제1회 와쿠(AGCU)컵

Official Solutions

출제

- ✓ bamgoesn
- ✓ invrtd_h
- ✓ martinok1103
- ✓ movegreen
- ✓ nflight11
- ✓ shjohw12
- ✓ surung9898

검수

- ✓ bnb2011
- ✓ cgiosy
- ✓ eaststar
- ✓ jyheo98
- ✓ lms0806
- ✓ parkky
- ✓ pjshwa
- ✓ qawbecrdtey
- ✓ tony9402
- ✓ yooyou7

후원

- ✓ StartLink
- ✓ pjshwa
- ✓ utilforever
- ✓ 대회에 후원해주셔서 정말 감사드립니다!!!

문제		의도한 난이도	출제자
A	초코바	**Very Easy**	bamgoesn
B	사격 내기	**Very Easy**	movegreen
C	고양이는 많을수록 좋다	**Easy**	surung9898
D	오렌지먹은지오랜지	**Easy**	martinok1103
E	합금 주화	**Easy**	nflight11
F	콰트로치즈피자	**Easy**	nflight11
G	N결수	**Easy**	kiwiyou
H	△	**Easy**	shjohw12

문제		의도한 난이도	출제자
I	고추장 괄호 문자열	**Easy**	martinok1103
J	사사의 사차원 사탕 봉지	**Medium**	bamgoesn
K	I LOVE JavaScript	**Medium**	invrtd_h
L	OX	**Medium**	shjohw12
M	강아지는 많을수록 좋다	**Medium**	surung9898
N	악보는 거들 뿐	**Medium**	kiwiyou
O	지연 평가	**Medium**	invrtd_h

A. 초코바

`math, arithmetic`
출제진 의도 – **Very Easy**

- ✓ 제출 785번, 정답 576명 (정답률 74.268%)
- ✓ 처음 푼 사람: **sr3gm**, 31초
- ✓ 출제자: bamgoesn

A. 초코바

- ✓ 100원 동전 N개로는 가격이 $100N$원 이하인 초코바만을 살 수 있습니다.
- ✓ 따라서 $M \leq 100N$이면 초코바를 살 수 있습니다.

B. 사격 내기

`math`, `bitmasking`
출제진 의도 – **Very Easy**

- ✓ 제출 619번, 정답 463명 (정답률 76.090%)
- ✓ 처음 푼 사람: **jhnah917**, 75초
- ✓ 출제자: `movegreen`

B. 사격 내기

- ✓ A와 B의 점수가 결정되면, 그 사실만으로 정확히 어떤 과녁을 맞추었는지 결정할 수 있습니다.
- ✓ 이는 과녁의 점수가 2의 연속된 승수로 이루어져 있으므로 A와 B의 점수를 2진법으로 풀면 어떤 과녁을 맞추었는지에 대한 정보를 얻을 수 있으며, 하나의 정수를 2진법으로 푸는 방법은 하나뿐이기 때문입니다.
- ✓ 따라서 A와 B의 점수에서 512점, 256점, ... 순서로 뺄셈을 시도하는 방법을 떠올릴 수 있습니다.
- ✓ 만약 뺄셈에 성공하면 해당 과녁을 맞췄다고 볼 수 있으며, 해당 과녁의 점수를 A와 B 중에 한 명만 맞췄는지를 확인하면 됩니다.

B. 사격 내기

- ✓ 그런데 과녁 하나에 대해 A와 B 둘 중 하나만 과녁을 맞추었는지를 확인하는 것은 boolean XOR 연산입니다.
- ✓ 컴퓨터는 이미 2진수 형태로 정보를 저장하고 있으므로, 각 자릿수마다 XOR 연산을 적용하면 C의 점수를 2진수로 표현한 결과를 얻을 수 있습니다.
- ✓ 이는 곧 bitwise XOR 결과와 같습니다.
- ✓ 따라서 A와 B의 bitwise XOR 값이 C의 점수가 됩니다.

C. 고양이는 많을수록 좋다

`greedy`, `math`
출제진 의도 – **Easy**

- ✓ 제출 870번, 정답 323명 (정답률 37.931%)
- ✓ 처음 푼 사람: **jhnah917**, 226초
- ✓ 출제자: surung9898

C. 고양이는 많을수록 좋다

- ✓ 최소의 행동 횟수로 고양이를 총 N 마리를 집에 들이기 위해서는 **한 번의 마법으로 가능한 한 많은 고양이를 생성**해야 합니다. 두 가지 마법 중 어떤 마법을 사용하는 것이 더 유리할까요?
- ✓ 현재 마도카의 집에 고양이가 k 마리 있다고 가정한다면, 생성 마법을 사용하였을 때 결과는 $k+1$ 마리가 됩니다.
- ✓ 반면, 복제 마법은 현재 집에 있는 고양이를 0 마리부터 k 마리까지 대상으로 복제할 수 있으므로, 결과적으로 k 마리부터 $2k$ 마리 중 임의의 고양이들이 집에 있을 것입니다.

