제 2회 PNU CodeRace

문제 해설

PULSE
부산대학교 알고리즘 동아리

SE:LON
부산대학교 정보컴퓨터공학부 학생회

부산대학교
정보컴퓨터공학부

University innovation
대학혁신
지원사업

STARTLINK

한빛미디어
HanbitMedia, Inc.

문제		의도한 난이도	출제자
A	찾았다 악질	Easy	Rche
B	열 정렬정렬 정	Easy	whquddn55
C	땅 두 배로 따먹기	Medium	tykr0001
D	pqbd	Hard	Rche
E	터트려라 풍선	Medium	whquddn55
F	첨탑 부수기	Hard	tykr0001
G	고장난 통신탑	Medium	kanght1219
H	Next Level	Medium	kmiiiaa

검수자: jh05013, lky7674, tony9402, chansol

A. 찾았다 악질

알고리즘 구분: 구현, 반복문 순회, 유한합

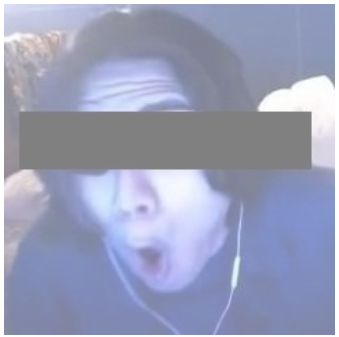
의도한 난이도: **Easy**

- 제출 횟수 238번, 정답 54명 (정답률 22%)
- 출제자: Rche

A. 찾았다 악질

- 스트리머의 수 N, 시청자 변화 추이의 길이 M에 대해서 각 시청자 기록을 순회하며 각 열 원소의 차에 대한 **절댓값의 유한합을 구하는 문제**입니다.
- 랄파를 제외한 나머지 스트리머를 모두 순회하므로 **실제 루프는 $(N-1) \times M$ 번 순회**합니다.
- $\sum|L_i - K_{ij}|$ 의 값을 i마다 구해주고, 2000을 넘는지 확인해주면 됩니다.
- 계산 도중 시청자 차이의 합이 2000을 넘는 경우 jump(continue) **함으로써 코드의 효율을 높일 수 있습니다.**

- TMI : 랄파는 유명 스트리머 두 분이 모티브지만 엄연히 현실에 존재하지 않는 가상의 인물입니다.
- 출제자는 클린한 인터넷 팬문화를 지향합니다. ^^7



B. 열 정렬정렬 정

알고리즘 구분: 정렬, 그리디, 분리 집합, 순열 사이클 분할

의도한 난이도: **Easy**

- 제출 횟수 140번, 정답 14명 (정답률 10%)
- 출제자: whquddn55

B. 열 정렬정렬 정

- 배열을 정렬 상태로 만들기 위해 적용해야하는 swap 연산의 **최소 횟수**를 구하는 문제입니다.
- 정렬되지 않은 상태의 배열을 앞에서 부터 보겠습니다.
- 원래 배열의 0번째 위치의 값을 보면 정렬했을 때의 값과 동일합니다.
- 즉, 0번째 위치의 값은 이미 정렬된 상태이므로 건드리지 않는 것이 좋을 것 같습니다.

Index	0	1	2	3	4	5
array	0	2	1	54	21	23
sorted	0	1	2	21	23	54

B. 열 정렬정렬 정

- 이제 1번째 위치의 값을 보겠습니다.
- 정렬되었을 때의 값은 1 입니다. 하지만, 현재는 2가 위치해 있습니다.
- 정렬 상태로 만들기 위해서는 1번째 위치의 값을 1로 바꾸어야합니다.
- 따라서 1과 swap합니다.

Index	0	1	2	3	4	5
array	0	2	1	54	21	23
sorted	0	1	2	21	23	54

B. 열 정렬정렬 정

- 이제 2번째 위치의 값을 봅시다.
- 운이 좋게도 방금 한 swap덕분에 정렬된 값으로 되었네요.
- 이러한 방식으로 앞에서 부터 현재 위치의 값이 정렬된 값과 다를 때 swap 연산을 적용하여 그리디하게 진행할 수 있습니다.
- 정렬에 $O(N \log N)$, 배열 순회에 $O(N)$의 복잡도를 갖기 때문에 $O(N \log N)$에 풀 수 있습니다.

Index	0	1	2	3	4	5
array	0	1	2	54	21	23
sorted	0	1	2	21	23	54

B. 열 정렬정렬 정

- $a[i]$는 $a[j]$의 위치에, $a[j]$는 $a[k]$의 위치에, $a[k]$는 $a[i]$의 위치에… 이런 식으로 정렬하기 위해 이동해야하는 원소들이 cycle을 이루게 됩니다.
- cycle을 풀어주기 위해서는 최소 $\mathbf{len(cycle) - 1}$만큼의 swap이 필요합니다.
- 서로 다른 cycle c_i와 c_j에 속한 원소 끼리 swap을 하게 될 경우 cycle의 크기가 더 커지게 됩니다. 따라서 같은 cycle에 있는 원소끼리만 swap해주는 것이 최적입니다.
- 배열을 순회하면서 해당 *Index*에 위치해야하는 원소와 swap하는 것이 cycle을 풀어주는 것과 동일한 작업이기 때문에 사이클 계산 없이 그리디하게 간단하게 풀 수 있습니다.

Index	0	1	2	3	4	5
array	0	2	1	54	21	23

C. 땅 두 배로 따먹기

알고리즘 구분: 누적 합, 동적 프로그래밍, 게임 이론, 완전 탐색

의도한 난이도: Medium

- 제출 횟수 32번, 정답 0명 (정답률 0%)
- 출제자: `tykr0001`

C. 땅 두 배로 따먹기

- 각 플레이어는 두 행동 중 어떤 행동을 하더라도 남아있는 땅 중 가장 큰 땅을 골라 먹는 것이 최선의 행동이므로 땅의 크기를 기준으로 내림차순 정렬하여 큰 순서대로 먹습니다.
- 각 플레이어가 2번 행동을 언제 했는지가 주어진다면 **각 참가자가 먹은 땅의 크기를 누적 합을 이용하여 $O(1)$로 계산 가능**합니다.

