2022 연세대학교 신학기맞이 프로그래밍 경진대회

목차

문제		의도한 난이도	출제자
A	녹색거탑	Beginner	정기웅 QuqqU
B	현대모비스와 함께하는 부품 관리	Beginner	이국렬 lky7674
C	엠비티아이	Easy	박선종 standingbell
D	미적분학 입문하기 2	Easy	이국렬 lky7674
E	인지융	Medium	김병철 klm03025
F	팬케이크맛쿠키	Medium	박선종 standingbell
G	심각한 계단 중독입니다	Medium	김병철 klm03025
H	GGG	Hard	이국렬 lky7674
I	XOR ABC	Hard	김태윤 ystaeyoon113
J	잠자는 규리	Hard	김태윤 ystaeyoon113
K	이것도 XOR해 보시지	Challenging	이승재 pom0319

A. 녹색거탑

`math`

출제진 의도 – **Beginner**

- ✓ 제출 77번, 정답 51명 (정답률 66.234%)
- ✓ 처음 푼 사람: **yup0927**, 0분
- ✓ 출제자: QuqqU

A. 녹색거탑

- 코딩괴물이 어느 곳에 있든, 항상 왼쪽 혹은 오른쪽으로 내려가는 2 가지 방법만 있습니다.
- 따라서 N 층의 녹색거탑을 내려오는 경우의 수는 2^N 이고, 이것을 출력하면 정답입니다.
- 2^N 을 계산하는 방법은 여러가지이지만, 출제자가 생각하는 가장 좋은 방법을 소개합니다.
- > printf(“%d”, 1 « n);

B. 현대모비스와 함께하는 부품 관리

`implementation`
출제진 의도 – **Beginner**

- ✓ 제출 94번, 정답 47명 (정답률 51.064%)
- ✓ 처음 푼 사람: **developerhan**, 3분
- ✓ 출제자: lky7674

NAVER D2 HYUNDAI MOBIS

- ✓ 지문을 다시 한 번 읽어보도록 합시다. 특히 예제를 다시 보시길 바랍니다.
- ✓ 입력 값에 상관없이, 테스트 케이스의 개수만큼 분류가 끝났다는 메세지를 출력하면 됩니다.

C. 엠비티아이

`implementation`, `brute force`
출제진 의도 – **Easy**

- 제출 72번, 정답 38명 (정답률 52.778%)
- 처음 푼 사람: **luciaholic**, 10분
- 출제자: standingbell

C. 엠비티아이

- ✓ 백준 및 Problem Solving 분야에서는 영어문제를 마주할 일이 매우 많고, 이번 대회 취지에 따라 준비해본 영어문제입니다.
- ✓ MBTI 조합은 2으로 총 16가지 조합이 나올수 있습니다.
- ✓ ENFP, ENFJ, ENTP, ENTJ, ESFP, ESFJ, ESTP, ESTJ
- ✓ INFP, INFJ, INTP, INTJ, ISFP, ISFJ, ISTP, ISTJ
- ✓ 2차원 보드판에서 위의 16가지 단어를 찾으면 정답의 개수를 추가해줍시다

C. 엠비티아이

- ✓ $4 \leq N$
  세로방향 탐색 : 위 $\rightarrow$ 아래, 아래 $\rightarrow$ 위
- ✓ $4 \leq M$
  가로방향 탐색 : 왼쪽 $\rightarrow$ 오른쪽, 오른쪽 $\rightarrow$ 왼쪽
- ✓ $4 \leq N$, $4 \leq M$
  대각선방향 탐색 : 왼쪽 위 $\rightarrow$ 오른쪽아래 , 왼쪽 아래 $\rightarrow$ 오른쪽 위
  오른쪽 위 $\rightarrow$ 왼쪽 아래, 오른쪽 아래 $\rightarrow$ 왼쪽 위
- ✓ 조건을 만족하지 않으면 해당 방향 탐색을 생략하고 구현해주면 됩니다.
- ✓ Index와 탐색하는 방향 구현, N, M 이 각각 4보다 작을때를 주의하면서 구현하면 8가지를 쉽게 구현할 수 있습니다.
- ✓ 전탐색을 하더라도 $O(32NM)$ 이기 때문에, 여유롭게 통과할 수 있습니다.