C. 고양이는 많을수록 좋다

- ✓ 그렇다면 생성 마법의 결과인 $k+1$과 복제 마법 결과의 최댓값인 $2k$ 중 언제 한 쪽이 더 큰 값이 될까요?
- ✓ 만약 $k+1 \leq 2k$라고 가정한다면, 결과적으로 $1 \leq k$임을 얻을 수 있습니다. 즉, 현재 마도카의 집에 고양이가 1마리 이상 있다면, 무조건 복제 마법을 사용하는 것이 같은 행동 횟수로 더 많은 고양이를 집에 들일 수 있다는 것입니다.
- ✓ 따라서 가장 처음 1마리의 고양이를 집에 들이기 위한 1번의 생성 마법 이후에는, 언제나 복제 마법을 사용하는 것을 통해 행동 횟수를 최소한으로 줄일 수 있습니다.

C. 고양이는 많을수록 좋다

- ✓ 문제를 풀 때 지켜야 할 또 한 가지의 요구 사항은 **정확히** N 마리의 고양이를 들이는 것입니다.
- ✓ 다만 이는 복제 마법으로 0 마리부터 k 마리까지 중 임의의 수의 고양이를 더할 수 있는 사실을 통해 해결할 수 있습니다.
- ✓ 즉, 만약 현재 상황에서 k 마리를 추가했을 때 N 마리와 같거나 더 많이 고양이를 들이게 된다면, k 보다 작은 어떤 양의 정수를 더함으로써 N 마리의 고양이를 만들 수 있다 는 것입니다.
- ✓ 이는 수식적으로, k 에 $[0, k]$ 중 임의의 수를 더할 수 있으므로, 만약 $N \leq 2k$ 라면 $0 \leq t \leq k$ 인 어떤 t 에 대하여 $k + t = N$ 임을 의미합니다.

C. 고양이는 많을수록 좋다

- ✓ 따라서 $2^0 = 1$ 부터 2 씩 곱해나가면서, 처음으로 $2^x \geq N$ 를 만족하는 x 를 찾는 방식으로 $O(\log N)$ 에 문제를 해결할 수 있습니다.
- ✓ 예를 들어, 예제 2 번인 6 의 경우 $2^2 = 4 < 6$ 이고 $2^3 = 8 \geq 6$ 이므로 답은 $3 + 1 = 4$(처음 생성 마법을 1 번 사용했다는 것을 잊지 마세요!)가 됩니다.
- ✓ $\log N$ 값을 이용하여 수학적으로 $O(1)$ 에 해결하는 것도 물론 가능합니다.
- ✓ 0 의 경우를 예외 처리해야 함 을 잊지 마세요!



D. 오렌지먹은지오랜지

`bruteforcing`, `string`
출제진 의도 – **Easy**

- ✓ 제출 497번, 정답 184명 (정답률 38.632%)
- ✓ 처음 푼 사람: **sweetlay**, 5분
- ✓ 출제자: `martinok1103`

D. 오렌지먹은지오랜지

- ✓ $N = 2000$ 으로 $O\left(N^2\right)$ 풀이를 허용하므로 브루트포스 알고리즘을 사용할 수 있습니다.
- ✓ 앞과 뒤에서부터 각각 길이 1 부터 길이 N 까지의 부분 문자열을 가져옵니다.
  - 가져온 부분 문자열을 앞에서부터 같은 위치에서 1 글자씩 비교해 다른 문자가 1 개라면 YES 를 출력하고 프로그램을 종료합니다.
- ✓ 이 과정은 한 번 할 때마다 $O(N)$ 이고, 최대 $O(N)$ 번 반복됩니다.
- ✓ 모든 길이에서 비교를 했는데 1 개만 다른 것이 없다면 NO를 출력하고 프로그램을 종료합니다.

E. 합금 주화

`arithmetic, physics?`
출제진 의도 – **Easy**

- ✓ 제출 277번, 정답 171명 (정답률 63.177%)
- ✓ 처음 푼 사람: **ainta**, 5분
- ✓ 출제자: `nflight11`

E. 합금 주화

- ✓ 문제에서 요구하는 것은 합금 주화의 밀도이고, 이를 구하기 위해서는 합금 주화의 질량과 부피를 알아야 합니다. 과연 이 두 가지를 모두 알아야만 할까요?
- ✓ **밀도는 물질의 양에 영향을 받지 않습니다.** 따라서 질량이나 부피 어느 한 쪽을 고정하고 문제의 정보를 이용하여 다른 쪽의 값을 계산하면 됩니다.
- ✓ 입력에서 제공하는 것은 퍼센트로 나타낸 질량 비율이므로, 주화의 질량을 100이라고 두면 계산이 쉬워집니다.
- ✓ 엄밀한 계산이 필요하다면 주화의 질량을 M로 두고 이후 이어지는 과정을 똑같이 따라합시다.

