서강대학교
SOGANG UNIVERSITY

# Sogang Programming Contest 2019

## Official Solutions

Sogang University

STARTLINK

# 대회 운영진

## 운영 총괄

- 박수현 / 서강대학교 컴퓨터공학과 학부과정 @shiftpsh

## 출제

- 박건 / 서강대학교 컴퓨터공학과 학부과정 @lvalue
- 박수현 / 서강대학교 컴퓨터공학과 학부과정 @shiftpsh
- 손정연 / 서강대학교 컴퓨터공학과 학부과정 @sjy366
- 윤기영 / 서강대학교 컴퓨터공학과 학부과정 @craclog
- 이준석 / 서강대학교 컴퓨터공학과 학부과정 @semteo04
- 정진욱 / 서강대학교 컴퓨터공학과 학부과정 @ZZangZZang

## 검수

- 김영재 / 서강대학교 컴퓨터공학과 교수
- 김영현 / 서울대학교 컴퓨터공학부 학부과정 @kipa00
- 심유근 / 서울대학교 컴퓨터공학부 학부과정 @cozyyg
- 최재민 / KAIST 전산학부 학부과정 @jh05013

## 온사이트 스탭

- 김승채 / 서강대학교 전자공학과 학부과정
- 박한나 / 서강대학교 컴퓨터공학과 학부과정

## Master A. 1998년생인 내가 태국에서는 2541년생?!

**정답자** 15명 (100%)

**첫 정답** 김준서, 2분

**출제자** @shiftpsh (박수현)

문제 제목과 예제를 통해 불기 연도와 서기 연도의 차이를 계산할 수 있습니다. $2541 - 1998 = 543$이므로, 불기 $y$년은 서기 $y - 543$년입니다. 이를 계산해 출력하는 프로그램을 작성하면 정답입니다.

## Master B. 도깨비불

**정답자** 5명 (33%)

**첫 정답** 김동민, 53분

**출제자** @ZZangZZang (정진욱)

두 글자 사이에서 도깨비불 현상이 일어날 수 있는 조건을 잘 생각해 보면, 다음 글자의 초성으로 들어가야 할 글자가 임시적으로 이전 글자의 종성으로 들어가는 경우 도깨비불 현상이 일어남을 알 수 있습니다. 그렇게 되는 경우들은 아래와 같습니다.

1. 자음 → 모음 → (자음) → 모음이 순서대로 오는데 괄호 안의 자음이 ㅃ, ㅉ, ㄸ 중 하나가 아닌 경우, 예를 들어 '삿' → '사자'
2. 자음 → 모음 → 모음 → (자음) → 모음이 순서대로 오는데 괄호 안의 자음이 ㅃ, ㅉ, ㄸ 중 하나가 아닌 경우, 예를 들어 '확' → '화가'
3. 자음 → 모음 → (자음 → 자음) → 모음이 순서대로 오는데 괄호 안의 자음이 겹받침을 이룰 경우, 예를 들어 '낛' → '난조'
4. 자음 → 모음 → 모음 → (자음 → 자음) → 모음이 순서대로 오는데 괄호 안의 자음이 겹받침을 이룰 경우, 예를 들어 '괅' → '관저'

1번과 2번 케이스에서 ㅃ, ㅉ, ㄸ는 종성에는 등장할 수 없기 때문에(예를 들어 '가' → '가ㅉ' → '가짜'), 도깨비불 현상을 일으킬 수 없어서 제외해야 합니다.

또한 조금 더 생각해 보면 1번과 2번, 그리고 3번과 4번을 아래와 같이 각각 한 케이스로 묶을 수 있습니다.

a. 모음 → (자음) → 모음이 순서대로 오는데 괄호 안의 자음이 ㅃ, ㅉ, ㄸ 중 하나가 아닌 경우
b. 모음 → (자음 → 자음) → 모음이 순서대로 오는데 괄호 안의 자음이 겹받침을 이룰 경우

이런 경우를 모두 세 주면 정답입니다. 띄어쓰기가 들어옴에 주의합시다.

