2023 Soongsil Programming Contest

Official Problem Set

Sponsored By:

HYUNDAI MOBIS STARTLINK

언어 가이드

- 채점은 Intel Xeon E5-2666v3 프로세서를 사용하는 AWS EC2 c4.large 인스턴스에서 진행합니다.
- 채점 서버의 운영체제는 Ubuntu 16.04.7 LTS 입니다.
- 아래 언어 중 원하는 언어를 선택해 사용할 수 있습니다.
  - C11: `gcc 11.1.0`
  - C++17: `g++ 11.1.0`
  - Java 15: `OpenJDK version "16.0.1" 2021-04-20`
  - Python 3: `Python 3.11.0`
  - PyPy3: `Python 3.9.12, PyPy 7.3.9 with GCC 10.2.1 20210130 (Red Hat 10.2.1-11)`
  - 컴파일과 실행 옵션은 `https://help.acmicpc.net/language/info`에서 확인할 수 있습니다.
- C11/C++17에서 `scanf_s`와 `Windows.h`등의 비표준 함수를 사용할 수 없습니다.
- Java를 사용하는 경우, `main` 메소드를 포함하는 클래스의 이름은 `Main`이어야 합니다.
- Python에서 `numpy`와 같은 외부 모듈을 사용할 수 없습니다.
- 채점 사이트에서 컴파일 에러를 받은 경우, '컴파일 에러' 글씨를 누르면 오류가 발생한 위치를 볼 수 있습니다.
- 아래 코드는 표준 입력(standard input)을 통해 공백으로 구분된 두 정수를 입력으로 받아서 표준 출력(standard output)을 통해 합을 출력하는 코드입니다.
  - C11

```c
#include <stdio.h>

int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d\n", a + b);
    return 0;
}
```

  - C++17

```cpp
#include <iostream>
using namespace std;

int main() {
    int a, b;
    cin >> a >> b;
    cout << a + b << endl;
    return 0;
}
```

– Java 15

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println(a + b);
        sc.close();
    }
}
```

– Python 3 / PyPy3

```python
a, b = map(int, input().split())
print(a + b)
```

- 입출력 양이 많을 때는 위 코드를 사용한 입출력이 너무 오래 걸리기 때문에 다른 방식으로 입출력해야 합니다.
- C11/C++17에서 `scanf`와 `printf`를 사용하는 경우, 입출력 속도는 문제를 해결할 수 있을 정도로 충분히 빠릅니다.
- C++17에서 `cin`과 `cout`을 사용하는 경우, 입출력 전에 `ios_base::sync_with_stdio(false);`와 `cin.tie(nullptr);`를 사용하여야 합니다. 단, 이 이후에는 `cin`, `cout` 계열 함수와 `scanf`, `printf` 계열 함수를 섞어서 사용하면 안 됩니다. 또한, 개행문자로 `std::endl` 대신 `"\n"`을 사용해 주세요.
- Java 15에서는 `BufferedReader`와 `BufferedWriter`를 사용하여야 합니다.
- Python 3 및 PyPy3 에서는 `input()` 대신 `sys.stdin.readline().rstrip("\n")`을 사용하여야 합니다. 코드의 가장 위 부분에 `import sys` 와 `input = lambda: sys.stdin.readline().rstrip("\n")` 을 사용하여야 합니다.
- 아래 코드는 표준 입력(standard input)을 통해 문제의 개수 T를 입력받은 다음 T줄에 걸쳐 공백으로 구분된 두 정수를 입력으로 받아 표준 출력(standard output)을 통해 두 정수의 합을 총 T줄에 걸쳐 출력하는 코드입니다.

– C++17

```cpp
#include <iostream>
using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int T;
```

```
    cin >> T;
    for(int i=1; i<=T; i++){
        int a, b;
        cin >> a >> b;
        cout << a + b << "\n"; // do not use endl
    }
    return 0;
}
```

– Java 15

```
import java.util.*;
import java.io.*;

public class Main{
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));

        int T = Integer.parseInt(br.readLine());
        for(int i=1; i<=T; i++){
            String[] temp = br.readLine().split(" ");
            int a = Integer.parseInt(temp[0]);
            int b = Integer.parseInt(temp[1]);
            bw.write(String.valueOf(a + b) + "\n");
        }
        br.close();
        bw.close();
    }
}
```

– Python 3 / PyPy3

```
import sys
input = lambda: sys.stdin.readline().rstrip("\n")