E. 합금 주화

- ✓ 우선 두 금속의 밀도를 $d, D\ (d < D)$로 재배열하여 줍시다.
- ✓ 밀도가 d인 금속이 합금 주화에서 차지하는 질량은 $100 - \chi$이므로, 이 금속의 혼합 전 부피는 질량을 밀도로 나눈 $\dfrac{100-\chi}{d}$ 만큼이 됩니다. 마찬가지로 밀도 D인 금속은 질량으로 χ만큼 들어갈 테니, 이 금속의 혼합 전 부피는 $\dfrac{\chi}{D}$ 만큼이 됩니다.
- ✓ 주화의 부피는 혼합 전 두 금속의 합이며, 질량은 처음에 100이라고 두었으니, 합금 주화의 밀도를 얻을 수 있습니다.
- ✓ 나눗셈의 결과가 정수가 되지 않을 수 있으니, 정수 나눗셈 대신 실수 나눗셈을 사용할 필요가 있습니다. 계산이 단순한 데 비해 실수 오차는 넉넉하게 주어졌으니, 실수 오차로 인한 오답은 크게 걱정하지 않아도 됩니다.

F. 콰트로치즈피자

`string`
출제진 의도 – **Easy**

- ✓ 제출 392번, 정답 232명 (정답률 60.459%)
- ✓ 처음 푼 사람: **kirvy810**, 143초
- ✓ 출제자: `nflight11`

F. 콰트로치즈피자

- ✓ 문제에서 요구하는 것은 **서로 다른 치즈가 4종 이상 존재하는가?** 입니다.
- ✓ [string] Cheese 식의 문자열인 토핑만이 치즈이니, 토핑 중 치즈인 것을 판별하는 것이 필수적입니다.
- ✓ 이는 각 토핑마다 해당 토팡의 마지막 6개 문자만을 검사하는 것으로 쉽게 판별할 수 있습니다. (Python에서는 endswith 함수를 통하여 훨씬 쉽게 검사할 수 있습니다.)
- ✓ 단, 5글자 이하의 토핑에 대해서는 예외 처리를 해 주어야 합니다.

F. 콰트로치즈피자

- ✓ 치즈들 중 서로 다른 것이 4종 이상 존재하는지 판별하는 방법은 여러 가지가 있습니다.
- ✓ 모든 토핑을(혹은 모든 치즈를) 순회하며 이미 등장한 치즈들이 아닐 경우 카운트를 하나씩 늘리는 방법이 가장 쉬운 방법입니다.
- ✓ set 자료구조를 알고 있다면 모든 치즈를 특정한 set에 집어 넣고 이 set의 길이가 4 이상인지 확인하는 간단한 풀이를 사용할 수 있습니다.
- ✓ 토핑의 최대 개수가 100 으로 그렇게 크지 않기 때문에, 4중 반복문을 사용하는 $O(N^4)$ 방법 외에는 대부분의 풀이과정이 시간제한에 걸리지 않습니다.

G. N결수

`number_theory`
출제진 의도 – **Easy**

- ✓ 제출 210번, 정답 102명 (정답률 50.952%)
- ✓ 처음 푼 사람: **ainta**, 11분
- ✓ 출제자: `kiwiyou`

G. N결수

- ✓ N 결수를 만드는 방법부터 생각해 봅시다.
- ✓ 가장 간단한 방법으로 문자열을 이용할 수 있습니다. 하지만 긴 문자열을 수로 바꾸는 데 오랜 시간이 걸립니다.
- ✓ 문자열을 거치지 않고 수를 만들 수 있을까요?

G. N결수

- ✓ 현재 수의 자리 수를 알고 있다면, $N-1$ 결수에 $10^{\text{N의 자리 수}}$ 를 곱하고 N 을 더해 N 결수를 만들 수 있습니다.
- ✓ N 의 자리 수는 문자열을 이용해도 되고, 10 으로 여러 번 나눠 구할 수 있습니다.
- ✓ 두 방법 모두 $O(\log N)$ 시간이 걸리므로, **곱셈과 덧셈을 상수 시간으로 가정했을 때** $O(N \log N)$ 입니다.
- ✓ 그런데 N 결수는 매우 큰 수가 될 수 있으므로, 이 가정은 깨지고 시간 초과를 받습니다.

