2024 KSAAC Winter

Official Solutions

by

Automata

2024년 2월 24일

문제		의도한 난이도	출제자
A	소수가 아닌 수 3	어려워요	flakepowders
B	KSA 문자열	어려워요	sooahn07
C	당근 클릭 게임	어려워요	eric00513
D	기숙사 비밀번호 구하기	어려워요	davidkim0
E	시험공부 계획 세우기	어려워요	aaron0526
F	냥냥이와 삼색이	어려워요	hwpark12
G	사인은 B0	어려워요	davidkim0
H	선형대수학: 개념과 방법	어려워요	flakepowders
I	탐색 게임	어려워요	aaron0526
J	히스토그램에서 가장 큰 직사각형과 쿼리 2	어려워요	aaron0526
K	좋은 수열	어려워요	mickeyjung

Thanks to



- ✓ 후원
  - utilforever
  - ai4youej
- ✓ 운영진
  - aaron0526
  - davidkim0
  - eric00513
  - flakepowders
  - hwpark12
  - mickeyjung
  - sooahn07
- ✓ 검수진
  - ai4youej
  - alex0211
  - anchor
  - dohoon
  - edward3378
  - gs20036
  - gs22059
  - hjroh0315
  - jyheo98
  - kiwiyou
  - naeby
  - parkky
  - runnie0427
  - shiftpsh
  - utilforever
  - vkdldjvkdnj
  - wizardrabbit

A. 소수가 아닌 수 3

`math`

출제진 의도 – **어려워요**

- 제출 1402번, 정답 283명 (정답률 21.612%)
- 처음 푼 사람: **dabbler1**, 1분
- 출제자: flakepowders

A. 소수가 아닌 수 3



- ✓ 모든 한 자리 수는 111을 곱해서 그 숫자만 사용한 소수가 아닌 수로 만들 수 있습니다.
- ✓ 000이나 11 등을 출력하지 않도록 유의해야 합니다.

B. KSA 문자열

`string`, `greedy`
출제진 의도 – **어려워요**

- 제출 1062번, 정답 131명 (정답률 12.806%)
- 처음 푼 사람: **dabbler1**, 6분
- 출제자: sooahn07

B. KSA 문자열

- KSAKSAK... 형태의 문자열을 **K 문자열**, SAKSAKS... 형태의 문자열을 **S 문자열**, AKSAKSA... 형태의 문자열을 **A 문자열**이라고 합시다.
- 또한, 이 셋을 묶어서 **KSA 문자열**이라고 합시다.
- 문자는 맨 앞과 맨 뒤에만 추가할 수 있으므로 X 에서 문자를 적절히 삭제하여 **KSA 문자열**을 만드는 것이 핵심입니다.
- 어떤 문자열 X 에 대해 X 에서 0 개 이상의 문자를 삭제하여 만든 문자열을 X 의 부분수열이라고 합시다.

B. KSA 문자열

- **서브태스크 2**에서는 **K 문자열**인 X 의 부분수열 중 가장 긴 것의 길이만 찾으면 됩니다.
- **서브태스크 3**에서는 이를 확장하여 **K 문자열**, **S 문자열**, **A 문자열** 각각에 해당하는 X 의 부분수열 중 가장 긴 것에 대해 답을 계산하면 됩니다.
- **K 문자열**의 최장 길이를 N 이라고 하면 정답은 $2(|X| - N)$ 입니다.
- 대신, **S 문자열**이나 **A 문자열**은 앞에 몇 개의 문자를 더 붙여야 하므로 예외처리가 필요합니다.
- 이외에도 X 앞에 K와 KS를 붙인 후 **K 문자열**의 최장 길이를 구하는 풀이가 있습니다.

C. 당근 클릭 게임

dp
출제진 의도 – **어려워요**

- ✓ 제출 434번, 정답 89명 (정답률 21.429%)
- ✓ 처음 푼 사람: **yijw0930**, 12분
- ✓ 출제자: eric00513

C. 당근 클릭 게임

- **서브태스크 1**에서 $K \leq 2$이므로 스피드 효과를 아예 구매하지 않는 것이 최선입니다. 따라서 답은 K입니다.
- **서브태스크 2**는 매초 가능한 $K+1$가지의 선택을 모두 시도해보면 브루트포스 알고리즘으로 해결할 수 있습니다.
- **서브태스크 4**에서는 $dp[i][j]$를 "i초에 s의 값이 j일 때 가질 수 있는 당근의 최대 개수"로 정의합시다.
- 스피드 효과를 구매하지 않고 한 번 클릭하면 $dp[i+1][j] = dp[i][j] + j$입니다.
- 반면, k번째 스피드 효과를 구매하면 $dp[i+1][j+B_k] = dp[i][j] - A_k$입니다.
- 위와 같이 dp 배열을 계산하면 $\mathcal{O}\left(NK^2 \cdot \max(B)\right)$에 해결할 수 있습니다.

D. 기숙사 비밀번호 구하기

`constructive`, `linear_algebra`, `modular_multiplicative_inverse`
출제진 의도 – **어려워요**

- 제출 283번, 정답 46명 (정답률 17.314%)
- 처음 푼 사람: **potato167**, 23분
- 출제자: davidkim0

D. 기숙사 비밀번호 구하기

- 문제가 인터랙터 이슈로 사라질 뻔 했으나 kiwiyou님의 도움으로 생존했습니다.
- 모든 N에 대하여 계수의 행렬의 행렬식이 0이 아니게 설계를 하면 가우스 소거법으로도 할 수 있으나, 이 경우 $\mathcal{O}\left(N^3\right)$으로 **서브태스크 1**을 해결할 수 있습니다.
- 만일 계수행렬이 단순하게 **항등행렬(identity matrix)**이라면, 계수만 받으면 되기에 $\mathcal{O}\left(N^2\right)$ 안에 해결은 가능하지만, 이 경우 문제에서 원하는 계수의 조건을 충족하지 못합니다.
- 따라서 구성적으로 두 식의 차이 등으로 한 문자를 구하는 방법을 생각해 보아야 합니다.
- 이후부터는 모듈로 곱셈 역원 등의 방법을 통해 직접 계산을 구현하면 되며, 가능한 방법이 다양하기에 검수자 한 분의 풀이를 예시로 제시하겠습니다.