# Master C. 블랙 프라이데이

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @sjy366 (손정연)

배열 $A$에서 세 수를 적절히 골라 합을 $C$가 되게 할 수 있는지를 판단해야 하는 문제입니다. 식으로 표현하면 $A_i + A_j + A_k = C$가 되는 $i \neq j \neq k \neq i$가 존재하는지 찾으면 됩니다. 해결하는 방법은 여러 가지가 있으나, 몇 가지만 소개하도록 하겠습니다.

## 방법 1

배열을 정렬한 후, $i$와 $j$를 $i < j$가 되도록 for-루프를 이용해 골라서 고정시킵니다. 이렇게 하면 $A_i = C$가 되는 경우, $A_i + A_j = C$가 되는 경우는 쉽게 찾을 수 있습니다. 이제 $j < k$에 대해 $A_i + A_j + A_k = C$가 되는 $A_k$가 있는지 찾으면 됩니다.

$A_i + A_j + A_k = C$라면 $A_k = C - (A_i + A_j)$이므로, 배열이 정렬되어 있음을 이용해 $A_k$는 이분 탐색으로 $\mathcal{O}(\log N)$만에 찾을 수 있습니다. 모든 $i$와 $j$의 쌍을 고르는 시간 복잡도가 $\mathcal{O}(N^2)$이므로 총 시간 복잡도는 $\mathcal{O}(N^2 \log N)$이며, 이 정도로도 문제를 해결하기엔 충분합니다.

이진 탐색 트리 등의 자료구조를 이용해도 위와 같은 시간 복잡도로 해결 가능합니다.

## 방법 2

해시 집합을 이용해 문제를 해결할 수도 있습니다. 우선 모든 $A_i$의 값들을 해시 집합에 전부 집어넣습니다.

이제 $i$와 $j$를 $i < j$가 되도록 for-루프를 이용해 골라서 고정시킵니다. 마찬가지로 $A_i = C$가 되는 경우, $A_i + A_j = C$가 되는 경우는 쉽게 찾을 수 있습니다.

역시 $A_i + A_j + A_k = C$라면 $A_k = C - (A_i + A_j)$입니다. 문제에서 같은 무게의 상품이 존재하지 않는다고 했으므로, $C - (A_i + A_j)$가 $A_i$와 $A_j$ 중 어느 것과도 같지 않은데 해시 집합에 존재한다면, 이는 $A_k$가 될 수 있음을 의미합니다. 이 방법을 사용하면 $\mathcal{O}(N^2)$만에 문제를 해결 가능합니다.

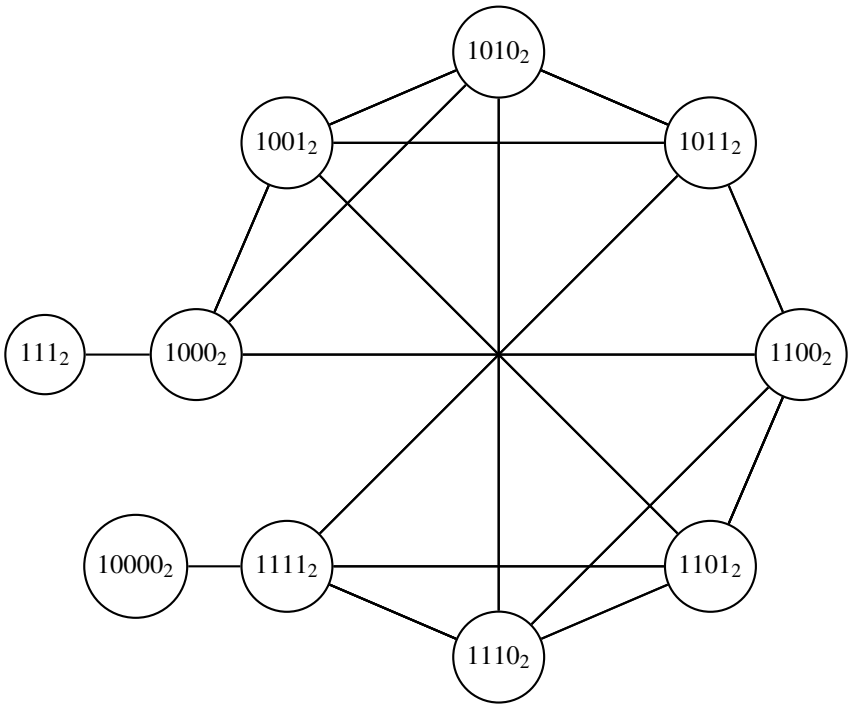
# Master D. 이진수 게임

**정답자** 1명 (7%)

**첫 정답** 이길원, 154분

**출제자** @sjy366 (손정연)

이진수들을 그래프의 노드로 생각하고, 각각의 이진수들에 대해 그 이진수에 '가능한 동작'들을 취해 만들어질 수 있는 이진수들을 인접한 노드로 갖도록 합시다.




