2023 SCCC Contest #2

Official Problem Set

Sponsored By:

HYUNDAI MOBIS

대회 중 유의 사항

- 2023 SCCC 내부 대회 유의 사항입니다.
- 대회 진행 관련
  - 대회는 2시간 동안 8문제로 진행됩니다.
  - 대회 도중에는 대회 운영진을 제외한 타인과 대화할 수 없습니다.
  - 인터넷 검색을 허용하나, 제출하는 모든 코드는 대회 시간 도중에 본인이 직접 작성한 코드여야 합니다.
  - ChatGPT, Copilot, Bard 등 AI를 활용하여 생성한 코드도 사용할 수 없습니다.
  - 단, 외국어 문제 번역을 위해 GPT 등의 AI를 활용하는 것은 가능합니다.
  - 문제와 관련된 질문은 슬랙(21 나정휘) 또는 디스코드(jhnah917) DM을 이용해야 합니다.
- 문제 관련
  - 문제는 운영진들이 생각하는 난이도순으로 정렬되어 있습니다.
  - 모든 문제는 C++17로 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.
  - 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.
- 대회 상품
  - 1등: 5만원
  - $solve^3$ 가중치로 3명 추첨: BBQ 황금올리브치킨 + 콜라 1.25L
  - $solve^2$ 가중치로 10명 추첨: 스타벅스 아이스 카페 아메리카노 T 2잔 + 부드러운 생크림 카스텔라
  - $solve^1$ 가중치로 10명 추첨: 스타벅스 아이스 카페 아메리카노 T
- 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

언어 가이드

- 채점은 Intel Xeon E5-2666v3 프로세서를 사용하는 AWS EC2 c4.large 인스턴스에서 진행합니다.
- 채점 서버의 운영체제는 Ubuntu 16.04.7 LTS 입니다.
- BOJ에서 지원하는 모든 언어를 자유롭게 사용할 수 있습니다.
- 각 언어의 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

```
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

```
#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)
```

2023 SCCC Contest #2

Problem List

#	Problem Name	Time limit	Memory limit	Page
A	BOJ 28960 보자기	2 seconds	1024MiB	7 – 7
B	BOJ 28967 암호 분석	2 seconds	1024MiB	8 – 8
C	BOJ 28836 심부름	2 seconds	1024MiB	9 – 10
D	BOJ 28837 평행 우주	2 seconds	1024MiB	11 – 11
E	BOJ 28729 쓸어담기	2 seconds	1024MiB	12 – 12
F	BOJ 9788 배열 자르기	2 seconds	128MiB	13 – 13
G	BOJ 28754 격자 게임	2 seconds	1024MiB	14 – 14
H	BOJ 28947 부분 문자열과 쿼리	2 seconds	1024MiB	15 – 15

문제지에 있는 문제가 총 8문제가 맞는지 확인하시길 바랍니다.

모든 문제는 C++17로 풀 수 있음을 보장합니다.

A. 보자기 (BOJ 28960)

숭실이는 추석 선물로 여러 음식과 그것을 포장하는 보자기를 선물 받았다. 선물 받은 보자기를 다른 곳에 사용하기 위해 빨래를 한 숭실이는, 마당에 있는 빨랫줄에 보자기를 널어서 건조하려고 한다.

빨랫줄은 바닥으로부터 H 센티미터 떨어진 높이에 바닥과 평행하게 있으며, 길이는 L 센티미터다. 보자기는 직사각형 형태이며, 서로 마주 보지 않는 두 변의 길이는 각각 A 센티미터와 B 센티미터다.

숭실이는 보자기를 바닥에 끌리지 않도록 빨랫줄에 널고싶어 한다. 다행히 빨랫줄이 매우 튼튼해서 보자기를 널어도 형태가 변형되지 않고, 빨랫줄과 보자기가 접촉하는 면적을 무시할 수 있을 정도로 아주 가늘다.

빨랫줄의 높이와 길이, 그리고 보자기의 크기가 주어지면 보자기를 바닥에 끌리지 않도록 빨랫줄에 널 수 있는지 확인하는 프로그램을 작성하자. 보자기가 바닥에 닿는 면적이 0인 경우에는 바닥에 끌리지 않는 것으로 간주한다.

입력 형식

첫 번째 줄에 빨랫줄이 위치한 높이 H, 빨랫줄의 길이 L, 보자기의 평행하지 않은 두 변의 길이 A, B가 공백으로 구분되어 주어진다.

