씨앗

all

Week 4

Date : 23.05.02

Made by 은재민

참고 : pf.최영규-파이썬 알고리즘

오늘의 진도

1. 알고리즘의 복잡도
2. 알고리즘의 유형
3. 시간 복잡도
4. 빅오 표기법
5. 자료구조
6. 알고리즘 설계
7. 설계 예시

경고?

- 오늘 내용은 이해하기 어려울 수 있습니다.
- 평소보다 강의 시간이 길어질 수 있습니다.
- 이해가 안가는 내용은 언제든지 말씀해주세요.

알고리즘의 정의

알고리즘 : 주어진 문제를 해결하기 위한 단계적인 절차
알고리즘의 동작을 이해하려면 , 컴퓨터적인 사고 방식이 필요

C언어, JAVA 같은 프로그래밍 언어, 자연어, 기호, 유사코드 등으로 표현.
어떤 방법으로 표현했는 지는 상관없음!

알고리즘은 0개 이상의 입력, 1개 이상의 출력을 가져야하며
명확해야하고, 유한해야하고, 유효해야 한다.

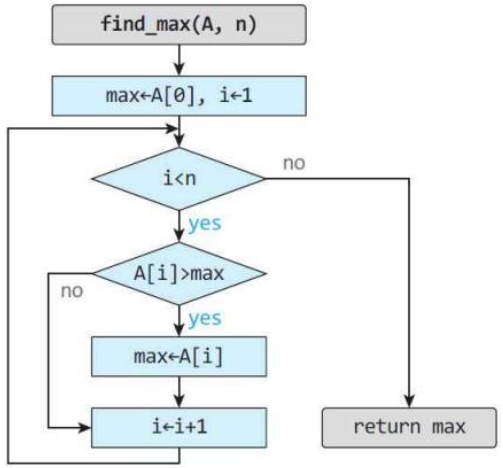
알고리즘의 표현

① 영어나 한국어와 같은 자연어를 사용하는 방법

find_max(A)

1. 리스트 A의 첫 번째 항목을 변수 max에 복사한다.
2. 리스트 A의 다음 항목들을 차례대로 max와 비교하여, max보다 더 크면 그 값을 max로 복사한다.
3. 배열 A의 모든 요소를 비교했으면 max를 반환한다.

② 흐름도(flowchart)로 표시하는 방법



③ 유사 코드(pseudo-code)로 기술하는 방법

```
01  find_max( A )
02        max←A[0]                    // ← 는 대입 연산을 의미함
03        for i←1 to size(A) do
04           if A[i]>max then
05              max←A[i]
06        return max
```

④ 특정한 프로그래밍 언어(예: C언어)

```
01  int find_max(int A[], int n) {     // 배열의 길이도 전달해야 함
02        int i, tmp=A[0];              // 사용할 변수를 미리 선언해야 함
03        for( i=1 ; i<n ; i++ ) {      // 블록의 시작을 '{'로 표시함
04           if( A[i] > tmp ) {
05              tmp = A[i];            // =를 대입 연산자로 사용하고,
06           }                         // 비교 연산자로는 ==를 사용함
07        }                            // 중괄호 { }를 맞추어 주어야 한다.
08        return tmp;
09  }
```

같은 동작을 하는 다른 알고리즘

문제 : 두 정수 A, B의 최대 공약수는 ?

아이디어 1.

A의 약수를 모두 구하고, B의 약수를 모두 구해서
약수들끼리 비교

아이디어 2.

유클리드 공제법을 사용

결론!

1. 어떤 결과를 얻기 위한 해결법은 여러가지가 있다.

2. 어떤 방법을 사용하냐에 따라 필요한 자원은 천차만별이다.

3. 절대적으로 우수한 알고리즘은 없다.
-> 어떤 알고리즘도 이게 더 좋다! 라고 단정지을 수 없다.

4. 상황에 따라 적절한 방법을 떠올리고 구현하는 것이 가장 중요!

문제의 유형

1. 정렬
2. 탐색 -> 어떤 자료구조에서 어떻게 값을 가져올 것인가?
3. 문자열
4. 그래프
5. 기하학
6. 조합문제
7. 기타 등등

정렬과 탐색

데이터가 매번 무작위로 흩어져 있다면?

-> 매번 데이터 전체를 뒤져봐야 함.

=> 효율성이 떨어지게 됨.