C. 땅 두 배로 따먹기

- 땅의 갯수가 13(= N)개인 아래의 표를 예시로 들어 산지니가 2번 행동을 사용할 땅의 *Index*를 4, 현기가 2번 행동을 사용할 땅의 *Index*를 8이라고 한다면 각 플레이어가 먹을 땅의 크기는 아래와 같이 계산됩니다. (산지니가 먹을 땅의 크기 = A, 현기가 먹을 땅의 크기 = B)

$$A = (a[0] + a[2] + a[4] + a[5]) \times 2 + a[7] + a[10] + a[12]$$

$$B = (a[1] + a[3] + a[6] + a[8] + a[9]) \times 2 + a[11]$$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
a	8	7	6	5	5	4	4	4	3	3	2	1	1

C. 땅 두 배로 따먹기

- 홀수 번째 땅의 크기의 합을 누적하여 저장한 odd와 짝수 번째 땅의 크기의 합을 누적하여 저장한 $even$을 사용하여 각 플레이어가 먹을 땅의 크기를 다시 표현하면 아래와 같습니다.

$$A = (even[4] + a[5]) \times 2 + (odd[7] - odd[5]) + (even[12] - even[8])$$

$$B = (odd[3] + even[8] - even[4] + a[9]) \times 2 + (odd[12] - odd[9])$$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
a	8	7	6	5	5	4	4	4	3	3	2	1	1
odd	0	7	7	12	12	16	16	20	20	23	23	24	24
even	8	8	14	14	19	19	23	23	26	26	28	28	29

C. 땅 두 배로 따먹기

- 이전 슬라이드의 식을 땅의 갯수가 N, 산지니가 2번 행동을 할 땅의 $Index$를 i, 현기가 2번 행동을 할 땅의 $Index$를 j라고 했을 때 일반화하면 아래와 같습니다. (이때, $i < j$)

$$A = (even[\,i\,] + a[\,i+1\,]) \times 2 + (odd[\,j\,] - odd[\,i+1\,]) + (even[\,N-1\,] - even[\,j\,])$$

$$B = (odd[\,i\,] + even[\,j\,] - even[\,i\,] + a[\,j+1\,]) \times 2 + (odd[\,N-1\,] - odd[\,j+1\,])$$

Index	0	1	2	3	4	5	6	7	8	9	10	11	12
a	8	7	6	5	5	4	4	4	3	3	2	1	1
odd	0	7	7	12	12	16	16	20	20	23	23	24	24
even	8	8	14	14	19	19	23	23	26	26	28	28	29

C. 땅 두 배로 따먹기

- 현기가 산지니보다 먼저 2번 행동을 수행하는 경우에도 유사한 과정을 거쳐 아래와 같은 수식을 만들 수 있습니다. 즉, 현기가 2번 행동을 할 땅의 $Index$가 i, 산지니가 2번 행동을 할 땅의 $Index$가 j라고 했을 때 $i < j$ 인 경우입니다.

$$B = (odd[\,i\,] + a[\,i+1\,]) \times 2 + (even[\,j\,] - even[\,i+1\,]) + (odd[\,N-1\,] - odd[\,j\,])$$

$$A = (even[\,i\,] + odd[\,j\,] - odd[\,i\,] + a[\,j+1\,]) \times 2 + (even[\,N-1\,] - even[\,j+1\,])$$

C. 땅 두 배로 따먹기

- 한 플레이어가 먼저 $Index\ i$에서 2번 행동을 소모했다고 가정하면 다른 참가자는 그 이후에 **어떤 $Index$에서 2번 행동을 하는 것이 최선인지 선형적으로 탐색**$O(N)$하여 찾으면 됩니다.
- 예를 들어 i가 짝수일 경우, 산지니가 먼저 2번 행동을 소모했으므로 현기는 본인이 먹을 땅의 크기가 최대이면서 산지니가 먹을 땅의 크기는 가능한 한 작아지게 되는 $Index$를 탐색하여 찾을 수 있습니다. 이때의 $Index$를 j라고 하면 산지니/현기가 먹을 땅의 크기를 각각 A_j와 B_j 로 표현할 수 있습니다.

C. 땅 두 배로 따먹기

- 따라서 $DP[i][0]$, $DP[i][1]$를 각각 “어떤 플레이어가 먼저 2번 행동을 소모한 $Index$가 i 이상일 때 산지니/현기가 최선의 행동으로 먹을 수 있는 땅의 크기”라고 정의하면 아래와 같은 의사 코드를 작성할 수 있습니다.

if i is an even number	if i is an odd number
$DP[i][0] = A_j$	$DP[i][0] = A_j$
$DP[i][1] = B_j$	$DP[i][1] = B_j$
if $DP[i][0] < DP[i+1][0]$	*if* $DP[i][1] < DP[i+1][1]$
$DP[i][0] = DP[i+1][0]$	$DP[i][1] = DP[i+1][1]$
$DP[i][1] = DP[i+1][1]$	$DP[i][0] = DP[i+1][0]$
else if $DP[i][0] == DP[i+1][0]$	else if $DP[i][1] == DP[i+1][1]$
$DP[i][1] = \min(DP[i][1], DP[i+1][1])$	$DP[i][0] = \min(DP[i][0], DP[i+1][0])$

C. 땅 두 배로 따먹기

- 시간복잡도는 각 i에 대하여 선형탐색을 하므로 $O(N^2)$ 입니다.
- 공간복잡도는 $O(N)$ 입니다.

D. pqbd

알고리즘 구분: 문자열, 다이나믹 프로그래밍, Manacher's Algorithm, Map(Dictionary)

의도한 난이도: Hard

- 제출 횟수 1번, 정답 22명 (정답률 4%)
- 출제자: Rche

D. pqbd

- Longest palindromic substring을 구하는 문제의 응용판입니다.
- $O(N)$에 구해야 하기 때문에 **Manacher's Algorithm**을 사용합니다.
- Palindrome을 구하되 갱신 조건을 거울 대칭 / 점 대칭으로 하고, **가장 긴 거울 대칭 회문과 가장 긴 점 대칭 회문을 각각 구해서** 더 긴 것을 찾으면 됩니다.
  ("pop" 대신 "poq", "pod"를, "puup" 대신 "pund"를 찾는 방식으로 전환)
- $mirror['p'] = 'q'$, $turn['n'] = 'u'$와 같이 정의, $S[i] == S[n\text{-}i\text{-}1]$ 대신 $S[i] == mirror[S[n\text{-}i\text{-}1]]$ or $S[i] == turn[S[n\text{-}i\text{-}1]]$로 조건문 수정
- 처음에는 Parsing 문제로 구상했으나 그렇게 풀면 stack에서 탈출하는 endpoint를 판별하는데 구현상의 문제가 생길 것으로 예상됩니다. (ex : "poooq") 혹시 파싱으로 푸신 분 있으면 코드 꼭 보고 싶습니다.

