# 마법 구슬 찾기

당신은 모양과 질량이 완전히 동일한 $k+1$개의 구슬을 갖고 있다. 이 중 $k$개의 구슬은 일반적인 구슬이고, 1개는 **마법 구슬**이다. 당신은 마법 구슬을 찾아 마법의 성에 들어가려고 한다.

마법 구슬과 일반 구슬을 육안으로 구별할 수 있는 방법은 없지만, 마법 구슬을 찾아내는 데에 사용할 수 있는 $M$ ($M \geq 2$)개의 주머니가 있다. 주머니에는 0부터 $M-1$까지의 번호가 붙어 있다.

주머니를 활용하여 마법 구슬을 찾을 수 있는 방법은 아래와 같다.

1. 갖고 있는 모든 구슬을 $M$개의 주머니에 나눠 담는다.
   - 어떤 주머니에도 넣지 않은 구슬이 있으면 안 된다.
   - 구슬을 담지 않은 빈 주머니는 있어도 된다.
   - 주머니 속에는 구슬만 담을 수 있으며, 다른 주머니를 담을 수는 없다.
2. 주문을 외운다.
3. 주문을 외운 직후:
   - 마법 구슬이 들어 있지 않은 주머니 속의 구슬들은 모두 소멸된다.
   - 마법 구슬이 들어 있는 주머니 속의 구슬들은 마법 구슬의 보호를 받아서 소멸되지 않는다. 하지만, 주문으로 인한 부수 효과를 수습해야 하고, 이 과정에서 비용이 든다. 마법 구슬이 $i$번 주머니에 들어 있었고, $i$번 주머니에 구슬 $j$개가 들어 있었다면, $A[i] \times j + B[i]$원 ($A[i] \geq 0$, $B[i] \geq 1$)이 든다.

마법 구슬은 절대로 소멸되지 않으므로, 위의 과정을 마법 구슬 1개만 남을 때까지 반복하면 마법 구슬을 찾을 수 있다.

당신은 구슬들을 주머니에 나눠 담는 전략을 수립하여, 최악의 경우에 마법 구슬을 찾는 데에 드는 비용을 최소화하고자 한다. 즉, $k+1$개의 구슬 중 어떤 구슬이 마법 구슬이더라도 총 $w$원 이하를 들여 마법 구슬을 찾을 수 있는 최소한의 $w$를 찾고자 한다.

0 이상 $N-1$ 이하의 모든 $k$에 대해 이 문제를 해결하는 함수를 작성하라.

## 함수 목록 및 정의

여러분은 아래 함수를 구현해야 한다.

```
vector<long long> find_minimum_costs(int N, vector<int> A, vector<int> B)
```

- $N$: 구슬의 최대 개수
- $A$, $B$: 길이가 $M$인 배열. 모든 $i$ ($0 \leq i \leq M-1$)에 대해, 마법 구슬이 $i$번 주머니에 들어 있고, $i$번 주머니에 구슬 $j$개가 들어 있다면, $A[i] \times j + B[i]$원의 비용이 든다.
- 이 함수는 길이가 $N$인 배열 $C$를 반환해야 한다. 모든 $k$ ($0 \leq k \leq N-1$)에 대해, $C[k]$는 일반 구슬 $k$개와 마법 구슬 1개가 있을 때, 최악의 경우 마법 구슬을 찾기 위해 드는 최소 비용(원 단위)이다.

제출하는 소스 코드의 어느 부분에서도 입출력 함수를 실행해서는 안 된다.

## 예제 1

$N = 3$, $M = 2$, $A = [1, 3]$, $B = [2, 1]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
find_minimum_costs(3, {1, 3}, {2, 1})
```

$k = 0$일 때는, 갖고 있는 유일한 구슬이 마법 구슬이므로, 0원의 비용을 들여서 마법 구슬을 찾을 수 있다.

$k = 1$일 때는, 두 개의 주머니에 구슬을 하나씩 나눠 담는 전략을 수립할 수 있다. 마법 구슬이 0번 주머니에 있었다면 $A[0] \times 1 + B[0] = 1 \times 1 + 2 = 3$원이 들고, 1번 주머니에 있었다면 $A[1] \times 2 + B[1] = 3 \times 1 + 1 = 4$원이 든다. 따라서, 최악의 경우 4원의 비용이 든다.

$k = 2$일 때는, 다음과 같은 전략을 수립할 수 있다. 편의상 세 개의 구슬을 X, Y, Z라고 부른다.

- 0번 주머니에 X, Z를 담고, 1번 주머니에 Y를 담은 후, 주문을 외운다.
  - 0번 주머니에 마법 구슬이 있는 경우: $A[0] \times 2 + B[0] = 1 \times 2 + 2 = 4$원이 들고, X, Z만 남았다. 이제 0번 주머니에 Z를 담고, 1번 주머니에 X를 담은 후, 주문을 외운다.
    - 0번 주머니에 마법 구슬이 있는 경우: $A[0] \times 1 + B[0] = 1 \times 1 + 2 = 3$원이 들고, Z만 남았다.
    - 1번 주머니에 마법 구슬이 있는 경우: $A[1] \times 1 + B[1] = 3 \times 1 + 1 = 4$원이 들고, X만 남았다.
  - 1번 주머니에 마법 구슬이 있는 경우: $A[1] \times 1 + B[1] = 3 \times 1 + 1 = 4$원이 들고, Y만 남았다.

이 전략에서 X가 마법 구슬일 경우 총 $4 + 4 = 8$원이, Y가 마법 구슬일 경우 총 4원이, Z가 마법 구슬일 경우 총 $4 + 3 = 7$원이 들기 때문에, 최악의 경우 8원의 비용이 든다.

함수는 $[0, 4, 8]$을 반환해야 한다.

## 예제 2

$N = 13$, $M = 2$, $A = [1, 3]$, $B = [2, 1]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
find_minimum_costs(13, {1, 3}, {2, 1})
```

함수는 $[0, 4, 8, 11, 13, 17, 18, 20, 24, 25, 27, 29, 30]$를 반환해야 한다.

## 예제 3

$N = 12$, $M = 3$, $A = [5, 3, 0]$, $B = [1, 4, 6]$인 경우를 생각해 보자.

그레이더는 다음과 같이 함수를 호출한다.

```
find_minimum_costs(12, {5, 3, 0}, {1, 4, 6})
```

함수는 $[0, 6, 7, 12, 13, 16, 17, 18, 19, 20, 22, 23]$를 반환해야 한다.

## 제약 조건

- $2 \leq N \leq 1\,000\,000$
- $2 \leq M \leq 100\,000$
- $0 \leq A[i] \leq 10^9$ (모든 $0 \leq i \leq M-1$)
- $1 \leq B[i] \leq 10^9$ (모든 $0 \leq i \leq M-1$)

## 부분문제

1. (3점) $A[0] = A[1] = \cdots = A[M-1]$, $B[0] = B[1] = \cdots = B[M-1]$
2. (4점) $N \leq 5\,000$, $M = 2$
3. (7점) $M = 2$
4. (12점) $A[i] = 0$, $B[i] \leq 1\,000$ (모든 $0 \leq i \leq M-1$)
5. (25점) $N \leq 10\,000$, $M \leq 10$
6. (17점) $M \leq 100$
7. (32점) 추가 제약 조건 없음

## Sample grader

Sample grader의 입력 형식은 아래와 같다.

- Line 1: $N$ $M$
- Line $2+i$ ($0 \leq i \leq M-1$): $A[i]$ $B[i]$

Sample grader의 출력 형식은 아래와 같다.

- Line $1+k$ ($0 \leq k \leq N-1$): $C[k]$