T = int(input())
for _ in range(T):
    a, b = map(int, input().split())
    print(a + b)
```

대회 중 유의 사항

- 2023 숭실대학교 프로그래밍 대회(2023 SCON) 유의 사항입니다.
- 대회 정보
  - 이 대회는 숭실대학교 IT대학이 주최, 컴퓨터학부 문제해결 소모임 SCCC가 주관합니다.
  - 이 대회는 현대모비스와 스타트링크의 후원을 받아 진행됩니다.
- 사진 촬영 안내
  - 대회 당일 현장 스태프가 대회 현장을 촬영하여 사진으로 기록 및 온라인 게시할 예정입니다.
  - 사진은 추후 SCCC의 홍보와 SCCC가 주관하는 대회 홍보에 사용될 수 있습니다.
- 대회 진행 관련
  - 대회는 3시간 동안 10문제로 진행됩니다.
  - 6문제 이상 해결한 팀은 스코어보드 갱신이 중단됩니다.
  - 대회 종료 1시간 전부터 모든 팀의 스코어보드 갱신이 중단됩니다.
  - 대회 도중에는 대회 스태프와 팀원을 제외한 타인과 대화할 수 없습니다.
  - 대회 사이트 및 언어 공식 레퍼런스 사이트를 제외한 모든 인터넷의 사용은 금지됩니다.
    - `BOJ Help : https://help.acmicpc.net/language/info`
    - `C/C++ : https://en.cppreference.com/w/`
    - `Java : https://docs.oracle.com/en/java/javase/15/docs/api/`
    - `Python : https://docs.python.org/3/`
  - 화장실 이용 시 스태프와 동행해야 하며, 한 번에 한 명씩 이용 가능합니다.
  - 대회 도중 휴대전화 사용이 불가능합니다. 불가피한 경우 스태프의 감독하에 사용 가능합니다.
  - 문제와 관련된 질문은 대회 페이지의 '질문' 탭을 이용해야 합니다. 현장 스태프는 문제에 대한 질문을 받지 않습니다.
  - 대회 공지는 대회 페이지의 '공지' 탭을 이용해 전달합니다. 주기적으로 확인해 주시길 바랍니다.
- 문제 관련
  - 문제는 운영진들이 생각하는 난이도순으로 정렬되어 있습니다.
  - 모든 문제는 C++과 Java로 해결할 수 있음이 보장됩니다.
  - 제출한 프로그램은 문제에 명시된 제한 시간 내에 정답을 출력하고 정상적으로 종료되어야 합니다. 이는 return code가 0이어야 함을 의미하여, 이외의 exit code는 런타임 에러가 발생합니다.
  - 제출한 프로그램은 문제에 명시된 제한 메모리보다 많은 메모리를 사용할 수 없습니다.
  - 언어별 추가 시간과 추가 메모리가 주어지지 않습니다.
  - 제출한 프로그램은 표준 입력(standard input)을 통해 입력받아서 표준 출력(standard output)을 통해 정답을 출력해야 합니다.
  - 표준 입출력을 제외한 파일 입출력, 네트워킹, 멀티 스레딩 등의 시스템 콜은 사용할 수 없습니다.
- 팀의 등수는 다음과 같은 방법을 이용해 계산합니다.
  - 문제의 패널티 = (대회가 시작한 시점으로부터 처음으로 **맞았습니다!!**를 받기까지 걸린 분 단위 시간) + (제출 횟수 - 1) * 20분

  - 팀의 패널티 = **맞았습니다!!**를 받은 모든 문제의 패널티의 합
  - 팀의 등수 = (더 많은 문제를 푼 팀의 수) + (푼 문제의 개수가 동일하면서 패널티가 더 작은 팀의 수) + 1
  - 컴파일 에러는 패널티에 포함되지 않습니다.

간식 안내

- 제공되는 간식에 알레르기 유발 물질이 포함되어 있으니 주의해서 섭취하시길 바랍니다.
- 영양성분표 및 원재료명 관련 문의는 대회 페이지의 질문 기능을 통해 질문하면 답변드리겠습니다.
- 오예스
  - 우유, 밀, 계란, 대두 함유
  - 땅콩을 사용한 제품과 같은 제조시설에서 제조
- 트윅스
  - 밀, 우유, 대두 함유
  - 헤이즐넛 혹은 아몬드를 사용한 제품과 같은 제조시설에서 제조
- 쿠크다스 케이크
  - 밀, 우유, 대두, 계란, 돼지고기 함유
- 비쵸비 비스켓
  - 밀, 우유, 대두, 쇠고기 함유
  - 달걀, 땅콩, 호두, 복숭아, 토마토, 돼지고기, 닭고기, 오징어, 새우, 게, 조개류(굴, 홍합) 혼입 가능

2023 Soongsil Programming Contest

Problem List

문제지에 있는 문제가 총 10문제가 맞는지 확인하시길 바랍니다.

모든 문제는 C++17, Java 15, PyPy3으로 풀 수 있음을 보장합니다. (단, Python 3는 보장하지 않음)

I번 문제와 J번 문제는 입출력 양이 매우 많습니다. 2–4페이지에 있는 언어 가이드를 참고하시길 바랍니다.

A. 정보섬의 대중교통

숭실대학교 정보과학관은 숭실대입구역으로부터 멀리 떨어져 있는 대신, 바로 앞에 *숭실대별관앞*이라는 명칭의 버스 정류소가 자리 잡고 있다.