D. pqbd

- Manacher's Algorithm에 대한 간단한 설명
- **가장 긴 홀수 길이 팰린드롬을 찾는 해법**입니다.
- 문자열 S에 대해서 $A[i]$는 $S[i]$를 중심으로 **가장 긴 회문의 반지름 크기**라고 정의합니다.
  (S = "bananac"일 경우 $A[4] = 2$)
- 따라서 **$S[i-A[i]]$ 부터 $S[i+A[i]]$는 회문입니다.** ("anana")
- $S[i-A[i]-1]$ 부터 $S[i+A[i]+1]$은 회문이 아닙니다. ("bananac")

D. pqbd

- index i에 대해서 $j<i$인 모든 j에 대해서 $A[j]$가 계산되어있다고 가정합니다.
- $r = max(j + A[j])$이고, 이때의 j를 p라고 정의합니다.
- $i > r$일 경우, **i가 가장 긴 회문 범위에 포함되지 않으므로** $A[i]$의 초기값은 0 입니다.
- 아닐 경우, $S[p]$를 중심으로 $S[p-A[p]] \sim S[p+A[p]]$는 회문이므로 해당 회문 내에서 $\boldsymbol{A[2*p-i] = A[i]}$ ***입니다.*** (예: “segaages” 라는 회문에서, “segaag”를 뒤집은 “gaages”를 겹치면 원래 회문이 됨)
- “bananac” 에서 i=4일때 p=3, r=5가 되며, “anana”가 회문이므로, 대칭인 두 n을 중심으로 하는 회문은 “ana”, “ana”로 동일함
- 이때, 계산된 회문 범위인 r을 넘길 수 있습니다. r 이후의 값은 i를 중심으로 회문이 되는지 확인되지 않았습니다. 따라서 $A[i]$의 초기값은 $\boldsymbol{min(r-i, A[2*p-i])}$ 입니다.
- 초기값이 정해진 뒤 Palindrome 조건을 양옆을 탐색하며 갱신합니다.

D. pqbd

- Q: 홀수 팰린드롬만 구할 수 있다면 짝수 팰린드롬은 어떻게 구하나요?
- A: **길이를 약 두배로 불립니다.**

  (ex: “bananac” → “*b*a*n*a*n*a*c*”, “common” → “*c*o*m*m*o*n*”)
- “common”에 Manacher’s Algorithm 적용시 : 정답은 “o” 또는 “m”
- “*c*o*m*m*o*n*”에 Manacher’s Algorithm 적용시 : 정답은 “*o*m*m*o*”
- ***을 다 지워주면 놀랍게도 짝수 팰린드롬이 나옵니다.**
- 더 자세한 설명은 구글에 검색하세요.
- 참조한 글 : https://www.crocus.co.kr/1075

E. 터트려라 풍선

알고리즘 구분: 오프라인 쿼리, 분리 집합

의도한 난이도: Medium

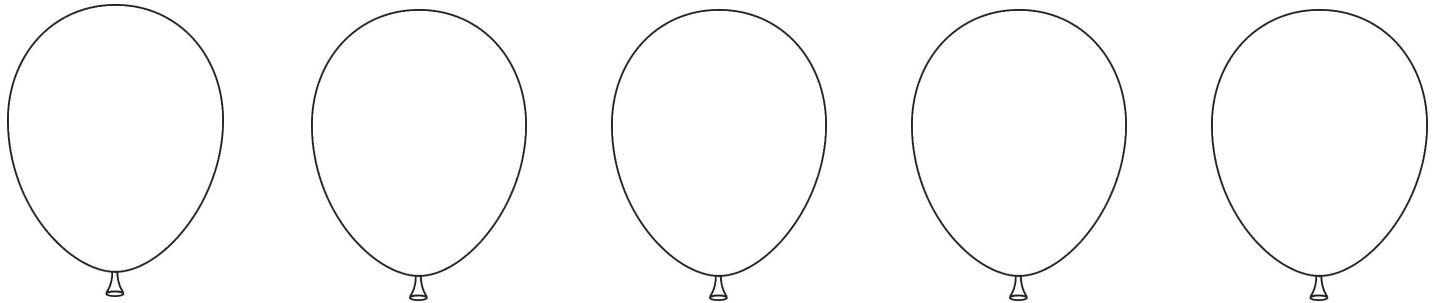
- 제출 횟수 41번, 정답 0명 (정답률 0%)
- 출제자: whquddn55

E. 터트려라 풍선

- 풍선 다트 게임에서 매 라운드 터트린 풍선의 순서가 주어지고, **최종 점수**를 계산하는 문제입니다.
- 문제에서 설명한대로 풍선을 터트릴 때 마다 *segment*에 대한 정보를 갱신하면 풀 수 있을 겁니다. 하지만, 구현해야할 양이 상당히 많습니다.
- 매번 풍선을 터트릴 때 마다 해당 풍선이 포함되었던 *segment*를 빠르게 판별할 수 있어야합니다. 따라서 각 *segment*의 시작점과 끝점을 관리하고 있어야합니다.
- 또한, 매번 풍선을 터트릴 때 마다 점수를 계산해야하기 때문에 각 *segment*에 포함된 풍선 점수의 합과 모든 *segment*의 점수의 합을 관리하고 있어야합니다.
- 직접 해보시면 아시겠지만, 구현이 굉장히 까다롭습니다.