예를 들어, $1010_2$에 인접한 노드의 목록은

1. 한 자리 숫자를 바꾼 $1110_2$, $1000_2$, $1011_2$,
2. 1을 더한 $1011_2$,
3. 1을 뺀 $1001_2$

가 됩니다. 그러면 이 문제는 너비 우선 탐색(BFS)를 이용해, 시작 노드와 목표 노드 간의 최단 거리를 구하는 방식으로 해결할 수 있습니다.

이진수의 최대 길이가 10이므로, 노드 개수는 $2^{10} = 1024$개입니다. 따라서 BFS를 사용하면 시간 안에 문제를 해결할 수 있습니다.

## Master E. 그르다 김가놈

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @craclog (윤기영)

길이 $x$의 김밥으로 만들 수 있는 김밥조각의 개수 $m(x)$는 다음과 같이 정의됩니다.

$$m(x) = \begin{cases} 0 & \text{if } x \le K \\ \lfloor (x-K)/P \rfloor & \text{if } x < 2K \\ \lfloor (x-2K)/P \rfloor & \text{otherwise} \end{cases}$$

$i$번째 김밥의 길이를 $A_i$라고 하면, 아래의 식을 만족하는 $P$의 최댓값 $P_M$을 구하면 됩니다.

$$\sum_{1 \le i \le N} m(A_i) \ge M$$

여기서 $m(x)$는 $P$가 감소할수록 증가하고, $P$가 증가할수록 감소하는 함수임에 주목한다면 $P \le P_M$인 모든 $P$에 대해 위 식을 만족할 수 있음을 알 수 있습니다. 따라서 $P$의 최댓값은 이분 탐색으로 찾을 수 있습니다.

$N, K, M$은 모두 $10^9$ 이하의 정수이지만 중간 계산 값 $\sum_{1 \le i \le N} m(A_i)$는 32비트 정수 범위를 초과할 수 있음에 주의합시다.

## Master F. 7-세그먼트 디스플레이

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @shiftpsh (박수현)

이 문제는 다이나믹 프로그래밍으로 해결할 수 있습니다.

$d(n, r)$을 '$n$개의 7-세그먼트 디스플레이를 사용해 만들 수 있는 수 중 $m$으로 나눴을 때의 나머지가 $r$인 가장 큰 수'로 정의합시다. 그러면

1. $d(n-1, r)$ 뒤에 0–9를 하나 붙인다고 생각할 경우

   $$d(n, (10r+0) \bmod m), d(n, (10r+1) \bmod m), \cdots, d(n, (10r+9) \bmod m)$$

   를 업데이트할 수 있고,
2. $d(n-1, r)$ 뒤에 11을 하나 붙인다고 생각할 경우

   $$d(n, (100r+11) \bmod m)$$

   를 업데이트할 수 있습니다.

이렇게 아래에서 위로 최댓값을 갱신해 나가다 보면, 정답은 $n$개의 7-세그먼트 디스플레이를 사용해 만들 수 있는 수 중 $m$으로 나눴을 때의 나머지가 0인 가장 큰 수, 즉 $d(n, 0)$이 됩니다. 이를 계산하는 시간 복잡도는 $\mathcal{O}(nm)$입니다.

테스트 케이스가 25개까지 들어올 수 있으므로, 배열 $d$를 초기화하는 과정에서 많은 시간이 걸릴 수 있음에 주의합시다. C/C++에서는 `memset` 등의 함수 사용을 권장합니다.

