Sogang ICPC Team

정보통신기획평가원

서강대학교 SOGANG UNIVERSITY

2020 The 0x10th

Sogang Programming Contest

2020 서강대학교 프로그래밍 대회

Champion
Official Problemset

STARTLINK

대회 규칙

- 대회 중 제출한 소스코드는 채점 서버에 의해 자동으로 채점되며, 실시간으로 결과를 알 수 있습니다. 문제를 제출하였을 때 '맞았습니다!!'를 받으면 문제를 푼 것으로, 이외의 결과를 받으면 틀린 것으로 생각합니다.
- 문제를 풀 때마다 패널티 점수가 누적됩니다. 패널티 점수는 모든 맞은 문제에 대해, 대회 시작 시간부터 그 문제를 풀기까지 걸린 시간을 t분, 처음으로 문제를 맞기 직전까지 제출한 횟수를 w번이라고 할 때 $(t+20w)$점입니다.
- 순위는 푼 문제가 많은 순서대로, 푼 문제가 같을 경우에는 패널티 점수의 합이 적은 순서대로 결정됩니다.
- 사용 가능 언어는 C, C++, Java, Python 3, Kotlin입니다. 모든 문제는 출제진이 C++과 Java(혹은 Kotlin), PyPy 3으로 정답을 작성했음이 보장됩니다. 각 언어의 컴파일 옵션과 컴파일러 버전은 아래와 같습니다.

C11 gcc (GCC) 10.2.0

컴파일 `gcc Main.c -o Main -O2 -Wall -lm -static -std=gnu11 -DONLINE_JUDGE -DBOJ`

실행 `./Main`

C++11 gcc (GCC) 10.2.0

컴파일 `g++ Main.cc -o Main -O2 -Wall -lm -static -std=gnu++11 -DONLINE_JUDGE -DBOJ`

실행 `./Main`

C++14 gcc (GCC) 10.2.0

컴파일 `g++ Main.cc -o Main -O2 -Wall -lm -static -std=gnu++14 -DONLINE_JUDGE -DBOJ`

실행 `./Main`

C++17 gcc (GCC) 10.2.0

컴파일 `g++ Main.cc -o Main -O2 -Wall -lm -static -std=gnu++17 -DONLINE_JUDGE -DBOJ`

실행 `./Main`

C++20 gcc (GCC) 10.2.0

컴파일 `g++ Main.cc -o Main -O2 -Wall -lm -static -std=gnu++20 -DONLINE_JUDGE -DBOJ`

실행 `./Main`

Java 8 openjdk 15 2020-09-15

컴파일 `javac -release 8 -J-Xms1024m -J-Xmx1920m -J-Xss512m -encoding UTF-8 Main.java`

실행 `java -Xms1024m -Xmx1920m -Xss512m -Dfile.encoding=UTF-8 -XX:+UseSerialGC -DONLINE_JUDGE=1 -DBOJ=1 Main`

Python 3 Python 3.8.2

컴파일 `python3 -c "import py_compile; py_compile.compile(r'Main.py')"`

실행 `python3 Main.py`

PyPy3 PyPy 7.3.2-alpha0 with GCC 7.3.1 20180303 (Red Hat 7.3.1-5) (Python 3.7.4)

컴파일 `python3 -c "import py_compile; py_compile.compile(r'Main.py')"`

실행 `pypy3 Main.py`

Kotlin/JVM kotlinc-jvm 1.4.10 (JRE 1.8.0_201-b09)

컴파일 `kotlinc-jvm -J-Xms1024m -J-Xmx1920m -J-Xss512m -include-runtime -d Main.jar Main.kt`

실행 `java -Xms1024m -Xmx1920m -Xss512m -Dfile.encoding=UTF-8 -XX:+UseSerialGC -DONLINE_JUDGE=1 -DBOJ=1 -jar Main.jar`

- 시간 제한에 '추가 시간 있음'이 표시된 일부 문제들은 **언어별로 추가 시간 및 메모리를 제공합니다.** 변경하여 적용되는 시간 및 메모리 제한은 언어별로 다음과 같습니다.
  - **Java 8, Kotlin/JVM**: 시간 $\times 2 + 1$초, 메모리 $\times 2 + 16$MB
  - **Python 3**: 시간 $\times 3 + 2$초, 메모리 $\times 2 + 32$MB
  - **PyPy3**: 시간 $\times 3 + 2$초, 메모리 $\times 2 + 128$MB
- 네트워크 사용은 금지됩니다. 단, 솔루션을 제출하거나 언어 레퍼런스를 확인하는 것은 가능합니다. 접속이 허용된 사이트의 예는 다음과 같습니다.

