9 월 7 일(수)

알고리즘

- 문제해결을 위한 절차/방법
- 어떠한 문제를 해결하기 위한 여러 동작들의 모음

자료구조 (주기억 장치에 일어나는 일)

- 자료를 효율적으로 이용할 수 있는 방법론
- 데이터를 구조적으로 표현하는 방식
- 놀고 있는 장소를 활용
- 한정된 공간을 어떻게 효율적으로 사용할까

알고리즘과 자료구조는 뗄수 없는 사이

- 적절한 모양의 블럭(자료구조)을 요리조리 돌리고 옮겨(알고리즘)서 클리어!

자료구조

- **원시구조**
  정수,실수,문자
  0101 하나의 자체/ 한공간에 하나/

- **선형구조**
  배열, 연결리스트. 스택, 큐, 덱

- **배열**
  원시자료의 묶음
  양수의 시작이 0 이기 때문에 0 부터 시작
  물리적으로 데이터들을 분리하고 데이터 하나당 크기가 같다.
  배열의 문제: 추가와 삽입 삭제는 어렵다(예:신발장이 물리적)
  데이터가 낭비 될 수 있다.

- **단순 연결 리스트(Singly Linked List)**
  배열의 문제점을 해결한다.
  다음친구가 어디있는지 알고있다.(데이터 하나당 주소를 부여한다(논리적))
  추가 삽입 삭제가 가능하고 자료형에 대해서 자유롭다.

문제점
- Head 노드를 참조하는 주소를 잃어버릴 경우 데이터 전체를 못 쓰게 되는 단점이 있다. 다음 노드를 참조하는 주소 중 하나가 잘못되는 경우에도 체인이 끊어진 것마냥 거기부터 뒤쪽 자료들을 유실한다. 따라서 안정적인 자료구조는 아니다.

- **이중 연결 리스트 (Doubly Linked List)**
  다음 노드의 참조뿐만 아니라 이전 노드의 참조도 같이 가리키게 하면 이중 연결 리스트가 된다. 뒤로 탐색하는 게 빠르다는 단순한 장점 이외에도 한 가지 장점이 더 있는데, 단순 연결 리스트는 현재 가리키고 있는 노드를 삭제하는 게 한 번에 안 되고 O(n)이 될 수밖에 없는데 비해[4]이중 연결 리스트에서 현재 노드를 삭제하는 것은 훨씬 간단하다.

  문제점
  관리해야 할 참조가 두 개나 있기 때문에 삽입이나 정렬의 경우 작업량이 더 많고 자료구조의 크기가 약간 더 커진다.

- **원형 연결 리스트 (Circular linked list)**
  단순 연결 리스트에서 마지막 원소가 널 대신 처음 원소를 가리키게 하면 원형 연결 리스트가 된다. 이와 비슷하게, 이중 연결 리스트의 처음과 끝을 서로 이으면 이중 원형 연결 리스트를 만들 수 있다. 스트림 버퍼의 구현에 많이 사용한다. 이미 할당된 메모리 공간을 삭제하고 재할당하는 부담이 없기 때문에 큐를 구현하는 데에도 적합하다.

- **청크 리스트(Chunked List)**
  배열과 리스트의 장점을 합친 것. 리스트의 멤버가 배열이다. CPU 에 캐시 기능이 있는 경우 지역성 Locality 이 떨어지는 연결 리스트는 심각한 성능 저하를 불러온다. 이를 보완하기 위해 리스트의 멤버를 레코드의 배열로 하는 것이다. 이 청크 리스트의 발전형이 바로 B+tree 다.

- 리스트는 컴퓨터의 속도가 워낙 빨라져 속도문제를 느낄수 없으므로 언어에서 자주 채택한다.

자료구조 유형 (리스트를 넣고 빼기 규칙)

- **스택(stack) (LIFO - Last In First Out)**
  먼저 넣은데이터를 나중에 꺼내는 -> 나중에 넣은걸 일찍꺼냄
  PUSH POP

- **큐(Queue)(FIFO - First In First Out)**
  먼저들어온게 먼저 나감(대기열을 사용 할 때)
  PUT GET

- **덱(Dequeue)**
  스택과 큐의 합
  PUSH POP

트리(tree)

- 리스트의 변화
- 마인드맵(나무) 가계도 조직도 처럼 리스트
- 데이터간의 상하구조 관계에 있다, 종속관계 표현
- root node child node sibling node
- sibling node 연결 x
- 비선형구조(하나의 결과만이 있는것이 아니다.)

트리순회

- 트리 구조에서 각각의 노드를 정확히 한 번만, 체계적인 방법으로 방문하는 과정을 말한다. 이는 노드를 방문하는 순서에 따라 분류된다.