- DFS로 푼다면 더 간단하게 풀 수 있습니다.
- $N \times M$ 칸에서 각각 8방향으로 DFS 탐색을 합니다.
- 만약 각 순서에서 등장해야하는 알파벳이 등장하지 않는다면 탐색을 멈춥니다.
- 첫번째 depth는 E, I 만 찾고, 두번째 depth 때는 N,S 만 찾는 등의 방식을 선택하면 됩니다.
- 해당 방법으로 풀이시 N, M 조건에 상관없이 한번에 풀 수 있습니다.
- DFS를 모르시는 분들이라면 해당문제를 통해 연습해봅시다.

D. 미적분학 입문하기 2

`math`, `calculus`

출제진 의도 – **Easy**

- ✓ 제출 53번, 정답 22명 (정답률 41.509%)
- ✓ 처음 푼 사람: **coconut99**, 20분
- ✓ 출제자: lky7674

D. 미적분학 입문하기 2

- ✓ 지문 상의 회전체의 공식을 통해서 좀... 많이 귀찮은 계산을 통해서 답을 구할 수도 있습니다.
- ✓ 이번에는 학교에서 귀찮아서 가르치지 않은 공식을 소개하고, 이를 통해서 풀어보도록 하겠습니다.

- Pappus's centroid theorem : 평면 상의 넓이가 A인 영역이 주어졌을 때, 해당 영역의 무게 중심에서 d만큼 떨어진 축을 중심으로 회전했을 때 나오는 회전체의 부피 V는 $2\pi dA$다.
- 삼각형의 무게 중심 : $(x_c, y_c) = (\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3})$
- 삼각형의 넓이 A : 신발끈 공식, 헤론의 공식 등 구하는 방법은 여러 가지 있습니다.
- 따라서 x 축, y 축을 기준으로 삼각형을 회전했을 때 나오는 회전체의 부피는 각각 $2\pi y_c A$, $2\pi x_c A$가 됩니다.
- 1학년 학생들은 이 공식 알아두는 것을 추천합니다. 공학수학 1에서 유용하게 사용할 것입니다.

E. 인지융

ad-hoc

출제진 의도 – **Medium**

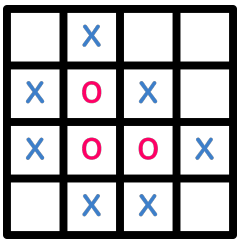
- ✓ 제출 93번, 정답 15명 (정답률 16.129%)
- ✓ 처음 푼 사람: **dreami63**, 56분
- ✓ 출제자: klm03025

E. 인지융

- ✓ 두 공간을 어떤식으로 채워야 최대한 겹치지 않는지 찾는 문제입니다.
- ✓ 어차피 두 구역을 최대한 닿지 않게 채워 놓기만 하면 남은 칸엔 모두 바리케이드를 놓으면 되기 때문에, 바리케이드의 크기는 고려하지 않아도 됩니다.

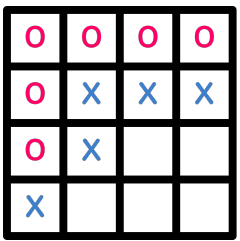
E. 인지융

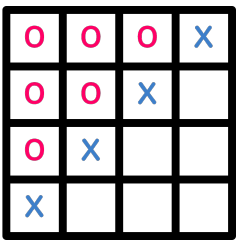
- ✓ 두 공간을 최대한 겹치기 않게 배치하기 위해선, 각각의 공간이 외부 공간과 닿는 칸의 수를 최소화 해야 합니다.
- ✓ 여기서, 외부 공간과 닿는 칸이란 아래 그림에서 파란색 X가 표시된 칸과 같습니다.
- ✓ 파란색 칸에는 다른 공간을 배치할 수 없기 때문에, 이 칸을 최소화 하는 방법을 찾는 것이 문제 풀이의 핵심입니다.