C/C++ https://en.cppreference.com/w/

Java https://docs.oracle.com/javase/8/docs/api/

Python https://docs.python.org/3/

Kotlin https://kotlinlang.org/docs/reference/

- 대회 종료 전에 퇴실할 수 없습니다.
- 책이나 개인이 준비한 인쇄된 참고자료를 열람할 수 있습니다. 참고자료의 양에는 제한이 없습니다.
- 대회 중에 휴대폰 및 전자기기는 사용할 수 없습니다.

문제 목록

문제지에 있는 문제가 총 8문제가 맞는지 확인하시기 바랍니다.

A 파일 정리

B 컨설팅

C 연료가 부족해

D 에어컨 설치

E 사탕 배달

F 폰친구

G Confuzzle

H 파인애플 피자

문제 A. 파일 정리

시간 제한	3 초
메모리 제한	1024 MB

친구로부터 노트북을 중고로 산 스브러스는 노트북을 켜자마자 경악할 수밖에 없었다. 바탕화면에 온갖 파일들이 정리도 안 된 채 가득했기 때문이다. 그리고 화면의 구석에서 친구의 메시지를 확인할 수 있었다.

> 바탕화면의 파일들에는 값진 보물에 대한 정보가 들어 있어. 하나라도 지우게 된다면 보물은 물론이고 다시는 노트북을 쓸 수 없게 될 거야. 파일들을 잘 분석해서 보물의 주인공이 될 수 있길 바랄게. 힌트는 "확장자"야.

화가 났던 스브러스는 보물 이야기에 금세 화가 풀렸고 보물의 정보를 알아내려고 애썼다. 하지만 파일이 너무 많은 탓에 이내 포기했고 보물의 절반을 보상으로 파일의 정리를 요청해왔다. 스브러스의 요청은 다음과 같다.

- 파일을 확장자 별로 정리해서 몇 개씩 있는지 알려줘
- 보기 편하게 확장자들을 사전 순으로 정렬해 줘

그럼 보물의 절반을 얻어내기 위해 얼른 스브러스의 노트북 파일 정리를 해줄 프로그램을 만들자!

입력

첫째 줄에 바탕화면에 있는 파일의 개수 N이 주어진다. $(1 \le N \le 50\,000)$

둘째 줄부터 N개 줄에 바탕화면에 있는 파일의 이름이 주어진다. 파일의 이름은 알파벳 소문자와 점(.)으로만 구성되어 있다. 점은 정확히 한 번 등장하며, 파일 이름의 첫 글자 또는 마지막 글자로 오지 않는다. 각 파일의 이름의 길이는 최소 3, 최대 100이다.

출력

확장자의 이름과 그 확장자 파일의 개수를 한 줄에 하나씩 출력한다. 확장자가 여러 개 있는 경우 확장자 이름의 사전순으로 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
8 sbrus.txt spc.spc acm.icpc korea.icpc sample.txt hello.world sogang.spc example.txt	icpc 2 spc 2 txt 3 world 1

노트

엄청난 보물의 정체는 바탕화면을 정리했다는 뿌듯함이라고 알려진다.

이 페이지는 공백입니다

문제 B. 컨설팅

시간 제한	2 초
메모리 제한	512 MB

Sogang ICPC Team에서는 학회원들을 돕기 위해 Sogang Program Consulting Team(이하 SPC Team)을 만들었다. SPC Team은 학회원들과 화목하게 지내게 될 날만을 상상하며 에러가 발생한 코드를 무료로 디버깅해주는 컨설팅을 바로 시작했다.

