6주차: 알고리즘 입문, 그리디 알고리즘

커리큘럼

1. 변수, 입출력
2. 조건문, 반복문
3. 리스트, 튜플, 세트, 딕셔너리
4. 함수
5. class
6. **알고리즘 입문, 그리디 알고리즘**
7. 재귀함수
8. 탐색
9. DP(Dynamic Programming)

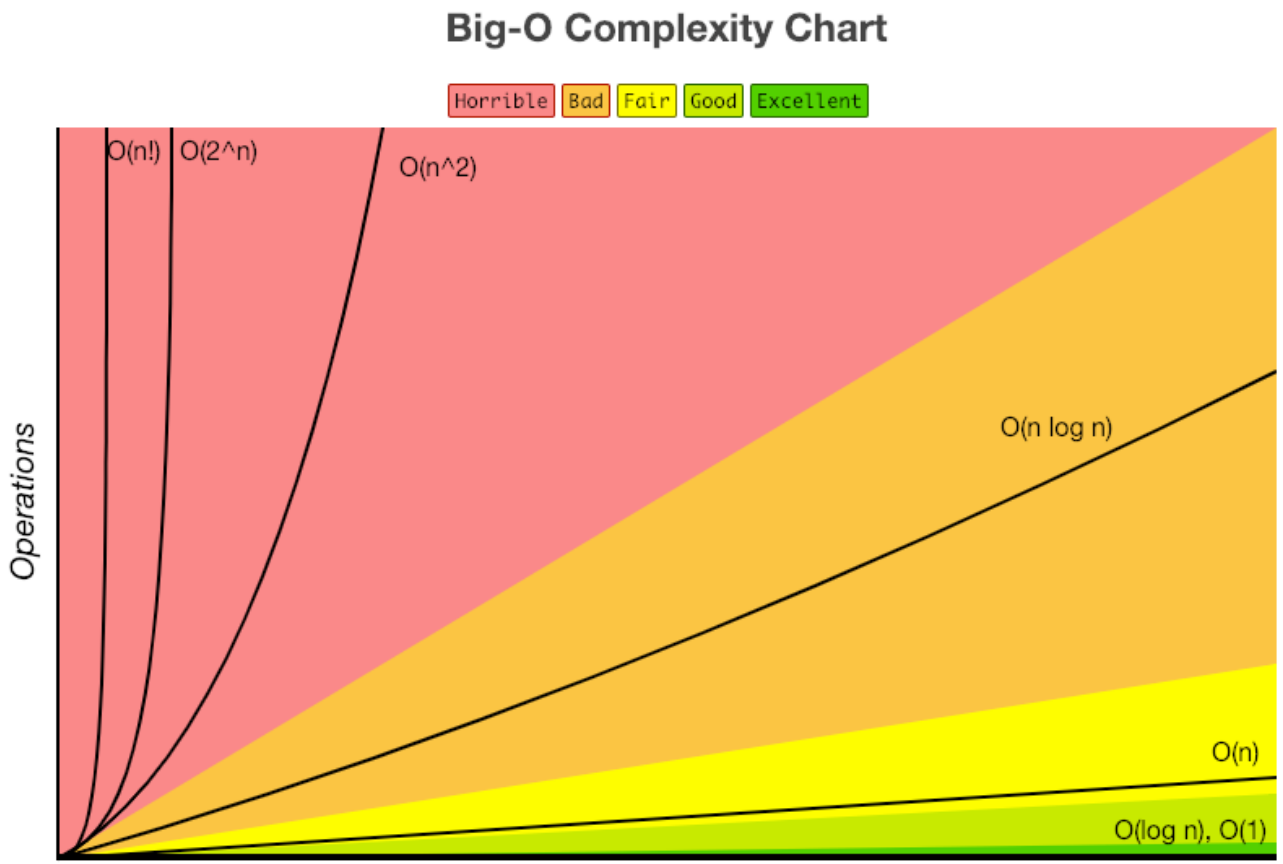
알고리즘

- 문제를 해결하기 위한 여러 동작들의 모임
- 어떤 기능이 일어나기 위해 내재된/독립된 단계적 명령어들의 집합
- 좋은 알고리즘의 분석 기준

  ✓ Correctness : 문제를 해결하는가
  ✓ Efficiency : 이를 효과적으로 하는가

  1. 정확성
  2. 작업량
  3. 기억 장소 사용량
  4. 최적성
  5. **복잡도(Big-O 표기법: 시간 복잡도)**

Big-O



- O(1) : n이 증가해도 수행횟수 불변
- O(log n) : 이진탐색(Binary Search)
- O(n) : 선형탐색(Linear Search)
- O(n log n) : 퀵정렬(Quick sort)
- O(n^2) : 이중 for 문

버블 정렬

```
a=list(map(int,input().split()))
for i in range(len(a)):
    for j in range(len(a)-i-1):
        if a[j]>a[j+1]:
            tmp = a[j]
            a[j] = a[j+1]
            a[j+1] = tmp
for i in a:
    print(i,end=' ')
```

시간 복잡도?

Ans : O(n^2)

선택 정렬

```
a=list(map(int,input().split()))
for i in range(len(a)):
    min_index = i
    for j in range(i,len(a)):
        if a[min_index] > a[j]:
            min_index = j
    tmp = a[min_index]
    a[min_index] = a[i]
    a[i] = tmp

for i in a:
    print(i,end=' ')
```

시간 복잡도?