학부 연구생 찬솔이는 야근을 마치고 대중교통을 이용해 집에 가려고 한다. 다행히 아슬아슬하게 막차가 끊기지 않은 상황인데, 구체적으로 A분 뒤에 숭실대별관앞 정류소에 집으로 가는 마지막 버스가 도착하고, B분 뒤에 숭실대입구역에 집으로 가는 마지막 열차가 도착한다.

찬솔이는 지금 버스 정류소에 서 있다. 그런데, 찬솔이는 지하철 역까지 걸어가서 지하철을 타는 것이 여기서 버스를 타는 것 보다 빠를 수도 있다는 사실을 알아차려 버렸다. 숭실대입구역의 지하철 승강장까지 걸어가는데는 N분이 걸린다. 버스와 지하철 중 더 먼저 탈 수 있는 것이 무엇인지 알려줘서 야근한 찬솔이의 피로를 회복시켜 주자.

단, 버스와 지하철이 도착한 뒤 탑승하는 데 걸리는 시간은 신경 쓰지 않고, 버스와 지하철 모두 도착한 직후에 승객을 태운 뒤 기다리지 않고 바로 떠난다. 또한 지하철 역에 도착하는 시간과 지하철 열차가 도착하는 시간이 정확히 같다면 지하철을 탈 수 있다.

입력 형식

첫째 줄에 N, A, B가 공백으로 구분되어 주어진다.

출력 형식

버스에 더 먼저 탑승할 수 있으면 `Bus`, 지하철에 더 먼저 탑승할 수 있으면 `Subway`, 버스와 지하철에 탑승하게 되는 시간이 동일하면 `Anything`을 출력한다.

제한

- $1 \leq A \leq 10^6$
- $1 \leq N \leq B \leq 10^6$
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	표준 출력(stdout)
10 5 15	Bus
1 1 1	Anything
1 100 1	Subway

첫 번째 예시에서 버스는 5분 후에 탑승할 수 있고, 지하철은 10분 동안 역에 걸어간 다음 5분을 더 기다려야 하므로 15분 후에 탑승할 수 있다.

B. 팀명 정하기

> 현대 모비스는 모빌리티 SW 해커톤, 알고리즘 경진대회, 채용 연계형 SW 아카데미 등 다양한 SW 인재 발굴 프로그램을 진행하고 있다. 지난 2월에 개최된 모빌리티 SW 해커톤은 국내 14개 대학의 소프트웨어 동아리 20개 팀, 70여 명이 참여해 모빌리티 소프트웨어 개발 실력을 겨뤘다.

숭실대학교 컴퓨터학부 문제해결 소모임 SCCC 부원들은 매년 모빌리티 SW 해커톤, SCON, ICPC와 같은 팀 대회에서 사용할 팀명을 정하기 위해 많은 고민을 한다. 졸업을 한 학기 남겨둔 성서는 더 이상 부원들이 팀명으로 고통을 받지 않도록 가이드라인을 만들었다.

성서의 가이드라인에 따르면 팀 이름을 짓는 방법은 두 가지가 있다.

1. 세 참가자의 입학 연도를 100으로 나눈 나머지를 오름차순으로 정렬해서 이어 붙인 문자열
2. 세 참가자 중 성씨를 영문으로 표기했을 때의 첫 글자를 백준 온라인 저지에서 해결한 문제가 많은 사람부터 차례대로 나열한 문자열

예를 들어 600문제를 해결한 18학번 안(AHN)씨, 2000문제를 해결한 19학번 이(LEE)씨, 6000문제를 해결한 20학번 오(OH)씨로 구성된 팀을 생각해 보자. 첫 번째 방법으로 팀명을 만들면 181920이 되고, 두 번째 방법으로 팀명을 만들면 OLA가 된다.

2000문제를 해결한 19학번 이(LEE)씨, 9000문제를 21학번 나(NAH)씨, 1000문제를 해결한 22학번 박(PARK)씨로 구성된 팀은 첫 번째 방법으로 팀명을 만들면 192122가 되고, 두 번째 방법으로 팀명을 만들면 NLP가 된다.

세 팀원의 백준 온라인 저지에서 해결한 문제의 개수, 입학 연도, 그리고 성씨가 주어지면 첫 번째 방법과 두 번째 방법으로 만들어지는 팀명을 차례대로 출력하는 프로그램을 작성하라.

입력 형식

첫째 줄에 첫 번째 팀원이 백준 온라인 저지에서 해결한 문제의 개수 P_1, 입학 연도 Y_1, 성씨 S_1이 공백으로 구분되어 주어진다.

둘째 줄과 셋째 줄에는 두 번째 팀원의 정보 P_2, Y_2, S_2와 세 번째 팀원의 정보 P_3, Y_3, S_3이 첫째 줄과 같은 형식으로 주어진다.

출력 형식

첫째 줄에 첫 번째 방법으로 만든 팀명을 출력한다.

둘째 줄에 두 번째 방법으로 만든 팀명을 출력한다.

제한

- $1 \le P_i \le 20\,000$ $(1 \le i \le 3)$
- P_i는 모두 서로 다르다. $(1 \le i \le 3)$
- $2010 \le Y_i \le 2099$ $(1 \le i \le 3)$
- Y_i는 모두 서로 다르다. $(1 \le i \le 3)$
- 성씨는 알파벳 대문자로만 구성되어 있으며, 최대 5글자이다.
- 모든 성씨는 서로 다르다.
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	표준 출력(stdout)
600 2018 AHN 2000 2019 LEE 6000 2020 OH	181920 OLA
1000 2022 PARK 9000 2021 NAH 2000 2019 LEE	192122 NLP

참고

181920은 2020 ICPC Asia Seoul Regional Contest 동상 수상팀, NLP는 2022 ICPC Asia Seoul Regional Contest 은상 수상팀이다.