좋은 탐색을 위해선

미리 데이터가 정리(정렬)되야 함

정렬 Sorting

정렬 : 레코드를 어떤 기준에 의해서 순서대로 재배열하는 문제

순서 : 오름차순(ascending) or 내림차순(descending)

record : 2학년 2반에 있는 30명의 학생

field : 키, 나이, 몸무게, 성적 등등

key / sort key : 정렬의 기준이 되는 필드 ex) 성적 순으로~

어떻게 정렬할 것 인가? (오름차순)

arr = [1, 4, 5, 9, 2, 6, 3, 7, 8]

아이디어 1. 가장 작은 값부터 하나씩 빼자.

➔ 선택정렬, selection sort

아이디어 2. 하나씩 정렬하면서 끝까지 진행하자.

➔ 삽입정렬, insertion sort

탐색 Searching

탐색 : 주어진 데이터에서 원하는 값을 찾는 작업

값을 어떻게 찾을 것인가?

1. 눈에 보이지 않는 주머니에서 빨간 공을 꺼내기
2. 정렬된 리스트에서 숫자 35가 있는 지 확인하기
3. 어떤 웹페이지에서 “알고리즘” 이라는 단어 찾기
4. 쿠팡에서 “키보드” 상품 검색하기

정렬과 탐색 / 알고리즘은 왜 중요할까

어떤 프로그램의 속도를 결정하는 것은

하나의 작업을 할 때 필요한 동작의 횟수!

정확하고 빠른 알고리즘 없이는 우수한 프로그램이 될 수 없다!

ex)

트럼프 카드에서 ♥5 를 찾아보자.

유튜브에서 영상을 일일이 하나씩 넘겨가며 찾는다면..?

빠르게 찾는 대신, 다른 영상을 찾아준다면?

알고리즘의 성능과 복잡도란?

복잡도 = 시간 복잡도 + 공간 복잡도

컴퓨터 성능 향상에 따라,
공간 복잡도의 중요성이 비교적 낮아짐

암호코드 성공 다국어

시간 제한	메모리 제한
2 초	128 MB

2 1009	시간 초과

우리를 괴롭히는 시간 초과는 시간 복잡도 때문!

시간 복잡도

입력의 크기?

N개의 숫자 중에서 가장 큰 숫자 찾기
자연수 N의 약수의 개수 구하기
길이가 N인 문장에서 길이가 M인 단어 찾기

정의 : 입력 N 의 크기에 따라 실행되는 조작의 수
⇒ 즉, 알고리즘의 성능

평가방법 = 점근적 표기법 : N이 무한대로 커질 때의 복잡도를 간단히 표현

최상의 경우 : 오메가 표기(Big-Ω Notation)
평균의 경우 : 세타 표기법 (Big-θ Notation)
=〉 평가하기가 너무 까다롭다.
최악의 경우 : 빅오 표기법 (Big-O Notation)
=〉 최악의 경우에 대해 알고리즘을 평가

최악의 경우란 무엇일까?

어떤 알고리즘이 수행되는데 있어서 최대 연산이 발생하게 되는 입력

ex)

리스트 A : 9 8 7 6 5 4 3 2 1

case1. A 에서 1 or 100을 찾기

case2. 삽입 정렬로 오름차순 정렬하기

case3. 최대값 / 최소값 / 총합 을 구하기

빅오 표기법 (Big-O Notation)

알고리즘의 연산횟수를 나타낸 함수 : $f(n)$

$x > m$ 에서 $kg(n) \geq f(n)$을 만족하는 양수 m, k 가 존재하면, $f(n) \in O(g(n))$

컴퓨터 공학에서의 밑이 표기되지 않은 log는 거의 대부분 밑이 2인 경우를 말함.