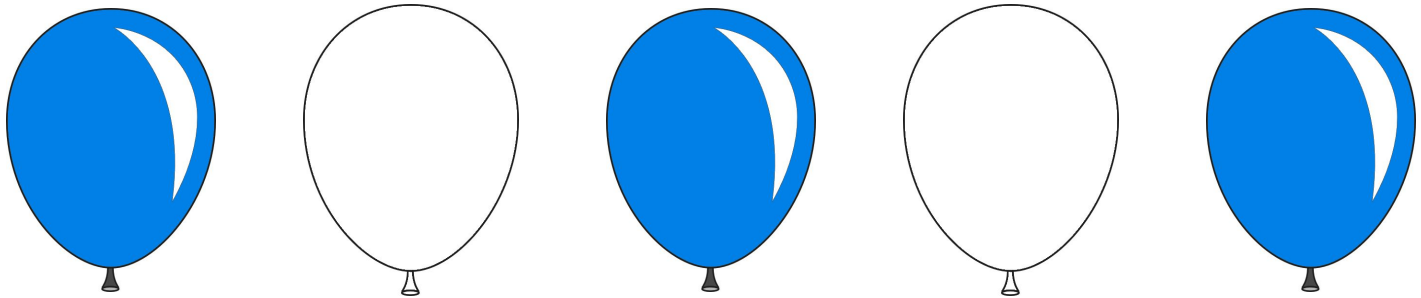
E. 터트려라 풍선

- 모든 풍선이 있는 상태에서 순서대로 하나씩 터트려 가는 것이 아니라
  모든 풍선이 터트려져 있는 상태에서 역순으로 하나씩 생성하면 쉽게 해결할 수 있습니다.
- 예제 1번으로 예시를 들어보겠습니다.
  아래는 풍선이 모두 터져있는 상태입니다.



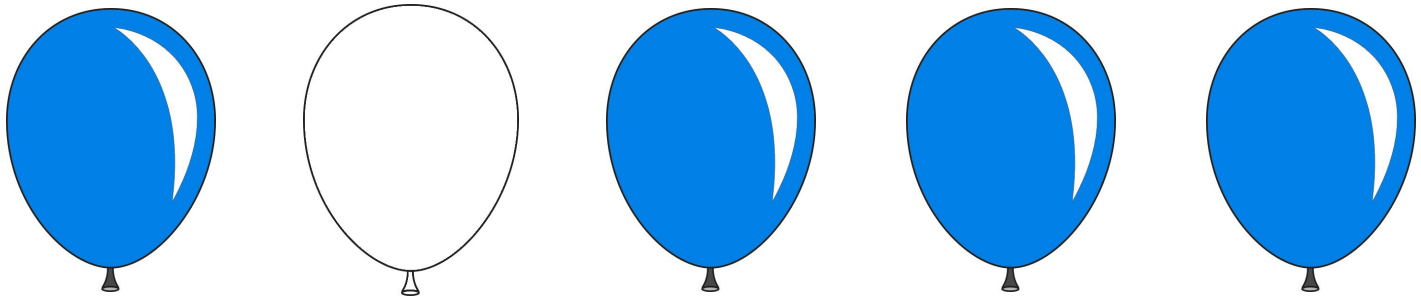
E. 터트려라 풍선

- 5번 풍선을 생성할 때 양 옆에 생성된 풍선이 존재하지 않으므로 *segment*는 (21)로 1개이므로 점수는 21점입니다.
- 3번 풍선을 생성할 때 양 옆에 생성된 풍선이 존재하지 않으므로 *segment*는 (8), (21)로 2개이므로 점수는 29점입니다.
- 1번 풍선을 생성하면 양 옆에 생성된 풍선이 존재하지 않으므로 *segment*는 (13), (8), (21)로 3개이므로 점수는 42점입니다.



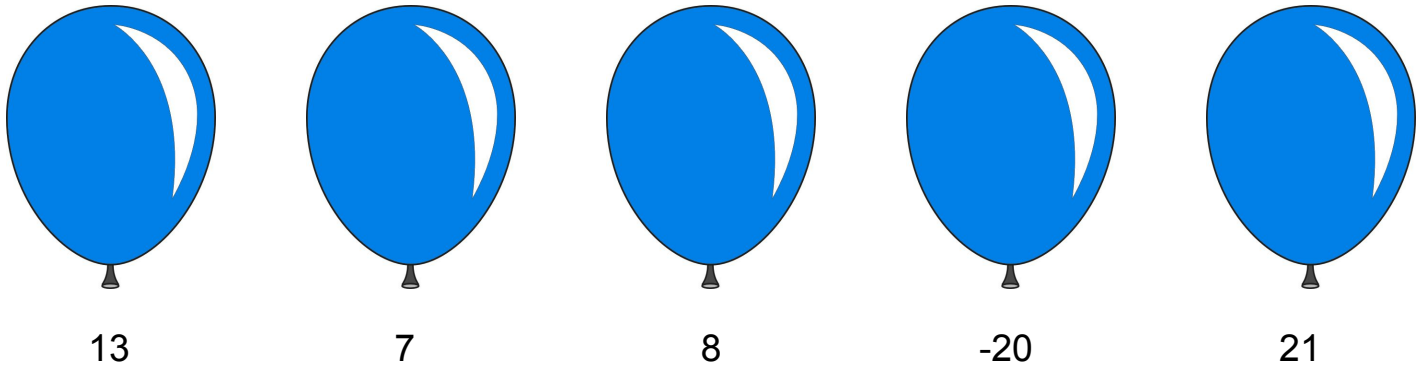
E. 터트려라 풍선

- 4번 풍선을 생성할 때 양 옆에 생성된 풍선이 존재하게 됩니다. 따라서 이들을 하나의 *segment*로 묶어주어야합니다. 이는 union-find 알고리즘을 사용하여 $O(1)$로 쉽게 구현할 수 있습니다. *segment*는 (13), (8, -20, 21)로 점수는 40점입니다.



E. 터트려라 풍선

- 마지막 5번 풍선을 생성할 때 역시 양 옆에 생성된 풍선이 존재합니다. 마찬가지로 이들을 하나의 *segment*로 묶어주어야합니다. (13), (8, -20, 21) *segment*와 새로 생성된 7이 하나의 *segment*로 묶여 (13, 7, 8, -20, 21) *segment*만 남게 됩니다.
  다만, **첫 다트를 던지기 전에는 점수가 계산되지 않는다고 했으므로**, 마지막 풍선 생성시 점수 계산은 생략합니다.