출력 형식

보자기가 바닥에 끌리지 않도록 빨래를 널 수 있으면 "`YES`", 널지 못하면 "`NO`"를 출력한다.

제한

- $1 \le H, L, A, B \le 200$
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고

첫 번째 예시는 보자기를 어떻게 걸더라도 항상 땅에 끌리지 않도록 걸 수 있다.

두 번째 예시는 길이가 10인 변이 빨랫줄과 평행하도록 걸면 보자기의 끝이 땅에 닿게, 즉 땅에 끌리지 않게 걸 수 있다.

세 번째 예시에서는 땅에 끌리지 않도록 빨래를 널 수 없다.

B. 암호 분석 (BOJ 28967)

정휘는 문자열 C를 갖고 있다. 이 문자열을 주원이에게 안전하게 전달할 방법을 고민하던 정휘는 문자열을 암호화해서 전달하기로 했다.

암호화 방법은 다음과 같다.

1. C를 0번 이상 이어서 적는다.
2. 원하는 위치에 원하는 문자를 삽입하는 동작을 원하는 만큼 0번 이상 반복한다.

예를 들어 C가 "ab"인 경우를 생각해 보자. 첫 번째 단계에서 C를 3번 반복해서 "ababab"를 만든 다음, 두 번째 단계에서 문자 6개를 적당히 추가하면 "abacabaaacbb"를 만들 수 있다.

문자열 C와 암호화된 결과 S가 주어지면, 첫 번째 단계에서 C를 적은 횟수로 가능한 최댓값을 구해보자.

입력 형식

첫 번째 줄에 문자열 C가 주어진다.

두 번째 줄에 암호화된 문자열 S가 주어진다.

출력 형식

암호화 과정의 첫 번째 단계에서 C를 적은 횟수로 가능한 최댓값을 출력한다.

제한

- $1 \le |C| \le 10^6$
- $1 \le |S| \le 10^6$
- 문자열은 알파벳 소문자와 대문자로만 구성되어 있다.

예제

예제는 채점 사이트 참고

C. 심부름 (BOJ 28836)

숭실이는 집에서 편하게 누워서 유튜브를 보고 있다. 하지만 숭실이의 어머니는 숭실이에게 마트에 들려서 식재료를 구매하고, 아파트 경비실에 가서 택배를 찾아오라고 시켰다. 숭실이는 최대한 빨리 집에 돌아와서 눕고 싶기 때문에 이동 시간이 최소가 되도록 이동 경로를 설계하려고 한다.

집에서 마트까지의 거리는 a미터, 집에서 경비실까지의 거리는 b미터, 마트에서 경비실까지의 거리는 c미터이며, 숭실이는 이렇게 세 가지 도로만 이용할 수 있고, 반대 방향으로도 이동할 수 있다.

숭실이의 속도는 숭실이가 들고 있는 물건에 따라 달라진다. 만약 숭실이가 아무것도 들고 있지 않으면 분당 v_0미터의 속도로 이동하며, 식재료와 택배 중 하나만 들고 있으면 분당 v_1미터, 식재료와 택배를 모두 들고 있으면 분당 v_2미터의 속도로 이동한다.

마트에 도착해서 음식을 구매하는 것과 경비실에 도착해서 택배를 찾는 데 걸리는 시간은 무시할 수 있을 정도로 짧으며, 이동 도중에 집에 들른다면 집에 들르는 즉시 들고 있는 물건을 내려놓을 수 있다.

숭실이가 식재료와 택배를 모두 집으로 갖고 돌아오는 데 걸리는 최소 시간을 구하는 프로그램을 작성해 보자.

입력 형식

첫 번째 줄에 여섯 개의 정수 a, b, c, v_0, v_1, v_2가 공백으로 구분되어 주어진다.

출력 형식

숭실이가 식재료와 택배를 모두 집으로 갖고 돌아오는 데 걸리는 최소 시간을 출력한다. 실제 정답과 출력값의 절대오차 또는 상대오차가 10^{-4} 이하이면 정답으로 처리한다.

