알고리즘과 입출력

최백준 choi@startlink.io

알고리즘

알고리즘

Algorithm

- In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed.
- https://en.wikipedia.org/wiki/Algorithm

알고리즘

Algorithm

- 알고리즘이란 어떠한 문제를 해결하기 위한 여러 동작들의 모임이다.
- https://ko.wikipedia.org/wiki/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98

알고리즘

Algorithm

- 알고리즘 공부에 가장 효과적인 것은
- 문제 풀이!

Baekjoon Online Judge

https://www.acmicpc.net

- 2016년 6월 15일 기준
  - 전체 문제 11833개
  - 채점 가능한 문제 8018개
  - 풀린 문제 5692개
  - 사용 가능한 언어: 50개

그 외의 사이트

Site

- 알고스팟: https://algospot.com
- Codeforces: http://codeforces.com
- Topcoder: http://topcoder.com

읽어보면 좋은 글

서문: 프로그래밍은 어렵다

- 알고리즘 문제 해결 전략 책의 서문
- http://book.algospot.com/hard.html

공부하는 방법

1. 먼저 알고리즘이나 문제를 푸는 방법을 이해
    - 완벽하지 않거나 일부만 이해했어도 성공!
2. 관련 문제를 풀어본다
    - 한 문제는 길어야 2시간 정도만 고민해본다
    - 모르겠으면 포기하고 정답 소스를 보거나 풀이를 본다
3. 1번과 2번에서 이해가 잘 가지 않는 부분이 있으면 질문한다.
    - 설마 이런 것을 질문해도 될까 고민 되는 것도 질문해야 한다
4. 다시 알고리즘을 이해해보고 문제를 다시 풀어본다.
    - 모르겠으면 포기하고 다시 풀이를 본다
    - 그래도 모르겠으면, 다른 일을 하거나, 놀러 나가거나, 다른 알고리즘이나 문제를 풀어본다

가장 중요한 점

- 프로그래밍을 많이 하는 것도 중요하지만
- 가장 중요한 것은 생각을 많이 하는 것

더 중요한 점

- 더 중요한 것은 모르겠으면 포기하고 풀이를 보는 것
- 스스로 생각해서 해결하지 않고 답을 보고 해결했어도 해결한것!
- 1개를 3일에 걸쳐서 푸는 것 보다 3개를 1일동안 푸는 것이 더 좋다
- 미적분 문제를 풀기 위해서 미적분을 스스로 생각해서 만들 필요가 없는 것과 비슷
- 여기서 포기라는 것은 최대한 노력을 해보고를 의미함
- 무작정 "아 모르겠다~ 포기하고 답봐야지" 가 아님
- 그래도 1~2시간 고민해도 모르겠으면 포기!
- 알고리즘에서 제일 중요한 것은 알고리즘을 만드는 것이 아니고, 생각의 과정을 배워서 나중에 스스로 그러한 생각을 해내는 것

프로그래밍 언어

Programming Language

- 프로그래밍 언어는 크게 상관이 없다
- 많은 사람들이 알고리즘 문제를 풀 때 사용하는 언어: C++ > C > Java
- C언어를 사용하는 경우에는 C++을 사용하는 것이 좋다.
- C++을 사용하는 경우에는 C++11, STL, scanf/printf를 사용하는 것이 좋다.
- Java를 사용하는 경우에는 Scanner를 이용해 입력을 편리하게 받는 것이 좋다.

```
ios_base::sync_with_stdio(false);
```

시간 복잡도

시간 복잡도

Time Complexity

- 시간 복잡도를 이용하면 작성한 코드가 시간이 얼마나 걸릴지 예상할 수 있다.
- 표기법으로 대문자 O를 사용한다.

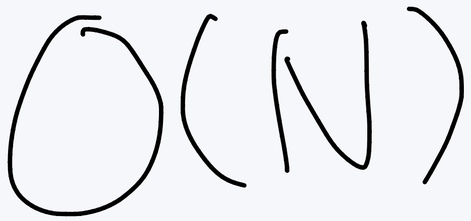
- 영어로는 Big O Notation
- 입력의 크기에 대해서 시간이 얼마나 걸릴지 나타내는 방법
- 최악의 경우에 시간이 얼마나 걸릴지 나타내야한다.