C. 등차수열의 합

수학과 전공과목인 조합론을 수강하는 정휘는 등차수열의 합 공식에 대해 배우고 있다. 2023 SCON 대회 개최가 일주일 남았지만, 아직 문제를 절반도 만들지 못해 발등에 불이 떨어진 정휘는 화장실에 가는 척을 하면서 정보과학관에 달려와 등차수열에 관한 문제를 만들었다.

길이가 N인 수열 A가 주어졌을 때, $1 \le i \le N$에 대해 $A_i = B_i + C_i$를 만족하고 길이가 N인 두 등차수열 B, C를 구하라.

등차수열의 정의는 다음과 같다.

- 어떤 수열 $A = \{A_1, A_2, \cdots, A_N\}$이 등차수열이라는 것은, $2 \le i \le N$인 모든 i에 대해 $A_i - A_{i-1}$이 모두 동일한 수열을 말한다. 정의에 따라 길이가 2 이하인 수열은 항상 등차수열이다.

입력 형식

첫째 줄에 수열 A의 길이 N이 주어진다.

둘째 줄에 수열 A의 원소 $A_1, A_2, \cdots, A_N$이 순서대로 공백으로 구분되어 주어진다.

출력 형식

만약 모든 $1 \le i \le N$에 대해 $A_i = B_i + C_i$인 길이가 N인 두 등차수열 B, C가 존재하지 않으면 첫째 줄에 `NO`를 출력한다.

그렇지 않으면 첫째 줄에 `YES`를 출력한다. 이후 둘째 줄에 B의 원소를, 셋째 줄에 C의 원소를 차례대로 공백으로 구분해서 출력한다. 가능한 수열 B, C가 여럿인 경우, 아무거나 하나만 출력한다. 수열 B, C가 존재할 경우, 문제의 제한을 만족하는 출력이 존재한다는 것을 증명할 수 있다.

제한

- $1 \le N \le 10^5$
- $-10^9 \le A_i \le 10^9$ $(1 \le i \le N)$
- $-10^{18} \le B_i, C_i \le 10^{18}$ $(1 \le i \le N)$
- A_i, B_i, C_i는 모두 정수 $(1 \le i \le N)$

예제

표준 입력(stdin)	표준 출력(stdout)
4 1 2 3 4	YES 2 4 6 8 -1 -2 -3 -4
3 1 2 1	NO
1 0	YES -3 3

D. 선택 정렬의 이동 거리

1부터 N까지의 정수가 한 번씩 등장하는 수열 A가 주어진다. 이 수열에서 선택 정렬 알고리즘을 수행할 때, 각 수의 이동 거리를 출력하라.

선택 정렬 알고리즘이 무엇인지 잘 모르는 친구들은 친절한 주원이가 준비한 아래 설명을 읽어보도록 하자.

- 길이가 N인 수열 $A = \{A_1, A_2, \cdots, A_N\}$을 오름차순으로 정렬하는 선택 정렬 알고리즘은 아래 동작을 $N-1$번 반복해서 수행한다.
  1. 지금이 i번째 동작이라면, $A_i, A_{i+1}, \cdots, A_N$ 중 최솟값 A_j를 찾는다.
  2. A_i와 A_j의 위치를 교환한다. 이때 A_i와 A_j의 이동 거리가 각각 $(j-i)$만큼 증가한다.

예를 들어 $\{1, 3, 5, 2, 4\}$와 같은 수열이 주어졌다고 하자. 처음에 모든 수의 이동 거리는 0으로 같다. 선택 정렬 알고리즘은 다음과 같은 과정을 거쳐 수행된다.

1. $A_1 = 1$과 $A_1 = 1$을 교환해서 $\{1, 3, 5, 2, 4\}$가 된다. 이때 1의 이동 거리는 0만큼 증가한다.
2. $A_2 = 3$과 $A_4 = 2$를 교환해서 $\{1, 2, 5, 3, 4\}$가 된다. 이때 2와 3의 이동 거리는 2만큼 증가한다.
3. $A_3 = 5$와 $A_4 = 3$을 교환해서 $\{1, 2, 3, 5, 4\}$가 된다. 이때 3과 5의 이동 거리는 1만큼 증가한다.
4. $A_4 = 5$와 $A_5 = 4$를 교환해서 $\{1, 2, 3, 4, 5\}$가 된다. 이때 4와 5의 이동 거리는 1만큼 증가한다.

따라서 1은 0만큼, 2는 2만큼, 3은 3만큼, 4는 1만큼, 5는 2만큼 이동한다.

입력 형식

첫째 줄에 수열의 길이 N이 주어진다.

둘째 줄에 수열의 원소 $A_1, A_2, \cdots, A_N$이 차례대로 공백으로 구분되어 주어진다.

출력 형식

첫째 줄에 N개의 정수를 공백으로 구분하여 출력한다. i번째 정수는 i의 이동 거리를 의미한다.

제한

- $1 \le N \le 5 \times 10^5$
- A에는 1부터 N까지의 정수가 정확히 한 번씩 등장한다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 1 2 3 4 5	0 0 0 0 0
5 1 3 5 2 4	0 2 3 1 2

E. prlong longf

성서는 **Bronze 5** 난이도의 문제를 풀다가 **틀렸습니다**를 받았다.

계산 도중 수가 너무 커져서 오버플로우가 발생했다고 생각한 성서는 코드 에디터의 "찾기 및 바꾸기" 기능을 사용해서 코드의 `int`를 모두 동시에 `long long`으로 바꾸었는데, `printf`도 모두 `prlong longf`로 바뀌는 사고가 일어났다!

