SCV/2021

저울

- Greedy -

안태진(taejin7824@gmail.com)
GitHub(github.com/taejin1221)
상명대학교 소프트웨어학과
201821002

목차

Problem

- 저울!
  - 문제!
    - 이거 아니다!
  - 진짜 문제!
    - 무려 골드 3!
    - 뭔가 대회에서 좋아할듯한 문제
    - 실제 KOI 문제!
    - 한국의 미래가 밝다..!

저울 성공 출처

골드 III

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	10521	4041	3254	37.798%

문제

하나의 양팔 저울을 이용하여 물건의 무게를 측정하려고 한다. 이 저울의 양 팔의 끝에는 물건이나 추를 올려놓는 접시가 달려 있고, 양팔의 길이는 같다. 또한, 저울의 한쪽에는 저울추들만 놓을 수 있고, 다른 쪽에는 무게를 측정하려는 물건만 올려놓을 수 있다.

무게가 양의 정수인 N개의 저울추가 주어질 때, 이 추들을 사용하여 측정할 수 없는 양의 정수 무게 중 최솟값을 구하는 프로그램을 작성하시오.

가 주어졌을 때, 이 추들로 측정할 수 없는 양의 정수 무게 중 최솟값은 21이다.

출처

Olympiad > 한국정보올림피아드 > KOI 2011 > 초등부 3번

Problem

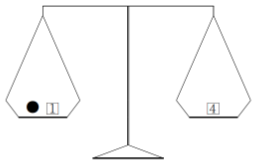
- 저울
  - KOI에서 저울을 주제로 문제를 하나 더 냈었네요
    - 이건 DP(Knapsack) 입니다!
    - 한번 풀어보세요!
    - https://www.acmicpc.net/problem/2629

문제

양팔 저울과 몇 개의 추가 주어졌을 때, 이를 이용하여 입력으로 주어진 구슬의 무게를 확인할 수 있는지를 결정하려고 한다.

무게가 각각 1g과 4g인 두 개의 추가 있을 경우, 주어진 구슬과 1g 추 하나를 양팔 저울의 양쪽에 각각 올려놓아 수평을 이루면 구슬의 무게는 1g이다. 또 다른 구슬이 4g인지를 확인하려면 1g 추 대신 4g 추를 올려놓으면 된다.

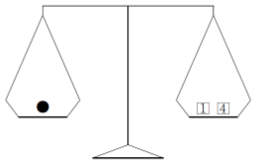
구슬이 3g인 경우 아래 <그림 1>과 같이 구슬과 추를 올려놓으면 양팔 저울이 수평을 이루게 된다. 따라서 각각 1g과 4g인 추가 하나씩 있을 경우 주어진 구슬이 3g인지도 확인해 볼 수 있다.



<그림 1> 구슬이 3g인지 확인하는 방법 (1은 1g인 추, 4는 4g인 추, ●은 무게를 확인할 구슬)

<그림 2>와 같은 방법을 사용하면 구슬이 5g인지도 확인할 수 있다. 구슬이 2g이면 주어진 추를 가지고는 확인할 수 없다.

추들의 무게와 확인할 구슬들의 무게가 입력되었을 때, 주어진 추만을 사용하여 구슬의 무게를 확인 할 수 있는지를 결정하는 프로그램을 작성하시오.



<그림 2> 구슬이 5g인지 확인하는 방법

목차

Approach

- Comment
  - Greedy를 Greedy인 것을 알고, 난이도가 쉽다면 딱 떠오르는 생각으로 풀텐데
  - 난이도가 어려워서... 오래걸렸다...

  - 대회를 준비해야해서 직관적인 해법이 왜 맞는지를 증명을 하고 푸느라 오래걸림
    - 사실 그냥 맞는 것 같은데 해볼까? 했는데 됨...
    - 이 피피티 만들면서 증명함 ㅋㅋㅋㅋ

  - 사실 증명이 다임 코드 짧음
    - 증명: 슬라이드 6개
    - 코드: 슬라이드 1개

문제	결과	메모리	시간
2437	맞았습니다!!	2020 KB	0 ms
2437	시간 초과		
2437	틀렸습니다		
2437	시간 초과		
2437	시간 초과		
2437	시간 초과		
2437	시간 초과		

목차

Approach

- Approach 1
  - 일단 정렬을 하자
    - 뭔가 정렬을 해야 할 것 같았음... 그냥 경험에서 나온 느낌이랄까..?
  - 생각

나랑 동갑이더라 ㅋㄷㅋㄷ

    - 내가 가지고 있는 추의 무게는 만들 수 있다.
    - 결국 못 만드는 것은 내가 가지고 있지 않은 추의 무게!
    - 그렇다면 무게를 1부터 1,000,000까지 늘려가며 그 무게를 내 추로 측정 수 있는지 판단!
      - 어떤 추의 무게가 1,000,000이라면 추의 최대 개수가 1,000개이기 때문에 그 사이가 무조건 빈다!
    - 어떻게 판단하는가?