시간 복잡도

Time Complexity

- 아래 소스는 1부터 N까지 합을 계산하는 소스이다.

```
int sum = 0;
for (int i=1; i<=N; i++) {
    sum += i;
}
```



- 시간 복잡도: O(N)

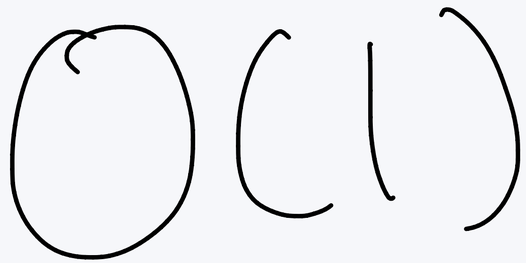
시간 복잡도

Time Complexity

- 아래 소스는 1부터 N까지 합을 계산하는 소스이다.

```
int sum = 0;
sum = N*(N+1)/2;
```

- 시간 복잡도: O(1)

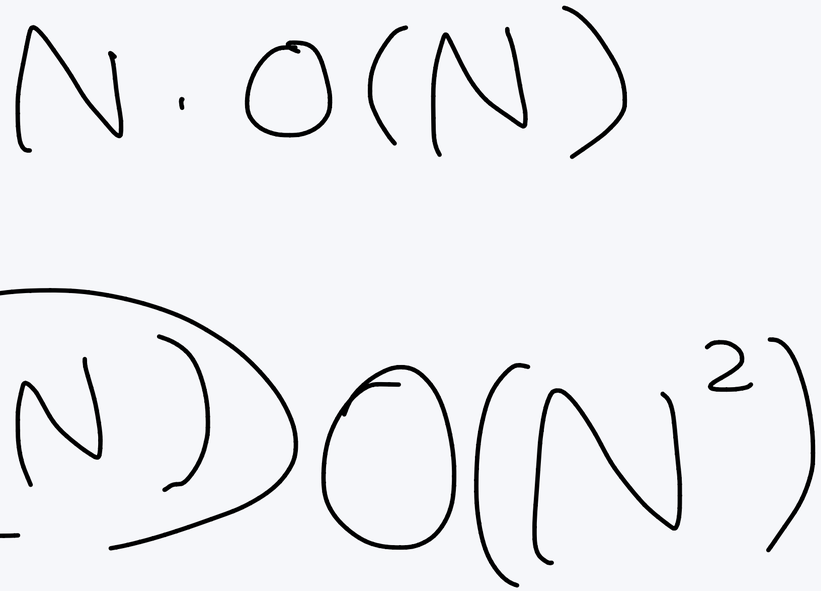


시간 복잡도

Time Complexity

- 아래 소스는 1부터 N까지 합을 계산하는 소스이다.

```
int sum = 0;
for (int i=1; i<=n; i++) {
    for (int j=1; j<=n; j++) {
        if (i == j) {
            sum += j;
        }
    }
}
```



- 시간 복잡도: O(N^2)

시간 복잡도

Time Complexity

- 대표적인 시간 복잡도는 아래와 같다.
- O(1)
- O(lgN)
- O(N)
- O(NlgN)
- O(N^2)
- O(N^3)
- O(2^N)
- O(N!)

시간 복잡도

Time Complexity

- 시간 복잡도 안에 가장 큰 입력 범위를 넣었을 때, 1억이 1초정도이다.