```
#include <stdio.h>

long long main(){
    long long n, res = 1;
    scanf("%d", &n);
    for(long long i = 1; i <= n; i++){
        res *= i;
    }
    prlong longf("%d\n", res);
    return 0;
}
```

스스로 코드를 고치기 귀찮았던 성서는 대회 참가자들에게 바뀐 코드를 주고 초기 상태로 복원해 달라고 부탁하려고 했지만, 주어진 코드에 따라 복원 방법이 유일하지 않을 수 있다는 사실을 깨달았다. 좋은 문제 아이디어를 발견한 성서는 2023 SCON에 다음과 같은 문제를 출제했다.

모든 `int`가 `longlong`으로 바뀐 문자열이 주어진다. 가능한 원래 문자열은 모두 몇 가지인가?

입력 형식

첫째 줄에 바뀐 문자열의 길이 N이 주어진다.

둘째 줄에 `int`가 모두 `longlong`으로 바뀐 길이 N의 문자열이 주어진다.

출력 형식

문자열의 초기 상태로 가능한 경우의 수를 출력한다.

제한

- $1 \le N \le 80$
- 주어진 문자열의 모든 문자는 알파벳 소문자이고, 공백을 포함하지 않는다.
- 주어진 문자열은 `int`를 부분 문자열로 갖지 않는다.

예제

표준 입력(stdin)	표준 출력(stdout)
15 prlonglonglongf	3
15 longestpathtowf	1
22 longlongdoublelonglong	4

첫 번째 예시에서 가능한 초기 상태는 `printlongf`, `prlongintf`, `prlonglonglongf`로 총 3가지이다.

F. 안전한 건설 계획

숭실대학교는 새로운 고층 건물을 설계하고 있다.

현재까지 쌓아 올린 건물의 골조는 수직 방향으로 세운 N개의 기둥, 그리고 서로 다른 두 기둥을 수평으로 연결하는 M개의 빔으로 구성되어 있다. 즉, 현재까지 쌓아 올린 건물 골조의 평면도는 아래 그림처럼 기둥을 점의 형태로, 빔을 점과 점을 연결하는 선분 형태로 나타낼 수 있다. 각 기둥은 1번부터 N번까지 번호가 붙어 있고 각 빔도 1번부터 M번까지 번호가 붙어 있다.

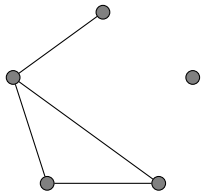


Figure 1: 5개의 기둥과 4개의 빔으로 구성된 골조

숭실대학교는 건물의 안정성을 강화하기 위해 몇 차례의 보강 작업을 진행해서 모든 서로 다른 두 기둥을 연결하는 빔이 존재하도록, 건물에 총 $N(N-1)/2$개의 빔을 설치하려고 한다. 즉, 기둥의 개수가 5개라고 하면 평면도가 아래 그림처럼 되도록 만들려고 한다.

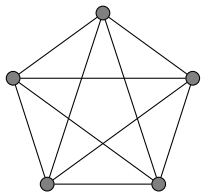
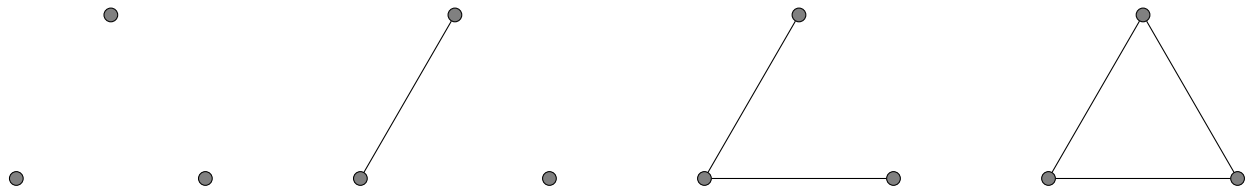


Figure 2: 5개의 기둥과 10개의 빔으로 구성된 골조

구조물의 하중을 지탱하기 위해, 보강 작업은 삼각형을 쌓아 나가는 방식으로 진행해야 한다. 구체적으로, 보강 작업은 서로 다른 세 기둥 a, b, c를 선택해서 $(a, b), (b, c), (c, a)$ 세 쌍을 모두 빔으로 연결하는 삼각형을 한 번에 쌓는 작업이다. 이때, 기존에 세 기둥 사이에 빔이 몇 개 있었는지에 따라 보강 작업에 필요한 비용이 달라진다.



(a) 작업 진행 불가능 (b) 작업 비용 1 (c) 작업 비용 0 (d) 작업 필요 없음

Figure 3: 빔 개수에 따른 보강 작업 비용

- 세 기둥 사이에 빔이 하나도 없는 경우: 작업의 위험도가 높으므로 진행할 수 없다.
- 세 기둥 사이에 1개의 빔이 있는 경우: 빔이 2개 추가되어 삼각형이 만들어지며, 이 작업에는 주의가 필요하므로 1만큼의 비용이 든다.
- 세 기둥 사이에 2개의 빔이 있는 경우: 빔이 1개 추가되어 삼각형이 만들어지며, 이 작업은 비교적 안전하므로 비용이 들지 않는다.

- 세 기둥 사이에 3개의 빔이 모두 있는 경우: 이미 삼각형이 구성되어 있으므로 아무것도 진행하지 않는다.

보강 작업을 0회 이상 원하는 만큼 진행해서, 모든 기둥 사이에 빔이 존재하도록 하는 최소 비용을 구하라. 모든 입력에 대해 서로 다른 두 기둥을 연결하는 빔이 항상 존재하도록 보강 작업을 진행할 수 있음이 보장된다.

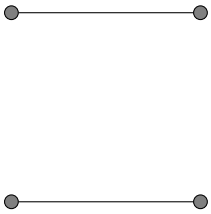
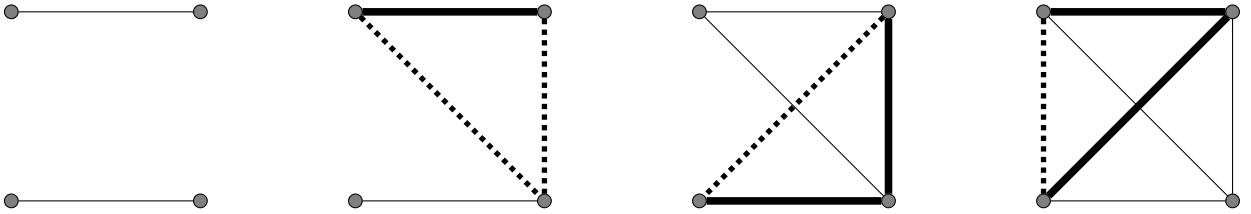


Figure 4: 예제 입력 1

예를 들어 골조의 평면도가 위 그림처럼 생겼고, 왼쪽 위에 있는 점부터 시계방향 순서대로 $1, 2, 3, 4$번 기둥이라고 해보자.

첫 번째 보강 작업에서 $1, 2, 3$번 기둥을 연결하는 삼각형을 만들고, 두 번째 보강 작업에서 $1, 2, 4$번 기둥을 연결하는 삼각형을 만들면 총합 $1 + 1 = 2$의 비용으로 완성할 수 있다.

반면, 첫 번째 보강 작업에서 $1, 2, 3$번 기둥, 두 번째 보강 작업에서 $2, 3, 4$번 기둥, 세 번째 보강 작업에서 $1, 2, 4$번 기둥을 선택하면 총합 $1 + 0 + 0 = 1$의 비용으로 완성할 수 있다.



(a) 작업 진행 전 (b) 첫 번째 작업 후 (c) 두 번째 작업 후 (d) 세 번째 작업 후