그러던 어느 날, 기세등등했던 SPC Team의 모두를 당황시킨 코드가 등장했다. 아무리 봐도 정상적인 코드인데, 원하는 데이터를 얻을 수 없었던 것이다. 하지만 포기를 모르는 SPC Team은 계속해서 디버깅을 시도한 끝에, 한 번에 여러 줄의 명령이 실행되고 있었다는 사실을 알게 되었다! 이 상황을 이해하기 위해 다음 예시를 살펴보자.

```
1: WRITE A TO B
2: WRITE B TO C
3: READ B
4: READ C
5: EXIT
```

위 코드는 문제가 된 학회원의 코드이다. 명령어가 순서대로 실행되면 전혀 문제가 없을 코드지만, 줄 1–4가 동시에 실행된다면 문제가 생긴다. 메모리 A에서 메모리 B로 데이터가 옮겨지지도 않았는데 두 번째 줄이 실행되면, 메모리 C에 무슨 데이터가 들어갈지 알 수 없다. 이러한 문제를 확인한 SPC Team은 다음과 같이 컨설팅을 해 주었다.

```
1: WRITE A TO B
2: WAIT
3: WRITE B TO C
4: WAIT
5: READ B
6: READ C
7: EXIT
```

위와 같이, 중간에 `WAIT`를 삽입하여 `WRITE A TO B`와 `WRITE B TO C`가 동시에 실행되는 것을 막아준다면, 메모리 C에 어떤 데이터가 들어갈지 명확해진다! 위 코드에 대한 컨설팅을 끝마친 SPC Team은 문제가 발생할 수 있는 경우를 다음과 같이 세 가지로 분류했다.

1. READ with WRITE
    - `WRITE A TO B`와 `READ B`가 동시에 실행되면, 메모리 B의 데이터가 확실하지 않으므로 두 명령어 사이에 `WAIT`가 있어야 한다.
    - `WRITE A TO B`와 `WRITE B TO C`가 동시에 실행되면, 메모리 C의 데이터가 확실하지 않으므로 두 명령어 사이에 `WAIT`가 있어야 한다.
2. WRITE with WRITE
    - `WRITE A TO C`와 `WRITE B TO C`가 동시에 실행되면, 메모리 C의 데이터가 확실하지 않으므로 두 명령어 사이에 `WAIT`가 있어야 한다.
3. 교착 상태
    - `WRITE A TO B`와 `WRITE B TO A`가 동시에 실행되면, 메모리 A와 메모리 B의 값이 확실하지 않으므로 두 명령어 사이에 `WAIT`가 있어야 한다.

이 문제를 겪고 있는 학회원들이 지속적으로 SPC Team에 컨설팅 문의를 신청하고 있다. 반복되는 작업에 지친 SPC Team은 위와 같은 상황을 알아서 탐지하여 컨설팅해주는 프로그램을 만들고자 한다. 하지만 너무나도 바쁜 나머지, 유능한 프로그래머인 당신에게 프로그램의 제작을 의뢰했다. 너무나도 마음이 상냥한 당신은 이 의뢰를 거절할 수 없다!

입력

입력으로 최대 10 000줄의 명령어가 주어지며, WRITE문, READ문, EXIT문으로 구성된다. EXIT문은 마지막에 한 번만 주어진다.

각 명령어는 다음과 같이 정의되며, 메모리 이름은 1–3글자의 알파벳 대문자로 구성되어 있다.

- `WRITE A TO B`: 메모리 A의 내용을 메모리 B로 옮긴다. 이 때, 메모리 A는 READ 상태가 된다.
- `READ A`: 메모리 A의 데이터를 읽는다.
- `EXIT`: 프로그램을 종료한다.

`WRITE A TO A`같이 동일한 메모리로 WRITE를 수행하는 경우는 없다.

출력

WAIT을 최소로 사용한 컨설팅 결과를 기존 명령어들의 순서를 유지하여 출력한다.