D. 기숙사 비밀번호 구하기

- i 번째 줄(1 부터 시작)의 경우, x_j 의 계수를 $2(i-1)+2n(j-1)$ 으로 설정합니다.
- 이렇게 설정한 행렬에 대하여, $i(i \geq 3)$ 번째 줄의 x_i 계수를 1 만큼 늘립니다. 이렇게 할 시 계수 행렬의 성분들이 서로 겹치지 않습니다.
- 첫 번째 줄 과 두 번째 줄의 결과의 차이가 $\left(\sum_{i=2}^{n} 2x_i\right) \bmod 998\,244\,353$ 임을 알 수 있습니다. 이를 바탕으로, x_3 부터 x_n 까지의 미지수의 값을 알아낼 수 있습니다.
- 첫 번째와 두 번째 미지수의 경우 $x_3, x_4, \cdots, x_n$ 을 소거한 후, 간단한 연립방정식을 해결하여 구할 수 있습니다.
- 계수를 우변으로 넘겨줄 때(이항) 모듈로 역원을 사용하면 되며, $998\,244\,353$ 이 소수이므로 페르마의 소정리를 사용하여 정리해 주면 됩니다.

E. 시험공부 계획 세우기

`ad_hoc`, `greedy`, `math`
출제진 의도 – **어려워요**

- 제출 109번, 정답 27명 (정답률 24.771%)
- 처음 푼 사람: **potato167**, 32분
- 출제자: aaron0526

- $A_i = j (i \neq j)$ 일 때 i 에서 j 로 가는 간선이 있는 그래프를 생각합시다. 그래프의 모든 사이클 $(c_1, \cdots, c_t)$ 에 대해 $s_{c_1} = \cdots = s_{c_t}$ 가 되어야 함을 알 수 있습니다. 적절한 방법을 통해 사이클을 추출한 후 사이클 내의 정점들을 가중치를 합하여 하나의 정점으로 관리합니다.
- 새롭게 만들어지는 그래프는 사이클이 없고 모든 정점의 진출 차수가 0 또는 1 인 그래프입니다. 즉, 진출 차수가 0 인 정점들을 일종의 루트라고 볼 때, 이 그래프는 Forest(여러 개의 분리된 트리로 구성된 그래프)입니다.
- 각 정점 v 에 대해, v 의 서브트리의 모든 가중치의 합을 $W[v]$ 라 둡니다. 본 문제는 그래프에 존재하는 정점들에 대해 $W[v]$ 값을 계산한 후, $W[v]$-가중치 합이 X 이상 Y 이하인 모든 양의 정수쌍의 개수를 세는 문제로 환원됩니다. 따라서 이 문제는 $\mathcal{O}(NK)$ DP로 해결할 수 있습니다.

F. 냥냥이와 삼색이

`math, implementation, case_work`
출제진 의도 – **어려워요**

- ✓ 제출 122번, 정답 15명 (정답률 13.115%)
- ✓ 처음 푼 사람: **cki86201**, 104분
- ✓ 출제자: hwpark12

F. 냥냥이와 삼색이

- 크사냥이는 **귀엽습니다.**
- **서브태스크 1**, **서브태스크 2**는 각각 $\mathcal{O}\left(N^3\right), \mathcal{O}\left(N^2\right)$ BFS로 해결할 수 있습니다.
- **서브태스크 1**은 가능한 냥냥이와 삼색이의 위치 N^2 개를 모두 관리하는 visited 배열에서 각 위치에 도달하기까지 지민이의 이동 경로 전체를 저장하면 $\mathcal{O}\left(N^3\right)$ 에 해결 가능합니다.
- **서브태스크 2**는 **서브태스크 1**에서 지민이의 이동 경로 전체가 아닌 각 위치의 이전 위치와 그 때 지민이가 이동한 점을 저장하면 역추적해 지민이의 전체 이동 경로를 $\mathcal{O}\left(N^2\right)$ 에 구할 수 있습니다.
- 나머지 서브태스크의 경우 $\mathcal{O}\left(N\right)$ 풀이의 아이디어에 따라 구현 방법이 크게 차이날 수 있습니다. 여기선 출제자의 풀이를 소개합니다.
- 설명을 위해 현재 냥냥이와 삼색이의 위치는 처음 위치인 A, B와 구분해 a, b로 표기합니다.

- **서브태스크 3**의 경우 다음과 같이 항상 원하는 구도를 만족시킬 수 있습니다.
- 0 또는 N 으로 이동해 $1 \leq |a-b| \leq 2$를 만족시킨 후 $|a-b|=1$인 경우 a로 이동해 $|a-b|=2$이도록 만듭니다.
- $\min(C,D) \leq a, b \leq \max(C,D)$ 이며, $a=C$ 또는 $b=C$이도록 만듭니다. 이는 $(a+b)/2$로 이동해 a에서 b방향으로 고양이들을 평행이동하는 방법을 사용합니다. 필요하다면 미리 0 또는 N 으로 이동해 a와 b를 바꿔 평행이동할 방향을 맞춰줘야 합니다.
- $b=C$라면 0 또는 N 으로 이동해 a와 b를 바꿔 $a=C$이도록 만듭니다. 이제 a로 이동해 $b=D$로 만들면 원하는 구도가 완성됩니다.