Figure 5: 예제 출력 1

이 밖에도 여러 방법이 있지만, 비용을 1보다 적게 사용하는 방법은 없으므로 1이 최소 비용이다.

입력 형식

첫째 줄에 건물의 골조를 구성하는 기둥의 개수 N과 빔의 개수 M이 공백으로 구분되어 주어진다.

둘째 줄부터 $M + 1$번째 줄까지, $i + 1$번째 줄에 i번째 빔이 연결하는 두 기둥의 번호 a_i, b_i가 공백으로 구분되어 주어진다.

출력 형식

보강 작업을 최적으로 진행할 때, 모든 서로 다른 기둥을 연결하는 빔이 존재하도록 만드는 최소 비용을 출력한다.

제한

- $3 \leq N \leq 40$
- $1 \leq M \leq N(N-1)/2$
- $1 \leq a_i < b_i \leq N$ $(1 \leq i \leq M)$
- $i \neq j$ 이면 $(a_i, b_i) \neq (a_j, b_j)$이다. $(1 \leq i, j \leq M)$ 즉, 연결하는 두 기둥의 쌍이 동일한 빔이 여러 개 존재하지 않는다.
- 서로 다른 두 기둥을 연결하는 빔이 항상 존재하도록 보강 작업을 진행할 수 있음이 보장된다.
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	**표준 출력**(stdout)
4 2 1 2 3 4	1
4 6 1 2 1 3 1 4 2 3 2 4 3 4	0
4 3 1 2 2 3 3 4	0

세 번째 예시는 첫 번째 보강 작업에서 $1, 2, 3$번 기둥, 두 번째 보강 작업에서 $1, 3, 4$번 기둥, 세 번째 보강 작업에서 $1, 2, 4$번 기둥을 선택하면 0만큼의 비용이 든다.

G. Traveling SCCC President

숭실대학교 컴퓨터학부 문제해결 소모임 SCCC의 회장인 찬솔이는 대회를 성공적으로 개최하기 위해 학교의 여러 건물에 들려 업무를 보고 있다.

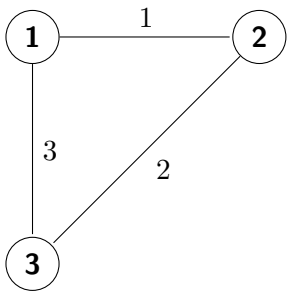
숭실대학교의 캠퍼스는 1번부터 N번까지 번호가 붙어 있는 N개의 건물과 서로 다른 두 건물을 연결하고 1번부터 M번까지 번호가 붙어 있는 M개의 도로로 구성되어 있다. i번 도로는 u_i번 건물과 v_i번 건물을 연결하고, 이 도로를 이용해 한 쪽 건물에서 다른 쪽 건물로 이동하는데 w_i분이 걸린다. 모든 건물은 도로를 통해 이어져 있다.

대회를 개최하기 위해서는 N개의 건물에서 한 번씩 **차례대로** 회의를 진행해야 한다. 구체적으로, 회의를 진행해야 하는 건물의 순서 $A_1, A_2, \cdots, A_N$이 주어진다. i번째로 진행해야 하는 회의는 A_i번 건물에서 진행되며, 모든 A_i는 서로 다르다.

4차 산업혁명이 도래한 21세기에 매번 도로를 통해 이동하는 것은 비효율적이기 때문에, 찬솔이는 순간 이동 장치를 만들었다. 순간 이동 장치를 사용하면 지금까지 방문했던 건물 중 원하는 곳으로 순식간에 이동할 수 있다.

찬솔이는 S번 건물에서 출발해서 N개의 회의를 마친 다음 다시 S번 건물로 돌아와야 한다. 시간이 부족한 찬솔이를 위해 N개의 회의를 차례대로 모두 마치고 S번 건물로 돌아오는데 걸리는 최소 시간을 구해주자.

예를 들어 숭실대학교 캠퍼스가 아래 그림과 같은 형태이고 1번 건물에서 출발하며, $1, 3, 2$번 건물에서 차례대로 회의를 진행해야 한다고 하자.



찬솔이는 먼저 1번 건물에서 회의를 진행한 뒤, 도로를 통해 2번 건물을 거쳐 3번 건물로 이동해서 회의를 진행한다. 이후 순간 이동 장치를 사용해 2번 건물로 이동해 회의를 진행한 뒤, 순간 이동 장치를 써서 1번 건물로 돌아올 수 있다. 이 과정에서 소요되는 총 시간은 3분이다.

입력 형식

첫째 줄에 건물의 개수 N, 도로의 개수 M, 출발하는 건물의 번호 S가 공백으로 구분되어 주어진다.

둘째 줄부터 M개의 줄에 걸쳐, i번 도로가 연결하는 두 건물의 번호 u_i, v_i와 도로를 통행하는 데 걸리는 시간 w_i가 한 줄에 하나씩 공백으로 구분되어 주어진다.

$M+2$번째 줄에는 N개의 정수 $A_1, A_2, \cdots, A_N$이 주어진다. 이는 i번째 회의가 A_i번 건물에서 진행됨을 의미한다.

출력 형식

S번 건물에서 출발해서 N개의 회의를 **차례대로** 마친 다음에 다시 S번 건물로 돌아오는 데 필요한 최소 시간을 출력한다.

제한

- $2 \le N \le 2\,000$
- $N - 1 \le M \le 5\,000$
- $1 \le S \le N$
- $1 \le u_i < v_i \le N$ $(1 \le i \le M)$
- $1 \le w_i \le 5\,000$ $(1 \le i \le M)$
- A에는 1부터 N까지의 정수가 정확히 한 번씩 등장한다.
- 모든 건물은 도로를 통해 이어져 있다.
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	**표준 출력**(stdout)
4 6 1 1 2 1 2 3 2 3 4 3 1 4 4 2 4 5 1 3 2 1 2 3 4	6
3 3 1 1 2 1 2 3 2 1 3 3 1 3 2	3

H. SCCC 신입 부원 모집하기