한 줄에 하나의 명령어만 출력해야 하며, 만약 그러한 컨설팅 결과가 여러 개라면 그 중 하나를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
WRITE A TO B WRITE B TO C READ B READ C EXIT	WRITE A TO B WAIT WRITE B TO C WAIT READ B READ C EXIT
WRITE EAX TO ECX WRITE ESP TO ESI READ EAX WRITE EDX TO EBP WRITE EDI TO ESI READ ECX WRITE ECX TO EAX READ EBP WRITE ESP TO EBP WRITE EBP TO EDX EXIT	WRITE EAX TO ECX WRITE ESP TO ESI READ EAX WRITE EDX TO EBP WAIT WRITE EDI TO ESI READ ECX WRITE ECX TO EAX READ EBP WAIT WRITE ESP TO EBP WAIT WRITE EBP TO EDX EXIT

문제 C. 연료가 부족해

시간 제한	1 초 **(추가 시간 있음)**
메모리 제한	1024 MB

피라미드를 직접 보는 게 소원이었던 향빈이는 사막 투어 여행패키지를 신청하게 되었다. 그리고 여행 둘째 날에 사막 입구에 도착해서 사막 투어용 자동차에 탑승했다.

출발하기 전 들뜬 마음으로 자동차 안에 앉아있던 향빈이는 사막 투어 가이드북을 발견했다. 가이드북 안의 $R \times C$ 크기의 지도에는 연료 보관소 N곳의 위치와 연료 보관소마다 보관하고 있는 연료량이 표시되어 있었다.

평소에 자동차 덕후였던 향빈이는 모든 자동차의 연비를 외우고 있었고, 현재 탑승한 자동차가 1만큼의 거리를 움직일 때 연료를 1만큼 소비한다는 것을 알고 있다. 또한, 자동차는 지도의 x축 또는 y축과 평행한 방향으로만 주행한다. 따라서 예를 들어 자동차가 $(0,0)$에서 (i,j)까지 최단 거리로 움직인다면 연료는 $i+j$만큼 소비할 것이다.

현재 자동차의 연료가 없는 관계로, 여기에 있는 주유소에서 연료를 충전하고 나서 출발하려 한다. 하지만 주유소에서 파는 연료는 매우 비싸기 때문에, 향빈이는 여기서는 연료를 되도록 최소로 충전하고 이후에는 이동하면서 방문하는 연료 보관소에서 연료를 충전할 계획이다.

향빈이는 얼른 피라미드를 보고 싶기 때문에 운전사분께 피라미드와 멀어지는 방향으로는 운전하지 말아 달라고 부탁했다. 즉, 자동차는 오직 오른쪽이나 아래쪽으로만 이동한다.

주유소에서 충전 가능한 연료량에는 제한이 없으며, 현재 위치와 피라미드가 있는 위치에는 연료 보관소가 없다. 또한, 한 위치에 두 개 이상의 연료 보관소가 있는 경우는 없다.

문제는 연료 보관소마다 위치와 보관된 연료량이 다르기 때문에, 연료보관소들을 어떤 순서로 경유해서 이동하는가에 따라 처음 충전해야 하는 연료량이 달라질 수 있다는 점이다. 현 위치인 $(1,1)$에서 피라미드가 있는 지점인 (R,C)까지 가기 위해 주유소에서 충전해야 하는 연료량의 최솟값을 구해보자.

입력

첫째 줄에 지도의 세로 길이와 가로 길이를 나타내는 정수 R, C가 주어진다. $(2 \le R,C \le 3\,000)$

둘째 줄에 지도에 표시된 연료 보관소의 개수를 나타내는 정수 N이 주어진다. $(0 \le N \le 1\,000)$

셋째 줄부터 N개 줄에 각 연료 보관소의 위치를 나타내는 정수 좌표 (r,c)와 보관중인 연료량을 나타내는 정수 f가 주어진다. $(1 \le r \le R, 1 \le c \le C, 0 \le f \le 100)$

출력

향빈이가 현 위치인 $(1,1)$에서 피라미드가 있는 지점인 (R,C)까지 가기 위해 주유소에서 충전해야 하는 연료량의 최솟값을 출력한다.

(뒷면에 입출력 예시가 있습니다.)