- **서브태스크 4**부터는 원하는 구도를 만족할 수 없는 경우가 등장합니다.
- $C = D$인 경우
  - C와 D가 1 또는 $N - 1$인 경우, $(A, B) = (C, D)$가 아니면 만족할 수 없습니다.
  - 이외의 경우 $(a, b) = (C + 1, D - 1)$이도록 **서브태스크 3**의 방법으로 이동한 후 0으로 이동해 원하는 구도를 완성합니다.
- $|C - D| = 1$인 경우
  - 원하는 구도가 양 끝인 경우, 즉 $C + D$가 3 또는 $2n - 3$인 경우는 $(A, B) = (C, D)$ 또는 $(A, B) = (D, C)$가 아니면 만족할 수 없습니다.
  - 이외의 경우 $C < D$라면 $(a, b) = (C - 1, D + 1)$, $C > D$라면 $(a, b) = (C + 1, D - 1)$이도록 **서브태스크 3**의 방법으로 이동한 후 각각 N, 0으로 이동해 완성합니다.

- **서브태스크 5**는 $|A-B| \leq 2$인 경우 또한 고려해야 합니다. 다음은 **서브태스크 4**에 추가해야 하는 부분들입니다.
- (C, D)가 $(1, 1)$ 또는 $(N-1, N-1)$인 경우 $(A, B) = (C, D)$ 일때, $C+D$가 3 또는 $2n-3$인 경우 $(A, B) = (C, D)$ 또는 $(A, B) = (D, C)$ 일때 만족할 수 있음을 처리해줍니다.
- $A = B$인 경우
  - A와 B가 1 또는 $N-1$인 경우 만족할 수 없습니다.
  - 0으로 이동해 $(a, b) = (A-1, B+1)$으로 만든 후 **서브태스크 4**의 방법으로 완성합니다.

- ✓ $|A-B|=1$인 경우
  - $A+B$가 3 또는 $2n-3$인 경우 다음 과정을 거쳐 **서브태스크 4**의 방법으로 완성합니다.
    1. $(A,B)=(2,1)$이라면 0으로, $(A,B)=(n-2,n-1)$이라면 N으로 이동합니다.
    2. $(A,B)=(1,2)$이거나 $(A,B)=(n-1,n-2)$이라면 1로 이동합니다.
  - 이외의 경우 $A<B$라면 0으로 이동해 $(a,b)=(A-1,B+1)$, $A>B$라면 N으로 이동해 $(a,b)=(A+1,B-1)$이도록 만든 후 **서브태스크 4**의 방법으로 완성합니다.

F. 냥냥이와 삼색이



- 총 이동 횟수에 유의미한 영향을 미치는 과정들은 모두 **서브태스크 3**의 경우에 있고, 순서대로 다음과 같습니다.
  1. $1 \leq |a - b| \leq 2$이도록 만들 때
  2. $\min(C, D) \leq a, b \leq \max(C, D)$이며, $a = C$ 또는 $b = C$이도록 만들 때
  3. a와 b를 바꿀 때
  4. $b = D$로 만들 때
- 1번과 2번 과정에서 냥냥이나 삼색이 중 하나는 한 방향으로만 이동하고, 3번과 4번 과정에선 삼색이가 한 방향으로만 이동합니다. 따라서 위 모든 과정의 이동 횟수는 $2N$ 이하입니다.
- 이외의 나머지 과정들은 이동 횟수를 모두 합쳐도 N 이하이기 때문에 총 이동 횟수는 항상 $3N$ 이하이고, 시간복잡도는 $\mathcal{O}(N)$입니다.

G. 사인은 B0

`math, combinatorics`
출제진 의도 – **어려워요**

- 제출 132번, 정답 21명 (정답률 15.909%)
- 처음 푼 사람: **sharaelong**, 34분
- 출제자: davidkim0

G. 사인은 B0

- 한별이는 **귀엽습니다.**
- 직접 $\binom{n}{i}$ 의 값을 구하기에는 N 이 너무 크기에, 비교를 할 다른 방법을 고안해 보아야 합니다.
- 양수 A, B 에 대해 $A \geq B$ 는 $A/B \geq 1$ 과 동일합니다. 이를 이용하여 비교를 간편하게 진행할 수 있습니다.
- 일반적으로 $(ax + by + c)^n$ 에서 $x^i y^j$ 의 계수는 $a^i b^j c^{n-i-j} \times \binom{n}{i}\binom{n-i}{j}$ 이 됨을 알 수 있습니다. (물론, $i + j \leq n$)
- $M(i, j)$ 에서 $M(i + 1, j)$ 로 이동하려면 $i + j < n$ 이면서 $\frac{a}{c} \times \frac{n-i-j}{i+1} \geq 1$ 이여야 합니다.
- $M(i, j)$ 에서 $M(i, j + 1)$ 로 이동하려면 $i + j < n$ 이면서 $\frac{b}{c} \times \frac{n-i-j}{j+1} \geq 1$ 이여야 합니다.

- 즉 크기관계를 알기 위해 큰 수의 값을 구하지 않고 $\mathcal{O}(1)$ 안에 인접한 두 수를 비교할 수 있습니다.
- 가장 기본적인 아이디어는, 이동가능한 모든 $M(i,j)$에 대해 최단 경로 길이를 찾아 해결하는 것입니다. DP를 사용하여 $M(i,j)$으로 이동하기 위해서는 반드시 $M(i,j-1)$와 $M(i-1,j)$로 이동해야 하는 점을 고려하면 됩니다. 이 경우 시간복잡도는 $\mathcal{O}\left(TN^2\right)$ 입니다.
- 시간복잡도를 더 줄이기 위해서는, 탐색 자체에 소요되는 시간을 줄여야 합니다. 그런데 **'ㄱ'자 혹은 'ㄴ'자 모양의 경로가 최적임**을 아래와 같이 증명할 수 있습니다.