## Master G. 단어 암기

**정답자** 1명 (7%)

**첫 정답** 윤성호, 179분

**출제자** @semteo04 (이준석)

현재 알고 있는 알파벳의 목록이 주어질 때 완전히 아는 단어의 수를 쿼리하는 것은 (단어의 길이) × (알파벳의 종류) × (단어의 개수) 만큼의 시간이 걸리고, 이렇게 풀면 시간 안에 문제를 해결할 수 없습니다.

알파벳 개수가 26개임을 고려하면, 비트마스킹을 이용할 수 있습니다. 단어에 $a$가 포함되어 있다면 0번째 비트가 켜져 있고, $b$가 포함되어 있다면 1번째 비트가 켜져 있고, $\cdots$, $z$가 포함되어 있다면 25번째 비트가 켜져 있는 비트마스크를 고려합시다. 그러면 모든 단어들을 비트마스크로 만들 수 있습니다. 현재 알고 있는 단어 목록도 비슷하게 관리할 수 있습니다.

어떤 단어를 '완전히 알고' 있으려면

(어떤 단어에 등장한 알파벳의 집합) $\subset$ (알고 있는 알파벳의 집합)
$\Rightarrow$ (어떤 단어에 등장한 알파벳의 집합) $\cap$ (알고 있는 알파벳의 집합) = (어떤 단어에 등장한 알파벳의 집합)

이 되면 됩니다. 따라서 현재 알고 있는 알파벳들의 비트마스크를 $A$, $i$번째 단어의 비트마스크를 $W_i$라고 하면,

$$W_i \wedge A = W_i$$

이 성립할 경우 $i$번째 단어를 완전히 알고 있는 상태임을 의미하게 됩니다. $\wedge$는 bitwise AND 연산자입니다.

따라서 $\mathcal{O}(NM)$만에 모든 쿼리를 처리할 수 있습니다.

## Master H. 색깔 하노이 탑

**정답자** 1명 (7%)

**첫 정답** 이지훈, 105분

**출제자** @semteo04 (이준석)

원판을 최소로 이동시킬 때 필요한 이동 횟수는 총 $(2^{N+2}-5)$ 회입니다.

1. 먼저 **색깔에 관계없이** $N$번째 빨간 원판까지를 다른 기둥으로 옮기는 경우를 생각합시다. 이런 경우의 최소 횟수를 $a_N$이라고 한다면, 최종 정답은 $(2a_N+1)$회가 됩니다.
   - $N$번째 검은 원판을 옮기기 위해서는 $N$번째 빨간 원판까지를 다른 기둥으로 옮겨야 하기 때문에, $(2a_N+1)$번보다 더 적은 수로 색깔 하노이 탑을 해결할 수는 없습니다. 만일 $(N-1)$번째 검은 원판까지만을 옮겼다면 $N$번째 빨간 원판이 어디에 있든 최종 상태에는 $N$번째 원판의 색깔 순서가 뒤집혀 있게 됩니다.

- 색깔에 관계없이 $N$번째 빨간 원판까지를 $a_N$번의 횟수로 옮기는 이동 $\tau(i,o,a)$가 있다고 가정한다면, $\tau(i,o,a)$가 처음 실행될 때 어떤 크기의 원판을 서로 바꾸어 넣어 놓든지 다시 실행할 경우 원래대로 되돌아오게 됩니다. 따라서 $\tau(i,a,o)$를 시행하고, $N$번째 검은 원판을 옮긴 뒤 다시 $\tau(i,o,a)$를 실행하면 됩니다.

2. 마찬가지로 색깔에 관계없이 $(N-1)$번째 검은 원판까지를 다른 기둥으로 옮기는 최소 횟수를 $b_N$이라고 한다면, 1과 비슷한 방법으로 $a_N = 2b_N + 1$임을 보일 수 있습니다. 따라서 최종 정답은 $(2(2b_N+1)+1) = (4b_N+3)$회가 됩니다.

3. 마지막으로 하노이 탑의 최소성에 의해 $b_N = 2\cdot\left(2^{N-1}-1\right)$이 성립합니다. 따라서 정답은 $4\cdot 2\cdot\left(2^{N-1}-1\right)+3 = 2^{N+2}-5$입니다.

최소 이동 횟수를 $10^9+7$로 나눈 나머지를 출력해야 함에 유의하면서 계산합시다.