F. 첨탑 부수기

알고리즘 구분: 정수론, 오일러 피 함수(Euler's phi function)

의도한 난이도: Hard

- 제출 횟수 34번, 정답 0명 (정답률 0%)
- 출제자: `tykr0001`

F. 첨탑 부수기

- 먼저 꼭대기 층에서 마주치는 괴물의 강함을 수식으로 나타내면 아래와 같습니다.

$$Strength(N) = g(N)^{g(N-1)^{g(n-2)^{\cdots}}}$$

- 하지만 N 제한이 10^9이므로, 이 급수를 모두 계산하려고 하면 시간 제한에 걸리게 됩니다.
- 또한 `Overflow`를 방지하기 위해 필연적으로 계산 도중 `modulo` 연산을 적용해야 하는데, 이 과정 또한 문제가 됩니다.

F. 첨탑 부수기

- 예를 들어 아래의 수식을 계산한다고 해보겠습니다.

$$6^{8^3} \bmod 7$$

- 이를 단순히 계산하면 다음과 같습니다.

$$6^{8^3} \bmod 7 = 6^{512} \bmod 7 = 1$$

F. 첨탑 부수기

- 하지만 계산 중간에 모듈로 연산을 섞으면 다른 값이 나오게 됩니다.

$$6^{8^3} \bmod 7 = 6^{512} \bmod 7 = 1$$

$$6^{(8^3 \bmod 7)} \bmod 7 = 6^1 \bmod 7 = 6$$

- 따라서, 단순히 모듈로 연산을 적용하는 것은 올바른 풀이가 아닙니다.

F. 첨탑 부수기

- 이 문제를 해결하기 위해서는 오일러 피 함수(Euler's Phi function)에 대해 알아야 합니다.
- **오일러 피 함수 $\phi(n)$는 1부터 n 까지의 정수 중 n과 서로소인 수의 개수**를 뜻합니다.
- 예시를 몇 개 들면 아래와 같습니다.

$$\phi(5) = 4$$

$$\phi(9) = 6$$

$$\phi(13) = 12$$

F. 첨탑 부수기

- 오일러 피 함수를 이용하여 이 문제를 해결할 때 필요한 보조정리는 아래와 같습니다.

보조정리 1. $\gcd(a, m) = 1$ 인 두 자연수 a, m 에 대하여 $a^{\phi(m)} \equiv 1 \bmod m$ (오일러 정리)

보조정리 2. 임의의 자연수 a, m에 대하여 $a^{\phi(m)} \equiv a^{2\,\phi(m)} \bmod m$

보조정리 3. 3 이상의 자연수 m에 대하여 $\phi(m)$는 짝수이다.

보조정리 4. 3 이상의 짝수인 m에 대하여, $\phi(m) \le m/2$

F. 첨탑 부수기

- 보조정리 1 증명
  - m 이하의 자연수 중 m과 서로소인 수만 모아놓은 집합을 S라고 하면

$$S = \{ b_1, \cdots, b_{\phi(m)} \}$$

  - S의 각 원소에 m과 서로소인 정수 a를 곱한 집합을 aS라고 하면

$$aS = \{ ab_1, \cdots, ab_{\phi(m)} \}$$

F. 첨탑 부수기

- 보조정리 1 증명
  - 집합 aS의 원소들 또한 m과 서로소

$$aS = \{ab_1, \cdots, ab_{\phi(m)}\}$$

  - 이때, 집합 aS의 원소들을 m으로 나눈 나머지는 서로 같지 않다.
    - $ab_i \equiv ab_j \bmod m$ 인 서로 다른 정수 i, j 가 있다고 가정해보면 $a(b_i - b_j)$ 는 m의 배수이다. (단, $b_i > b_j$)
    - 이때 a와 m이 서로소이므로 $(b_i - b_j)$가 m의 배수가 되어야 한다.
    - b_i와 b_j 모두 1이상 m이하인 수들이므로 $b_i - b_j \le m - 1$ 인데, 이 범위에서 m의 배수는 0이므로 $b_i = b_j$

F. 첨탑 부수기

- 보조정리 1 증명
  - 정수를 그 정수와 서로소인 수 m으로 나누었을 때, 나머지도 마찬가지로 m과 서로소이다.
  - 그러므로 집합 aS의 원소들을 m으로 나눈 나머지의 집합은 S와 같음.
  - 따라서 S의 모든 원소의 곱과 aS의 모든 원소의 곱은 m으로 나눈 나머지가 같다.

$$b_1 b_2 \cdots b_{\phi(m)} \equiv a^{\phi(m)} b_1 b_2 \cdots b_{\phi(m)} \bmod m$$

$$a^{\phi(m)} \equiv 1 \bmod m$$

F. 첨탑 부수기

- 보조정리 2 증명
  - 먼저 a와 m을 각각 소인수분해하여 아래와 같이 나타낸다.

$$m = \prod_{i=1}^{k} p_i^{e_i}, a = \prod_{j=1}^{t} q_j^{f_j}$$

  - 그리고 a와 m의 공통 소인수를 아래와 같이 나타낸다.

$$\{p_1, p_2, \ldots, p_u\} = \{q_1, q_2, \ldots, q_u\} = \{p_1, p_1, \ldots, p_k\} \cap \{q_1, q_2, \ldots, q_t\}$$

  - 보조정리 2의 증명은 모든 j에 대해 $q_j^{\phi(m)} \equiv q_j^{2\phi(m)} \bmod m$ 임을 증명하는 것과 동치이다.