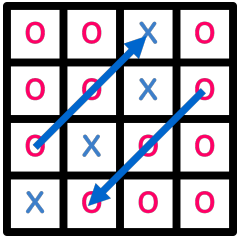
E. 인지융

- ✓ 쉽게 생각해보면, 아래 그림과 같이 두 모서리를 끼고 있으면 외부 공간과 닿는 칸의 수가 최소화가 될 것입니다.
- ✓ 하지만 왼쪽 그림처럼 모서리만 채우게 될 경우, 가운데가 비게 되어 외부 공간과 닿는 칸의 수가 늘어납니다.
- ✓ 결국, 비어있는 칸을 채우는 방식으로 접근하게 되면, 오른쪽 그림과 같이 계단 방식으로 채우는 것이 가장 효율적이라는 것을 알 수 있습니다.





- ✓ 물론, 반대쪽에도 대각선으로 값을 채운다는 것을 인지해야 합니다.
- ✓ 이 경우 두 영역이 최대한 만나지 않기 위해선, 칸을 채우는 방향을 서로 반대로 하면 됩니다.



F. 팬케이크맛 쿠키

`Class` `Implementation`, `Greedy`
출제진 의도 – **Medium**

- 제출 17번, 정답 3명 (정답률 17.647%)
- 처음 푼 사람: **luciaholic**, 100분
- 출제자: standingbell

F. 팬케이크맛 쿠키

- 문제 지문이 매우 길기 때문에, 차근차근 한줄씩 분석해봅시다.
- 문제의 맵은 제 1사분면 좌표평면으로 만들어져 있으며, 예시 그림과 같이 바닥 $y = 0$ 인 지점에 대해서는 젤리가 생성되어 있지 않습니다.
- 모든 시점 t에서 최대 점수를 획득할 수 있는 경로의 문구에 주목해봅시다.
- 처음부터 maxC만큼의 능력치를 모두 쓰고, curC가 채워지는 족족 사용해서 u만큼 떠오르면 됩니다.
- 즉 maxC 소모, 한칸 d만큼 감소, 다시 u만큼 증가를 반복하면 됩니다.
- 또한, 두개의 경로가 같다면 더 빠르게 능력을 사용하는 경로를 우선시한다고 했기 때문에, 무조건 능력을 앞선 시점 t에서 써버리는게 정답입니다.

F. 팬케이크맛 쿠키

- ✓ setT가 호출되는 횟수는 최대 1500회이기때문에, 시간이 넘어갈 일은 없습니다.
- ✓ 생성자에서도 모든 변수를 초기화해주는걸 잊지말아야 합니다.
- ✓ 또한, 미리 reset() 함수를 구현해둘때, curC까지 초기화를 해주고 세터마다 reset()을 호출해주면 구현이 쉬워집니다.
- ✓ 모든 분들이 처음 겪어볼 백준 시스템 기능중 하나인 '클래스 구현' 이기 때문에 구현에 대한 설명이 길뿐, 해당문제의 의도는 Greedy 하게 능력을 초반에 써버리는 것이었습니다.
- ✓ 또한, 상반기에 열리는 대회인만큼 신입생분들은 class에 대한 개념을 숙지하시길 바라고, class에 대해서 모르시는 다른 분들께서도 객체지향 프로그래밍이 매우 중요한 개념이기 때문에 잘 숙지하셨으면 좋겠습니다.

G. 심각한 계단 중독입니다

`Constructive`
출제진 의도 – **Medium**

- ✓ 제출 38번, 정답 6명 (정답률 15.789%)
- ✓ 처음 푼 사람: **dlguq0107**, 111분
- ✓ 출제자: klm03025

G. 심각한 계단 중독입니다

- ✓ 주어진 데이터로 원형 계단 수를 만들 수 있는지 판단하는 문제입니다.
- ✓ 먼저, 원형 계단 수의 특성을 생각해 봅시다.

G. 심각한 계단 중독입니다

- ✓ 어떤 수 K에 대해, K + 1 또는 K - 1이 양 옆에 있어야 합니다.
- ✓ 즉, 어떤 수 K가 n개 있다면, K + 1과 K - 1이 2n개 있어야 합니다.