1억 = 1초

시간 복잡도

Time Complexity

N = 1000

- 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

1.25억

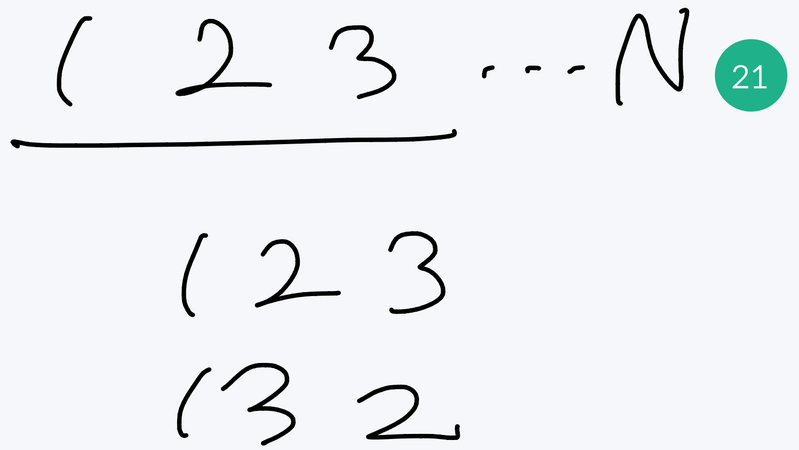
$2^{20} = 1048576$

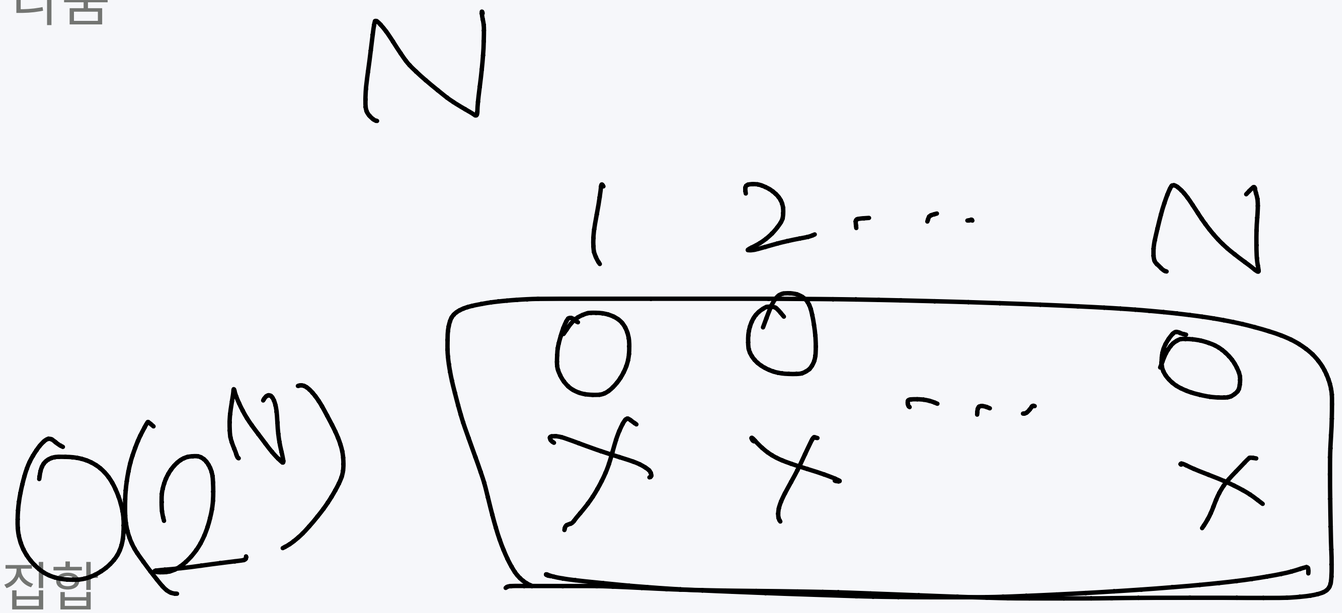
$N! = 10! = 3628800$

시간 복잡도

Time Complexity

- 시간 복잡도의 의미 (대부분의 경우, 항상은 아님)
- O(1): 단순 계산 (a + b와 같은 연산, 배열에 접근하는 연산)
- O(lgN): N개를 절반으로 계속해서 나눔
- O(N) : 1중 for문
- O(NlgN) :
- O(N^2) : 2중 for문
- O(N^3) : 3중 for문
- O(2^N) : 크기가 N인 집합의 부분 집합
- O(N!) : 크기가 N인 순열