G. 사인은 B0

- $M(i,j)$ 에서 $M(i+1,j)$ 로 이동하기 위한 조건을 i 에 대해서 표현하자면 $i \leq \frac{an-aj-c}{a+c}$ 가 됩니다. 즉, (정수형으로 바꾸기 전의) 최댓값은 j 가 1 씩 커질수록 $\frac{a}{a+c}$ 씩 감소함을 관찰할 수 있습니다.
- 마찬가지로 $M(i,j)$ 에서 $M(i,j+1)$ 로 이동하기 위한 조건을 j 로 나타내자면 i 가 1 씩 커질수록 정수형으로 바꾸기 전 최댓값이 $\frac{b}{b+c}$ 씩 감소함을 관찰할 수 있습니다.
- 간단한 그래프를 그려 $M(i,j)$ 의 가능한 영역 중 $i+j$ 가 최대가 되는 지점을 확인하면, 임의의 점에서 도달하기 위해서 굳이 구불구불하게 갈 필요없이 'ㄱ'자, 'ㄴ'자 모양으로만 이동하면 충분함을 알 수 있습니다.
- 따라서 언제 꺾을 것인지를 x 축, y 축에 대해서 각각 $\mathcal{O}(N)$ 으로 확인해주면 됩니다. 이 경우 시간복잡도는 $\mathcal{O}(TN)$ 이며, **서브태스크 2**를 해결할 수 있습니다.

G. 사인은 B0

- 경로의 길이의 최댓값을 구하는 것이니, $\mathcal{O}(TN)$ 에서 $\mathcal{O}(T \log N)$ 으로 이분탐색을 활용하여 시간을 간단하게 줄일 수 있습니다. 이 경우 **서브태스크 3**까지 풀리지만, 더 나은 시간복잡도가 추가적으로 존재합니다.
- 아까 $\mathcal{O}(TN)$ 알고리즘에 대해 $i+j$ 가 최대가 되는 지점을 확인하면, 첫 번째 가로줄에서 최대한 가서 'ㄱ'자 모양으로 이동하는 것 혹은 첫 번째 세로줄에서 최대한 이동한 후 'ㄴ'자 모양으로 이동하는 것이 최대임을 관찰할 수 있습니다.
- 따라서 이를 바탕으로 첫 번째 가로줄과 첫 번째 세로줄에서 갈 수 있는 최장 경로에 대해, 'ㄱ'자 혹은 'ㄴ'자의 최대를 구하면 됩니다. 이 경우 시간복잡도는 $\mathcal{O}(T)$ 입니다.

H. 선형대수학: 개념과 방법

`constructive`
출제진 의도 – **어려워요**

- ✓ 제출 85번, 정답 4명 (정답률 4.706%)
- ✓ 처음 푼 사람: **physics0523**, 113분
- ✓ 출제자: flakepowders



- ✓ 서술 편의상 사용하는 기호 (a, b, c) 는 아래와 같이 정의됩니다.
  - $c > 0$ 이라면 초항이 a 이고 b 미만의 수들로 구성된 공차가 c 인 등차수열
  - $c < 0$ 이라면 초항이 a 이고 b 초과의 수들로 구성된 공차가 c 인 등차수열
- ✓ $N < 10$ 이거나 $N - 1$ 이 소수라면 답이 존재하지 않습니다.
- ✓ 그 외에는 N 을 6 으로 나눈 나머지에 따라 경우를 나눠서 구성하면 간단합니다.
  - $0, 2, 4$ 중 하나라면 $(N\%4+2,N+1,4),((N+2)\%4+2,N+1,4),(1,N+1,4),(3,N+1,4)$
  - 1 이라면 $((N+1)\%4+2,N-1,4),((N-1)\%4+2,N-1,4),1,(N-2,1,-4),(N-4,1,-4),N,N-1$
  - 3 이라면 $(10,N-5,4),(4,N-5,4),2,N-5,1,(N-2,1,-4),(N-4,1,-4),N,N-1,6,N-3$
  - 5 이라면 $((N-1)\%4+2,N+1,4),((N+1)\%4+2,N+1,4),(1,N+1,4),(3,N+1,4)$

- 다른 풀이로, 조건에 맞는 수열이 꽤 많다는 것을 이용해 앞에서부터 수열을 구성하며 남은 수들 중 조건에 맞는 수들을 무작위로 이어붙이는 과정을 여러 번 시도해도 됩니다.
- 또 다른 풀이로, N 이 홀수더라도 $N-2, N-4, N-6$ 중 적어도 하나는 소수가 아님을 이용해서 구성하는 것도 가능합니다. 쌍둥이 소수 때문에 반례가 생기지 않도록 유의해야 합니다.
- 그 외에도 다양한 풀이들이 있습니다.
- 출제자는 시험 기간에 이런 문제를 생각하다가 결국 선형대수 재수강을 하게 되었습니다.