제한

- $1 \le a, b, c \le 100$
- $1 \le v_2 \le v_1 \le v_0 \le 100$
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고

첫 번째 예시는 마트에서 식재료를 산 다음 우체국에서 택배를 받아서 집으로 돌아가는 것이 최적이며, 이때 걸리는 시간은 $\frac{a}{v_0} + \frac{c}{v_1} + \frac{b}{v_2} = \frac{1}{10} + \frac{2}{10} + \frac{2}{10} = \frac{5}{10}$이다.

두 번째 예시는 우체국에서 택배를 받은 다음 마트에서 식재료를 사고 다시 우체국을 거쳐서 집으로 돌아가는 것이 최적이며, 이때 걸리는 시간은 $\frac{b}{v_0} + \frac{c}{v_1} + \frac{c}{v_2} + \frac{b}{v_2} = \frac{6}{5}$이다.

세 번째 예시는 우체국에서 택배를 받은 다음 마트에서 식재료를 사서 집으로 돌아가는 것이 최적이며, 이때 걸리는 시간은 $\frac{b}{v_0} + \frac{c}{v_1} + \frac{a}{v_2} = \frac{3}{7} + \frac{4}{6} + \frac{2}{5} = \frac{314}{210}$이다.

네 번째 예시는 상점을 거쳐서 우체국에 가서 택배를 받은 다음 마트에 가서 식재료를 사고 집으로 돌아가는 것이 최적이며, 이때 걸리는 시간은 $\frac{a}{v_0} + \frac{c}{v_0} + \frac{c}{v_1} + \frac{a}{v_2} = \frac{1}{7} + \frac{3}{7} + \frac{3}{6} + \frac{1}{5} = \frac{267}{210}$이다.

다섯 번째 예시는 우체국에 가서 택배를 받고 집에 가져다 놓은 다음 마트에 가서 식재료를 사고 집으로 돌아가는 것이 최적이며, 이때 걸리는 시간은 $\frac{b}{v_0} + \frac{b}{v_1} + \frac{a}{v_0} + \frac{a}{v_1} = \frac{3}{10} + \frac{3}{9} + \frac{2}{10} + \frac{2}{9} = \frac{95}{90}$이다.

참고

절대오차는 실제 정답과 출력값의 차이, 상대오차(오차율)는 절대오차를 실제 정답으로 나눈 값을 의미한다. 즉, 실제 정답을 a, 출력값을 b라고 하면, 절대오차는 $|a-b|$, 상대오차는 $\frac{|a-b|}{a}$로 계산한다. 따라서 이 문제는 실제 정답이 x일 때 $\min\left\{x - 10^{-4}, (1-10^{-4})x\right\}$ 이상 $\max\left\{x + 10^{-4}, (1+10^{-4})x\right\}$ 이하인 실수를 출력하면 된다.

D. 평행 우주 (BOJ 28837)

숭실이는 오랜 연구 끝에 평행 우주를 여행할 수 있는 장치를 개발했다. 숭실이는 이 장치를 이용해 A번 우주에서 B번 우주로 이동하려고 한다.

장치에는 각각 양의 정수가 적힌 N개의 버튼이 있다. 만약 숭실이가 x번 우주에 있을 때 a_i가 적힌 버튼을 누르면 $x \vee a_i$번 우주로 이동한다. 이때 $\vee$은 bitwise or 연산을 의미한다.

A번 우주에서 출발해 버튼을 100번 이하로 눌러서 B번 우주로 이동할 수 있을지 판별하는 프로그램을 작성해 보자.

입력 형식

첫 번째 줄에 버튼의 개수와 출발 위치, 도착 위치를 의미하는 세 정수 N, A, B가 공백으로 구분되어 주어진다.

두 번째 줄에는 N개의 버튼에 적힌 수 $a_1, a_2, \cdots, a_N$이 공백으로 구분되어 주어진다.

출력 형식

만약 숭실이가 버튼을 100번 이하로 눌러서 B번 우주로 갈 수 없으면 "`-1`"을 출력한다.

그렇지 않은 경우, 첫째 줄에 버튼을 눌러야 하는 횟수 K를 출력한다. 두 번째 줄에는 눌러야 하는 버튼의 번호 K개를 차례대로 출력한다. 이때 버튼에 적힌 수가 아닌, 입력에서의 버튼의 위치(인덱스)를 출력해야 함에 주의하라.

가능한 답안이 여러 가지라면 그중 아무거나 출력해도 된다.

제한

- $1 \le N \le 10^5$
- $0 \le A, B \le 10^9$
- $1 \le a_i \le 10^9$
- 입력으로 주어지는 수는 모두 정수이다.