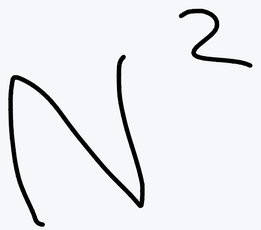


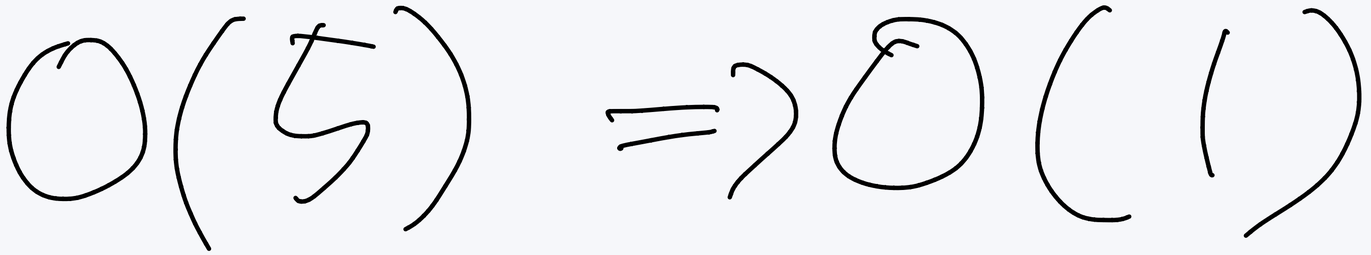


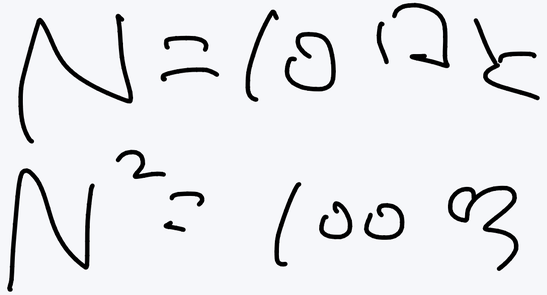
시간 복잡도 계산

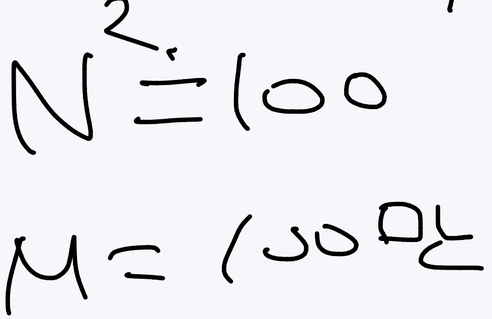
Time Complexity

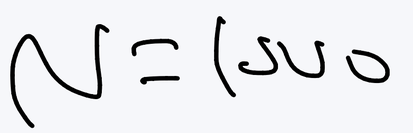
- Big O Notation 에서 상수는 버린다.
- O(3N^2) = O(N^2)
- O(1/2 N^2) = O(N^2)
- O(5) = O(1)
- 두 가지 항이 있을 때, 변수가 같으면 큰 것만 빼고 다 버린다.
- O(N^2 + N) = O(N^2)
- O(N^2 + NlgN) = O(N^2)
- 두가지 항이 있는데 변수가 다르면 놔둔다.
- O(N^2 + M)











입/출력

Hello World

https://www.acmicpc.net/problem/2557

- Hello World!를 출력하는 문제

printf

Cout

System.out.pritln

print( )

A+B

A+B

- 두 수를 입력받고 A+B를 출력하는 문제
- https://www.acmicpc.net/problem/1000
- https://www.acmicpc.net/problem/2558
- https://www.acmicpc.net/problem/10950
- https://www.acmicpc.net/problem/10951
- https://www.acmicpc.net/problem/10952
- https://www.acmicpc.net/problem/10953
- https://www.acmicpc.net/problem/11021
- https://www.acmicpc.net/problem/11022

A+B

https://www.acmicpc.net/problem/1000

- C: https://gist.github.com/Baekjoon/c8007caf8d5410ddc162
- C++: https://gist.github.com/Baekjoon/09d2a162fde90e36ac37
- Java: https://gist.github.com/Baekjoon/0c7632bd9b01909c7e6f

A+B - 3

https://www.acmicpc.net/problem/10950

- C: https://gist.github.com/Baekjoon/6277d6ffca7ba1fe3079
- C++: https://gist.github.com/Baekjoon/288036bcb8b085dc6c0a
- Java: https://gist.github.com/Baekjoon/e167a3a1ea322788add4

A+B - 3

https://www.acmicpc.net/problem/10950

- 테스트 케이스 형식으로 주어지는 경우에는
- 각각을 독립적인 문제로 생각하고 풀면 된다.
- 전체 테스트 케이스를 입력 받은 다음에, 풀지 않아도 된다.