- **전위 순회(preorder)**
  다음과 같은 방법으로 진행한다. 루트 노드에서 시작해서,
  1. 노드를 방문한다.
  2. 왼쪽 서브 트리를 전위 순회한다.
  3. 오른쪽 서브 트리를 전위 순회한다.
  전위 순회는 깊이 우선 순회(depth-first traversal)라고도 한다.

- **중위 순회(Inorder Traversal)**
  다음의 순서로 진행된다.
  1. 왼쪽 서브 트리를 중위 순회한다.
  2 .노드를 방문한다.
  3. 오른쪽 서브 트리를 중위 순회한다.
  중위 순회는 대칭 순회(symmetric traversal)라고도 한다음과 같은 방법으로 진행한다.

- **후위 순회(postorder)**
  다음의 순서로 진행된다.
  1. 왼쪽 서브 트리를 후위 순회한다.
  2. 오른쪽 서브 트리를 후위 순회한다.
  3. 노드를 방문한다

그래프(Graph)

- 연결관계
- 데이터들의 관계
- 페이스북 친구 추천 알수도있는 사람
- 상하관계가 없다 모두가 root node 가 따로 정해져 있지 않다.

알고리즘

- 일을 처리하는 방법 (어떠한 상황과 데이터를 가지고 있는데 어떻게 처리할래?)
- 대표적 알고리즘 – 정렬 , 탐색, 재귀 등

정렬 알고리즘

- 분포도와 가지고 있는 자료의 양, 형태에 따라 알고리즘이 다름
  상황에 따른 알고리즘 선택

- **선택정렬(selection sort)**
  전체를 읽고 가장 작은것을 앞으로 반복 왼쪽(작은것)으로 보낸다.

  big O 표기법: $O(n^2)$

- **버블정렬(bubble sort)**
  처음부터~n 번째 앞뒤로 비교하고를 반복 오른쪽(큰거)으로 보낸다.

  big O 표기법: $O(n^2)$

- **삽입정렬(insertion sort)**
  앞뒤 중앙의 자리에 삽입해줌
  앞뒤 중간값에 비교한후 맞는부분에 넣는다. 왼쪽으로 보냄

k 번째 원소를 1 부터 k-1 까지와 비교해 적절한 위치에 끼워넣고 그 뒤의 자료를 한 칸씩 뒤로 밀어내는 방식으로, 평균적으론 O(n2)중 빠른 편이나 자료구조에 따라선 뒤로 밀어내는데 걸리는 시간이 크며, 작은 게 뒤쪽에 몰려있으면(내림차순의 경우 큰 게 뒤쪽에 몰려있으면) 효율성이 떨어진다.

big O 표기법: $O(n^2)$

- **병합정렬(Merge sort)**
  반을 나누고 정렬하고 후에 합친다.
  개발자는 존 폰 노이만으로 원소 개수가 1 또는 0 이 될 때까지 두 부분으로 자른 뒤 자른 순서의 역순으로 크기를 비교해 병합해 나간다. 병합된 부분 안은 이미 정렬되어 있으므로 전부 비교하지 않아도 제자리를 찾을 수 있다. 대표적인 분할 정복 알고리즘으로 존 폰 노이만의 천재성을 엿볼 수 있는 알고리즘이다.
  성능은 아래의 퀵정렬보다 전반적으로 뒤떨어지고, 데이터 크기만한 메모리가 더 필요하지만[17] 최대의 장점은 데이터의 상태에 별 영향을 받지 않는다는 점과 stable sort 라는 점이다.

  big O 표기법: 평균적으로 O(n log n)

- **퀵정렬(Quick Sort)**
  중심점(기준)을 잡아서 작은건 왼쪽 그리고 큰건 오른쪽 (평균적으로 가장빠르다)
  절반이 아니고 중간위치 또는 평균을 대강 뽑음 기준보다 큰놈 오른쪽 작은쪽 왼쪽