Ans : O(n^2)

삽입 정렬

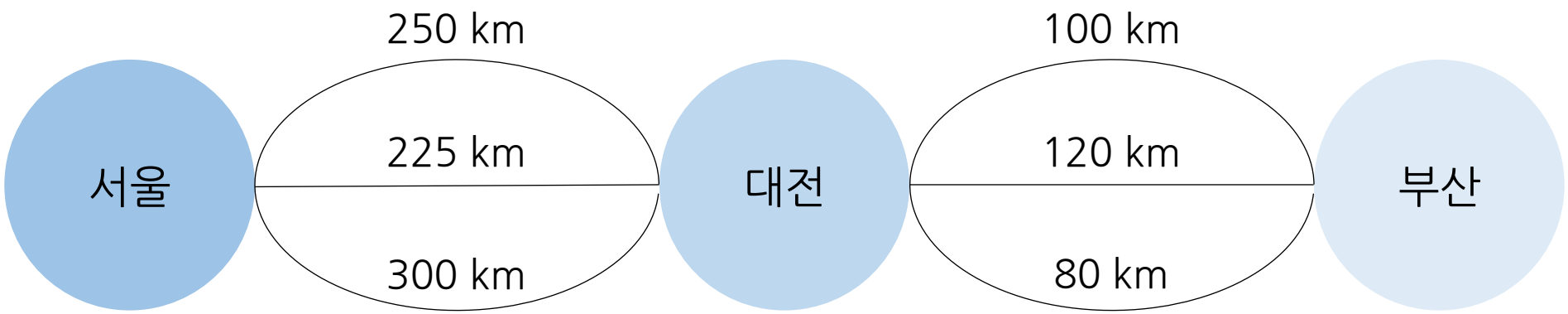
```
a=list(map(int,input().split()))
for i in range(1,len(a)):
    for j in range(i,0,-1):
        if a[j] < a[j-1]:
            tmp = a[j]
            a[j] = a[j-1]
            a[j-1] = tmp

for i in a:
    print(i,end=' ')
```

시간 복잡도?

Ans : O(n^2)

그리디 알고리즘(Greedy Algorism)

- ✓ 현재의 상황에서 가장 좋은 것(최선의 상황)을 선택하는 알고리즘
- ✓ 특정한 경우에 대해서는 최적의 해, 그렇지 않은 경우도 있음



동전 문제

- ✓ 지불해야 하는 값이 N원 일 때 1원, 50원, 100원, 500원 동전으로 동전의 수가 가장 적게 지불하는 방법을 찾아 보시기 바랍니다.

- ✓ 지불해야 하는 값이 N원 일 때 10원, 40원, 50원, 100원 동전으로 동전의 수가 가장 적게 지불하는 방법을 찾아 보시기 바랍니다.

예제 1

0~9 로만 이루어진 문자열이 주어지고, 왼쪽부터 오른쪽으로 하나씩 모든 숫자를 확인하여 모든 숫자 사이에 곱하기 또는 더하기 연산자를 넣어 만들어질 수 있는 가장 큰 수가 무엇인지 구해 보시기 바랍니다. 단, 모든 연산은 일반적인 연산 순서가 아닌 왼쪽부터 순서대로 이루어집니다.

예를 들어, 20145는 2+0+1*4*5일 때 최대이고, 왼쪽에서 부터 차례대로 계산하면 60입니다.

입력 예시	출력 예시
20145	60

예제 2

N개의 회의에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다.

```
9
1 3
2 5
4 7
1 8
5 9
8 10
9 11
11 14
13 16
```

입력 예시

```
4
```

출력 예시


과제

6-1

동네 편의점의 주인은 N개의 동전을 가지고 있습니다. 이때 N개의 동전을 이용하여 만들 수 없는 양의 정수 금액 중 최솟값을 구하는 프로그램을 작성하시기 바랍니다.

예를 들어, N=5이고, 각 동전이 3,2,1,1,9원 동전이라고 가정하면, 주인이 만들 수 없는 최소 양의 정수는 8입니다.

```
5
3 2 1 1 9
```

입력 예시
첫째 줄: 동전의 개수
둘째 줄: 각 동전의 화폐단위(자연수)

```
8
```

출력 예시

6-2

+와 -의 연산으로 구성된 어떤 식을 적당히 괄호를 쳤을 때 이 식의 최댓값과 최솟값을 구해 보시기 바랍니다.

예를 들어, 55-50+40의 경우 최댓값은 (55-50)+40 = 45이고, 최솟값은 55-(50+40) = -35가 됩니다.

```
55-50+40
```

입력 예시

```
45 -35
```

입력 예시

6-3

정수로 이루어진 수열이 주어지고, 그 수열 각 숫자의 합을 구하고자 합니다. 하지만, 그냥 그 수열의 합을 모두 더해서 구하는 것이 아니라, 수열의 두 수를 묶으려고 합니다. 어떤 수를 묶으려고 할 때, 위치에 상관없이 묶을 수 있고 수열의 합을 구할 때 묶은 수는 서로 곱한 후에 더합니다. 이 때, 입력 받은 수열의 합의 최댓값을 구하시기 바랍니다.

예를 들어, [-1, 2, 1, 3]은 -1+1+(2*3)일 때 최대이고, 계산하면 6입니다.

```
-1 2 1 3
```

입력 예시

```
6
```

입력 예시