A+B - 3

https://www.acmicpc.net/problem/10950

```
int t;
int a[100],b[100];
scanf("%d",&t);
for (int i=0; i<t; i++) {
    scanf("%d %d",&a[i],&b[i]);
}

for (int i=0; i<t; i++) {
    printf("%d\n",a[i]+b[i]);
}
```

- 이렇게 입력을 다 받고, 모아서 하나씩 출력하지 않고

A+B - 3

https://www.acmicpc.net/problem/10950

```
int t;
int a,b;
scanf("%d",&t);
while (t--) {
    scanf("%d %d",&a,&b);
    printf("%d\n",a+b);
}
```

- 이렇게 하나 하나 입력받고 풀면 된다.
- 앞 페이지의 방법 처럼 구현하면, T개수를 모를때 배열의 크기를 정하기 어렵다.
- 또, 전체 입력이 매우 큰 경우에 매우 큰 크기의 배열을 필요하게 된다.

A+B - 4

https://www.acmicpc.net/problem/10951

- C: https://gist.github.com/Baekjoon/e27c3613b54c0052febf
- C++: https://gist.github.com/Baekjoon/f239f1196d5f2b0ce65d
- Java: https://gist.github.com/Baekjoon/c236b060f64c9f6eebda

A+B - 4

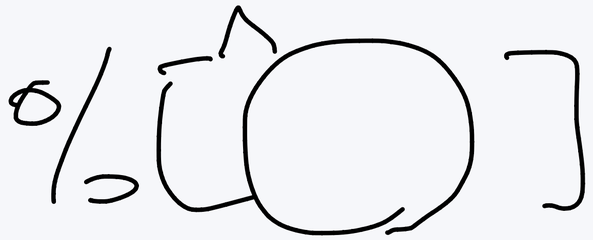
https://www.acmicpc.net/problem/10951

- 이 문제 처럼 입력이 몇 개인지 주어지지 않은 경우에는
- 입력을 EOF까지 받으면 된다.
- C: `while (scanf("%d %d",&a,&b) == 2)`
- C++: `while (cin >> a >> b)`
- Java: `while (sc.hasNextInt())`

- scanf의 리턴값은 성공적으로 입력받은 변수의 개수다.

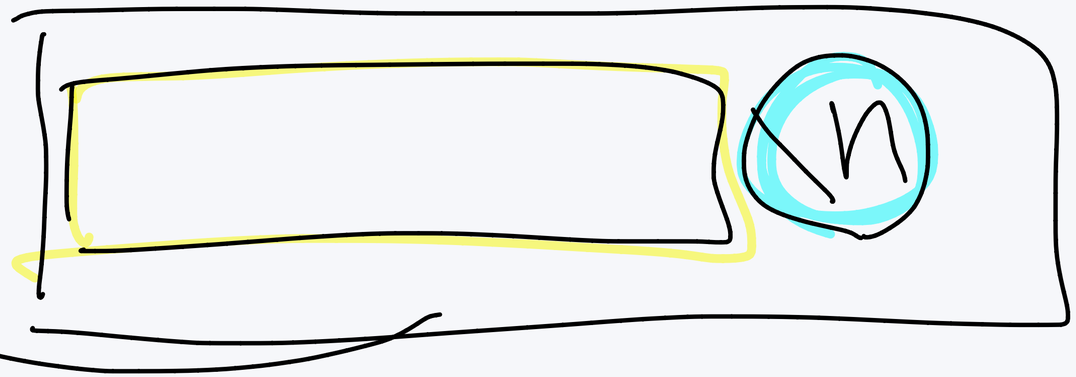
한 줄 입력받기

Line



```
scanf("%s",s);
cin >> s;
```

- 위의 두 방법은 한 줄을 입력받을 수 없다.



```
fgets(s, 100, stdin);
scanf("%[^\n]\n", s);
getline(cin, s);
```

- 위의 세 방법은 모두 한 줄을 전체로 입력받을 수 있다.
- fgets는 줄 바꿈까지 입력 받기 때문에, 조심해야 한다.

nextLine()