F. 첨탑 부수기

- 보조정리 2 증명
  - q_j가 a와 m의 공약수인 경우
    - 보조정리 1(오일러 정리)에 의해 임의의 $i\,(\neq j)$ 에 대해 아래의 식이 자명하다.

$$q_j^{\phi(p_i^{e_i})} \equiv q_j^{\phi(m)} \equiv q_j^{2\phi(m)} \equiv 1 \mod p_i^{e_i}$$

    - $\phi(m) \geq \phi(p_j^{e_j}) \geq e_j$이므로 $q_j^{\phi(m)} \equiv q_j^{2\phi(m)} \equiv 0 \mod p_j^{e_j}$ 다.
    - $q_j^{\phi(m)}$와 $q_j^{2\phi(m)}$는 $(p_1^{e_1}, p_2^{e_2}, \ldots, p_k^{e_k})$로 나눈 나머지가 1 혹은 0으로 같으므로 중국인의 나머지 정리에 의해 m으로 나눈 나머지도 같다.

$$m = \prod_{i=1}^{k} p_i^{e_i}, a = \prod_{j=1}^{t} q_j^{f_j}$$

F. 첨탑 부수기

- 보조정리 3 증명
  - $\gcd(m, k) = 1$ 이면 $\gcd(m, m - k) = 1$ 도 성립한다. (이때, $m > k$)
  - 여기서 m과 $m - k$를 한 쌍으로 묶을 수 있다.
  - 묶이지 않는 경우는 k가 $m/2$인 경우 밖에 없으나, 이때 $\gcd(m, m/2) = m/2$이므로 고려하지 않아도 됨.
  - 따라서 m 이하의 자연수 중 m과 서로소인 수의 개수는 짝수다.

F. 첨탑 부수기

- 보조정리 4 증명
  - 2이상 m 이하의 모든 짝수들은 m과 공약수로 2를 가지므로 서로소가 아니다.
  - 따라서 서로소는 많아봐야 $n - n/2 = n/2$개이다.

F. 첨탑 부수기

- 아래의 증명된 보조정리를 이용해 문제를 **압축**할 수 있습니다.

보조정리 1. $\gcd(a, m) = 1$ 인 두 자연수 a, m 에 대하여 $a^{\phi(m)} \equiv 1 \bmod m$ (오일러 정리)

보조정리 2. 임의의 자연수 a, m에 대하여 $a^{\phi(m)} \equiv a^{2\,\phi(m)} \bmod m$

보조정리 3. 3 이상의 자연수 m에 대하여 $\phi(m)$는 짝수

보조정리 4. 3 이상의 짝수인 m에 대하여, $\phi(m) \le m/2$

F. 첨탑 부수기

- 보조정리 2를 이용해 문제에서 주어진 점화식을 아래와 같이 바꿀 수 있습니다.

$$Strength(N) = g(N)^{Strength(N-1)} \bmod m$$
$$= g(N)^{(Strength(N-1) \bmod \phi(m)) + \phi(m)} \bmod m$$

- 이때, $Strength(N-1)$이 $\phi(m)$보다 이상일 때만 이 식이 성립하는데, $k(\neq 1)$층에서의 $g(k)$는 $Seed$에 따라 최소 10 이상의 값을 가지므로 이므로 3층 이상에서의 $Strength$는 최소 10^{10} 이상입니다.

F. 첨탑 부수기

- 재귀적으로 연산을 수행하면 modulus는 $m \to \phi(m) \to \phi(\phi(m)) \to \cdots \to 1$ 이 됩니다.
- **모든 자연수에 대해 modulo 1은 0 이므로 그 이후의 연산은 생략**가능하고, 그 전까지의 연산이 충분히 짧은 시간 내에 끝난다는 것은 보조정리 3, 4를 통해 증명가능합니다.
- 보조정리 3에 의해 $\phi(m), \phi(\phi(m)), \phi(\phi(\phi(m)))$ 등은 모두 짝수입니다.
- 보조정리 4에 의해 재귀를 한번 거칠 때 마다 modulus는 절반 이상 감소합니다.
- 따라서 **$\log m$에 비례한 횟수 내에 연산이 가능**합니다.

F. 첨탑 부수기

- 각 재귀마다 오일러 피 함수의 계산이 필요합니다.
- 에라토스테네스의 체와 유사한 과정으로 Naive하게 계산하면 $O(\sqrt{m})$ 이며, 이 문제를 해결하기에 충분히 짧은 시간입니다.
- 전체 시간복잡도는 $O(\sqrt{m}\log m)$입니다.

- 참조한 글: https://lego0901.tistory.com/14, https://rkm0959.tistory.com/181
- TMI: 본 문제는 “Slay the Spire” 라는 게임에서 모티브를 얻어 출제했습니다.

G. 고장난 통신탑

알고리즘 구분: 정수론, 에라토스테네스의 체, 많은 조건 분기

의도한 난이도: Medium

- 제출 횟수 73번, 정답 0명 (정답률 0%)
- 출제자: kanght1219

G. 고장난 통신탑

- 통신탑과의 관계들을 그래프로 생각해본다면, 고장난 통신탑의 세 가지 특징과 우리가 구해야 할 경로의 조건을 다음과 같이 그래프로 표현할 수 있습니다.

  - 이 그래프는 1부터 무한하게 번호가 붙어있는 정점들로 이루어져 있다.

  - 어떤 정점 번호의 쌍 (a, b)이 있을 때 두 수의 쌍이 약수 관계일때만 그 사이에 간선이 양방향으로 a에서 b로 존재한다. 약수 관계란 두 수의 쌍 (a, b) 중 큰 수를 작은 수로 나눴을 때 나머지가 0인 관계다.

  - 간선들 중 1번 정점에 연결된 간선 중 짝수 번호와 연결된 간선의 가중치는 모두 2이며, 그 외 나머지 모든 간선들은 모두 가중치가 1이다.

  - 우리가 구해야 할 경로는 a번 정점에서 b번 정점까지 지나가는 간선들의 가중치의 합이 최소이고, 정점 번호들의 합이 최소인 경로이다.

G. 고장난 통신탑

1. 주어지는 두 수 (a, b)가 약수 관계일 때

1) $a \neq 1, b \neq 1$일 때

- a번 정점에서 b번 정점으로 1 대 1 통신이 가능하므로 가중치의 합이 1로 최소 로 유일합니다. 그러므로 “a - b”가 정답입니다.

2) 주어지는 두 수 중 둘 중 하나가 1일 때

- 나머지 하나가 홀수일 때는 가중치가 1이므로 1)과 같습니다.

- 나머지 하나가 짝수일 때는 가중치가 2이지만, 다른 경로는 약수 관계로 이어진 간선을 타지 않으면 최소 2개 이상의 간선을 거쳐야 하므로 가중치의 최소는 만족시킬 수 있어도, 정점 번호들의 합에서는 “a - b”보다 최소일 수 없습니다. 그러므로 “a - b”가 정답입니다.