## Champion A. solved.ac

**정답자** 27명 (84%)

**첫 정답** 이기현, 7분

**출제자** @shiftpsh (박수현)

간단한 문제입니다. 입력받은 수 $n$개를 정렬해서, $\text{round}(n \times 0.15)$개만큼의 원소를 앞뒤에서 잘라내고 평균을 구하면 됩니다. 시간 복잡도는 $\mathcal{O}(n \log n)$입니다.

들어오는 수들이 1 이상 30 이하라는 점을 고려한다면 인덱스 소트를 이용해 $\mathcal{O}(n)$만에 해결할 수도 있습니다.

## Champion B. 마인크래프트

**정답자** 10명 (31%)

**첫 정답** 임지환, 23분

**출제자** @lvalue (박건)

높이 $x$를 목표로 땅을 고른다고 생각해 봅시다. 일단 높이 $x$보다 높은 곳에 있는 흙을 전부 제거해 인벤토리에 넣고, 높이가 $x$가 되지 않는 땅에 인벤토리에 있는 흙을 놓는다고 생각할 수 있습니다. 이를 식으로 쓰면,

- $o(x)$ : 높이 $x$보다 높은 곳에 있는 흙의 수
- $u(x)$ : 높이 $x$보다 낮은 곳에 있는 흙의 수
- $f(x)$ : 높이 $x$이 되도록 땅을 고르는 데 걸리는 시간

라고 할 때,

$$f(x) = \begin{cases} 2[o(x)+B]+u(x) & \text{if } o(x)+B < u(x) \\ \infty & \text{otherwise} \end{cases}$$

가 됩니다.

$o(x)$와 $u(x)$를 구하는 시간은 각각 $\mathcal{O}(NM)$인데, 땅의 높이가 256블록을 초과할 수 없다는 조건 때문에 $f(0)$부터 $f(256)$까지를 전부 계산해 그 중 가장 작은 것을 출력해도 무방합니다.

## Champion C. 카드 놓기

**정답자** 26명 (81%)

**첫 정답** 이기현, 19분

**출제자** @semteo04 (이준석)

카드를 놓는 동작을 역순으로 한다고 생각합니다. 원래 카드들의 순서 $B$가 있었다면, $B$를 맨 왼쪽 끝과 맨 오른쪽 끝에서 차례차례 복구할 수 있습니다. doubly-linked list 또는 이와 비슷한 방법을 이용해 배열을 복구 가능합니다.

초기 상태에 유일한 노드 $S = E$가 있고, 값 1이 저장되어 있는 doubly-linked list를 생각합시다.

1. 만약 $A_i = 1$이라면, 노드 $S$의 **이전** 위치에 값 $n - i + 1$이 저장된 노드를 삽입하는 것으로 생각할 수 있습니다.
2. 만약 $A_i = 2$라면, 노드 $S$의 **다음** 위치에 값 $n - i + 1$이 저장된 노드를 삽입하는 것으로 생각할 수 있습니다.
3. 만약 $A_i = 3$이라면, 노드 $E$의 **다음** 위치에 값 $n - i + 1$이 저장된 노드를 삽입하는 것으로 생각할 수 있습니다.

이 동작을 반복하면 $S$로 시작하는 doubly-linked list에는 원래 카드들의 순서가 저장되어 있게 됩니다. $A_1 = 2$인 경우에 노드 삽입 연산의 구현에 주의합시다.

## Champion D. 로봇 조립

**정답자** 11명 (34%)

**첫 정답** 이기현, 27분

**출제자** @craclog (윤기영)

Disjoint set 자료구조를 사용하면 되는 문제입니다.

Disjoint set $S$와 초기값이 전부 1인 배열 $V$를 만들어, 두 부품 $u$와 $v$가 같은 로봇에서 왔을 때 union$(u, v)$를 수행합니다. $u \neq v$일 때 트리 구조인 $S$에서 내부적으로 $u$의 부모를 $v$로 설정했다면, $V_v = V_u + V_v$가 되게 해 줍니다.

어떤 부품 $u$에 대해 robot$(i)$의 부품이 몇 개인지 알고 싶을 때는 $V_{\text{find}(u)}$를 계산하면 됩니다.

