알고리즘 멘토링

-주민찬-

Orientation

멘토 : 주민찬

진행방식 수업 30분, 실습 1시간

실습은 백준 (온라인 문제 풀이 사이트) 에서 3문제 정도 푸는 것으로 하고 실습 시간에 질의응답 같이 진행

멘토링 시간 이후 질문이 있을 시 카톡으로 질문해도 자세히 알려드리겠습니다. 꼭 멘토링 관련이 아니어도 질문하셔도 돼요~

강의계획서

알고리즘 멘토링 계획		
날짜	강의 내용	강의자
09월 23일	OT, 시간복잡도, Prefix Sum	주민찬
09월 30일	Binary Search	주민찬
10월 07일	자료구조(Stack, Queue, Deque)	주민찬
10월 28일	자료구조(Graph), BFS, DFS	주민찬
11월 04일	자료구조(Heap), Dikstra	주민찬
11월 11일	Bellman Ford, Floyd-Warshall	주민찬
11월 18일	Dynamic Programming	주민찬
11월 25일	Union-Find	주민찬
12월 02일	Segment Tree	주민찬

시간복잡도

N개의 리스트의 합

1	6	8	3	5	16	9	27	11	34

리스트 정렬 (선택정렬)

1	3	5	6	8	9	11	16	27	34

O - 표기

- $O(n^2)$: 최고차항의 차수가 n^2을 넘지 않는 모든 함수의 집합

n	n^2	n^2+10n	$2n^2+10n$
1	1	11	12
10	100	200	300
100	10000	11000	21000
1000	1000000	1010000	2010000
10000	100000000	100100000	200100000

O - 표기

시간/n	1	2	3	4	8	16	32	64	1000
1	1	1	1	1	1	1	1	1	1
log n	0	1	1.58	2	3	4	5	6	9.97
n	**1**	**2**	**3**	**4**	**8**	**16**	**32**	**64**	**1000**
n log n	0	2	4.75	8	24	64	160	384	9966
n^2	1	4	9	16	64	256	1024	4096	1000000
n^3	1	8	27	64	512	4096	32768	262144	1000000000
2^n	2	4	8	16	256	65536	4294967296	약 1.844×10^{19}	약 1.07×10^{301}
n!	1	2	6	24	40320	20922789888000	약 2.63×10^{35}	약 1.27×10^{89}	약 4.02×10^{2567}

O - 표기

- 1초가 걸리는 입력의 크기
- O(1)
- O(lgN)
- O(N) : 1억
- O(NlgN) : 5백만
- O(N^2) : 1만
- O(N^3) : 500
- O(2^N) : 20
- O(N!) : 10

$2^{20} = 1048576$

$N! = 10! = 3628800$

시간복잡도

```python
import heapq

def Floyd(graph,V):
    for i in range(1,V+1):
        for j in range(1,V+1):
            for k in range(1,V+1):
                graph[i][j]=min(graph[i][j], graph[i][k]+graph[k][j])

    Min = float("inf")
    for i in range(1,V+1):
        Min = min(Min,graph[i][i])

    if Min == float("inf"):
        return -1
    else:
        return Min
```

시간복잡도

```python
def binary_search(n_list,target):
    s=0
    e=len(n_list)
    while s<=e:
        mid=(s+e)//2
        if target==n_list[mid]:
            return mid
        elif target>n_list[mid]:
            s=mid+1
        else:
            e=mid-1
    return 0
```

시간복잡도

```python
def prefix_sum(A,N):
    A = [0]+A
    S = [0 for i in range(N+1)]
    for i in range(1,N+1):
        S[i]=S[i-1]+A[i]
    return S
```

Prefix Sum

0	1	2	3	4	5	6	7	8	9	…
5	9	2	3	7	5	1	4	3	2	…

인덱스 i부터 j까지의 합 -> $O(n)$

이 과정을 n번 반복 -> $O(n^2)$

Prefix Sum 알고리즘을 이용하면 그대로 $O(n)$

Prefix Sum

0	1	2	3	4	5	6	7	8	9	…
5	9	2	3	7	5	1	4	3	2	…

길이가 (n+1)인 새로운 리스트를 생성
ex) prefix_sum = [0 for _ in range(n+1)]

prefix_sum[i]는 0번째 원소부터 i-1번째 원소까지의 합 (i >= 1)

그러면 리스트의 i번째 원소부터 j번째 원소의 합은
prefix_sum[j+1] - prefix_sum[i]

따라서 처음 prefix_sum 리스트를 만드는 과정은 n번 시행
그 후 i부터 j까지의 합은 1번에 시행 가능

이 과정을 n번 하더라도 독립적이기 때문에 시간복잡도는 $O(n)$

Prefix Sum

```python
import sys
input = sys.stdin.readline

N, M = map(int,input().split())
ls = list(map(int,input().split()))

prefix_sum = [0 for _ in range(N+1)]

for i in range(1,N+1):
    prefix_sum[i] = prefix_sum[i-1] + ls[i-1]

for k in range(M):
    i, j = map(int,input().split())
    print(prefix_sum[j+1]-prefix_sum[i])
```


실습!!