I. 탐색 게임

`binary_search`, `dp`, `knuth`
출제진 의도 – **어려워요**

- 제출 11번, 정답 0명 (정답률 0%)
- 처음 푼 사람: **대회 중 없음, yijw0930**
- 출제자: aaron0526

- 정답 후보로 $(i, i+1, \cdots, j)$가 있을 때 총 t개의 정수를 입력하여 질문을 1회 한 후, 그 후로부터는 K개의 정수를 입력하며 질문한다고 했을 때의 점수 기댓값을 $f(i,j,t)$라 두고, $g(i,j,t)$를 $(P_i + P_{i+1} + \cdots + P_j) \cdot f(i,j,t)$라 둡니다.
- 최적의 전략 상에서 $(i, i+1, \cdots, j)$가 후보로 남았을 때 t개의 정수를 입력해야 한다면 골라야 하는 정수 중 가장 작은 값으로 가능한 최솟값을 $c(i,j,t)$라 둡니다.
- 이 때 $g(i,j,t) = P_{c(i,j,t)} + (K+1) \cdot g(i, c(i,j,t)-1, K) + g(c(i,j,t)+1, j, t-1)$입니다.
- 서술의 편의를 위해, $g(i,j,0) = (K+1) \cdot g(i,j,K)$이라 두고 $g(i, i-1, t) = 0$이라 정의합니다. 또, $c(i,j,0) = c(i,j,K)$라 둡니다. 그러면 위에서 언급한 점화식은 모든 $1 \le i \le j \le N$과 $1 \le t \le K$에 대해 성립합니다.
- 점화식을 이용해 $c(i,j,t)$로 가능한 모든 값을 시도하면 **서브태스크 2**를 해결할 수 있습니다.

- **서브태스크 3**의 경우 DP 최적화 기법의 일종인 '크누스 최적화' 또는 '분할 정복을 이용한 최적화'를 통해 해결할 수 있습니다. 각각의 방법으로 DP 배열을 채우는 데에 총 $\mathcal{O}\left(N^2K\right)$, $\mathcal{O}\left(N^2K\log N\right)$ 이 드는데, 본 풀이에서는 이 중 전자를 살펴 봅니다.
- $c(i,j-1,t)\le c(i,j,t)\le c(i+1,j,t)$ 임이 확인된다면, $j-i$ 의 크기가 작은 (i,j,t) 쌍부터 함수 g 와 c 값을 채워 나갑니다.
- 이 때 $j-i$ 값이 동일한 위치의 DP값을 전부 계산하는 데에 $\mathcal{O}\left(NK\right)$ 가 듭니다. 따라서 총 시간복잡도는 $\mathcal{O}\left(N^2K\right)$ 입니다.
- 아래는 크누스 최적화를 위해 부등식 $c(i,j-1,t)\le c(i,j,t)\le c(i+1,j,t)$ 을 증명하는 과정입니다.

- ✓ Claim 1: $t \geq 1$일 때 $g(i, j, t-1) \geq g(i, j, t)$
  - $j - i$의 크기에 대한 수학적 귀납법으로 증명하자. $j - i = -1$인 경우 위 식은 자명하다.
  - $j - i < n$일 때 위 식이 성립한다고 가정한 후, $j - i = n$인 임의의 쌍 (i, j, t)을 잡자.
  - $t > 1$인 경우, 첫 선택으로 $t-1$개의 정수를 입력하는 것보다는 t개의 정수를 입력하는 것이 항상 점수를 낮추는 데에 더 유리하므로 $g(i, j, t-1) \geq g(i, j, t)$이다.
  - $t = 1$인 경우, $s = c(i, j, K)$라 뒀을 때, $j - (s+1) \leq j - i + 1 < n$이므로 귀납 가정에 의해 $(K+1) \cdot g(s+1, j, K-1) \geq (K+1) \cdot g(s+1, j, K) = g(s+1, j, 0)$다. 이를 이용하면
    $g(i, j, 0) = (K+1) \cdot g(i, j, K) = (K+1)P_s + (K+1)^2 \cdot g(i, s-1, K) + (K+1) \cdot g(s+1, j, K-1) \geq P_s + (K+1) \cdot g(i, s-1, K) + g(s+1, j, 0) \geq g(i, j, 1)$이다.
  - 따라서 수학적 귀납법의 원리에 의해 $t \geq 1$일 때 $g(i, j, t-1) \geq g(i, j, t)$이다.

- ✓ Claim 2: $i \leq j+1 \leq k+1$일 때 $g(i,k,t) - g(i,j,t) - g(j+1,k,t) \geq 0$
  - $i = j+1$인 경우 $g(i,j,t) + g(j+1,k,t) = g(i,k,t)$이므로 자명하다. $i \leq j \leq k$인 경우에 대해서만 증명하자.
  - $k-i$의 크기에 대한 수학적 귀납법으로 위 주장을 증명할 것이다. $k-i=0$이라면, 자명하다.
  - 이제 $j=k$인 모든 경우에 대해서는 위 식이 성립함을 간단히 확인할 수 있으므로, $j \neq k$인 경우만 고려하여 증명하자. $k-i<n$일 때 위 식이 항상 성립한다고 가정한 후, $k-i=n$인 임의의 쌍 (i,j,k,t)를 잡자.
  - $s = c(i,k,t)$라 두자. t와 s의 값에 따라 아래의 세 가지 케이스로 나눠서 증명한다.

- ✓ Claim 2 Induction - Case 1: $t \neq 0, s \leq j$
  - $g(i,k,t) = P_s + (K+1) \cdot g(i, s-1, K) + g(s+1, k, t-1)$.
  - $g(i,j,t) \leq P_s + (K+1) \cdot g(i, s-1, K) + g(s+1, j, t-1)$.
  - $s+1 \leq j+1 \leq k+1$ 이고 $k-(s+1) < k-i = n$ 이므로 귀납 가정에 의해 $g(s+1,k,t-1) - g(s+1,j,t-1) - g(j+1,k,t-1) \geq 0$.
  - 또한 Claim 1에 의해 $g(j+1,k,t-1) \geq g(j+1,k,t)$.
  - 앞서 얻은 네 개의 부등식을 연립하면 $g(i,k,t) - g(i,j,t) - g(j+1,k,t) \geq 0$ 이다.