입출력 예시

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

$(1,1) \to (2,2) \to (4,3) \to (5,5)$의 경로를 선택하면, $(1,1)$에서 $(2,2)$까지 가는 데 2만큼의 연료가 필요하고, $(2,2)$에서 $(4,3)$로 갈 때는 $(2,2)$의 연료 보관소에서 1만큼의 연료를 충전할 수 있으므로 추가로 2만큼의 연료가 필요하다. 마지막으로 $(4,3)$에서 $(5,5)$로 갈 때는 $(4,3)$에 있는 연료 보관소에서 4만큼의 연료를 충전할 수 있으므로 추가 연료가 필요하지 않다. 따라서 주유소에서는 연료를 4만큼만 충전하면 되며, 이 경우가 최소이다.

문제 D. 에어컨 설치

시간 제한	1 초 **(추가 시간 있음)**
메모리 제한	1024 MB

에어컨 사업을 하는 주현이에게, 새롭게 지어지는 뉴욜라 도서관에서 에어컨 설치 문의가 들어왔다. 도서관 측에서는 설계 도면을 주고 적당한 곳에 에어컨을 설치해달라고 요청했다. 오랜만에 떼돈을 벌 기회를 잡은 주현이는 신나는 마음으로 도면을 확인했다.

도서관 측의 남다른 건축학적 이념을 담은 뉴욜라 도서관의 설계 도면은 독특한 구조로 되어 있는데, 3차원 도면의 정수 좌표에 해당하는 곳에만 방을 만들었고 두 방의 좌표 사이의 거리가 1인 방 사이에는 항상 복도 또는 계단이 있다는 점이다. 이때, 방의 부피는 고려하지 않는다.

최대한 에어컨을 많이 팔고 싶었던 주현이는 모든 방에 에어컨을 설치해야 한다고 강력하게 말하고 싶었지만, 아쉽게도 에어컨은 인접한 방까지 냉방이 가능한 에어컨 뿐이었다. 다시 말해, 에어컨 한 대는 에어컨이 설치된 방을 포함하여, 그 방과 연결된 복도와 계단, 그리고 그 복도와 계단으로 이어져 있는 방까지 냉방이 가능하다.

물론 도서관 관계자들을 속여 모든 방에 에어컨을 설치해도 되겠지만, 바가지를 씌우면 미래의 사업에 차질이 생길 수도 있기 때문에 모든 방과 복도, 계단의 냉방이 가능한 최소 수의 에어컨만을 팔고자 한다.

주현이는 도서관에 몇 대의 에어컨을 팔 수 있을까?

입력

첫째 줄에 도서관의 방의 개수 N이 주어진다. $(1 \le N \le 1\,500)$

둘째 줄부터 N개 줄에 각 방의 정수 좌표 x_i, y_i, z_i가 주어진다. $(-200 \le x_i, y_i, z_i \le 200,\ (x_i, y_i, z_i) \ne (x_j, y_j, z_j)$ if $i \ne j)$

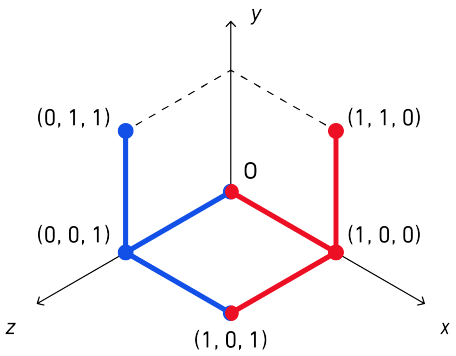
출력

주현이가 팔 수 있는 에어컨의 최소 수를 출력한다.

(뒷면에 입출력 예시가 있습니다.)

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
6 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 1	2



위 그림과 같이 $(0,0,1)$과 $(1,0,0)$에 에어컨을 설치하면, 도서관의 모든 방, 복도, 계단을 냉방할 수 있다. 두 개보다 적은 수의 에어컨으로는 불가능하다.

표준 입력(stdin)	표준 출력(stdout)
6 0 0 0 0 0 1 0 1 0 5 5 8 5 4 8 5 5 7	2

문제 E. 사탕 배달

시간 제한	3 초
메모리 제한	1024 MB

윤제가 사는 마을에는 N개의 사탕 가게가 있다. 사탕 가게는 1번부터 N번까지 번호가 매겨져 있다. 각 가게에서는 다섯 가지 종류의 사탕 중 하나만을 판매한다. 또한 사탕 가게들을 잇는 도로 $N-1$개로 이루어진 도로망이 있다. 모든 도로는 지나는 데에 1의 시간이 필요하며, 도로망을 이용해 한 사탕 가게에서 다른 모든 사탕 가게로 이동할 수 있다.