SCCC의 회장인 찬솔이는 2023년 1학기 신입 부원 선발 과정을 진행하고 있다. SCCC는 2023년 1학기에 정원이 X명인 스터디 그룹을 K개 운영하고, 이번 학기에 새로 들어오는 신입 부원들은 K개의 스터디 그룹 중 정확히 하나의 그룹에 참가해야 한다. 참가할 수 있는 스터디 그룹이 없으면 합격할 수 없으므로 모든 지원자는 본인이 참가할 수 있는 스터디 그룹을 한 개 이상 선택했다.

찬솔이는 공정하고 엄격한 심사를 통해 N명의 지원자들에게 서로 다른 점수를 매겼다. 이제 신입 부원들을 스터디에 배정하기 위해 점수가 높은 사람에서 낮은 사람 순으로 정렬한 다음, 한 명씩 스터디 그룹에 배정하려고 한다.

구체적으로, 점수가 i번째로 높은 사람을 스터디 그룹에 배정하려고 하는 상황을 생각해 보자. 만약 지금까지 스터디에 배정된 사람들의 집합을 유지하면서 i번째 사람을 스터디에 넣을 수 있으면 i번째 사람을 스터디에 배정한다. 반대로, 지금까지 스터디에 배정된 사람들을 어떻게 이동시키더라도 i번째 사람이 들어갈 수 있는 자리가 없다면 i번째 사람을 스터디에 배정될 수 없다.

예를 들어 $N = 5, K = 2, X = 2$이고, 점수가 높은 사람부터 각 지원자가 참가할 수 있는 스터디 그룹이 $\{1\}, \{1, 2\}, \{1\}, \{1\}, \{2\}$라고 하자.

점수가 가장 높은 사람과 두 번째로 높은 사람이 모두 1번 그룹에 배정되었고, 세 번째로 높은 사람을 배정해야 하는 상황을 생각해 보자. 두 번째 사람의 스터디 그룹을 2번 그룹으로 옮긴 뒤 세 번째 사람을 1번 그룹에 배정하면, 이전까지 스터디에 배정된 사람들의 집합을 유지하면서 세 번째 사람을 스터디에 넣을 수 있다.

반면, 1번 그룹에 첫 번째와 세 번째 사람, 2번 그룹에 두 번째 사람이 배정된 상황에서 네 번째 사람을 배정할 방법은 존재하지 않는다. 하지만 점수가 가장 낮은 사람은 2번 그룹에 들어갈 수 있으므로 최종 결과는 1번 그룹에 첫 번째와 세 번째 사람, 2번 그룹에 두 번째 사람과 다섯 번째 사람이 배정되는 것이다.

찬솔이는 이 규칙에 따라 지원자들을 스터디 그룹에 배정하려고 한다. 이때 스터디 그룹에 배정되는 사람의 수와, 각 스터디 그룹에 배정된 지원된 사람의 목록을 구해야 한다. 신입 부원 선발 외에도 많은 일을 처리해야 하는 찬솔이를 위해 스터디 배정 프로그램을 대신 작성해 주자.

입력 형식

첫째 줄에 지원자 수 N, 스터디 그룹의 개수 K, 각 스터디 그룹의 정원 X가 공백으로 구분되어 주어진다.

둘째 줄부터 N개의 줄에 걸쳐, i번째 줄에 i번째 지원자의 정보가 주어진다. $(1 \le i \le N)$

지원자의 정보는 한 줄로 구성되어 있다. 각 줄의 처음에는 참가할 수 있는 스터디의 개수 C_i가 주어지고, 이어서 참가할 수 있는 C_i개의 스터디 그룹의 번호 $A_{i,1}, A_{i,2}, \cdots, A_{i,C_i}$가 공백으로 구분되어 주어진다.

$N + 2$번째 줄에는 각 학생의 점수 $B_1, B_2, \cdots, B_N$이 공백으로 구분되어 주어진다.

출력 형식

첫째 줄부터 K개의 줄에 걸쳐, i번째 줄에 i번 그룹에 배정된 사람의 정보를 출력한다. $(1 \le i \le K)$

각 줄의 처음에는 해당 스터디 그룹에 배정된 사람의 수, 그리고 이어서 배정된 사람의 번호를 공백으로 구분하여 출력한다.

답이 여러 개 존재하면 아무거나 출력해도 되며, 배정된 사람의 번호를 출력하는 순서는 무관하다.

제한

- $1 \leq N, K, X \leq 15$
- $1 \leq K \times X \leq 15$
- $1 \leq C_i \leq K$
- $1 \leq A_{i,j} \leq K$ $(1 \leq i \leq N, 1 \leq j \leq C_i)$
- $j \neq k$ 이면 $A_{i,j} \neq A_{i,k}$이다. $(1 \leq i \leq N, 1 \leq j, k \leq C_i)$ 즉, 각 행의 원소는 모두 서로 다르다.
- $1 \leq B_i \leq 10^9$ $(1 \leq i \leq N)$
- $i \neq j$ 이면 $B_i \neq B_j$이다. $(1 \leq i, j \leq N)$ 즉, B의 원소는 모두 서로 다르다.
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	**표준 출력**(stdout)
5 2 2 1 1 2 1 2 1 1 1 1 1 2 5 4 3 2 1	2 1 3 2 2 5

I. 산책과 쿼리

성서는 숭실대학교 근처 자취 후보 장소를 N개 선정해 1부터 N까지 번호를 붙였다. 각 장소에는 자취방이 정확히 하나씩 있다. 자취방을 구할 때 고려할 사항은 여러 가지가 있지만, 성서는 그중에서도 산책의 자유도를 가장 중요하게 생각한다.

산책로 (a, b)는 서로 다른 두 장소 a와 b를 잇는 길로, a에서 b로 걷는 데 1의 시간이 걸리고, b에서 a로 걷는 데에도 1의 시간이 걸린다. 일단 걷기 시작하면 중간에 방향을 꺾거나 멈추지 않고 정확히 1의 시간 동안 걸어서 반대편에 도착해야 한다.

산책은 어떤 장소 u에서 출발해 산책로를 따라 걸어 다니다가 마지막에 다시 u로 되돌아오는 행동이다. 산책 도중에 어떤 장소나 산책로를 여러 번 방문해도 되지만, 산책 도중에 잠시라도 걸음을 멈추면 안 된다.