## Champion E. 분수

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @lvalue (박건)

$\frac{a}{b}$의 소수점 아래 $k$번째 자리 숫자는 아래와 같이 구할 수 있습니다.

$$\left\lfloor \frac{(a \cdot 10^{k-1} \bmod b) \cdot 10}{b} \right\rfloor$$

$10^{k-1} \bmod b$는 분할 정복을 사용한 거듭제곱(exponentiation by squaring)으로 $\mathcal{O}(\log k)$만에 계산할 수 있습니다. 오버플로에 주의하도록 합시다.

## Champion F. 대농부 김상혁

**정답자** 3명 (9%)

**첫 정답** 이기현, 163분

**출제자** @ZZangZZang (정진욱)

반경 $r$의 농토 안에 있는 어떤 농작물 $i$에서 얻는 수익의 함수 $f_i(r)$은 기울기가 $w_i$인 일차함수입니다. 또 원점에서 농작물 $i$까지의 거리를 $d_i$라고 하면 지문의 그림에서 볼 수 있듯 $c_i = r - d_i\,(r \geq d_i)$입니다.

따라서

$$f_i(r) = \begin{cases} w_i(r-d_i) = w_i r - w_i d_i & \text{if } r \geq c_i \\ 0 & \text{otherwise} \end{cases}$$

임을 알 수 있습니다. 그러므로 전체 이득은

$$\begin{aligned} g(r) &= \sum f_i(r) - Ar^2 \\ &= \sum_{d_i \leq r} f_i(r) - Ar^2 \\ &= \sum_{d_i \leq r} w_i r - \sum_{d_i \leq r} w_i d_i - Ar^2 \\ &= -Ar^2 + \left[\sum_{d_i \leq r} w_i\right] r - \left[\sum_{d_i \leq r} w_i d_i\right] \end{aligned}$$

입니다. 여기서 $f_i(r)$은 연속함수이므로 $g(r)$도 연속함수입니다.

$d_i \leq r$이라는 조건을 보면 $d_i$를 오름차순이 되게 농작물들을 정렬하는 것이 효율적임을 알 수 있습니다. 따라서 $0 = d_0 \leq d_1 \leq d_2 \leq \cdots \leq d_N < d_{N+1} = \infty$이고 $d_k \leq r < d_{k+1}$이면

$$\begin{aligned} g(r) = h_k(r) &= -Ar^2 + \left[\sum_{d_i \leq r} w_i\right] r - \left[\sum_{d_i \leq r} w_i d_i\right] \\ &= -Ar^2 + \left[\sum_{i=1}^{k} w_i\right] r - \left[\sum_{i=1}^{k} w_i d_i\right] \\ &= -Ar^2 + B_k r - C_k \end{aligned}$$

이 되는데, 이 때 $h_k(r)$은 $r$에 대한 간단한 이차함수 꼴로 표현됩니다.

각 구간 $[d_1, d_2], [d_2, d_3], \cdots, [d_N, \infty]$마다 서로 다른 형태의 이차함수들이 있고, $g(r)$은 연속함수이므로, 결국 구간마다의 이차함수가 연결된 형태의 함수가 됩니다. 따라서 각 구간의 이차함수들마다 최댓값 $M_k$을 구한다면 그것들의 최댓값인 $\max_{1 \leq k \leq N} M_k$가 $g(r)$의 답이 됩니다. 이차함수의 최댓값을 구할 때는 포물선의 축이 $[d_k, d_{k+1}]$ 안에 없을 수도 있다는 점에 주의합니다.

$B_k$와 $C_k$를 누적합(prefix sum)으로 구한다면 모든 구간에 대한 $h_k(r)$를 $\mathcal{O}(N)$만에 계산할 수 있으므로, 답을 구하는 전체 시간 복잡도는 $\mathcal{O}(N \log N)$이 됩니다.

(참고로, $g(r)$이 최대가 되는 점은 항상 극대점이 될 수 밖에 없기 때문에 $M_k$가 극댓값이 아닌 경우, 즉, 포물선의 축이 $[d_k, d_{k+1}]$ 안에 없는 경우 $M_k$는 무시해도 됩니다.)