G. 심각한 계단 중독입니다

- 하지만 반대로, K + 1과 K - 1의 입장에선 K가 필요합니다.
- 이때, K + 1과 K - 1은 K의 왼쪽에 붙을 수도 있고, 오른쪽에 붙을 수도 있기 때문에 하나의 수가 2개의 수 처럼 동작합니다.
- 첫 번째 케이스와 두 번째 케이스를 모두 고려했을 때, 결과적으로 2를 곱할 필요 없이 존재하는 수의 개수 그 자체로 관리하도록 합시다.

G. 심각한 계단 중독입니다

- ✓ 가장 작은 수가 n개 있다면, 그 수보다 1 큰 수는 최소 n개 있어야 합니다.
- ✓ 하지만, 정확히 n개가 있게 된다면 그 수보다 1 더 큰 수와 연결될 숫자가 없어집니다.
- ✓ 이렇게 되면, 전체 수가 이어지지 않게 되니, 결과적으로 가장 작은 수 보다 1 큰 수는 n개가 아닌 n + 1개 이상 필요합니다.

G. 심각한 계단 중독입니다

- 이런 식으로, 어떤 수보다 1 더 큰 수는 남은 어떤 수의 개수보다 최소 1 더 많은지 확인해주고, 매칭되었다고 가정하며 수를 계속 지워줍니다.
- 마지막으로, 다 지우고 난 뒤 가장 큰 수보다 1 더 작은 수의 개수는 가장 큰 수와 동일해야 합니다.
- 앞에서 말한 모든 조건을 만족하면 1을, 그렇지 않으면 -1을 출력하면 됩니다.

H. GGG

`math`

출제진 의도 – **Hard**

- ✓ 제출 28번, 정답 3명 (정답률 10.714%)
- ✓ 처음 푼 사람: **starwh03**, 167분
- ✓ 출제자: lky7674

H. GGG

- ✓ 원래 G번으로 출제될 문제였지만, 사정상 H번으로 옮기면서 이상한 이름이 된 문제였습니다.
- ✓ 벅스 교수님 수업에서 아이디어를 가져와서 문제를 출제했습니다.

H. GGG

n	0	1	2	3
$a_n = f(n)$	**7**	13	29	61
Δ_1	-	-	-	-
Δ_2	-	-	-	-
Δ_3	-	-	-	-

예시 : $f(x) = x^3 + 2x^2 + 3x + 7$

- ✓ 우선 N 차 다항식 $f(x)$ 가 있을때, $f(0)$ 부터 $f(N)$ 까지의 값을 구해봅시다.
- ✓ 각 함수값은 그냥 단순하게 구하게 되면, 시간 복잡도가 $O(N^3)$ 이 되기에 최적화를 해야 합니다.
- ✓ Horner's Rule($f(x) = f_0 + x(f_1 + x(\cdots + x(f_{N-1} + xf_N)))$)을 이용해서 시간 복잡도를 $O(N^2)$ 으로 줄일 수 있습니다.

H. GGG

n	0	1	2	3
$a_n = f(n)$	**7**	13	29	61
Δ_1	-	**6**	16	32
Δ_2	-	-	**10**	16
Δ_3	-	-	-	**6**

예시 : $f(x) = x^3 + 2x^2 + 3x + 7$

- ✓ 구한 함수값을 바탕으로 $\{a_n\}$의 제 N계 계차 수열까지 구해봅시다.
- ✓ 이는 시간 복잡도 $O(N^2)$에 간단히 할 수 있습니다.
- ✓ 각 수열의 첫 항이 GGG 함수의 입력 값이 됩니다.

I. XOR-ABC

`math`, `combinatorics`

출제진 의도 – **Hard**

- ✓ 제출 51번, 정답 9명 (정답률 17.647%)
- ✓ 처음 푼 사람: **coconut99**, 97분
- ✓ 출제자: ystaeyoon113

NAVER D2 HYUNDAI MOBIS

- xor 연산의 특징을 관찰해 조건을 만족하는 (A, B, C) 쌍을 조합론적으로 계산하는 문제입니다.
- 먼저, $A \text{ xor } B = C$ 를 만족하는 음이 아닌 일반적인 정수 쌍 (A, B, C) 을 생각해 봅시다.
- 주어진 정수 쌍 (A, B, C) 은 $B \text{ xor } A = C$ 를 만족합니다. 이는 쉽게 떠올릴 수 있습니다.
- 주어진 정수 쌍 (A, B, C) 가 $A \text{ xor } C = B$ 를 만족한다는 것이 이 문제의 첫번째 핵심 관찰입니다. 먼저 그것을 보이겠습니다.