G. N결수

- ✓ 나머지 연산의 특징은, 곱셈과 덧셈 중간마다 나머지를 취해도 결과의 나머지는 같다는 것입니다.
- ✓ 따라서 매 연산마다 나머지를 취해 주면 수가 커지지 않고 원하는 $O(N \log N)$ 시간복잡도를 얻습니다.

H. △

`constructive, trees`
출제진 의도 – **Easy**

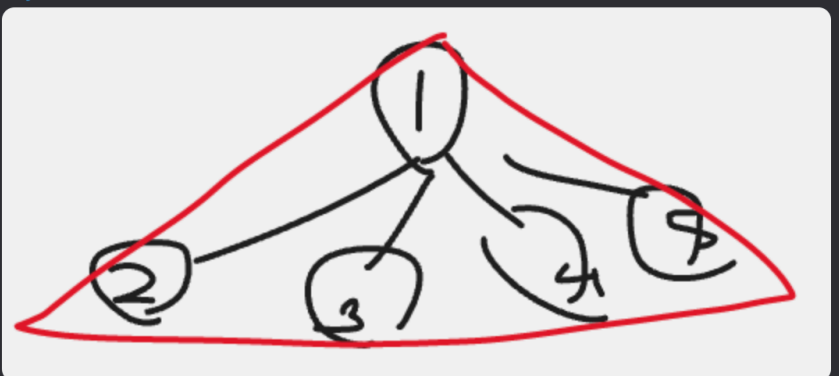
- ✓ 제출 192번, 정답 110명 (정답률 59.375%)
- ✓ 처음 푼 사람: **ainta**, 13분
- ✓ 출제자: `shjohw12`

qawbecrdtey 2023.03.14. 오후 12:55

그나저나 이 문제는 왜 제목이 삼각형인가요?

2023년 3월 15일

shjohw12 2023.03.15. 오전 12:06



H. △

- ✓ 직관적으로, 한 정점을 중심으로 다른 모든 정점들이 연결되어 있는 성형(star) 구조의 트리가 답이 될 것임을 예측할 수 있습니다. 증명은 다음과 같습니다.
- ✓ 트리에서 간선을 통해 직접 연결되어 있는 정점 사이의 거리는 1 이고, 그렇지 않은 정점 사이의 거리의 하한은 2 입니다.
- ✓ 즉, $(N-1) + 2 \times (\binom{N}{2} - (N-1)) \leq \sum_{1 \leq i < j \leq N} \text{dist}(i,\ j)$
- ✓ 성형 구조의 트리는 직접 연결되어 있지 않은 정점 사이의 거리가 2 이므로 최솟값을 갖는 트리입니다.
- ✓ 따라서 최솟값은 $(N-1)^2$ 이고, 한 정점을 중심으로 다른 모든 정점들이 연결되어 있는 트리를 아무거나 하나 출력하면 됩니다.

H. △

- ✓ 그렇다면, 성형 구조의 트리 말고도 문제의 조건을 만족하는 트리가 존재할까요?
- ✓ 결론부터 말하면, 존재하지 않습니다. 즉, 문제의 조건을 만족하는 트리와 성형 구조의 트리는 동치입니다.
- ✓ 해당 증명도 어렵지 않게 할 수 있습니다. 직접 고민해보시길 바랍니다! :)

I. 고추장 괄호 문자열

`string`, `bruteforcing`, `backtracking`
출제진 의도 – **Medium**

- ✓ 제출 140번, 정답 93명 (정답률 67.143%)
- ✓ 처음 푼 사람: **index**, 223초
- ✓ 출제자: `martinok1103`

I. 고추장 괄호 문자열

- ✓ 주어지는 문자열의 길이가 최대 20이므로 $O\left(2^N\right)$ 을 허용하기 때문에, 브루트포스로 해결할 수 있습니다.
- ✓ 주어진 문자열로부터 만들 수 있는 모든 원본 문자열을 생성한 뒤, 그것들이 올바른 괄호 문자열인지 판정합니다.
- ✓ 원본 문자열의 생성은 재귀함수를 통한 백트래킹으로 구현하는 것이 가장 일반적입니다.
- ✓ 올바른 괄호 문자열인지 판정하는 것은 스택을 사용하는 것이 가장 일반적입니다. 그러나 이 문제에는 괄호가 한 종류밖에 없으므로 스택을 대신하는 정수 하나만 들고 있어도 됩니다.