- $0 \le K \le 100$
- K를 최소화하지 않아도 된다.

예제

예제는 채점 사이트 참고

첫 번째 예시에서 첫 번째 버튼을 누르면 $2 \vee 3 = 3$번 우주로 이동하고, 이후 네 번째 버튼을 누르면 $3 \vee 10 = 11$번 우주로 이동한다.

E. 쓸어담기 (BOJ 28729)

N개의 물건이 일렬로 배치되어 있는 가게가 있다. 왼쪽에서부터 i번째에 위치한 물건은 하나에 A_i원이다.

숭실이는 **쓸어담기**라는 방법을 이용해 물건을 구매하려고 한다. 쓸어담기 방법이란, $1 \le L \le R \le N$을 만족하는 두 정수 L, R을 정한 다음, L번째 물건부터 R번째 물건까지를 하나씩 구매하는 것을 의미한다. 이때 물건을 구매하더라도 가게에서 없어지지 않는다. 즉, 물건이 구매될 때마다 새로운 물건이 다시 리필된다.

숭실이는 쓸어담기 방법을 총 M번 이용하려고 한다. 첫 번째 구매에서는 L_1번째부터 R_1번째 물건을 구매하고, 두 번째 구매에서는 L_2번째부터 R_2번째 물건을 구매하고, M번째 구매에서는 L_M번째부터 R_M번째 물건을 구매하려고 한다.

숭실이는 가게 주인과의 흥정을 통해 원하는 물건의 가격을 1 만큼 낮추는 것을 최대 K번 할 수 있는 기회를 얻었다. 즉, i번째 물건의 가격을 A_i원에서 $B_i(\le A_i)$원으로 바꿀 수 있으며, 이때 $\sum_{i=1}^{N} A_i - B_i \le K$를 만족해야 한다. 단, 물건의 가격을 음수로 만들 수는 없다.

숭실이는 돈을 아끼고 싶기 때문에 최대한 지불하는 금액의 총합이 작아지도록 가격을 깎으려고 한다. M번의 쓸어담기를 통해 물건을 구매할 때 지불해야 하는 금액의 최솟값을 구하는 프로그램을 작성해 보자.

입력 형식

첫 번째 줄에 물건의 개수 N, 쓸어담기의 수행 횟수 M, 흥정 가능 횟수 K가 공백으로 구분되어 주어진다.

두 번째 줄에 각 물건의 가격 $A_1, A_2, \cdots, A_N$이 공백으로 구분되어 주어진다.

세 번째 줄부터 M개의 줄에 걸쳐 쓸어담기를 수행할 구간의 끝점 L_i, R_i가 공백으로 구분되어 주어진다.

출력 형식

지불해야 하는 금액의 최솟값을 출력한다.

제한

- $1 \le N \le 10^5$
- $1 \le M \le 10^6$
- $0 \le K \le 10^{12}$
- $0 \le A_i \le 10^7$
- $1 \le L_i \le R_i \le N$
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고

첫 번째 예제에서 세 번째 물건과 네 번째 물건의 가격을 1씩 깎으면 $(1+2+2+3)+(2+3)=13$원만 지불하면 된다.

F. 배열 자르기 (BOJ 9788)

N개의 정수로 구성된 배열 $L = \{l_1, l_2, \cdots, l_N\}$이 주어진다. 배열을 인덱스 $i(1 \le i < N)$에서 분할한다는 것은, 배열 L을 두 개의 부분 배열 $L_1 = \{l_1, l_2, \cdots, l_i\}$와 $L_2 = \{l_{i+1}, l_{i+2}, \cdots, l_N\}$으로 나누는 것을 의미한다. 비슷하게, 배열 L을 $K-1$번 분할하면 $L_1 = \{l_1, \cdots, l_{i_1}\}$, $L_2 = \{l_{i_1+1}, \cdots, l_{i_2}\}$, $L_3 = \{l_{i_2+1}, \cdots, l_{i_3}\}$, $\cdots, L_K = \{l_{i_{k-1}+1}, \cdots, l_N\}$으로 분할할 수 있다. 이때 각 부분 배열의 길이는 1 이상이 되어야 한다.

