21982번 - D번 - 상자 빌리기 25점 서브태스크

한국어

시간 제한	메모리 제한
3 초 (하단 참고)	512 MB

문제

Albert는 높이가 서로 다른 n개의 상자를 갖고 있는데 각 상자는 직육면체 모양이다. 편의상 0번부터 n-1번까지 번호가 붙어있다. i번째 상자의 높이는 H[i]이고 밑면의 넓이는 W[i] 이다. 이 때, i번째 상자의 부피는 H[i] × W[i] 이다 (여기서 W[i]는 밑면의 "넓이"를 나타낸다). Albert는 언제나 상자를 높이 순으로 정렬해두기 때문에 H[0] < H[1] < ... < H[n-1]을 만족한다.

Alice는 Albert에게 상자 두 개를 빌리기로 했는데 Albert는 왠지 순순히 빌려주고 싶지 않아서 아래와 같은 조건을 달았다.

- 조건: Alice가 빌리는 상자 두 개의 높이 차이는 X를 넘을 수 없다 (즉, i번째 상자와 j번째 상자를 고른다면 | H[i] - H[j] | ≤ X 를 만족해야한다).

Alice는 흔쾌히 승낙했고, 물건을 옮길 때 사용해야 하기 때문에 두 상자의 부피의 합이 최대가 되는 한 쌍의 상자를 고르기로 했다.

예를 들어 n = 3, H = [3, 5, 8], W = [8, 16, 6], X = 3이라 하자.

- 각 상자의 부피는 순서대로 24, 80, 48이다.
- Albert가 제시한 조건을 만족하는 (i, j)쌍은 (0, 1)과 (1, 2)가 있다.
- 두 가지 방법 중 1번 상자와 2번 상자를 빌리면 부피의 합이 80 + 48 = 128이 된다.
- 같은 예에서 만약 X = 1이라면 Albert가 제시한 조건을 만족하는 쌍이 존재하지 않는다 (출력 항목 참고).

다른 예로, n = 4, H = [3, 10, 18, 20], W = [8, 11, 9, 3], X = 7이라 하자.

- 각 상자의 부피는 순서대로 24, 110, 162, 60이다.
- Albert가 제시한 조건을 만족하는 (i, j)쌍은 (0, 1)과 (2, 3)이 있다.
- 두 가지 방법 중 2번 상자와 3번 상자를 빌리면 부피의 합이 162 + 60 = 222이 된다.
- 같은 예에서 만약 X = 8이라면 1번과 2번 상자를 빌려 부피의 합이 272가 되도록 할 수 있다.

입력으로 n, X, 그리고 H[0] ... H[n-1], W[0] ... W[n-1] 이 주어졌을 때, Alice를 도와 Albert가 제시한 조건을 어기지 않으면서 부피의 합이 최대가 되는 한 쌍의 상자를 골라보자. 단, 이 문제의 경우 n이 크기 때문에 상자의 높이와 밑면 넓이가 직접 주어지지 않고, 이를 생성하는 방법이 입력으로 주어진다 (입력 항목 참고).

입력

첫 줄에 테스트 케이스의 수 T가 주어진다.

각 테스트 케이스는 세 줄에 걸쳐 주어진다.

첫 줄에 n과 X가 공백으로 구분되어 주어진다. 두 번째 줄에 네 개의 정수 h_s, h_a, h_b, h_c 가 공백으로 구분되어 주어진다. 세 번째 줄에 네 개의 정수 w_s, w_a, w_b, w_c 가 공백으로 구분되어 주어진다.

이를 이용해 H[i]와 W[i]를 구하기 위해 아래 공식을 이용한다 ($0 \le i \le n-1$ 임에 유의하자): ("%"는 통상 프로그래밍 언어에서 쓰이는 정수 나눗셈 연산의 나머지를 구하는 연산이다 (modulo))

- i = 0 일 때:
  - $H[i] = (h_s \% h_c) + 1$
  - $W[i] = (w_s \% w_c) + 1$
- $1 \le i \le n-1$ 인 i에 대하여:
  - $H[i] = H[i-1] + 1 + ((H[i-1] * h_a + h_b) \% h_c)$
  - $W[i] = (W[i-1] * w_a + w_b) \% w_c + 1$

위의 연산시 integer overflow를 피하기 위해 64-bit 정수 타입을 사용하는 것을 권장한다.