J. 사사의 사차원 사탕 봉지

`binary_search`, `prefix_sum`
출제진 의도 – **Medium**

- ✓ 제출 187번, 정답 94명 (정답률 51.872%)
- ✓ 처음 푼 사람: **ainta**, 18분
- ✓ 출제자: bamgoesn

- ✓ B_i 개의 사탕을 받고 싶어하는 아이에게 사탕을 줄 때 사탕을 꺼내야 하는 횟수는 $A_1 + A_2 + \cdots + A_x \geq B_i$ 인 가장 작은 x 입니다.
- ✓ 이러한 x 는 루프를 통해 $O(M)$ 의 시간 안에 계산할 수 있으며, 이를 모든 아이에 대해 계산하면 $O(NM)$ 의 시간 안에 계산됩니다.
- ✓ 하지만 N 과 M 이 최대 3×10^5 이기 때문에, 이와 같은 방법을 사용하면 시간 초과를 받습니다.
- ✓ 따라서 위 부등식을 만족하는 최소의 x 를 루프로 일일이 찾는 것보다는 빠르게 찾아야 합니다.

- ✓ $[A_1,\ A_1+A_2,\ A_1+A_2+A_3,\ \cdots,\ A_1+A_2+\cdots+A_M]$과 같은 배열을 생각해봅시다. 이 배열을 만드는 데에는 $O(M)$의 시간이 들어갑니다.
- ✓ 모든 i에 대해 $A_i \geq 1$이므로 이 배열은 증가합니다.
- ✓ 따라서, 이 배열에서 값이 M 이상인 가장 작은 값은 이분탐색을 통해 $O(\log M)$의 시간 안에 찾을 수 있습니다.
- ✓ 즉, 찾고자 하는 최소의 x는 위와 같은 배열을 한 번 만들어놓고 나면 아이 하나당 $O(\log M)$의 시간 안에 사탕을 꺼내야 하는 횟수를 찾을 수 있게 됩니다.
- ✓ 추가로, 아이를 쫓아내야 하는지는 위 배열의 마지막 원소의 값을 꺼내야 하는 사탕의 수와 비교해서 판단할 수 있습니다.
- ✓ 이와 같은 방법으로 풀면 최종 시간복잡도는 $O(M+N\log M)$입니다.

K. I LOVE JavaScript

`string`, `parsing`, `implementation`, `ad_hoc`, `stack`
출제진 의도 – **Medium**

- ✓ 제출 144번, 정답 89명 (정답률 68.750%)
- ✓ 처음 푼 사람: **jhnah917**, 23분
- ✓ 출제자: `invrtd_h`

- ✓ 형식을 보면 알 수 있듯이, 실제 Python이나 JavaScript 등지에서 사용하는 Array/List 표현을 약간 (더 분석하기 쉽게) 변형한 것입니다.

- ✓ 일단 주어진 하나의 문자열을 더 작은 단위로 분리합시다.
- ✓ ASON 객체의 표기를 구성하는 것으로는 `'['` 문자, `']'` 문자, 정수 표기, 문자열 표기 등이 있습니다. (이렇게 하나의 표기를 구성하는 더 작은 구성 요소를 토큰(Token)이라고도 합니다.)
- ✓ 운 좋게도, 문자열을 토큰 단위로 분리하는 것(Tokenizing)은 상당히 쉽습니다. 모든 토큰이 스페이스 하나 단위로 미리 분리되어 주어지기 때문입니다.
- ✓ 이것은 Python에서는 한 줄 입력을 받은 뒤 `str.split()` 메서드를 사용하여 해결할 수 있고, C++에서는 `cin`을 사용하여 입력이 끝날 때까지 계속 입력받으면 됩니다.
- ✓ Tokenizing의 결과로 우리는 문자열로 구성된 Array를 갖게 됩니다.

- ✓ 이제 정수와 문자열은 주어진 매뉴얼대로 처리하면 되는데, ASON 객체 안에 다른 ASON 객체가 들어 있을 수도 있다는 사실이 상당히 까다롭습니다.
- ✓ 이를 다양한 방법으로 해결할 수 있습니다.

1. $O(N^2)$ 풀이

- ✓ 주어진 ASON 객체가 다른 ASON 객체를 포함하고 있다고 합시다.
- ✓ 만약 그렇다면, 그 ASON 객체를 따라 들어갑니다. 만약 이 객체도 다른 ASON 객체를 포함하고 있다면, 계속해서 ASON 객체를 따라 들어갑니다.
- ✓ 이 절차는 언젠가 멈추는데, ASON 객체의 크기는 유한하기 때문입니다. 그렇게 되면 최종적으로는 다른 ASON 객체를 포함하지 않는 어떤 ASON 객체를 얻습니다.
- ✓ 이 객체의 크기는 쉽게 계산할 수 있고, 계산한 객체를 전체 Array에서 지웁니다. 이 과정은 $O(N)$에 처리 가능합니다.
- ✓ 전체 ASON 객체의 개수도 $O(N)$개까지 있을 수 있으므로, 최종 시간복잡도는 $O(N^2)$입니다.