Approach

- Approach 1
  - 내가 가진 추로 특정 무게를 측정할 수 있는가?

    Q: Greedy하게 큰 추부터 저울에 넣으면 그 무게를 정확히 측정할 수 있을까?

    A: 된다!
    - 이유는?
      - 측정하려는 무게를 1부터 늘려간다.
      - (찾으려는 수) - (큰 무게)는 찾으려는 수보다 작다
      - 만약 ↑를 못 만들었다면 그게 답이었을 것이다.
      - 하지만 내가 지금 이 무게를 측정한다는 것은?
        - 그 수를 만들 수 있는 것!

Approach

- Approach 1
  - 문제점
    - 시간 복잡도: $O(1{,}000{,}000n) = 1{,}000{,}000 \times 1{,}000 = 10^9$ => TLE
      1. 1부터 1,000,000까지 매번 n개의 추를 찾아야한다. => TLE
      2. 그럼 대충 100,000까지 찾으면 답 나오지 않을까? => WA
      3. 그럼 대충 500,000은? => TLE
      - 반례: 추가 range(1, 1001)라면 답은 500,501
      - 따라서 최소 500,501까지는 찾아야하는데
      - 그러면 5×10^8이므로 시간 초과가 뜬다...
  - 따라서 옳은 Solution이 아님!

문제	결과	메모리	시간
2437	맞았습니다!!	2020 KB	0 ms
2437	시간 초과 → 3번		
2437	틀렸습니다 → 2번		
2437	시간 초과		
2437	시간 초과		
2437	시간 초과		
2437	시간 초과		

1번

목차

Approach

- Approach 2
  - 다시 생각
    - 계속 걸렸던 생각
      - 추들의 무게를 담은 배열을 arr이라고 하자
      - 일단 기본적으로 $sum(arr[:i+1])+1$은 측정하지 못한다
      - e.g., $arr=\{1,1,2,3,6,7,30\}, i=5$ 일 때 $sum(arr[:i+1])+1=21$ 따라서 측정하지 못한다, $arr=range(1,1001), i=1{,}000$ 일 때 $sum(arr[:i+1])+1=500{,}501$ 따라서 측정하지 못한다

      - 이게 답이 되려면 $sum(arr[:i+1])$ 보다 작은 값들은 측정할 수 있다는 것을 보장해야한다.
        - 그렇지 않으면 $sum(arr[:i+1])+1$은 당연히 답이 될 수 없기 때문!

      - 증명해보자!
        - 우선 귀찮으니 $sum(arr[:i+1])$ 을 S_i라 하자

Approach

- Approach 2
  - 증명: S_i보다 작은 것들은 당연히 측정할 수 있는가?
    - S_i는 측정할 수 있다.
    - $S_i - 1$를 측정할 수 있는가?
      - 만약 기존의 추들로 1을 측정할 수 있다면, S_i를 만든 추에서 단순히 그 추를 빼면 $S_i - 1$가 된다.
      - 없다면 그게 답이었을 것이다.
    - $S_i - 2$를 측정할 수 있는가?
      - 만약 기존의 추들로 2를 측정할 수 있다면, '' ...
    - ⋮
    - $S_i - k$를 측정할 수 있는가?
      - 만약 k를 측정할 수 없었다면 k가 답이었을 것이고 그 수는 이미 걸러졌을 것이다.
      - 하지만 그 수가 답이 아니었다는 것은 그 수를 측정할 수 있었을 것이고

      그럼 당연히 $S_i - k$도 측정 가능하다.

Approach

- Approach 2
  - 따라서 S_i보다 작은 숫자들은 모두 측정 가능하다.
  - 최종적으로 $S_i + 1$이 $arr[i+1]$보다 작다면 그게 답이다.
  - 탕탕탕
  - 이게 내 결론이다!
  - 시간 복잡도 $O(n)$
  - 인 것 같지만 정렬하기 때문에 $O(\mathrm{n} \log n)$

목차

Code

- Let's coding
- 입력 및 정렬

- Algorithm 적용
  - 만약 $arr[i+1]$이 $S_i + 1$보다 크다면 그게 답
  - 아니라면 계속 갑시다.

  - 1부터 S_n까지 만들 수 있다면 $S_n + 1$이 답

```
int n;
cin >> n;

int arr[1'000];
for (int i = 0; i < n; i++)
    cin >> arr[i];

sort(arr, arr + n);
```

```
int ans = -1, sum = 0;
for (int i = 0; i < n; i++) {
    if (sum + 1 < arr[i]) {
        ans = sum + 1;
        break;
    }

    sum += arr[i];
}

if (ans == -1)
    ans = sum + 1;
```

감사합니다!