- xor 연산은 각 자릿수에 대해 독립적으로 적용되므로 하나의 자릿수에 대해서 분석하면 이를 모든 자릿수에 대해 적용시킬 수 있습니다. 따라서 먼저 한 자리 이진수 A, B, C 에 대해 분석해 보겠습니다.

A	0
B	0
C	0

A	0
B	1
C	1

A	1
B	0
C	1

A	1
B	1
C	0

- 위와 같이 한 자리 이진수 A, B, C 에 대해 $A \operatorname{xor} B = C$ 이면 $A \operatorname{xor} C = B$ 를 만족합니다. 물론 역도 성립합니다.

✓ 한 자릿수에서 얻어진 관찰을 모든 자릿수에 적용하면 다음과 같은 결과를 얻을 수 있습니다.

A	A_k	A_{k-1}	...	A_3	A_2	A_1
B	B_k	B_{k-1}	...	B_3	B_2	B_1
C	C_k	C_{k-1}	...	C_3	C_2	C_1

A	A_k	A_{k-1}	...	A_3	A_2	A_1
C	C_k	C_{k-1}	...	C_3	C_2	C_1
B	B_k	B_{k-1}	...	B_3	B_2	B_1

✓ 따라서, 음이 아닌 정수 쌍 (A, B, C)에 대해 $A \text{ xor } B = C$ 이면, $A \text{ xor } C = B$를 만족합니다.

- ✓ 따라서 다음과 같은 사실을 도출할 수 있습니다.
- ✓ $A \text{ xor } B = C$ 인 음이 아닌 정수 쌍 (A, B, C) 에 대해서, 다음 6가지 식이 모두 성립합니다.
  1. $A \text{ xor } B = C$
  2. $A \text{ xor } C = B$
  3. $B \text{ xor } A = C$
  4. $B \text{ xor } C = A$
  5. $C \text{ xor } A = B$
  6. $C \text{ xor } B = A$
- ✓ 이는 (A, B, C) 를 순서를 고려해서 나열하는 방법과 같습니다.

- 알아낸 사실을 통해 xor 연산의 특징을 좀더 살펴봅시다.
- $A \text{ xor } 0 = A$ 입니다. 따라서 $A \text{ xor } A = 0$ 이고, $A \text{ xor } B = 0$ 을 만족하는 B 는 A 가 유일합니다.
- 따라서 양의 정수 A, B 에 대해 $A \neq B$ 라면, $A \text{ xor } B \neq 0$ 입니다. A 와 B 가 양수이기 때문에 $A \text{ xor } B \neq A$ 이고, $A \text{ xor } B \neq B$ 입니다.
- 결국 양의 정수 $A, B\,(A \neq B)$ 에 대해 $A \text{ xor } B = C$ 이면 C 는 A, B 가 아닌 양의 정수입니다.
- 또한 A 와 B 가 K 자리의 이진수라면 C 또한 K 자리의 이진수이므로 $1 \leq C \leq 2^K - 1$ 입니다.

I. XOR-ABC

✓ 결론에 앞서 $K = 4, 1 \leq A, B \leq 2^K - 1, A \neq B$인 (A, B)에 대해서 $A \text{ xor } B$를 관찰해 봅시다.

		A						B	C					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

		A						C	B					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

		C						B	A					
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

✓ 이전에 알아낸 6가지의 등호가 모두 성립하기 때문에, 집합 $\{A, B, C\} = \{3, 9, 10\}$이 등장하는 경우는 $(A, B, C) = (3, 9, 10), (3, 10, 9), (9, 3, 10), (9, 10, 3), (10, 3, 9), (10, 9, 3)$ 총 6가지입니다.