2. $O(N)$ 풀이 1

- ✓ Parsing에서 자주 사용하는 stack을 이용하면 $O(N)$에 문제를 해결할 수 있습니다.
- ✓ 각 토큰을 순차적으로 훑으며 알고리즘을 진행합니다.
- ✓ `']'` 토큰이 아닌 토큰을 발견할 때마다 스택에 그 토큰을 집어넣습니다.
- ✓ `']'` 토큰을 발견하면 스택에서 원소를 빼기 시작합니다. `'['` 토큰을 발견할 때까지 원소를 뺍니다.

- ✓ 토큰을 뺄 때, 정수를 뺄 때마다 8byte를 더해 주고, 문자열을 뺄 때마다 (문자열의 길이)+12byte를 더해 줍니다.
- ✓ 마지막으로 '[' 토큰을 발견하면 그것까지만 제거한 후 스택에서 원소를 빼는 행동을 멈추고, 8byte를 더해 줍니다.
- ✓ 우리가 스택에 ']' 토큰을 넣은 적이 없으므로, 이것만 해도 우리는 모든 행동을 다 고려한 것입니다.

3. $O(N)$ 풀이 2

- ✓ 스택을 사용하는 알고리즘을 개선하면 사실 스택 없이도 이 알고리즘이 돌아간다는 사실을 알 수 있습니다.
- ✓ 알고리즘의 리턴 값을 ret에 저장한다고 했을 때, ret에 값이 더해지는 경우는 다음의 3가지 경우뿐입니다.
- ✓ 1. 정수 토큰이 있을 때 (+8byte)
- ✓ 2. 문자열 토큰이 있을 때 (+12+len(s)byte)
- ✓ 3. '[' 토큰이 있을 때 (+8byte)
- ✓ 따라서 ASON 객체들끼리 어떻게 포함 관계를 이루고 있는지와는 전혀 상관없이, 단순히 이것들의 개수를 세기만 하면 됩니다.

- ✓ 이 사실은 다른 방법으로도 증명 가능합니다.
- ✓ (전체 용량) = (정수, 문자열 원소가 차지하는 용량) + (ASON 객체들이 추가적으로 차지하는 용량)
- ✓ = (정수, 문자열이 차지하는 용량) + (ASON 객체의 수) * 8byte
- ✓ = (정수, 문자열이 차지하는 용량) + [토큰의 수 * 8byte

L. OX

`math, dp`
출제진 의도 – **Medium**

- ✓ 제출 133번, 정답 94명 (정답률 71.429%)
- ✓ 처음 푼 사람: **cozyyg**, 15분
- ✓ 출제자: `shjohw12`

sol1) math

- ✓ 이진수에서 1을 빼는 과정을 생각해 봅시다. 가장 낮은 1인 비트를 0으로 바꾸고, 그보다 낮은 비트를 전부 1로 바꾸게 됩니다.
- ✓ 주어진 문자열에서 O를 1로, X를 0으로 생각하면 이진수를 거꾸로 뒤집은 형태라는 것을 알 수 있습니다.
- ✓ 따라서 왼쪽에서 i번째에 있는 O를 2^i로 생각하여 모두 더한 값이 답이 됩니다.

sol2) dp

- ✓ 가장 왼쪽에 있는 O의 위치가 i 번째일 때 X로 만들기까지 시행하는 연산의 횟수를 dp_i 로 정의합니다.
- ✓ $dp_1 = 1$
- ✓ $dp_i = 1 + dp_1 + dp_2 + \cdots + dp_{i-1}$
- ✓ O인 위치의 dp 값을 모두 더한 값이 답이 됩니다.

M. 강아지는 많을수록 좋다

dp
출제진 의도 – **Medium**

- ✓ 제출 132번, 정답 85명 (정답률 64.394%)
- ✓ 처음 푼 사람: **nick832**, 21분
- ✓ 출제자: surung9898

- ✓ 예제에서 유추할 수 있듯이, 본 문제는 강아지를 많이 생성하는 마법을 먼저 쓰는 등의 그리디 알고리즘으로는 풀 수 없습니다.
- ✓ 그러나, **부분 문제의 답으로 원래 문제의 답을 구할 수는 있습니다.** 일례로, 3-생성 마법과 4-생성 마법이 있다면, 7마리의 강아지를 집에 들이기 위한 최소의 행동 횟수는 $min(3$마리를 들이는 데의 최소 행동 횟수 $+1, 4$마리를 들이는 데의 최소 행동 횟수 $+1)$ 이 될 것입니다.