출력

각 테스트 케이스에 대하여 Alice가 상자 한 쌍을 빌릴 수 없다면 -1을 출력한다. 빌릴 수 있다면, 가능한 모든 쌍 중 부피의 합의 최대값을 출력한다.

제한

- $1 \leq T \leq 10$
- $2 \leq n \leq 5{,}000{,}000$
- $1 \leq X, h_s, h_a, h_b, h_c, w_s, w_a, w_b, w_c \leq 10^9$
- 각 테스트 케이스의 모든 $0 \leq i \leq n-1$ 인 i에 대하여 $1 \leq H[i], W[i] \leq 10^9$

서브태스크 1 (9점)

- $2 \leq n \leq 100{,}000$

서브태스크 2 (16점)

- $2 \leq n \leq 5{,}000{,}000$

예제 입력 1 복사

```
7
3 1
2 3 7 5
7 3 8 17
3 3
2 3 7 5
7 3 8 17
4 7
2 3 7 10
7 3 8 11
4 8
2 3 7 10
7 3 8 11
10 20
1 17 31 23
7 4 8 41
6 1000000000
123456 1 1 180001
654321 1000000000 1 180161
6 1000000000
123456 1 1 180001
2021 1000000000 1 180161
```

예제 출력 1 복사

```
-1
128
222
272
6311
182109717128
178794695372
```

예제 1-4: 본문에서 다루었다.

예제 5: 이 예제의 H, W 값은 아래와 같다.

- H = [2,22,37,54,61,72,86,108,113,134]
- W = [8,41,9,4,25,27,35,26,31,10]

예제 6: 이 예제의 H, W 값은 아래와 같다.

- H = [123457,246916,313833,447667,535334,710668]

- W = [113839,172370,109296,122144,9432,179310]

예제 7: 이 예제의 H, W 값은 아래와 같다.

- H = [123457,246916,313833,447667,535334,710668]
- W = [2022,129668,123587,119610,146310,141374]

힌트

아래 표에 제공된 각 언어별 예제 코드는 H[0 ... n-1]과 W[0 ... n-1]를 생성하는 코드이다. 아래 코드를 그대로 사용하여도 되고, 이를 직접 구현해도 된다. 출제진이 의도한 해법은 아래 코드를 사용하며 제한 시간 내에 정답을 구한다.

C/C++

```
int n, x;
long long hs, ha, hb, hc,
          ws, wa, wb, wc;
// stdin에서 입력을 받는다.

long long H[n], W[n];

H[0] = hs % hc + 1;
W[0] = ws % wc + 1;
for(int i = 1; i <= n-1; i++) {
  H[i] = H[i-1] + 1
       + (H[i-1] * ha + hb) % hc;
  W[i] = (W[i-1] * wa + wb) % wc + 1;
}
```

Java

```
int n, x;
long hs, ha, hb, hc,
     ws, wa, wb, wc;
// stdin에서 입력을 받는다.

long[] H = new long[n], W = new long[n];

H[0] = hs % hc + 1;
W[0] = ws % wc + 1;
for(int i = 1; i <= n-1; i++) {
  H[i] = H[i-1] + 1
       + (H[i-1] * ha + hb) % hc;
  W[i] = (W[i-1] * wa + wb) % wc + 1;
}
```

Pypy3

```
n,x = # stdin에서 입력을 받는다.
hs,ha,hb,hc = # stdin에서 입력을 받는다.
ws,wa,wb,wc = # stdin에서 입력을 받는다.

H, W = [0 for i in range(n)], [0 for i in range

H[0] = hs % hc + 1
W[0] = ws % wc + 1
for i in range(1, n):
  H[i] = H[i-1] + 1 + (H[i-1] * ha + hb) % hc
  W[i] = (W[i-1] * wa + wb) % wc + 1
```

시간 제한

- Java 8: 3 초