## Champion G. 문자열 압축

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @lvalue (박건)

문자열 $S$의 $i$번째 문자까지 봤을 때, 최소 몇 개의 구간으로 쪼개서 압축 가능한지를 $T_i$라고 정의하면 다음이 성립합니다. ($L$ = 사전 단어의 최대 길이, $D$ = 사전 단어의 집합)

$$T_0 = 0$$
$$T_i = \min_{j<i,\ S_{j:i} \in D} T_j + 1$$

어떤 부분 문자열 $S_{j:i}$가 사전에 있는지, 즉 $S_{j:i} \in D$인지 검색하기 위해 trie를 naïve하게 구현하면, 시간 복잡도는 $\mathcal{O}(|S| \cdot L^2)$가 되어 시간 초과가 발생합니다. Trie의 상태 $L$개를 저장해 두고 각 DP 단계에서 재활용하거나, Aho-Corasick 알고리즘을 사용하면 $\mathcal{O}(|S| \cdot L)$를 얻을 수 있습니다. Trie는 메모리를 많이 사용하기 때문에, 최적화를 잘 하지 않으면 메모리 초과가 나올 수도 있음에 주의합니다.

사전 순으로 가장 먼저 오는 단어를 출력하는 것은 가능한 min 후보가 여러 개일 때 단어 번호가 가장 작은 것을 선택하는 것으로 처리할 수 있습니다.

## Champion H. 평행우주

**정답자** 0명 (0%)

**첫 정답** —

**출제자** @shiftpsh (박수현)

별자리는 $n$개의 별들이 $n-1$개의 선으로 연결되어 있다고 했으므로, 별을 정점으로, 선을 간선으로 생각하면 별자리는 트리 구조임을 알 수 있습니다. 따라서 이 문제는 트리의 위상이 같은지를 판단해야 하는 문제입니다. 다만, 트리의 개수가 최대 $10^5$개이기 때문에, 하나하나 비교해 보는 것은 불가능하고 해싱과 비슷한 방법으로 트리들을 관리해 줘야 함을 알 수 있습니다. 트리는 countable하기 때문에 이와 같은 일이 가능합니다.

우선 루트가 정해져 있는 트리에 고윳값을 부여하는 방법을 생각합니다. 루트가 $r$인 서브트리 $T_r$의 위상에 대한 고윳값을 $f(T_r)$이라고 생각한다면, $f(T_r)$은 $u \in N_{T_r}(r)$에 대해 재귀적으로 정의할 수 있을 것입니다.

한 예로, $f(T_r)$을 다음과 같이 정의하면 루트가 있는 트리의 위상에 대한 고윳값을 부여할 수 있습니다.

- 먼저, 노드 시퀀스 $(u_n)_{u_n \in N_{T_r}(r), n \in \mathbb{N}}^{||N_{T_r}(r)||}$을 정의합니다. 모든 $u_i$는 서로 다르고, 모든 $1 \le i < ||N_{T_r}(r)||$에 대해 $f(u_i) \le f(u_{i+1})$입니다. 쉽게 말하자면 $(u_n)$는 $r$의 모든 자식 노드 $u$들에 대해 $f(u)$의 값을 순서대로 정렬해 $u_1, u_2, \cdots, u_{||N_{T_r}(r)||}$ 순으로 번호를 붙여 둔 것입니다.
- 그러면 $f(T_r)$을 다음과 같이 정의할 수 있습니다. 여기서 $f$는 bit string이며, $+$는 문자열 연결(concatenation)을 의미합니다.

$$f(T_r) = \begin{cases} 10 & \text{if } ||T_r|| = 1 \\ 1 + \left(u_1 + \cdots + u_{||N_{T_r}(r)||}\right) + 0 & \text{otherwise} \end{cases}$$

  노드가 한 개인 트리의 경우 $f(T_r) = 10$이고, 노드가 여러 개인 트리의 경우, 그 서브트리들의 $f$ 값들을 정렬해 모두 이어붙인 bit string을 만든 뒤, 1과 0으로 감싸면 됩니다.

이와 같은 방법을 이용해 루트가 $r$인 서브트리 $T_r$에 대한 위상의 고윳값을 구할 수 있습니다. $f$가 트리의 위상이 될 수 있는가에 대한 증명은 trivial하므로 적지 않도록 하겠습니다.