M. 강아지는 많을수록 좋다

- ✓ 따라서, 동적 계획법으로 본 문제를 접근 해야 합니다.
- ✓ 이 경우, 점화식은 A-생성 마법과 B-생성 마법에 대해 상술한 내용을 수식화하여 $dp[k] = min(dp[k-A], dp[k-B]) + 1$ 로 나타낼 수 있습니다.
- ✓ 제약 사항이 없는 경우, 본 문제는 이런 방식으로 $O(N)$ 에 문제를 해결할 수 있습니다.

- ✓ **제약 사항에 해당하는 닫힌구간에 대해서 별도로 고려**해주어야 합니다.
- ✓ 다만, 닫힌구간의 최대 개수가 100 이므로, 특정 마리수가 어떤 닫힌구간에 속하는 지 전부 체크해도 제한 시간 내에 동작할 수 있습니다.
- ✓ 예를 들어, 각 마리수에 대해 적어도 1개 이상의 닫힌구간에 포함되는 지를 미리 계산해 놓은 뒤, $dp[k], dp[k-A], dp[k-B]$ 각각이 여기에 해당하는 지 체크한 후 점화식을 계산하는 방식으로 본 문제를 해결할 수 있습니다.
- ✓ 물론, BFS나 DFS 알고리즘으로도 DP와 유사하게 본 문제를 해결할 수 있으며, DP/BFS/DFS 모두 $O(NM)$ 의 시간 복잡도로 구현할 수 있습니다.

N. 악보는 거들 뿐

`greedy`, `implementation`
출제진 의도 – **Medium**

- ✓ 제출 225번, 정답 71명 (정답률 31.556%)
- ✓ 처음 푼 사람: **nick832**, 18분
- ✓ 출제자: `kiwiyou`

- ✓ 인접한 두 수의 상대적 크기가 유지되도록 하며, 수열의 최댓값을 최소화해야 합니다.
- ✓ 하나의 수가 두 번 이상 연속해서 나오는 경우는 수열을 결정하는 데 영향을 주지 않으므로, 문제를 단순화하기 위해 연속한 같은 수는 하나만 남기고 모두 지워버리겠습니다.
- ✓ 이 수열을 q_i 라 하고, q_i 의 어떤 구간 $[l, r]$ 이 $q_l < q_{l+1} < q_{l+2} < \cdots < q_r$ 이거나 $q_l > q_{l+1} > q_{l+2} > \cdots > q_r$ 을 만족한다면 "구간 $[l, r]$ 은 단조적이다" 라고 하겠습니다.
- ✓ 수열 q_i 는 단조적인 구간 몇 개의 합집합으로 나타낼 수 있습니다.
- ✓ 예를 들어, 수열 $\{1, 2, 5, 4, 3, 4, 1\}$ 은 단조적인 구간 $\{1, 2, 5\}$, $\{5, 4, 3\}$, $\{3, 4\}$, $\{4, 1\}$ 을 합한 것입니다.

- ✓ 모든 단조 구간 길이의 최댓값을 A라 하면, 각 구간의 최댓값 대신 A를, 최솟값 대신 1을 써도 단조성이 유지되도록 중간 수들을 바꿀 수 있습니다. 각 구간의 길이보다 항상 1부터 A까지의 정수가 더 많거나 개수가 같기 때문입니다.
- ✓ 따라서 답 N은 A 이상입니다.
- ✓ 그럼 $N < A$일 수 있을까요?
- ✓ 앞과 유사한 논리를 적용하면, 길이가 A인 구간은 1부터 N까지의 정수보다 더 많은 수를 포함해야 하므로 불가능하다는 것을 알 수 있습니다.
- ✓ 따라서 구하는 $N = A$이고, 이 A는 현재 수열의 증감 상태를 나타내는 변수와 이전 수를 기반으로 $O(N)$에 계산할 수 있습니다.

O. 지연 평가

`arithmetic`, `ad_hoc`, `implementation`
출제진 의도 – **Medium**

- ✓ 제출 156번, 정답 89명 (정답률 66.026%)
- ✓ 처음 푼 사람: **nick832**, 11분
- ✓ 출제자: `invrtd_h`

- ✓ Naive하게 모든 명령을 처리하는 것은 $O(NQ)$ where $N = 1\,234\,567\,890\,123$ 이므로 시간 초과가 나야 합니다. 사실 배열을 직접 만드는 순간 그 크기가 이미 1TB를 넘어가므로 메모리 초과를 받습니다.

- ✓ 밑에 Python의 range 객체의 구현 방법을 적어 놓은 힌트가 있습니다.
- ✓ “Python의 range 객체가 구현된 방법처럼 우리도 `start`, `stop`, `step` 3개의 변수만을 갖고 있어도 되겠다” 고 생각한 사람은 눈치가 빠른 것입니다.
- ✓ 사실 문제 조건을 잘 보면 S에서 모든 원소가 없어지는 일은 일어나지 않는다는 사실을 알 수 있으므로, `stop`을 들고 있는 것조차 필요 없습니다.