윤제는 오늘 M명의 친구를 한 명씩 순서대로 만나기로 했는데, 각 친구를 만나러 가는 길에 만날 친구가 좋아하는 종류의 사탕을 사다 주어야 한다. 이에 실패하면 친구들은 화가 나 떠나버린다.

윤제는 친구들을 빠르게 만나기 위해 한 친구를 만나고 나서 곧장 다음 친구를 만나러 출발한다. 시간을 최대한 단축하기 위해 언제나 최단 경로로 이동하며, 그 경로를 지나는 도중에 만나는 사탕 가게에 들러서 사탕을 산다. 친구가 있는 곳에 있는 가게에서 사탕을 사는 것도 가능하다. 다만, 윤제는 주머니가 작아서 사탕을 한 개만 들고 다닐 수 있다. 즉 어떤 친구 A를 만나러 가는 길에 또 다른 친구 B를 위한 사탕을 미리 사는 것은 불가능하다.

윤제는 자신을 떠날 친구들을 미리 파악하기 위해 각 친구에게 사탕을 사다 줄 수 있는지를 알고 싶어 한다. 윤제는 시작 위치를 자유롭게 선택할 수 있다. 윤제를 도와 각 친구가 좋아하는 사탕을 사다 줄 수 있는지 알려주는 프로그램을 작성하자.

입력

첫째 줄에 사탕 가게의 개수를 나타내는 정수 N이 주어진다. $(1 \le N \le 100\,000)$

둘째 줄에 각 사탕 가게에서 판매하는 사탕의 종류를 나타내는 N개의 정수가 공백으로 구분되어 주어진다. 사탕의 종류는 1과 5 사이의 양의 정수로 표현된다.

셋째 줄부터 $N-1$개의 줄에 걸쳐 도로의 정보를 나타내는 u, v가 주어진다. 이는 u번 사탕 가게와 v번 사탕 가게를 잇는 양방향 도로가 존재한다는 의미이다. $(1 \le u, v \le N, u \ne v)$

다음 줄에 친구의 수를 나타내는 정수 M이 주어진다. $(1 \le M \le 100\,000)$

다음 M개 줄에 걸쳐 각 친구가 서 있는 가게의 번호와 좋아하는 사탕의 종류가 약속 시간이 빠른 순으로 주어진다. 여러 명의 친구가 같은 가게 앞에 서 있을 수도 있다.

출력

약속 시간이 빠른 순서대로, 각 친구가 좋아하는 종류의 사탕을 사다 줄 수 있다면 PLAY를, 아니라면 CRY를 한 줄에 하나씩 출력한다.

(뒷면에 입출력 예시가 있습니다.)

입출력 예시

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

문제 F. 폰친구

시간 제한	1 초
메모리 제한	1024 MB

사탕을 배달하다 지친 윤제는 집에 돌아와 잠자리에 들었다.

사탕처럼 달콤한 꿈속에서 윤제는 가상의 친구 '폰친구' 들을 사귀었다. 폰친구들은 사탕을 좋아하지만, 사탕 배달을 요구하는 일은 없다. 대신 윤제가 있는 곳으로 직접 와주기 때문에 가만히 서서 사탕을 나누어 주기만 하면 된다!

꿈속에서 윤제는 같은 종류의 사탕 K개를 N명의 폰친구들에게 남김없이 나누어 주려고 한다. 폰친구들은 사탕을 어떻게 나누어 주어도 윤제를 떠나지 않지만, 각자 최소 m개 이상의 사탕을 받기를 원한다. 또한, 자신을 제외한 누군가가 M개보다 많은 사탕을 받는 것을 원하지 않는다.

윤제는 사탕을 Nm개 이상, NM개 이하로 갖고 있다. 윤제가 모든 폰친구들을 만족시키면서 사탕을 나누어 줄 수 있는 경우의 수를 구해보자.