- Claim 2 Induction - Case 2: $t \neq 0, s \geq j+1$
  - $g(i,k,t) = P_s + (K+1) \cdot g(i, s-1, K) + g(s+1, k, t-1)$.
  - $g(j+1,k,t) \leq P_s + (K+1) \cdot g(j+1, s-1, K) + g(s+1, k, t-1)$.
  - $i \leq j+1 \leq (s-1)+1$ 이고 $(s-1) - i < s - i \leq k - i = n$ 이므로 귀납 가정에 의해 $(K+1) \cdot g(i, s-1, K) - (K+1) \cdot g(i, j, K) - (K+1) \cdot g(j+1, s-1, K) \geq 0$.
  - Claim 1에 의해 $(K+1) \cdot g(i,j,K) = g(i,j,0) \geq g(i,j,1) \geq \cdots \geq g(i,j,t)$.
  - 앞서 얻은 네 개의 부등식을 연립하면 $g(i,k,t) - g(i,j,t) - g(j+1,k,t) \geq 0$ 이다.

- Claim 2 Induction - Case 3: $t = 0$
  - $K \neq 0$ 이므로 앞의 두 케이스에서 증명한 바에 따라
    $g(i,k,K) - g(i,j,K) - g(j+1,k,K) \geq 0$.
  - 따라서 $g(i,k,t) - g(i,j,t) - g(j+1,k,t) =$
    $(K+1) \cdot g(i,k,K) - (K+1) \cdot g(i,j,K) - (K+1) \cdot g(j+1,k,K) \geq 0$ 이다.
- 앞서 살펴본 세 가지 케이스에 대해 모두 증명하고자 하는 식이 성립한다. 따라서 수학적 귀납법의 원리에 의해 본 Claim은 참이다.

- Claim 3: $i \le j \le k+1 \le l+1$일 때 $g(i,l,t)+g(j,k,t)-g(i,k,t)-g(j,l,t) \ge 0$
  - $j = k+1$인 경우 $g(j,k,t)=0$이고 $i \le k+1 \le l+1$에서 Claim 2에 의해 $g(i,l,t)-g(i,k,t)-g(k+1,l,t) \ge 0$이므로 성립. 따라서 $i \le j \le k \le l$인 경우에 대해서만 증명하자.
  - $l-i$의 크기에 대한 귀납법으로 증명하자. $l-i=0$이라면 $i=j=k=l$이므로 자명하다.
  - $l-i<n$일 때 위 식이 성립한다고 가정한 후, $l-i=n$인 임의의 쌍 (i,j,k,l,t)를 잡자. $c_1=c(i,l,t), c_2=c(j,k,t)$라 두자. 이 때 다음의 두 식이 성립한다.
  - c_1, c_2와 t의 값에 따라 아래의 세 가지 케이스로 나눠서 증명한다.

- ✓ Claim 3 Induction - Case 1: $t \neq 0, c_1 \leq c_2$
  - $g(i,l,t) = P_{c_1} + (K+1) \cdot g(i, c_1 - 1, K) + g(c_1 + 1, l, t-1)$
  - $g(j,k,t) = P_{c_2} + (K+1) \cdot g(j, c_2 - 1, K) + g(c_2 + 1, k, t-1)$
  - $i \leq c_1 \leq k$ 이므로 $g(i,k,t) \leq P_{c_1} + (K+1) \cdot g(i, c_1 - 1, K) + g(c_1 + 1, k, t-1)$ 이다.
    비슷하게 $g(j,l,t) \leq P_{c_2} + (K+1) \cdot g(j, c_2 - 1, K) + g(c_2 + 1, l, t-1)$ 이다.
  - $c_1 + 1 \leq c_2 + 1 \leq k + 1 \leq l + 1, l - (c_1 + 1) < l - i = n$ 이므로 귀납 가정에 의해
    $g(c_1 + 1, l, t-1) + g(c_2 + 1, k, t-1) - g(c_1 + 1, k, t-1) - g(c_2 + 1, l, t-1) \geq 0$
  - 앞서 얻은 다섯 개의 부등식을 연립하면 $g(i,l,t) + g(j,k,t) - g(i,k,t) - g(j,l,t) \geq 0$
    이다.

- ✓ Claim 3 Induction - Case 2: $t \neq 0, c_1 > c_2$
  - $g(i,l,t) = P_{c_1} + (K+1) \cdot g(i, c_1 - 1, K) + g(c_1 + 1, l, t-1)$
  - $g(j,k,t) = P_{c_2} + (K+1) \cdot g(j, c_2 - 1, K) + g(c_2 + 1, k, t-1)$
  - $i \leq c_2 \leq k$ 이므로 $g(i,k,t) \leq P_{c_2} + (K+1) \cdot g(i, c_2 - 1, K) + g(c_2 + 1, k, t-1)$ 이다. 비슷하게 $g(j,l,t) \leq P_{c_1} + (K+1) \cdot g(j, c_1 - 1, K) + g(c_1 + 1, l, t-1)$ 이다.
  - $i \leq j \leq (c_2 - 1) + 1 \leq (c_1 - 1) + 1, l - (c_2 + 1) < l - i = n$ 이므로 귀납 가정에 의해 $g(i, c_1 - 1, K) + g(j, c_2 - 1, K) - g(i, c_2 - 1, K) - g(j, c_1 - 1, K) \geq 0$
  - 앞서 얻은 다섯 개의 부등식을 연립하면 $g(i,l,t) + g(j,k,t) - g(i,k,t) - g(j,l,t) \geq 0$ 이다.

- Claim 3 Induction - Case 3: $t = 0$
  - $K \neq 0$ 이므로 앞의 두 케이스에서 증명한 바에 따라 $g(i,l,K) + g(j,k,K) - g(i,k,K) - g(j,l,K) \geq 0$.
  - 따라서 $g(i,l,t) + g(j,k,t) - g(i,k,t) - g(j,l,t) = (K+1)\cdot g(i,l,K) + (K+1)\cdot g(j,k,K) - (K+1)\cdot g(i,k,K) - (K+1)\cdot g(j,l,K) \geq 0$ 이므로 증명이 완료된다.
- 앞서 살펴본 세 가지 케이스에 대해 모두 증명하고자 하는 식이 성립한다. 따라서 수학적 귀납법의 원리에 의해 본 Claim은 참이다.