그대로 출력하기

https://www.acmicpc.net/problem/11718

```
scanf("%[^\n]\n", s);
```

- 줄바꿈을 입력받지 않기 때문에, 편리한 방법이지만, 각 줄의 앞 뒤에 있는 공백은 무시하고 입력을 받게 된다.
- 따라서, 빈 줄을 입력받을 수 없다.
- 또, 공백으로 시작하는 경우 공백을 무시하고 문자부터 입력받게 된다.
- 이 문제는 위의 두가지 경우가 없기 때문에 사용 가능.
- C/C++: https://gist.github.com/Baekjoon/edc22d3d680a3bbd7ba7

그대로 출력하기 2

https://www.acmicpc.net/problem/11719

```
scanf("%[^\n]\n", s);
```

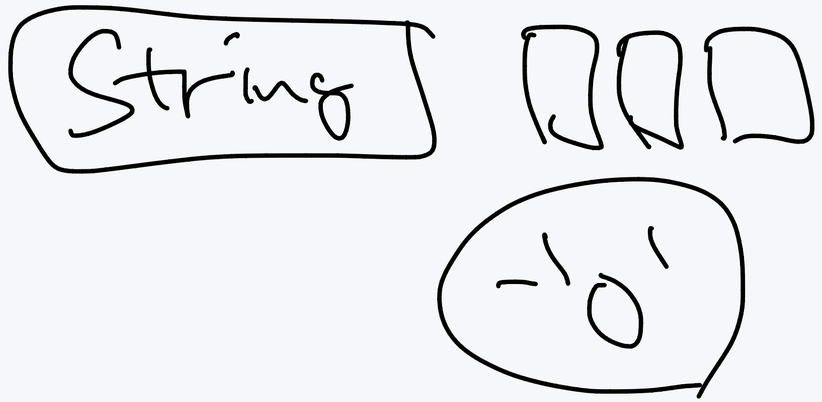
- 줄바꿈을 입력받지 않기 때문에, 편리한 방법이지만, 각 줄의 앞 뒤에 있는 공백은 무시하고 입력을 받게 된다.
- 따라서, 빈 줄을 입력받을 수 없다.
- 또, 공백으로 시작하는 경우 공백을 무시하고 문자부터 입력받게 된다.
- 이 문제는 위의 두가지 경우가 있기 때문에 사용 불가능.
- C/C++: https://gist.github.com/Baekjoon/66b4f9451a28dd416a33

숫자의 합

https://www.acmicpc.net/problem/11720

```
scanf("%1d",&x);
```

- %d 사이에 수를 넣으면, 그 길이 만큼 입력을 받게 된다.
- 12345에 1, 2, 3, 4, 5 따로 따로 입력 받을 수 있다.
- C/C++: https://gist.github.com/Baekjoon/0aba0c466380b9e10c2f



열 개씩 끊어 출력하기

https://www.acmicpc.net/problem/11721

```
scanf("%10s",s);
```

- %s의 경우도 개수를 지정해서 입력받을 수 있다.
- 만약, 입력받을 수 있는 것의 개수가 지정한 개수 보다 적으면 그만큼만 입력을 받게 된다.
- C/C++: https://gist.github.com/Baekjoon/90de3b2f72d6300e3e5e

연습하기

연습 문제

- N 찍기: https://www.acmicpc.net/problem/2741
- 기찍 N: https://www.acmicpc.net/problem/2742
- 구구단: https://www.acmicpc.net/problem/2739
- 2007년: https://www.acmicpc.net/problem/1924
- Sum: https://www.acmicpc.net/problem/8393
- 최소, 최대: https://www.acmicpc.net/problem/10818

별찍기 연습하기

연습 문제

- 별찍기 - 1: https://www.acmicpc.net/problem/2438
- 별찍기 - 2: https://www.acmicpc.net/problem/2439
- 별찍기 - 3: https://www.acmicpc.net/problem/2440
- 별찍기 - 4: https://www.acmicpc.net/problem/2441
- 별찍기 - 5: https://www.acmicpc.net/problem/2442
- 별찍기 - 7: https://www.acmicpc.net/problem/2444
- 별찍기 - 8: https://www.acmicpc.net/problem/2445
- 별찍기 - 9: https://www.acmicpc.net/problem/2446
- 별찍기 - 12: https://www.acmicpc.net/problem/2522
- 별찍기 - 16: https://www.acmicpc.net/problem/10991
- 별찍기 - 17: https://www.acmicpc.net/problem/10992