR

- 후위 : LRV



- BT compare → 비교할 거 두개 매개 변수로 받아서 같은지 체크, 재귀함수로 자식까지
- BT copy → temp 만들어서 동적 할당 받고 값 설정, 재귀함수로 자식까지

- Binary Search Tree (이진탐색트리) : Complete BT + 크기 순 배치
  - 탐색 : 그냥 하면 됨
  - 삽입 : 1. 들어갈 곳 찾기(parent 정해주면서), 2. temp 동적 할당 받고 값 설정해주고 설정해둔 parent에 값 크기에 따라 연결
  - 삭제 : 삭제하고 왼쪽에 가장 큰 값 or 오른쪽에 가장 작은 값 대체
- Performance of BST
  - h = O(log2n) (avg), O(n) (worst)
  - search, insert, delete : O(log2n) (avg), O(n) (worst)

- Heap → 각 노드가 자식의 노드보다 작지 않음, Complete BT

  ▼ Representation of Heap

        - 부모 : i / 2
        - 왼쪽 자식 : 2i
        - 오른쪽 자식 : 2i + 1
        - h = log2(n+1)
    - 삽입 → 들어갈 자리에 넣고 값 바꾸면서 자리 맞추기
    - 삭제 → 맨 위에꺼 삭제하고 자리 맞추기

8. Graphs

- 그래프 G = (V, E) → V(G) : Set of Vertex, E(G) : Set of Edge
- 무방향 (양방향) : (), 단방향 : <> 순서 O

- 무방향 그래프
    - v의 차수 : v에 부속한 간선의 수
    - 전체 v의 간선의 수의 합 : 간선의 수 * 2
- 방향 그래프
    - in-degree, out-degree → in, out으로 생각

- 완전 그래프 : 간선의 수가 최대인 그래프
    - 무방향 → n(n-1)/ 2, 방향 → n(n-1)
- 인접행렬 : 갈 수 있으면 1
    - 무방향 : 대칭, 방향 : 비대칭
- 인접리스트 : n개의 연결리스트
    - 무방향 : 2e개의 list node, 방향은 e개의 list node (e : edge) → 무방향은 1개의 edge에 node 2개

- 그래프 탐색 : 그래프의 한 정점에서 시작하여 연결된 모든 정점들을 차례로 방문
    - DFS(Depth First Search) : 인접 정점따라 계속 진행, 스택 (깊게깊게)

    - BFS(Breadth First Search) : 인접 리스트에 있는 모든 정점 방문, 큐 (넓게넓게)
- 신장트리 : 모든 정점을 포함하는 트리 (정점은 모두 포함, 간선은 일부) → DFS, BFS
- 최소 비용 신장트리 : 간선의 가중치 합이 최소인 신장트리 → Greedy Algorithms
    - Kruskal, Prim
- 단일 출발점, 모든 도착지 → Dijkstra Algorithm
- 모든 쌍의 최단경로 : cost 구하고 (A -1), 0 ~ n-1까지 경유해서 가는 최단경로 (A 0~n-1), A n-1이 최단경로 (마지막)
    - allcost algorithm : 경유해서 가는 거리 < 현재 최단거리 → 업데이트
- 이행적 폐쇄 행렬 : 경우해서라도 가는 길 있으면 1
    - allcost algorithm에서 가중치 대신 연결성

9. Sortings

- Bubble sort : 앞뒤로 비교 반복 O(n^2) (worst) → 1 ~ n 의 합 (모두 비교)
- Insertion sort : 앞에서부터 하나씩 뽑아서 삽입해서 정렬 O(n^2) (worst), O(n) (best)
- Quick sort : 피벗으로 리스트 분할 (partition 함수), 퀵정렬 재귀 두번 (앞뒤) O(nlogn) (avg), O(n^2) (worst) (하나와 여러개로 계속 분할)
- Merge sort : 분할 계속 하고 합병하면서 정렬 O(nlogn) (worst) (계속 같은 크기 분할)
- Heap sort : max heap에서 계속 deletemax → heap 구성, 전체에 대해 맨 위로 올리고 정렬
- Radix sort : 자릿수 따라 정렬 → 인접 리스트 처럼 정렬? (비교 X) 나눌 수 : d, 전체 수 : n, O(dn)

10. Hashings

- Symbol table : set of <key, value>

  ▼ ADT of Symbol table

    - Isin(key) : 특정 key가 테이블 내에 있는지 확인
    - Insert(key, value) : key, value를 테이블에 추가
    - delete(key) : 특정 key, value를 테이블에서 삭제

    - search(key) : 특정 key에 대한 value 반환

- Hashing : key값 저장 X, 해시 함수를 통해 key값으로 바로 저장위치 계산 O(1) gooood, O(n) (worst)
- Hash table : bucket → 담을 해시 값 개수?, slot → 각 해시 값 당 담을 수 있는 수
- 동의어, 충돌 → 다른 key가 같은 bucket에 ex> char, celi
- 오버플로우 : slot 꽉 찼는데 더 들어가려고 할 때

▼ Time Complexity of Hash table

- 삽입, 삭제, 검색 : O(1) (overflow X)
- 적재 밀도 : n/(s*b) 전체 공간 slot * bucket 대비 n

- Hash Function
    - 계산 간단, 충돌 적어야 함 (key가 bucket에 고르게 퍼져야함)
    - 균일 해싱 함수 (uniform hash function) : 1/b (전체 bucket에 균일하게)
    - 제산함수(division) : modular 연산 사용 x % M (table size) 0 ~ M - 1 bucket
    - 중간제곱함수(mid-square) : 값 제곱하고 중간의 r 비트 선택 0 ~ 2^r - 1 bucket
    - 접지(folding) : r개로 쪼개서 더함 0 ~ 2^r - 1 bucket
- Overflow handling
    - 선형 개방 주소법 : 선형 → 꽉 차면 바로 다음, 이차 → i^2 다음, 재해싱 → 새로운 해시 함수
    - 각 bucket을 synonym의 리스트로 → 충돌되면 linked list 연결해서 저장