G. 고장난 통신탑

2. a와 b가 둘 다 홀수일 때

- a와 b가 홀수일 때는 둘 다 1번 정점을 지나가야 가중치의 합과 식별 번호의 합의 최소를 만족할 수 있으므로 “a - 1 - b”가 정답입니다.

3. a와 b가 둘 다 짝수일 때

- a와 b가 짝수일 때는 1번 정점을 지나가면 가중치의 합의 최소를 만족할 수 없습니다.
- 대신 2번 정점을 지나가면 가중치의 합의 최소를 만족하면서 정점 번호들의 합의 최소를 만족합니다. 정답은 “a - 2 - b”가 됩니다.

G. 고장난 통신탑

4. 어떤 한 수가 홀수이고, 나머지 한 수가 짝수일때

- 두 수를 a, b로 잡을 때, 경로 “a - 1 - b”는 가중치의 합이 3이 되므로 최소를 만족하지 못합니다..

- 그 대신, g를 a와 b의 최대공약수라고 할 때, 1을 제외한 g의 최소 약수를 선택하면 됩니다. 1을 제외한 g의 최소 약수를 k라고 할 때, 경로는 “a - k - b”입니다.

- 만약 g가 1이면, a와 b의 최소공배수를 선택하면 됩니다. 이때 $lcm(a, b) = a * b / g$이고, g = 1이므로 경로 “a - (a * b) - b”가 정답입니다.

G. 고장난 통신탑

- 하지만, 우리는 최소 약수를 구하는 방법에 대해 다루지 않았습니다.

- 숫자 범위를 보면, 1 ≦ a, b ≦ 2 * 10 ^ 7이므로 $gcd(a, b) < 6666667$입니다.

- 이때, 6666667을 넘지 않는 수 중 가장 큰 소수는 6666617.

- 일반적으로 임의의 정수 n의 약수 찾기 방법을 사용하면 O(sqrt(n))의 시간 복잡도가 나오는데 sqrt(6666617) ≈ 2500이므로 최악의 경우에 각 테스트 케이스 당 대략 O(2500)의 시간 복잡도를 가지고, 모든 테스트 케이스를 처리하면 최악의 경우 대략 O(100000 * 2500)의 시간 복잡도를 가지게 되므로 1초로는 부족합니다.

G. 고장난 통신탑

- 일단 임의의 수를 소인수 분해하면

$$n = p_1^{k_1} \times p_2^{k_2} \times \cdots \times p_n^{k_n}$$

의 형태가 됩니다.(단, p_a는 소수)

- 일단 1)의 조건으로 소인수 분해를 정리했다고 했을 때 2)에 의해 임의의 수 n의 1을 제외한 최소 약수는 소수임이 증명됩니다.

1) $p_1 \leq p_2 \leq p_3 \leq \cdots \leq p_n$

2) $p_1 \leq p_1^i (1 \leq i \leq k_1)$

G. 고장난 통신탑

- 이때 에라토스테네스의 체의 방법을 응용해봅시다.

- 에라토스테네스의 체는 특정 범위 내의 소수를 구하기 위해 어떤 수 n에 도달하기 전 걸러지지 않았다면 소수이고, n의 배수를 거르는 방식으로 이루어집니다.

- 일단 크기가 6666667인 배열 X를 잡고, 배열을 $X[i] = i$로 초기화해줍시다.

G. 고장난 통신탑

- 2부터 탐색한다고 할 때, X[2] = 2이므로 2는 소수입니다. 그러므로 2의 배수를 탐색해줍시다. X[2] = 2, X[4] = 4, ... , X[2n] = 2n이므로 X[4], X[6], ..., X[2n]에 2를 넣어줍시다.

- X[3] = 3이므로 3은 소수입니다. 3의 배수를 탐색해봅시다. X[6] = 2이므로 그대로 놔둡니다. X[9] = 9이므로 3을 대입해줍니다. X[3n] = 3n인 X[3n]에 3을 대입합시다.

- X[4] = 2이므로 4는 소수가 아닙니다. 넘어갑시다.

- 위의 과정을 범위 끝(= 6666666)까지 반복해주면 X[i]에는 i의 1을 제외한 최소 약수가 들어있게 되고, g가 들어올 때마다 X[g]를 출력해주면 됩니다.

G. 고장난 통신탑

- 에라토스테네스의 체는 코드 돌리기 전에 한 번만 전처리해주면 되기 때문에 시간복잡도는 n을 6666667이라고 할 때, 에라토스테네스의 체를 처음 전처리하는 시간 O(n×log(log(n)))에 전체 테스트 케이스의 최대공약수를 구하는 시간 O(∑log(max(a□, b□))) 을 더해서 O(n×log(log(n)) + ∑log(max(a□, b□)))가 되어 C++, Pypy 기준 충분히 1초 안에 돌아갑니다.

- 이 풀이의 공간복잡도는 배열 X의 크기인 O(6666667)로 고정됩니다.

H. Next Level

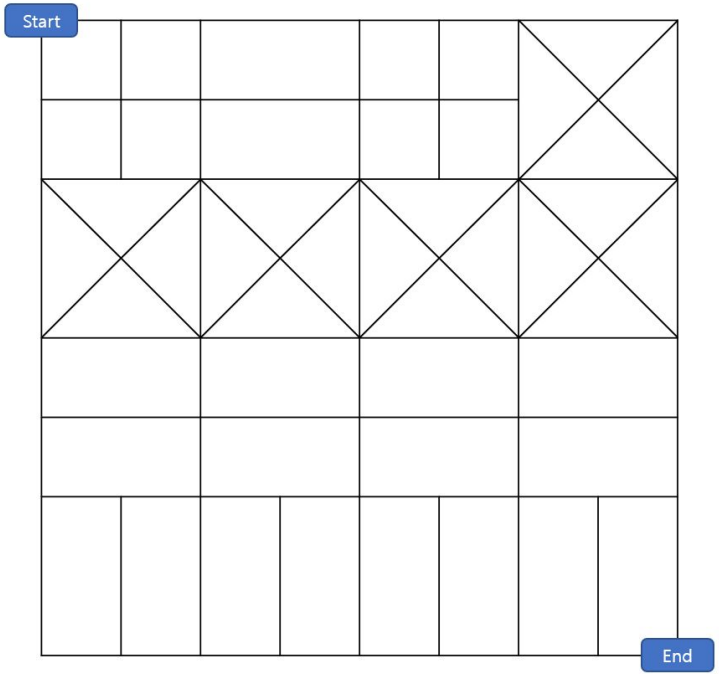
알고리즘 구분: 그래프, DFS, 탐색

의도한 난이도: Medium

- 제출 횟수 3번, 정답 0명 (정답률 0%)
- 출제자: kmiiiaa

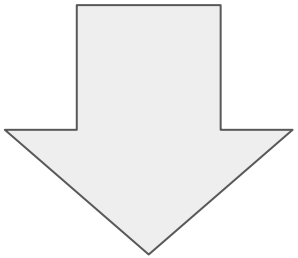
H. Next Level

- 주어진 지도 조각들을 바탕으로 지도를 만들고, 지도에서 목적지 KOSMO에 도달 가능한지 찾는 문제이다.