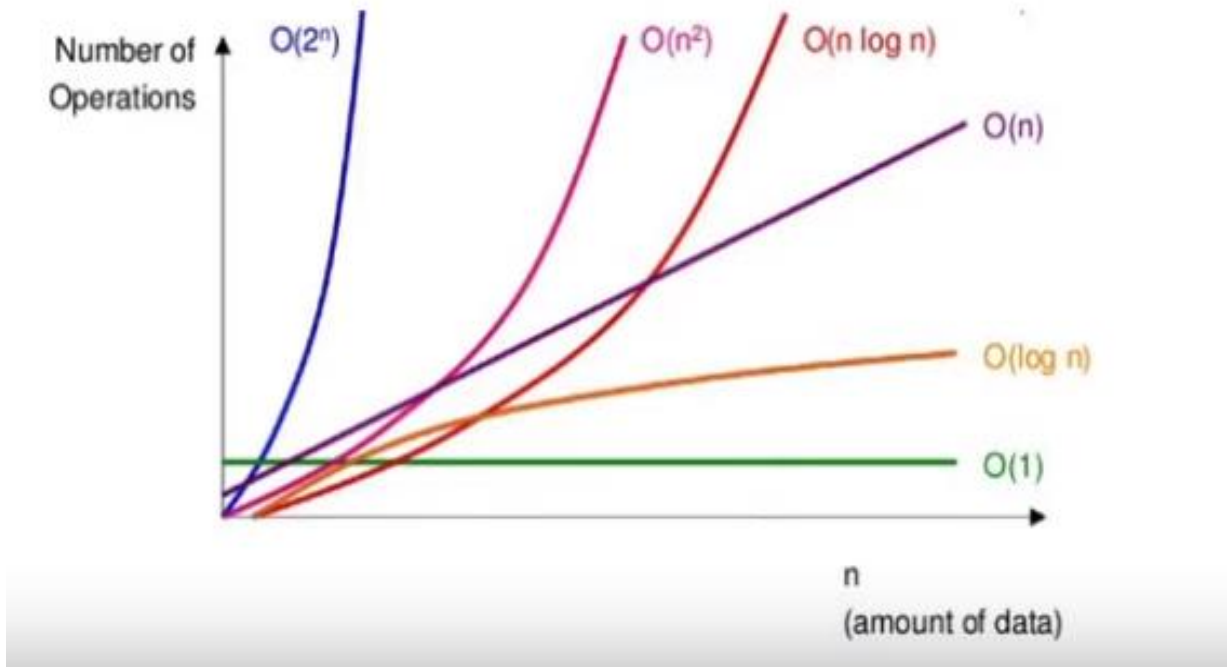
ex)

$f(n) = 3n^2 + 2n + 1 \rightarrow O(n^2)$

$g(n) = 300n^4 + 2500n \rightarrow O(n^4)$

$h(n) = 2^n + 3^n \rightarrow O(3^n)$

$\mathrm{k}(n) = 31251 \rightarrow O(1)$



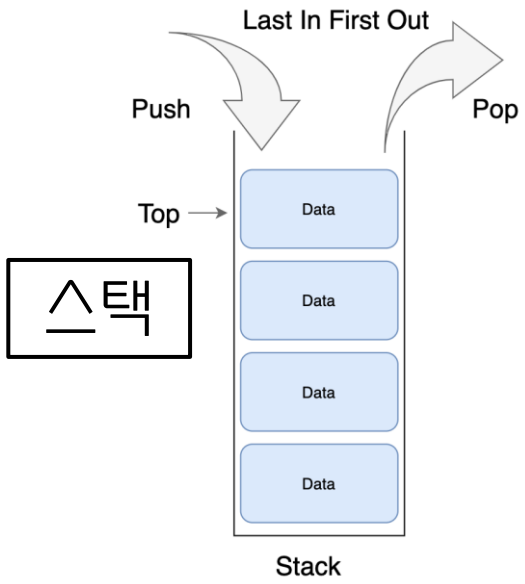
직접 풀어보자!

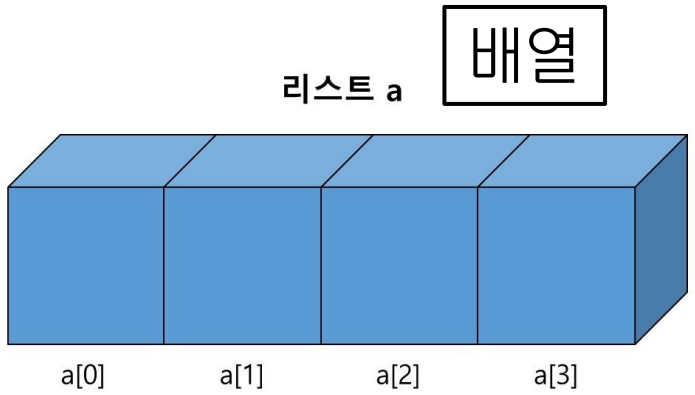
자료 : BigO_notation.py

자료구조

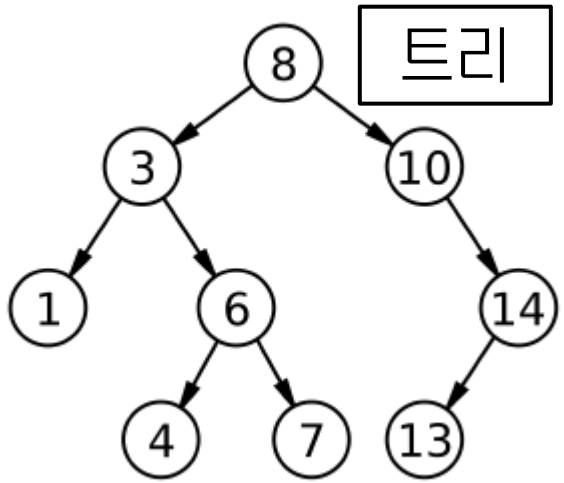
자료를 처리(탐색, 수정, 삭제)하기 위해
컴퓨터가 잘 다룰 수 있는 형태로
자료들을 정리하고 조직화하는 구조 ⇔ 자료를 저장하는 장소!