분할된 부분 배열 L_i의 최댓값을 M_i, 최솟값을 m_i라고 하자. N개의 정수로 구성된 배열 L과 정수 K가 주어지면, 배열을 $K-1$번 분할해서 얻을 수 있는 $\sum_{i=1}^{K}(M_i - m_i)$의 최솟값을 구하는 프로그램을 작성해 보자.

즉, 각 부분 배열의 최댓값과 최솟값의 차이의 합을 최소화하는 분할을 찾아보자.

입력 형식

입력의 첫 번째 줄에 테스트 케이스의 개수 T가 주어진다.

각 테스트 케이스는 빈 줄로 시작하고, 두 번째 줄에 두 정수 N, K가 공백으로 구분되어 주어진다.

테스트 케이스의 세 번째 줄에는 배열 L의 원소를 의미하는 N개의 양의 정수 $l_1, l_2, \cdots, l_N$이 공백으로 구분되어 주어진다.

출력 형식

각 테스트 케이스에 대해, $\sum_{i=1}^{K}(M_i - m_i)$의 최솟값을 한 줄에 하나씩 출력한다.

제한

- $1 \le T \le 120$
- $1 \le K \le N \le 400$
- $1 \le l_i \le 1\,000$
- 모든 테스트 케이스에서 $\sum N \le 6\,000$
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고

G. 격자 게임 (BOJ 28754)

정휘와 주원이는 모든 칸이 흰색으로 칠해져 있는 $N \times M$ 크기의 격자에서 게임을 하고 있다. 게임은 정휘부터 시작해서 두 사람이 번갈아 가며 진행한다.

각 사람은 자신의 차례에 흰색 칸으로만 구성된 넓이가 S 이하인 직사각형 영역을 선택한 다음, 선택한 영역에 포함된 모든 칸을 검은색으로 칠한다. 이때 직사각형 영역을 선택할 수 없는 사람이 게임에서 패배한다.

두 사람 모두 게임에서 승리하기 위해 최선을 다해 게임을 진행할 때, 정휘가 게임에서 이길 수 있는지 판별하는 프로그램을 작성해 보자.

입력 형식

첫 번째 줄에 게임판의 크기를 나타내는 두 정수 N, M과 선택할 수 있는 영역 넓이의 최댓값을 나타내는 정수 S가 공백으로 구분되어 주어진다.

출력 형식

정휘가 이기면 "`YES`", 주원이가 이기면 "`NO`"를 출력한다.

제한

- $1 \leq N, M \leq 1\,000$
- $1 \leq S \leq N \times M$
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고

H. 부분 문자열과 쿼리 (BOJ 28947)

숭실이는 N개의 문자열 $S_1, S_2, \cdots, S_N$을 갖고 있다. 아래 두 가지 쿼리를 처리하는 프로그램을 작성해 보자. 단, 처음에 N개의 문자열은 모두 빈 문자열이다.

- 1 l r s: $S_l, S_{l+1}, \cdots, S_r$의 맨 뒤에 s를 추가한다. s는 알파벳 소문자로 구성된 문자열이다.
- 2 i x y: S_i의 x번째 문자부터 y번째 문자까지를 출력한다. (단, 문자열의 인덱스는 1부터 시작한다.)

입력 형식

첫 번째 줄에 문자열의 개수와 쿼리의 횟수를 의미하는 두 정수 N, M이 공백으로 구분되어 주어진다.

이어서 M개의 줄에 한 줄에 하나씩 쿼리가 주어진다.

1번 쿼리의 경우, 줄의 맨 앞에 1이 주어진 다음 두 정수 l, r과 알파벳 소문자로 구성된 문자열 s가 공백으로 구분되어 주어진다. 2번 쿼리의 경우, 줄의 맨 앞에 2가 주어진 다음 세 정수 i, x, y가 공백으로 구분되어 주어진다.

출력 형식

2번 쿼리의 결과를 차례대로 한 줄에 하나씩 출력한다.

제한

- $1 \le N, M \le 2 \times 10^5$
- $1 \le l \le r \le N$
- s는 알파벳 소문자로 구성된 문자열이다.
- 첫 번째 쿼리에서 주어지는 문자열 s의 길이의 총합은 10^6을 넘지 않는다.
- $1 \le i \le N$
- $1 \le x \le y \le |S_i|$
- 두 번째 쿼리에서 출력하는 문자 개수의 총합은 10^6을 넘지 않는다.
- 입력으로 주어지는 수는 모두 정수이다.

예제

예제는 채점 사이트 참고