입력

첫째 줄에 정수 N, m, M, K이 주어진다. ($2 \le N \le 1\,000, 0 \le m \le M \le 1\,000, Nm \le K \le NM$)

N은 폰친구의 수, K는 윤제가 가진 사탕의 개수이다. m은 각 친구가 받아야 할 최소 사탕의 개수, M은 각 친구가 받을 수 있는 최대 사탕의 개수이다.

출력

윤제가 폰친구들에게 조건에 맞게 사탕을 나누어 줄 수 있는 경우의 수를 출력한다. 너무 많은 경우가 있을 수 있으므로 경우의 수를 10^9+7로 나눈 나머지를 출력한다.

입출력 예시

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

이 페이지는 공백입니다

문제 G. Confuzzle

시간 제한	3 초 **(추가 시간 있음)**
메모리 제한	1024 MB

N개의 정점으로 구성된 가중치 없는 트리가 주어진다. 트리 상의 두 정점 사이의 거리는 두 정점 사이의 간선의 개수로 정의한다.

각 정점에는 수가 적혀 있으며, 적어도 두 정점은 같은 값임이 보장된다. 이때, 서로 같은 값이 쓰여 있는 두 정점 쌍 중 가장 거리가 가까울 때의 거리를 구해보자.

입력

첫째 줄에 트리의 정점의 개수 N이 주어진다. $(2 \le N \le 100\,000)$

둘째 줄에 각 정점의 고유한 값인 정수 $c_1, c_2, \cdots, c_N$이 주어진다. $(1 \le c_i \le N)$

셋째 줄부터 $N-1$개 줄에 걸쳐 각 줄마다 트리를 이루는 간선을 나타내는 정수 u와 v가 주어진다. 이는 u번 정점과 v번 정점을 잇는 간선이 존재함을 의미한다. $(1 \le u, v \le N, u \ne v)$

출력

서로 같은 값을 갖는 두 정점 쌍 중 가장 거리가 가까울 때의 거리를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
7 1 2 3 1 4 2 2 1 2 1 3 3 4 3 5 4 6 5 7	2

이 페이지는 공백입니다

문제 H. 파인애플 피자

시간 제한	2 초 **(추가 시간 있음)**
메모리 제한	1024 MB

N개의 조각으로 이뤄진 파인애플 피자 한 판이 있다. 각 피자 조각의 파인애플 토핑의 개수는 시계 방향 순으로 $a_1, a_2, \cdots, a_N$개다. 파인애플 피자를 맛보기 위해 $K(\leq N)$명의 손님이 줄을 서서 기다리고 있다. 당신은 첫 조각을 고른 후, 시계 방향 순으로 피자를 한 조각씩 떼어 줄을 선 순서대로 손님에게 제공한다. 예를 들어, 4명의 손님에게 토핑 개수가 a_{N-1}개, a_N개, a_1개, a_2개인 피자 조각을 순서대로 나눠줄 수 있다.

각 손님의 나이는 줄을 선 순서대로 $b_1, b_2, \cdots, b_K$이다. 손님들은 자신보다 나이가 어리거나 같은 사람보다 파인애플 토핑을 적게 받으면 그 자리에서 밥상을 엎어버린다.

손님들이 밥상을 엎지 않도록 피자 조각을 고를 수 있는 방법은 몇 가지일까?

입력

첫째 줄에 피자 조각의 개수 N과 손님의 수 K가 공백으로 구분되어 주어진다. $(1 \leq K \leq N \leq 100\,000)$

둘째 줄에 각 피자 조각의 파인애플 토핑의 개수를 나타내는 정수 $a_1, a_2, \cdots, a_N$가 공백으로 구분되어 주어진다. $(0 \leq a_i \leq 10^9)$

셋째 줄에 손님들의 나이를 나타내는 정수 $b_1, b_2, \cdots, b_K$가 공백으로 구분되어 주어진다. $(0 \leq b_i \leq 10^9)$

출력

손님들이 밥상을 엎지 않도록 피자 조각을 고를 수 있는 방법의 수를 출력한다.

입출력 예시

표준 입력(stdin)	표준 출력(stdout)
5 4 1 2 3 2 1 2 1 1 2	1
5 4 1 1 1 1 1 9 9 9 9	5

이 페이지는 공백입니다