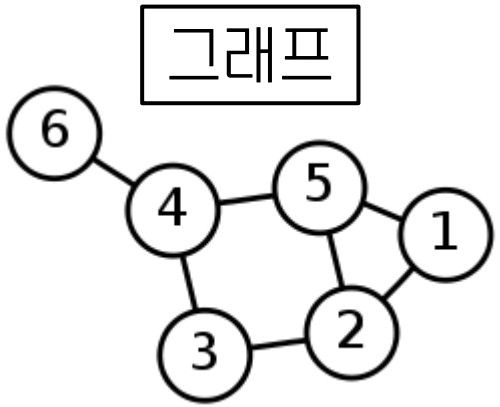
선형자료구조





비선형자료구조





적절한 자료구조의 선택

1. 동작에 따라 필요한 데이터(필드)가 달라짐
(가장 큰 값, 뒤에서 2번째 값, 가장 마지막에 입력한 데이터…)

2. 필요한 데이터를 가져오는데 적합한 자료구조는 모두 다름

Ex)

First in -> Queue

Last in -> Stack

Biggest -> Priority Queue

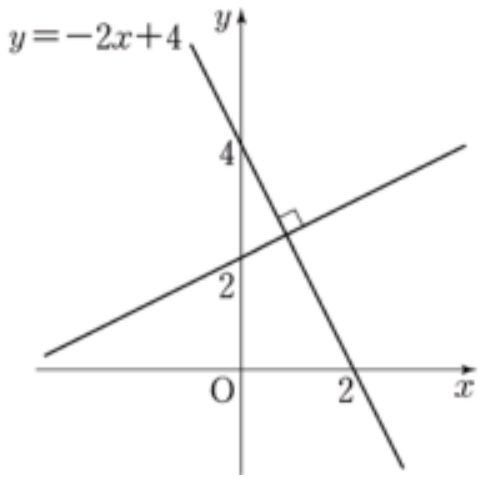
왜 그럴까?

알고리즘 설계

어떤 순서로 어떤 동작을 통해
문제를 해결할 것인지 설계하는 단계
➔논리구조를 만드는 과정!

문제를 풀기위한 지식(경험, 자료구조, 논리력)
+
주어진 조건

12. 직선 $y=-2x+4$와 수직이고 점 $(0, 2)$를 지나는 직선의 방정식은?



① $y=-2x+2$
② $y=-\frac{1}{2}x+2$
③ $y=\frac{1}{2}x+2$
④ $y=2x+2$

수학문제의
해결 과정을 떠올려보자

설계가 필수인 이유

아무리 뛰어난 프로그래머라고 해도
문제가 복잡하고
정확한 구현과정을 숙지하고 있지 않다면
구현 과정에서 실수가 발생하고
많은 수정을 거칠 수 밖에 없음.

따라서, 먼저 설계를 하고 설계를 따라서 구현, 코딩하는 것이
실수를 줄이고 쓸데없는 수정에 시간을 들이지 않는 방법!

설계 과정

1. 해결 아이디어 도출

2. 아이디어를 알고리즘으로 기술 (자연어 / 기호 를 추천)

-> 논리적 오류가 없어야 함.

-> 모두 구현 가능한 함수(동작)으로 구성 되어야 함.

-> 데이터를 어떻게 탐색할 것인가?

-> 데이터 삽입, 삭제, 참조 가 언제, 어떻게 이뤄지는 지 파악해야 함.

-> 시간 복잡도 분석.

3. 입출력 데이터를 어떻게 관리할 것인가

-> 적절한 자료구조 선정, 선정 이유는?

오늘의 실습

1. 백준 등 저지사이트에서 1개의 문제를 선정.
- 자신이 풀 수 있는 문제로 선정할 것.
- 못 고르겠으면 회장이 추천해드립니다.

2. 문제를 풀기위한 알고리즘을 설계해보자.

3. 설계를 바탕으로 프로그램을 구현해보자.

예시 : https://velog.io/@jm-kor-00/%ED%8C%8C%EC%9D%B4%EC%8D%AC%EB%B0%B1%EC%A4%80-1715%EB%B2%88-%EC%B9%B4%EB%93%9C-%EC%A0%95%EB%A0%AC%ED%95%98%EA%B8%B0