- $1 \leq A, B \leq 2^K - 1, A \neq B$ 인 모든 (A, B) 에 대해서 $(A, B, C) = (A, B, A \text{ xor } B)$ 를 관찰해 보겠습니다.
- 순서쌍의 개수는 A, B 를 정해주는 $(2^K - 1) \times (2^K - 2)$ 개가 존재합니다. 각 순서쌍은 어떤 집합에 대응되는데, 하나의 집합에 대응되는 순서쌍은 반드시 6개가 존재합니다.
- 각 집합에 대응된 순서쌍들 중 $(A < B < C)$ 를 만족시키는 순서쌍은 반드시 6개 중 하나입니다.
- 따라서 모든 조건을 만족시키는 (A, B, C) 쌍은 $\frac{(2^K - 1) \times (2^K - 2)}{6}$ 개 존재하게 됩니다.

- 40점은 반복문을 통해서 브루트포스로 풀 수 있습니다. $\mathcal{O}\left(4^K\right)$ 로 가능합니다.
- 80점은 그냥 $\frac{(2^K-1)\times(2^K-2)}{6}$ 을 적용하고 $1,000,003$으로 잘 나누면 됩니다. $\mathcal{O}\left(K\right)$ 로 가능합니다.
- 90점을 받기 위해서는 $(2^K-1)\times(2^K-2)$에 6의 $1,000,003$에 대한 곱셈 역원을 구해 곱해야 하기 때문에 페르마 소정리가 필요합니다.$\mathcal{O}\left(K+\log 1,000,003\right)$ 로 가능합니다.
- 100점을 받기 위해서는 2^K 연산을 빠르게 할 수 있기 위해 분할정복을 통한 거듭제곱법이 필요합니다. $\mathcal{O}\left(\log K+\log 1,000,003\right)$ 로 가능합니다.

J. 잠자는 규리

`stack`, `prefix sum`
출제진 의도 – **Hard**

- 제출 10번, 정답 1명 (정답률 10.000%)
- 처음 푼 사람: **starwh03**, 262분
- 출제자: ystaeyoon113

- ✓ 주어진 코드를 보고, 작동과정을 알아낸 후 코드의 시간복잡도를 줄이는 문제입니다.
- ✓ 주어진 코드의 시간복잡도는 $\mathcal{O}\left(N^3\right)$ 입니다.
- ✓ 이를 stack과 prefix sum을 이용해서 $\mathcal{O}\left(N\right)$ 까지 줄이는 것이 이 문제의 목표입니다.
- ✓ 먼저 규리가 짠 코드가 어떤식으로 작동하는지 간단하게 살펴보겠습니다.

J. 잠자는 규리

- ✓ 규리의 코드는 ans 배열을 $ans[1]$ 부터 $ans[N-1]$ 까지 차례대로 계산해 나갑니다.
- ✓ $ans[x]$ 를 계산하는 과정을 살펴보겠습니다. $y \in [0, x-1]$ 인 모든 정수 y 에 대해 $\min_{y \le i \le x} v[i]$ (회색 영역의 최소값)과, $ans[y]$ 를 곱해서 $ans[x]$ 에 더해주는 작업을 반복합니다.

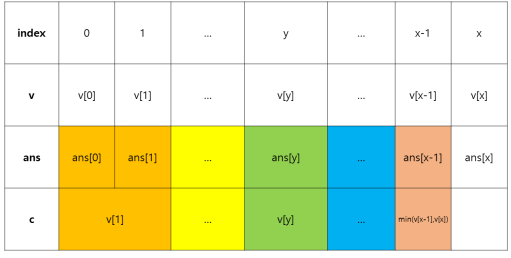
index	0	1	...	y-1	y	...	x	...	N-1
v	v[0]	v[1]	...	v[y-1]	v[y]	...	v[x]	...	v[N-1]
ans	ans[0]	ans[1]	...	ans[y-1]	ans[y]	...	ans[x]	...	ans[N-1]

index	0	1	...	y-1	y	...	x	...	N-1
v	v[0]	v[1]	...	v[y-1]	v[y]	...	v[x]	...	v[N-1]
ans	ans[0]	ans[1]	...	ans[y-1]	ans[y]	...	ans[x]	...	ans[N-1]