이제 루트가 없는 트리의 위상의 고윳값을 구하고자 합니다. 이를 위해 다음 정의들을 사용합니다.

**Definition 1.** 가중치가 없는 양방향 트리에서 두 노드 $u$, $v$ 사이의 **거리** $d(u, v)$는 $u$에서 $v$까지의 최단 경로를 구성하는 간선 수이다.

**Definition 2.** 가중치가 없는 양방향 트리 $T$에서 어떤 노드 $u$의 **이심률** $E_T(v)$는 다른 모든 노드까지의 최단 거리 중의 최댓값이며, 다음과 같이 정의된다.

$$E_T(u) = \max_{v \in V(T)} d(u, v)$$

$V(T)$는 $T$의 정점들의 집합을 의미한다.

**Definition 3.** 가중치가 없는 양방향 트리 $T$에서 **트리의 반지름** $R(T)$와 **트리의 중심** $C(T)$는 각각 다음과 같이 정의된다.

$$R(T) = \min_{u \in V(T)} E_T(u)$$
$$C(T) = \{u \mid E_T(u) = R(T)\}$$

이제 다음과 같은 Lemma를 증명합니다.

**Lemma 1.** 가중치가 없는 양방향 트리 $T$에서 트리의 중심은 항상 *1*개 혹은 *2*개이다.

*Proof.* 만약 $||T|| \le 2$라면, $T$의 노드들이 곧 중심이 되므로 이 경우 $T$의 중심의 갯수 $||C(T)|| = ||T||$입니다.

$T$가 $n \ge 3$개의 정점을 가지고 있다고 가정합시다.

어떤 정점 $u$로부터 다른 정점 $v$까지의 거리의 최댓값 $\max_{v \in V(T)} d(u, v)$은 $\deg(v) = 1$일 때만 가질 수 있습니다. 또한, 정점이 2개 이상인 모든 트리는 $\deg(u) = 1$이 되는 정점 $u$를 2개 이상 가집니다.

$T = T_0$에서 $\deg(u) = 1$인 정점 $u$를 모두 제거한 트리 $T_1$을 만든다고 생각합시다. 위의 사실로 인해,

$$E_{T_1}(u) = E_{T_0}(u) - 1 \qquad \forall u \in T_1$$

가 됩니다. 따라서 트리의 중심의 정의에 의해, $C(T_1) = C(T_0)$이 성립합니다.

위와 같은 방법으로 $T_k$에서 $\deg(u) = 1$인 정점 $u$를 모두 제거한 트리 $T_{k+1}$을 만들면, $||T_{k+1}|| \geq 1$일 때 $C(T_{k+1}) = C(T_k)$가 성립합니다. 따라서 이 과정을 $||T_n|| \leq 2$가 될 때까지 반복한다면 $C(T_0) = C(T_n)$이고, $||T_n|| \leq 2$일 경우 $||C(T_n)|| = ||T_n||$임을 위에서 보였으므로 $||C(T_0)|| \leq 2$가 항상 성립합니다. □

위 증명을 통해 모든 가중치가 없는 양방향 트리에 대해 트리의 중심은 하나 혹은 두 개만 존재함을 확인할 수 있습니다. 따라서 루트가 정해지지 않은 트리라면,

- 트리의 중심이 $c$로 하나인 경우, 루트를 트리의 중심으로 고른 후 위상에 고윳값을 부여할 수 있습니다.
- 트리의 중심이 $c_1$, $c_2$로 두 개인 경우, $c_1$이 루트인 트리에 대한 $f(T_{c_1})$과, $c_2$가 루트인 트리에 대한 $f(T_{c_2})$ 중 최댓값 혹은 최솟값을 고르면 됩니다.

문제에서 트리의 노드 수는 30 이하로 주어지므로, 위와 같이 위상을 계산한다면 64비트 정수 하나로 트리의 위상을 표현할 수 있게 됩니다. 이렇게 계산한 트리의 위상의 종류를 집합 등으로 관리하여, 개수를 출력하면 정답입니다.

이 알고리즘은 *The Design and Analysis of Computer Algorithms*에서 소개되어 Aho–Hopcroft–Ullman(AHU) 알고리즘이라고도 불리고 있습니다.