  big O 표기법: $O(n^2)$

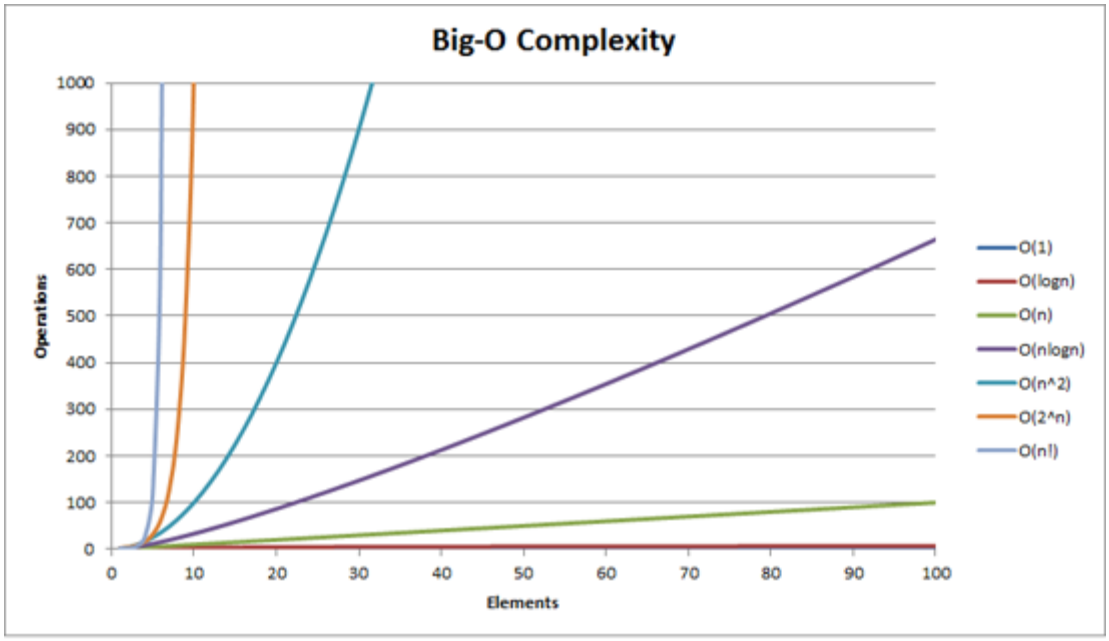
- **힙정렬(Heap sort)**
  - 원소들을 전부 힙에 삽입한다
  - 힙의 루트에 있는 값은 남은 수들 중에서 최소값(혹은 최대값)을 가지므로 루트를 출력하고 힙에서 제거한다.
  - 힙이 빌 때까지 2 의 과정을 반복한다.

  big O 표기법: O(n log n)

시간복잡도 (내가 만든 알고리즘의 효율성 테스트)

알고리즘이 실행되는데 소요되는 시간분석

- **점근 표기법(대문자 O 표기법) Big O**
  최대시간의 예측



- **빅세타표기법(최소)(Theta)**
  $\Theta(n^2)$

- **빅오메가 표기법(평균)(Big-Omega)**
  $\Omega\ (n^2)$

공간복잡도 (메모리를 차지하는 효율)

- 기존배열을 사용하느냐 아니면 새로운 배열을 만드는가
- 배열을 잃게 되고 얻게되는게 무엇일까
- 배열인가 리스트인가
- 얼마나 많은 주기억장치 메모리를 많이 차지 하는가

데이터베이스 (대부분 보조기억장치)

- 여러 사람에 의해 공유되어 사용될 목적으로 통합하여 관리되는 데이터의 집합
- 통합된 정보들을 저장하여 운영할 수 있는 공용 데이터들의 묶음
- 모두를 위한 데이터체계 집합

고객정보
열람권한
분류와 맞지 않음
데이터의 중복
철수와의 관계 바지와 원피스 연결 (관계 설정)
하나하나의 데이터 묶음을 테이블이라 부른다

DBMS(DataBase Management System)(설계, 조작, 관리)

- DBMS <-DB 와 다른거다
- DataBase 에 접근할 수 있는 기능을 제공하는 소프트웨어
- 즉, 데이터베이스계의 운영체제
- MySQL, PostgreSQL, SQLite, MariaDB,...

- SQL(Structured Query Language)
  DBMS 를 통해 데이터를 관리하기 위한 구조화된 질의문(왜 질의문이냐면 질문을 통한 대답)을 작성하기 위한 언어
  (추가, 삭제 명령을 SQL 을 통해 한다)
  관계형 데이터베이스 관리 시스템에서 사용

- NoSQL
  SQL 을 쓰지않아도 되는 관계형이아닌 DBMS
  사용자에게 쉽게 접근

과제 내용

- HTTP 와 HTTPS 는 무엇이며 그 차이는?
- 국내에 공인인증서가 생긴 배경과 그 위험성은?

학습링크

자료구조
https://goo.gl/fO7Vo, https://goo.gl/H1CKb0

알고리즘
https://goo.gl/GRz6tA, https://goo.gl/qdkZIF

시간복잡도
https://goo.gl/jQi2OF, https://goo.gl/rszprb, https://goo.gl/BmisjF

데이터베이스
https://goo.gl/tjPqqq, https://goo.gl/qNF8XH