알고리즘과 구조의 관계를 배웠다.
우리가 보통 이름과 관련된 알고리즘은 배열을 사용한다.

정렬과 탐색과 검색
검색: 내가 뭘 찾을지 정해져 있고 이 안에 그게 있다.
탐색: 특정한 목표가 아닌 좀더 빠르고 괜찮은 것을 찾자!

검색 알고리즘 : 무작위로 찾으면 찾는다는 보장이 없다.
순서대로 위에서부터 찾으면 찾아진다.
최악의 경우 O(n)만큼 걸린다.

이름이 나열되어 있다면 찾기가 쉽다.
2진 검색 : O(log2 n)만큼 걸린다.

정렬 : 일련의 데이터들을 그 데이터의 크기 순서대로 배치하는 알고리즘이다.
최소로 일하고 시험지의 점수를 점수별로 정렬할 수 있을까?

selection sort : 선택을 하나해서 sort한다.
n + n-1 + n-2 + ... + 1 = n(n+1)/2 -> O(n^2)

```
i=0
while(i <n-1){
   min = find_smallest(A[i, n-1])
   if (min != 1)
      swap(A[i], A[i+1])
}
```

find_smallest(A[i,n-1])
쪽 진행하면서 가장 작은 값을 찾아서 다른 곳에 저장해두는 것이다.


→ $n + (n-1) + \cdots + 1 = \frac{n(n+1)}{2} \rightarrow O(n^2)$

$(n-1) + \cdots + 1 = \frac{n(n-1)}{2} \rightarrow O(n^2)$

처음부터 1개를 빼고해도 똑같다.

p.203) 연습문제가 시험에 나왔다.

```
def is_sorted(A):
    n = len(A)
    for index, a in enumerate(A):
        if index < n - 1:
            if A[index] > A[index + 1]:
                return False
    return True
```

버블 소트

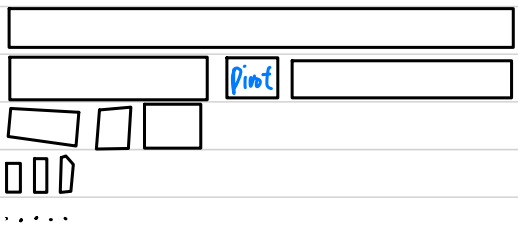
$n-1 + n-2 + n-3 + \cdots + 1 = \frac{n(n-1)}{2} \rightarrow O(n^2)$

삽입정렬

$1 + 2 + 3 + \cdots + (n-1) = \frac{n(n-1)}{2} \rightarrow O(n^2)$

$\log_2 1 + \log_2 2 + \log_2 3 + \cdots + \log_2 n-1$
$= \log_2 \prod_{i=1}^{n-1} i \rightarrow O(\_\_)$

퀵정렬 $\rightarrow O(n \log n)$




1개 → 1번
2개 → 2번
4개
i
n개 → $\log_2 n$번

n번의 비교
n번의 비교

$\rightarrow O(n \log_2 n)$

1명이 남도록 n번의 가위바위보를 하게한다.
$\therefore 2^n$명이 있는 것이다.

양말의 짝을 찾는 경우:

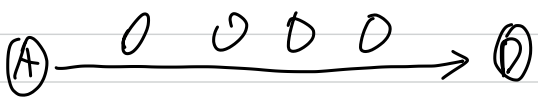
7.6 탐색(검색)

트리 구조 탐색 기법

검색 : 이미 정리된 목록에서 원하는 것을 찾는다. (이게 여기에 있는가 없는가)

탐색: 조건에 맞는 답을 찾아가능

(문제에 대한 답을 찾아가는 과정)




많은 경로들 중에서 가장 좋은 것을 찾는 알고리즘을 탐색 알고리즘이라고 할 수 있음

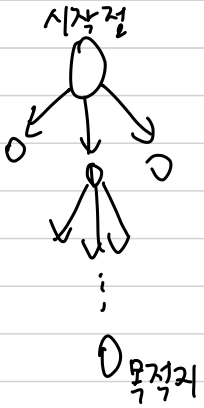
그 가장 좋은 것이 우리가 찾고자 하는 해답이다.

이러한 문제의 종류를 해결(problem solving)이라고 함

ex) 최단 거리 찾는 것,

오목 두는 것(돌을 놓을 수 있는 여러가지 위치 중에서 내가 이기도록 하는데 도움이 가장 많이 되는 위치를 찾는 것)

빨래 걷는 문제




목적지로 가는 가장 효율적인 경로를 찾는 문제!