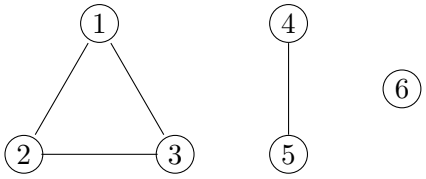
성서는 장소들 사이를 둘러보며 분위기가 좋은 산책로들을 자신만의 산책로 리스트에 추가할 계획이다. **만족스러운 산책**이란, 산책로 리스트에 적힌 산책로만을 사용하는 산책이다.

성서는 충분히 긴 시간 동안 산책을 하다가 원하는 시간에 맞춰 되돌아오고 싶다. 어떤 자취방 u가 **산책의 자유도가 높다**는 것은, 10^6 이상의 어떤 정수 t를 고르더라도, 장소 u에서 출발하면서 정확히 t의 시간이 걸리는 만족스러운 산책이 존재함을 의미한다.

처음에는 산책로 리스트가 비어 있지만, 성서는 분위기가 좋은 산책로를 발견할 때마다 그 산책로를 리스트에 추가할 것이다. 리스트가 변경될 때마다 산책의 자유도가 높은 자취방이 몇 개나 있는지 알아보자.

구체적으로, 다음과 같은 쿼리를 Q번 처리해야 한다.

- `a b`: 산책로 (a, b)를 산책로 리스트에 추가한다. 이후 산책의 자유도가 높은 자취방의 개수를 출력한다.



예를 들어 자취방이 6개이고 산책로 리스트가 $\{(1, 2), (2, 3), (1, 3), (4, 5)\}$라면 위 그림처럼 표현할 수 있다.

자취방 6에서는 10^6의 시간이 걸리는 만족스러운 산책을 할 수 없으므로 산책의 자유도가 높지 않다.

자취방 4와 5에서는 10^6이나 $10^6 + 2$ 등의 시간이 걸리는 만족스러운 산책은 할 수 있지만, $10^6 + 1$이나 $10^6 + 3$ 등의 시간이 걸리는 만족스러운 산책은 할 수 없으므로 산책의 자유도가 높지 않다.

반면에 자취방 $1, 2, 3$은 10^6이나 $10^6 + 1, 10^6 + 2, 10^6 + 3, \cdots$의 시간이 걸리는 만족스러운 산책이 모두 가능하므로 산책의 자유도가 높다.

따라서 위 그림에서 산책의 자유도가 높은 자취방의 개수는 3이다.

입력 형식

첫째 줄에 장소의 개수 N과 쿼리의 개수 Q가 공백으로 구분되어 주어진다.

둘째 줄부터 Q개의 줄에 걸쳐, i번째 줄에 i번째 쿼리에서 추가되는 산책로가 연결하는 두 장소 a_i, b_i가 공백으로 구분되어 주어진다.

출력 형식

쿼리가 주어질 때마다 한 줄에 하나씩 정답을 출력한다.

제한

- $2 \le N \le 3 \times 10^5$
- $1 \le Q \le 6 \times 10^5$
- $1 \le a_i < b_i \le N$ $(1 \le i \le Q)$
- 동일한 쿼리가 여러 번 주어지지 않는다.
- 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	표준 출력(stdout)
6 5 1 2 2 3 1 3 4 5 2 5	0 0 3 3 5
4 4 1 2 2 3 3 4 1 4	0 0 0 0

참고

입출력 양이 많으므로 문제지 2–4페이지의 언어 가이드에 있는 빠른 입출력을 사용하는 것을 권장한다.

J. 아이템

좌우로 무한히 긴 수직선 위에 N개의 아이템이 떨어져 있다. i번째 아이템의 위치는 X_i이며, 여러 개의 아이템이 한 곳에 있을 수 있다. 현재 위치 0에 있는 주원이는 최대한 많은 개수의 아이템을 줍고 싶다.

처음에 주원이는 한 번 이동할 때마다 1씩 왼쪽 또는 오른쪽으로 이동할 수 있다. 단, 아이템을 하나 주울 때마다 이동 거리가 2배씩 늘어난다. 즉, 현재까지 획득한 아이템이 k개라면 수직선 위에서 한 번 이동할 때마다 2^k씩 왼쪽 또는 오른쪽으로 이동할 수 있다. 단, 이동 중에는 중간에 멈출 수 없다.

아이템은 해당 아이템이 존재하는 위치에 멈춰야 주울 수 있으며, 아이템이 있는 위치에 멈췄더라도 아이템을 줍지 않을 수 있다. 아이템을 주우면 해당 아이템은 그 자리에서 사라진다.

주울 수 있는 아이템의 최대 개수를 구해보자.

입력 형식

첫째 줄에 아이템의 개수 N이 주어진다.

둘째 줄에 아이템의 위치 $X_1, X_2, \cdots, X_N$이 공백으로 구분되어 주어진다.

출력 형식

주울 수 있는 아이템의 최대 개수를 출력한다.

제한

- $1 \le N \le 2 \times 10^5$
- $0 \le X_i \le 10^{18}$ $(1 \le i \le N)$
- 입력으로 주어지는 수는 모두 정수이다.

예제

표준 입력(stdin)	표준 출력(stdout)
5 0 1 3 8 9	3
5 5 39 19 27 11	5
3 0 0 1000000000000000000	3

첫 번째 예시에서 $0 \to 1 \to 2 \to \mathbf{3} \to \mathbf{1} \to 5 \to \mathbf{9}$ 순서로 이동해서 3개의 아이템을 주울 수 있다. 굵게 표시된 수가 아이템을 주운 시점이다.

두 번째 예시에서 $0 \to 1 \to \cdots \to 4 \to \mathbf{5} \to 7 \to \cdots \to 37 \to \mathbf{39} \to 35 \to \cdots \to 23 \to \mathbf{19} \to \mathbf{27} \to \mathbf{11}$ 순서로 이동해서 5개의 아이템을 주울 수 있다. 굵게 표시된 수가 아이템을 주운 시점이다.

참고

입출력 양이 많으므로 문제지 2–4페이지의 언어 가이드에 있는 빠른 입출력을 사용하는 것을 권장한다.