- ✓ 따라서 a, step의 단 2개의 변수만을 들고 있으면 됩니다.
- ✓ a는 현재 가장 작은 원소가 무엇인가, step은 S 내부의 원소들끼리 얼마의 차이를 두고 떨어져 있는가를 나타냅니다.
- ✓ 예를 들어 S = [5, 11, 17, 23, ⋯]라면 a = 5, step = 6입니다.

- ✓ `0 x` 명령이 들어오면, a에 x만큼의 수를 더합니다. 더하는 명령이므로 `step`에는 변화가 없습니다.
- ✓ `1 x` 명령이 들어오면, a에 x를 곱합니다. 곱할 때 `step`의 값도 곱해진 만큼 증가하게 되므로, `step`에도 같이 x를 곱해 줍니다.
- ✓ `2 n` 명령이 들어오면, 가장 작은 n개의 수를 제거해야 하므로, `a, a + step, a + step + step` … 등의 수를 지워야 합니다. 이렇게 하고 나면 남은 수 중 가장 작은 수는 `a + step * n` 입니다. 따라서 a에 `step * n`을 더해줍니다.
- ✓ `3` 명령이 들어오면, a를 출력하면 됩니다.
- ✓ 모든 명령은 $O(1)$에 처리되므로, 전체 시간복잡도는 $O(Q)$ 입니다.

- ✓ 여기까지의 풀이는 힌트와 가장 가까운 풀이인데, 함수형 프로그래밍에 익숙하다면 다음과 같은 풀이도 생각할 수 있습니다.
- ✓ `a`, `b`, `idx`라는 3개의 64bit int 변수를 들고 있습니다.
- ✓ `idx`라는 변수는 현재 몇 번째 숫자에 관심이 있는지를 뜻하고, `a`, `b`는 일차함수 $f = ax + b$를 뜻합니다.
- ✓ 3번 명령이 들어올 때마다 `a * idx + b` $(= f(idx))$를 출력합니다.

- ✓ 다른 명령은 다음과 같은 방식으로 처리합니다.
- ✓ `0 n` 명령이 들어오면, 현재 갖고 있는 일차함수 $f = ax + b$에 일차함수 $g = x + n$을 합성시켜 얻은 함수 gf를 새로운 f로 합니다.
- ✓ `1 n` 명령이 들어오면, 현재 갖고 있는 일차함수 $f = ax + b$에 일차함수 $g = nx$를 합성시켜 얻은 함수 gf를 새로운 f로 합니다.
- ✓ `2 n` 명령이 들어오면, `idx`를 `n`만큼 늘립니다.
- ✓ 시간복잡도는 첫 번째 풀이와 같습니다.

- ✓ 이 풀이는, 배열 S를 (원본 배열 S_0 = `[1, 2, 3, ⋯]`)와 (배열에 씌인 Map $y = ax + b$)의 결합으로 보는 풀이입니다. 이 방법도 지연 평가가 적용되었습니다.
- ✓ Map이라는 이름에 익숙하지 않다면, Map이란 function의 다른 이름이라고 생각합시다.
- ✓ 우리가 만약 S_0에 있는 모든 원소에 함수 $y = ax + b$를 적용하고 싶다면 $1, 234, 567, 890, 123$번의 반복을 돌아야 할 것 같지만, 실제로 우리는 Map을 씌워만 놓고 S_0 내부의 원소는 아무것도 건드리지 않았습니다.
- ✓ Map이 효력을 발휘하는 시점은 Map을 씌우는 때가 아니라 배열에서 실제로 원소를 꺼낼 때이므로, 이것도 지연 평가라고 볼 수 있는 것입니다.
- ✓ 참고로 Python의 map 객체에도 비슷하게 지연 평가가 적용되어 있습니다.

- ✓ 당연하지만 S_0를 구현할 때 실제로 $1, 234, 567, 890, 123$개의 원소를 모두 만드는 것은 바람직하지 못합니다. Python의 range 객체처럼 필요할 때마다 그때그때 만들어서 주는 것이 효과적입니다.
- ✓ 또한 검수 과정 중에 다음의 특이한 풀이가 나온 적이 있는데, 위 풀이에서처럼 하나의 Map만을 관리하지 않고, 몇만 개의 Map을 모두 지연 평가하여 S_0 위에 씌워버린 것입니다.
- ✓ 이렇게 하면 S_0에서 원소 하나를 뽑을 때마다 몇만 개의 Map을 거쳐야 하므로 오히려 시간 초과가 납니다.
- ✓ 따라서 지연 평가가 빠른 소스 코드 실행에 언제나 도움이 되지는 않는다는 사실을 알 수 있습니다.