- ✓ x를 고정시키고, y 변화에 따른 $\min_{y \le i \le x} v[i]$를 관찰해 봅시다.
- ✓ x가 고정되면 $\min_{y \le i \le x} v[i]$는 y에 의해 결정되므로, $\min_{y \le i \le x} v[i] = c[y]$라고 정의하겠습니다.
- ✓ 만약 $v[y-1] \ge c[y]$이라면 $c[y-1] = c[y]$입니다. 반면, $v[y-1] < c[y]$이라면 $c[y-1] < c[y]$입니다. 따라서 모든 y에 대해 $c[y-1] \le c[y]$입니다.
- ✓ y를 $x-1$부터 0까지 순회하면 prefix sum을 구하는 것과 비슷하게 $c[y]$를 구할 수 있습니다.

J. 잠자는 규리

- $x-1$을 제외하고 c값이 같은 원소끼리 그룹으로 나눠보면, 그룹의 c값은 그룹에서 가장 index가 큰 원소의 v 값과 같습니다. 그리고 그 값은 그룹에 속한 원소들의 v값의 최소값입니다.
- 그룹으로 나뉘었다면 $ans[x]$에 $c[y] \times ans[y]$를 모든 y에 대해 각각 더하는 대신, c값이 같은 그룹끼리 다음과 같이 ans값을 더해서 한번에 $ans[x]$에 더해줄 수 있습니다.
- 이때 각 그룹마다 ans의 합, c값을 저장해 두겠습니다.



J. 잠자는 규리

- 이제 $ans[x+1]$ 을 계산해 보겠습니다. $\min_{y \le i \le x+1} v[i]$ 에 따라 새로운 그룹을 만들어야 합니다.
- 이때 이전에 $ans[x]$ 를 구하기 위해 만든 그룹들을 이용하여 새로운 그룹을 만들 수 있습니다.
- 먼저 $c[x] = min(v[x], v[x+1])$ 인 그룹을 만들고, x 를 그룹에 포함시킵니다. 이를 새 그룹이라고 하겠습니다. (회색 그룹)

index	0	1	...	y	...	x-1	x	x+1
v	v[0]	v[1]	...	v[y]	...	v[x-1]	v[x]	v[x+1]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]	ans[x+1]
c	v[1]		...	v[y]	...	min(v[x-1],v[x])	min(v[x],v[x+1])	

J. 잠자는 규리

- $c[x] \leq$ (인접한 그룹의 c값)일 때까지 인접한 그룹을 새 그룹에 연속적으로 포함시키고 새 그룹의 c값과 ans의 합을 update합니다. 새 그룹에 포함되지 않은 그룹은 그대로 유지합니다.

index	0	1	...	y	...	x-1	x	x+1
v	v[0]	v[1]	...	v[y]	...	v[x]	v[x]	v[x+1]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]	ans[x+1]
c	v[1]		...	v[y]	c[x]			

- 위 과정으로 얻어진 그룹들은 모든 y에 대해 $\min_{y \leq i \leq x+1} v[i]$을 계산하고 c값에 따라 나눈 그룹과 정확히 같습니다. 왜 그런지는 어렵지 않게 증명할 수 있습니다.

- 따라서, 얻어진 그룹들에 대해서 각각 c값과 ans 값의 합을 곱해서 $ans[x+1]$에 더하면 됩니다만.. $ans[x]$를 구할 때 썼던 정보를 이용하면 이마저도 최적화 할 수 있습니다.

index	0	1	...	y	...	x-1	x
v	v[0]	v[1]	...	v[y]	...	v[x-1]	v[x]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]
c	v[1]		...	v[y]	...	c[x-1]	

index	0	1	...	y	...	x-1	x	x+1
v	v[0]	v[1]	...	v[y]	...	v[x]	v[x]	v[x+1]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]	ans[x+1]
c	v[1]		...	v[y]	c[x]			

- 먼저, 새 그룹에 포함되지 않은 그룹들은 $ans[x]$를 구하는 데에도 이미 계산된 적이 있습니다. 따라서 다시 계산할 필요는 없습니다.