H. Next Level

- 길 찾는 것 자체는 기본적인 DFS에 약간의 조건을 추가하면 탐색을 진행할 수 있다.
- 이 문제의 핵심은 탐색을 진행할 그래프를 구현하는 것이다.



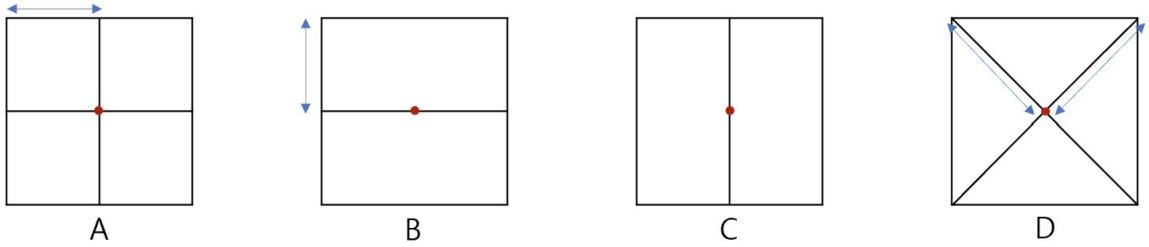
지도 조각의 공통된 규칙 찾기

H. Next Level

- 왜 공통된 부분을 찾아야 할까?
  - 근본적으로 객체지향 프로그래밍에서 객체를 추상화 하는 것과 동일
  - 4개의 독립적인 조각의 개별 구현하는 것이 아니라 하나의 공통된 부분을 구현하고 각 조각이 가진 약간의 차이를 추가 구현해 주는 방식으로 문제를 단순하게 재정의 가능

H. Next Level

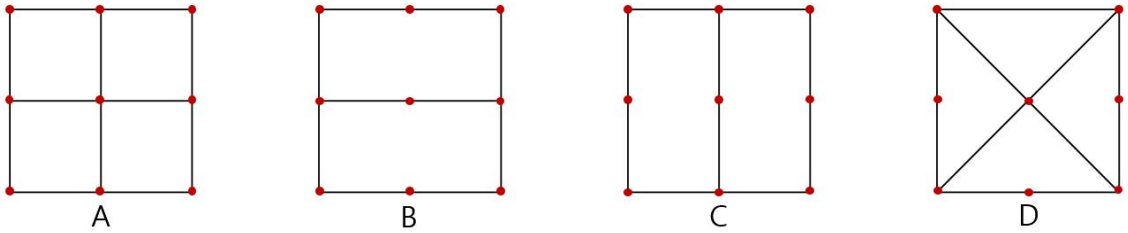
- 조각들의 공통점 찾기



1. 모든 조각은 정사각형을 기반으로 하고 있다.
2. 정사각형 내부의 선분은 모두 정사각형과 꼭짓점에서 만나거나 모서리 중앙에서 만난다.
3. 모든 조각들은 정사각형 중앙을 지나는 선분이 있다.

H. Next Level

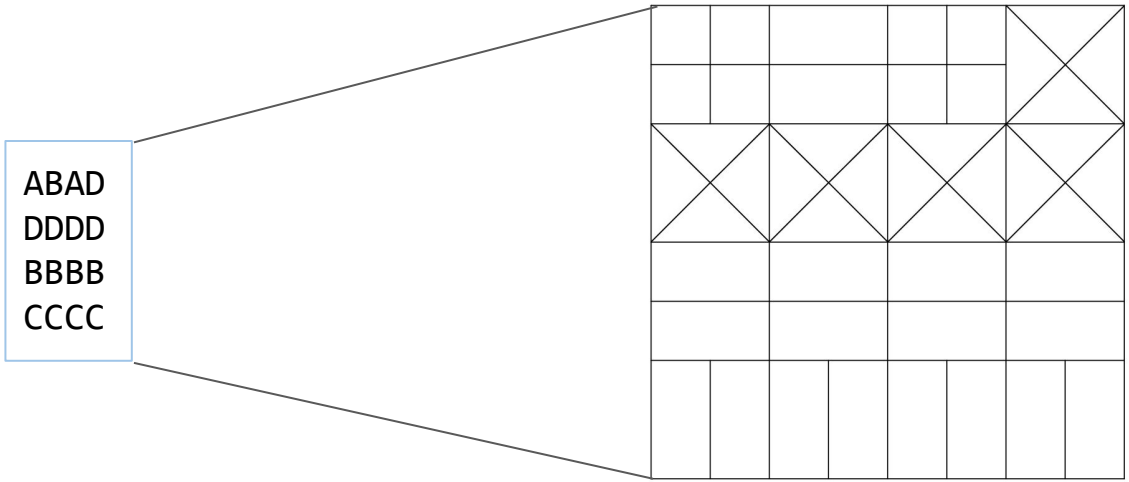
- 모든 조각은 9개의 노드를 가진 하나의 그래프이다.



-> 어떤 조각이 주어지든 9개의 노드를 가진 하나의 그래프로 정의하고, 각 노드간의 연결관계만 신경쓰면 됨.

H. Next Level

- 입력으로 주어진 N x N 조각들은 N x N개의 그래프를 간의 연결 관계이다
  - 어떤 조각이 주어 지든 전체 지도는 각 조각들은 모두 동일한 방식으로 연결되어 구성되어 있음



감사합니다