04 시뮬레이션

시뮬레이션

시뮬레이션은 프로그래밍 대회에서 가장 쉬운 문제이며 초기 상태와 어떤 작업을 수행할지 제공하고 최종 결과가 어떻게 될지 답하는 문제입니다.

이런 문제는 무엇을 해야 하는지 문제에 모두 쓰여 있습니다. 따라서 문제를 코드로 그대로 옮기면 쉽게 풀립니다. 설명만으로 어떤 문제인지 이해하는 것은 힘든 일입니다. 곧바로 구체적인 문제를 살펴보면서 시뮬레이션이 무엇인지 알아봅시다!

01 » 키위 주스

Easy

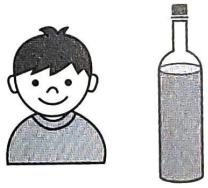
KiwiJuiceEasy SRM478 Div 2 Level 1
시뮬레이션

[문제]

Taro has prepared delicious kiwi fruit juice. He poured it into N bottles numbered from 0 to N-1. The capacity of the i-th bottle is capacities[i] liters, and he poured bottles[i] liters of kiwi juice into this bottle.
Now he wants to redistribute juice in the bottles. In order to do this, he will perform M operations numbered from 0 to M-1 in the order in which he will perform them. For the i-th operation, he will pour kiwi juice from bottle from Id[i] to bottle toId[i]. He will stop pouring when bottle fromId[i] becomes empty or bottle toId[i] becomes full, whichever happens earlier.
Return an int[] that contains exactly N elements and whose i-th element is the amount of kiwi juice in the i-th bottle after all pouring operations are finished.

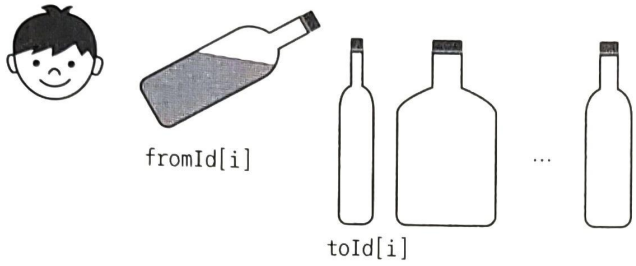
타로는 맛있는 키위 주스를 준비했습니다. 타로는 0부터 N-1이라 이름을 붙인 N개의 병에 키위 주스를 넣었습니다. 이때 i번째의 병의 용량은 capacities[i] 리터이며 타로가 i번째 병에 넣은 키위 주스의 양을 bottles[i] 리터라고 합니다.

그림 4-1 » 병의 용량과 주스의 양



타로는 병에 키위 주스를 재분배하려고 하며, 0부터 M-1까지 M회 조작합니다. i번째의 조작은 타로가 병 fromId[i]부터 병 toId[i]에 키위 주스를 넣는 것을 의미합니다. 병 fromId[i]가 비어 있거나 병 toId[i]가 꽉 차 있는 순간, 타로는 더 이상 키위 주스를 넣지 않습니다.

그림 4-2 » 주스 재분배



N개의 요소를 가진 정수 배열 int[]를 리턴해주세요. 배열의 i번째 요소는 모든 주스를 쏟는 작업이 완료되고 i번째 병에 남아 있는 키위 주스의 양입니다.

그림 4-3 » 각 병의 용량과 주스 용량의 관계

병의 용량	capacities[0]	capacities[1]	capacities[2]	...	capacities[N-1]
주스 용량	bottles[0]	bottles[1]	bottles[2]	...	bottles[N-1]

[정의 : 클래스와 함수 정의]

Class ▶ KiwiJuiceEasy

C# ▶
```
public int[] thePouring(int[] capacities, int[] bottles, int[] fromId,
        int[] toId)
```

C++ ▶
```
public: vector <int> thePouring(vector <int> capacities,
         vector <int> bottles,
         vector <int> fromId, vector <int> toId)
```

Java ▶
```
public int[] thePouring(int[] capacities, int[] bottles, int[] fromId,
        int[] toId)
```

[제약 조건 : 매개변수 범위]

capacities ▶ 2~50개의 요소가 있는 배열입니다. 각 요소는 1~1000000 사이의 값을 갖습니다.

bottles ▶ capacities와 같은 수의 요소가 있는 배열입니다. bottles[i]는 capacities[i]에 들어 있는 주스를 의미합니다.

fromId ▶ 1~50개의 요소가 있는 배열입니다.

todoId ▶ fromId와 같은 수의 요소가 있는 배열입니다.

변수 fromId와 toId는 0~(N-1) 사이의 값입니다. 이때 N은 변수 capacities의 항목 개수입니다. 변수 fromId[i]와 toId[i]는 서로 다른 값을 갖습니다.

문제 풀기 전에

문제의 의도를 잘 파악했나요?

한글 지문을 읽어도 무엇을 해야 하는지 정확하게 파악하지 못하고 완전히 다른 문제로 이해하는 사람도 많습니다. 우선 문제를 제대로 이해하는 것이 가장 중요합니다. 변수의 정의와 매개변수 범위라는 단어 자체를 이해하지 못한다면 1부로 돌아가 확인하세요. 문제를 어느 정도 이해했다면 계속 진행합시다.

문제를 읽고 바로 코드를 작성할 수 있나요? 대부분의 참가자에게는 힘든 일입니다(물론 할 수 있다면 즉시 프로그램을 작성해도 좋습니다).

그런 경우에는 문제를 한 번 더 생각해보세요. 문제를 생각할 때는 컴퓨터를 이용하지 않아도 됩니다. 우선 입력 사례를 기반으로 어떤 식으로 코드가 작동하는지 손으로 실험해보세요. 이후에 어느 정도 문제의 감이 잡히면 코드를 작성하는 것이 현명한 전략입니다.

1부에서 설명한 것처럼 TopCoder는 문제에 4, 5개 정도의 Example(입력 데이터와 출력 데이터)이 함께 나옵니다. 이러한 Example은 간단한 입력 데이터가 1, 2개 정도 있으므로 손으로 풀어볼 수 있습니다. 이러한 간단한 입력 데이터로 문제를 확실하게 이해하세요. 생각만으로 푸는 것은 어려울 수 있으므로 종이와 연필도 적극적으로 사용합시다!

[예시 : 입력 데이터와 출력 데이터]

0) ▶ capacities = {20, 20}
bottles = {5, 8}
fromId = {0}
toId = {1}
Returns: {0, 13}

일단 0번째 병의 주스를 1번째 병에 따릅니다. 주스를 모두 따르면 0번째 병은 비고 1번째 병은 13리터가 됩니다.