J. 잠자는 규리

index	0	1	...	y	...	x-1	x
v	v[0]	v[1]	...	v[y]	...	v[x-1]	v[x]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]
c	v[1]		...	v[y]	...	c[x-1]	

index	0	1	...	y	...	x-1	x	x+1
v	v[0]	v[1]	...	v[y]	...	v[x]	v[x]	v[x+1]
ans	ans[0]	ans[1]	...	ans[y]	...	ans[x-1]	ans[x]	ans[x+1]
c	v[1]		...	v[y]	c[x]			

- 어떤 그룹 G가 새 그룹에 포함된 그룹이라고 하겠습니다. 그러면 G는 $ans[x+1]$를 계산할 때 $ans[x]$에 더한 것보다 $(c[x]-$ 이전 G의 c값$) \times ($ 이전 G의 ans 합$)$ 만큼 더 더하게 됩니다.
- 따라서 $ans[x+1]$은 $ans[x]$에 새 그룹에 포함된 기존 그룹들의 변화치만 더해주면 됩니다.
- 이를 $x = 0$부터 $N-2$까지 반복하면 됩니다.

- ✓ 그룹의 ans 합, c 값을 stack을 이용해서 관리하면 효과적입니다.
- ✓ $ans[x]$ 를 채우는 데 고려해야 하는 그룹의 수는 x 마다 다르므로, ans 배열을 모두 채울 때까지 고려할 수 있는 그룹이 몇개나 되는지를 따져보겠습니다.
- ✓ 각 그룹의 가장 오른쪽 원소를 그룹의 대표라고 하면, 정확히 그룹의 수와 일치하게 됩니다.
- ✓ 각 원소는 최대 한번 그룹의 대표로 선정이 가능하고, 다른 그룹으로 포함되어 그룹의 대표에서 밀려나면 다시는 그룹의 대표로 선정되지 못합니다. 따라서 ans 배열을 다 채우는 데 고려하게 되는 그룹의 수는 $\mathcal{O}(N)$ 입니다.
- ✓ 따라서 알고리즘의 시간 복잡도는 $\mathcal{O}(N)$ 입니다.

K. 이것도 XOR 해 보시지

`ad_hoc`, `set`

출제진 의도 – **Challenging**

- ✓ 제출 5번, 정답 1명 (정답률 33.333%)
- ✓ 처음 푼 사람: **luciaholic**, 178분
- ✓ 출제자: pom0319

K. 이것도 XOR해 보시지

- ✓ 동전의 최대 무게조차 알 수 없으니, 동전의 최대 무게를 알아내는 동시에 각 동전의 무게도 알아내야 합니다.
- ✓ $a \oplus a = 0$인 점을 이용하면, $n - 1$번의 질의로 0번 동전과 모든 동전의 xor값을 알 수 있습니다.
- ✓ $b \neq c$면 $a \oplus b \neq a \oplus c$라는 점을 이용하면, 0번 동전의 무게와 관계없이 위의 질의를 통해 k개의 서로 다른 값을 얻을 것임을 알 수 있습니다.

K. 이것도 XOR해 보시지

- ✓ $k = 5$인 경우를 생각해봅시다. 0번 동전의 무게가 1g이면 $0, 2, 3, 4, 5$를 얻고, 2g이면 $0, 1, 3, 6, 7$을 얻는 것처럼, 0번 동전 무게에 따라 얻는 값의 집합이 달라집니다.
- ✓ 따라서 질의를 통해 얻은 값의 집합을 이용하면 0번 동전의 무게를 알 수 있습니다.

XOR		다른 동전의 무게				
		1	2	3	4	5
0번 동전의 무게	1	0	3	2	5	4
	2	3	0	1	6	7
	3	2	1	0	7	6
	4	5	6	7	0	1
	5	4	7	6	1	0

K. 이것도 XOR해 보시지

- ✓ 이제 주어진 정보는 0번 동전의 무게와, 0번 동전과 모든 동전의 무게의 xor 값입니다.
- ✓ $a \oplus b = c$ 면 $a \oplus c = b$ 이므로, 두 값을 다시 xor 하면 각 동전의 무게를 구할 수 있습니다.