- ✓ Claim 4-1: $i \leq j$ 이고 $t \geq 1$ 일 때 $c(i,j,t) \leq c(i+1,j,t)$
  - $c(i,j,t) = q, c(i+1,j,t) = p$ 라 두고, $p < q$ 이라 가정하자.
  - $i \leq (i+1) \leq (p-1)+1 \leq (q-1)+1$ 이므로 Claim 3에 의해 $g(i,q-1,K) + g(i+1,p-1,K) - g(i,p-1,K) - g(i+1,q-1,K) \geq 0$ 이다.
  - $P_p + (K+1) \cdot g(i,p-1,K) + g(p+1,j,t-1) > P_q + (K+1) \cdot g(i,q-1,K) + g(q+1,j,t-1)$ 이다. 여기에서 $p < q$ 이므로 등호가 없음에 유의하자.
  - 두 부등식을 연립하면 $P_p + (K+1) \cdot g(i+1,p-1,K) + g(p+1,j,t-1) > P_q + (K+1) \cdot g(i+1,q-1,K) + g(q+1,j,t-1)$ 이다. 이는 $c(i+1,j+1,t)$ 가 최적의 선택이 아니라는 뜻이므로 모순이다. 따라서 $p \geq q$.

- ✓ Claim 4-2: $i \le j$ 이고 $t \ge 1$ 일 때, $c(i, j-1, t) \le c(i, j, t)$
  - $c(i, j-1, t) = r$, $c(i, j, t) = q$ 라 두고, $q < r$ 이라 가정하자.
  - $q+1 \le r+1 \le (j-1)+1 \le j+1$ 이므로 Claim 3에 의해 $g(q+1, j, t-1) + g(r+1, j-1, t-1) - g(q+1, j-1, t-1) - g(r+1, j, t-1) \ge 0$ 이다.
  - $P_q + (K+1) \cdot g(i, q-1, K) + g(q+1, j-1, t-1) > P_r + (K+1) \cdot g(i, r-1, K) + g(r+1, j-1, t-1)$ 이다. 여기에서 $q < r$ 이므로 등호가 없음에 유의하자.
  - 두 부등식을 연립하면 $P_q + (K+1) \cdot g(i, q-1, K) + g(q+1, j, t-1) > P_r + (K+1) \cdot g(i, r-1, K) + g(r+1, j, t-1)$ 이다. 이는 $c(i, j, t)$ 가 최적의 선택이 아니라는 뜻이므로 모순이다. 따라서 $q \ge r$.

J. 히스토그램에서 가장 큰 직사각형과 쿼리 2

`dp`, `cht`, `segtree`, `stack`
출제진 의도 – **어려워요**

- 제출 42번, 정답 4명 (정답률 9.524%)
- 처음 푼 사람: **moonrabbit2**, 4분
- 출제자: aaron0526

- **서브태스크 1**의 경우, 잘 알려진 스택 사용 기법으로 쉽게 해결 가능합니다. 히스토그램에서 가장 큰 직사각형 (6549) 문제를 참조하세요.
- **서브태스크 2**의 경우, 답이 되는 직사각형의 오른쪽 끝 막대가 항상 r 번째 막대임을 이용합니다. l 번째 막대를 왼쪽 끝 막대로 두었을 때 직사각형의 넓이는 $H_l \cdot (x - l + 1)$ 꼴로 표현됩니다. 왼쪽 막대가 어떤 지점에 있어야 넓이가 최대가 되는지를 Convex Hull Trick으로 쉽게 계산할 수 있습니다.

- **서브태스크 3**은 세그먼트 트리와 CHT를 함께 활용하여 해결할 수 있습니다. 편의를 위해 $H_{i,j} = \min(H_i, H_{i+1}, \cdots, H_j)$라 정의합니다. 각 쿼리 (l, r)에 대해, 정답이 되는 직사각형 $\langle a, b \rangle$은 반드시 다음 중 적어도 하나의 성질을 갖습니다.
  1. $a = l, b = r$
  2. $a > l, H_{a-1} < H_{a,b}$
  3. $b < r, H_{b+1} < H_{a,b}$
- 1.의 경우 $H_{i,j}$를 빠르게 계산하는 세그먼트 트리를 제작하여 계산할 수 있습니다.
- 3.은 2.와 동일한 구조이므로 2.에 속하는 직사각형만 고려한 후, 뒤집어서 똑같은 과정을 한번 더 반복하면 됩니다.

J. 히스토그램에서 가장 큰 직사각형과 쿼리 2



- ✓ 쿼리를 l 내림차순으로 정렬하여 관리합니다. 배열 B 를 만든 후 오른쪽 막대기부터 스택에 삽입하며 스택을 관리합니다.
- ✓ 막대 l 보다 높은 막대들을 스택에서 삭제하고 막대 l 을 스택에 삽입한 후, 쿼리 (l, r) 의 답을 B_r 로 업데이트하는 과정을 반복합니다.
- ✓ 스택에서 원소가 하나 제거될 때마다 제거된 원소를 p, 스택의 Top에 남은 원소를 q 라 하면 다음과 같이 배열 B 를 업데이트합니다.
  - $B_r = \max\left(B_r, \begin{cases} H_p \cdot (r-l) & l < r < q \\ H_p \cdot (q-l-1) & r \geq q \end{cases}\right)$
- ✓ 위와 같은 업데이트는 리차오 트리를 통해 $\mathcal{O}\left(N \log^2 N\right)$ 에 해결할 수 있습니다.
- ✓ 본 문제에 대한 시간복잡도 $\mathcal{O}\left(N \log N\right)$ 풀이도 존재하니 도전해 보세요.

