자료구조는 선형과 비선형

선형 : 배열 리스트 스택 큐 비선형 : 트리 그래프

선형자료구조 : 하나의 자료 뒤에 하나의 자료.

! 랜덤접근가능 : 모든 자료에 O(1)로 접근이 보장되는 자료구조.

배열(array) : 정해진 크기의 메모리 먼저 할당받아 사용. 기억장소 추가 힘듬 데이터 삭제시 빈공간생겨 메모리 낭비. 원소의 인덱스값 알고있으면 O(1)에 바로 접근 가능. 논리적 저장순서와 물리적 저장순서가 일치해서 인덱스로 원소에 접근. 인덱싱만 O(1). 검색삽입삭제는 O(n)

해시 : 맨아래에 작성.

! 랜덤접근 불가능 : 모든자료에 O(1)로 접근 보장 x

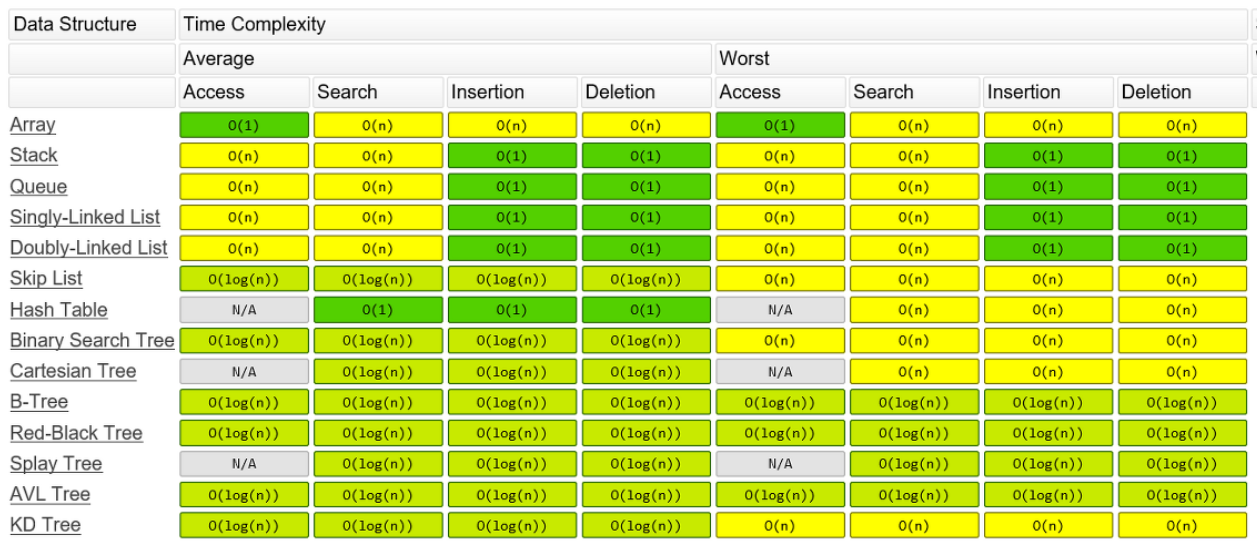
리스트 : 자료가 추가될 때 마다 메모리 할당. 논리적 물리적 위치 다름. 배열보다 소요시간큼.

스택 : LIFO. 후입선출. push. Pop.

큐 : FIFO. 선입선출. Enqueue Dequeue. 리스트와 배열로 구현가능. 배열은 보통 순환큐로 구현.

리스트스택큐 다 인덱싱 검색은 O(n). 삽입삭제는 O(1).

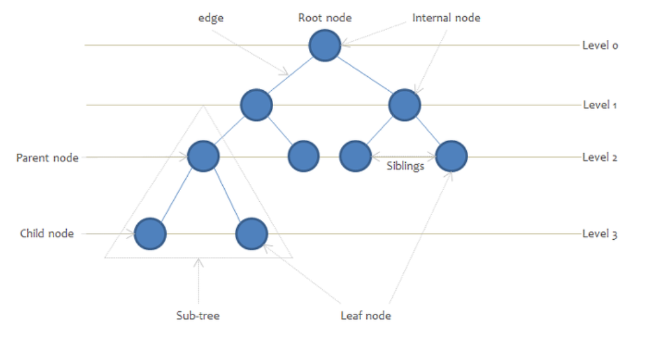
데크 : 스택과 큐의 융합형태. 양쪽 끝에서 삽입 삭제 모두 가능.



비선형 자료구조 : 하나의 자료 뒤에 여러 개 자료 존재할 수 있음.

트리 : 루트노드 부모자식형제노드. 단말(리프)노드-자식없는노드 내부(가지)노드 – 자식있는 노드

깊이 높이. 차수(degree)-자식개수. 모든 노드는 하나의 부모노드만 가짐. 노드가n개면 엣지는 n-1개



이진트리: 최대 2개의 자식. 정이진트리: 자식이 0또는2

완전이진트리 : 루트>왼쪽자식>오른쪽자식노드순으로 데이터 차례대로 삽입.

이진탐색트리 : 이진탐색이 동작할수있도록 고안된 자료구조.

왼쪽자식노드 < 부모노드 < 오른쪽자식노드. 데이터 탐색시 최악의경우 O(log n) 소요.

트리의 순회 : 한번씩 방문해 노드를 순회

전위순회 : 루트>좌>우 중위순회 : 좌>루트>우 후위순회 : 좌>우>루트

트리는 효율적 검색속도. 계층적 저장.

그래프(g) : 정점(vertex)과 간선(edge)의 집합. G=(v,e)

무방향(화살표x): 간선은 (정점, 정점), 방향(화살표o): 간선은 <정점, 정점>, 가중(숫자) 그래프

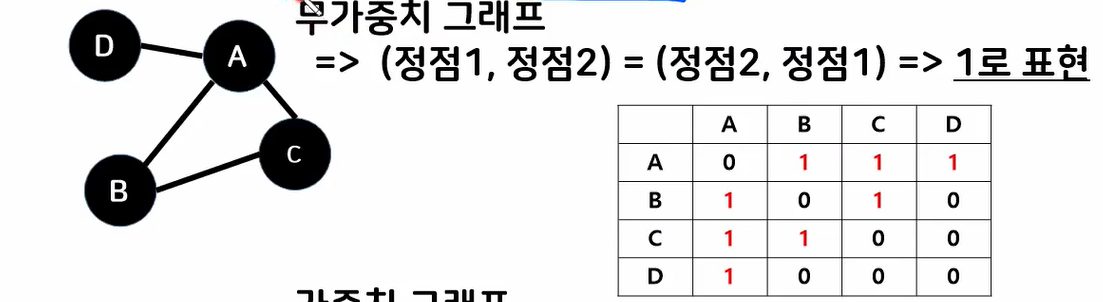
모든 정점이 다 연결되어 최대 간선 수 완전그래프. 아니면 부분그래프.

무방향그래프의 최대 간선 수 = k(k-1)/2. 방향그래프의 최대 간선 수 = k(k-1)

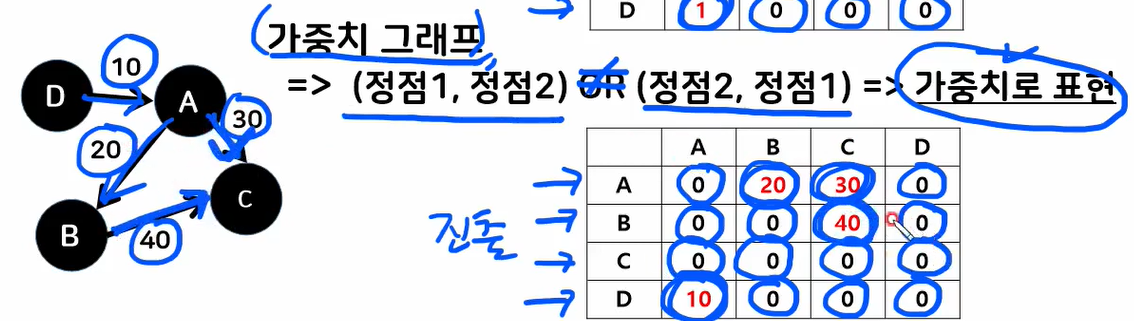
정점간에 다양한 경로와 루프가 가능. 사이클(a-b-c-a) 루프(a무한반복)

인접: 정점과 정점이 연결되어있을 때.

인접 행렬 표현 : 그래프 인접 관계를 행렬(표)로 표현. 무가중은 인접시 1, 아니면 0. 가중은 숫자



방향그래프는 시작(꼬리)점이 행방향에서 진출.



**그래프 순회** : 그래프의 모든 정점 순회하는 과정. 깊이우선탐색(DFS), 너비우선탐색(BFS)

깊이 우선 탐색(DFS) : 더 갈길없을때까지 쭉 파고 들어가고 끝이면 다음 경로. 스택이나 재귀로구현

너비우선탐색(BFS): 꼼꼼하게 좌우 살피며 다니자. 깊이 1인정점 다 하고 2인정점돌고 쭉. 큐로 구현

다익스트라알고리즘: gps. 최단경로탐색. 네비게이션. 거쳐온 최단경로 더하고 비교해서 선정.

해싱: 임의의 길이 값을 해시함수를 이용해 고정된 크기의 값으로 바꾸는 것

해시 테이블 : 해시함수를 사용해 변환한 값을 인덱스(색인)로 삼아 키와 데이터(value)를 저장하는 자료구조. 기본연산으로 탐색 삽입 삭제 가능. 충돌 하지 않으면 **검색시간이 O(1)**

데이터를 담을 테이블을 미리 크게 확보한 후 입력받은 데이터를 해시해 테이블 내의 주소를 계산하고 이 주소에 데이터를 담는 것. 리소스를 포기하고 속도를 취함.



해시함수는 나눠서 나머지쓰거나 자릿수접기등 다양한 방법 사용.

충돌 : 같은 인덱스에 여러 밸류 쌓이면 충돌남. **최악의경우 O(n).**

충돌대처 : 체이닝-충돌나는인덱스에 링크드리스트 삽입 리니어프로빙-데이터를 빈인덱스에 넣음

리사이징- 테이블 자체를 늘려버림

우선순위큐 : 큐에 우선순위개념도입. 우선순위 높은게 먼저나감. 힙으로 구현하는게 효율적

힙: 완전이진트리. 우선순위큐 위해 만들어짐. 힙사용시 삽입삭제에 O(log n)

항상 루트노드를 제거. 최소힙-루트노드가 가장작고 우선적으로제거. 최대힙-반대