K. 좋은 수열

`segtree`, `lazyprop`
출제진 의도 – **어려워요**

- 제출 7번, 정답 0명 (정답률 0%)
- 처음 푼 사람: **대회 중 없음, potato167**
- 출제자: mickeyjung

- 먼저 A에 존재하는 0을 모두 -1로 치환했을때 A_1 부터 A_i까지의 합을 S_i라고 정의합시다.
- A에 N이 존재하게 하기 위해서는 1들을 총 $N-1$번 합쳐야 하기 때문에 x, y, z, w 중 삭제되는 z는 무조건 0이어야 하고, 합쳐지는 x와 y는 무조건 각각 1 이상의 수여야 합니다.
- 따라서 설명의 편의상 문제를 다음과 같은 2가지 작업만을 시행하여 A의 길이를 2까지 줄이는 문제로 변형하도록 하겠습니다.
- $1\,1\,0\,0 \rightarrow 0\,1$
- $1\,1\,0\,1 \rightarrow 1\,1$

K. 좋은 수열

- 결론부터 말하자면 어떤 수열 A가 좋은 수열일 조건은 $1 \leq i \leq 2N-1$인 모든 i에 대해서 $S_i \geq 1$이어야 하고, 만약 $2 \leq i \leq 2N-1$인 모든 i에 대해 $S_i = 1$이라면 그때까지 연속한 0의 개수가 홀수 개여야 한다는 것입니다.
- 이제부터 두 가지 조건에 대한 증명을 하겠습니다.
- 첫 번째, $1 \leq i \leq 2N-1$인 모든 i에 대해서 $S_i \geq 1$이어야 합니다.
- 문제에서 주어진 작업이 $i, i+1, i+2, i+3$번 인덱스에 대해서 시행된다고 했을 때 X를 S_{i-1}로 정의합시다.
- 그러면 두 가지의 작업에 대해서 배열 S의 변화는 다음과 같습니다.
- $X+1, X+2, X+1, X \rightarrow X-1, X$
- $X+1, X+2, X+1, X+2 \rightarrow X+1, X+2$

- 즉, 마지막을 제외한 인덱스들에서 배열 S의 최솟값이 커지는 경우는 없다는 사실을 알 수 있습니다.
- $L = 2$일 때 가능한 경우는 $1\,1\,0\,0$으로 한가지 뿐이고, 이때의 배열 S 값을 살펴보면 $1\,2\,1\,0$입니다.
- 마지막 인덱스를 제외하면 $L = 2$일 때 S의 최솟값은 1이므로 처음 수열에서 마지막 인덱스를 제외한 S의 최솟값이 1 미만이라면 절대 $1\,1\,0\,0$의 상태에 도달할 수 없습니다.

K. 좋은 수열



- 두 번째, $2 \leq i \leq 2N-1$인 모든 i에 대해서 $S_i = 1$이라면 그 때까지 연속한 0의 개수는 홀수 개여야 합니다.
- 증명에 앞서, 어떤 수열에서 아무리 작업을 시행해도 오른쪽에서부터 연속하는 0의 개수의 기우성은 변하지 않는다는 사실을 관찰할 수 있습니다.
- 만약 $S_i = 1$인 어떤 인덱스 i가 존재하고, 그 때까지 연속한 0의 개수가 짝수 개라고 가정해봅시다.
- 그런 인덱스 i를 기준으로 수열 A를 쪼갭니다.
- $A_1, A_2, \cdots A_i$를 L, $A_{i+1}, A_{i+2}, \cdots A_{2N}$을 R이라고 하면 L과 R 사이에 걸치는 작업, 즉 L과 R에 동시에 영향을 미치는 작업이 있기 전까지는 모든 작업이 두 부분수열에서 독립적으로 시행됩니다.

- 또한, 위에서 말한 성질에 따라 L과 R 사이에 걸치는 작업이 시행되기 전까지 L의 오른쪽에서부터 연속하는 0의 개수는 항상 짝수로 유지됩니다.
- 그런데 L의 오른쪽에서부터 연속하는 0의 개수가 0인 경우, 즉 L의 가장 오른쪽 수가 1인 경우는 1번 조건에 위배되므로 불가능합니다.
- 따라서 L의 오른쪽에는 항상 0이 2개 이상 존재하게 되고, 이 경우 L과 R 사이에 걸치는 작업을 시행할 수 없음을 알 수 있습니다.
- L과 R 사이에 걸치는 작업이 시행되지 않는다면 L과 R 각각의 길이는 2까지 밖에 작아질 수 없고, 전체 길이는 4까지 밖에 작아질 수 없습니다.
- 결론적으로 L의 오른쪽에서부터 연속하는 0의 개수가 짝수인 경우는 길이를 2로 만드는 것이 불가능합니다.

K. 좋은 수열



- ✓ $1, 2$번 조건에 위배되지 않는 수열에 대해서는 가능한 앞에서부터 작업을 시행하면 항상 길이를 2까지 줄일 수 있음을 증명할 수 있습니다. (간단한 구성적 해도 존재합니다.)
- ✓ 이제 조건을 알았으니 쿼리를 어떻게 적용시킬 수 있을까요?



- **서브태스크 2**에서는 세그먼트 트리를 이용해 각 구간에서 필요한 정보들을 노드에 담는 방법을 사용할 수 있습니다.
- 어떤 구간에서 아래 정보들을 저장하면 그 정보들을 이용해 Merge를 할 수 있고 한 점에 대한 Update를 수행할 수 있습니다.
  - 전체 합
  - 가장 작은 Prefix Sum
  - 가장 작은 Prefix Sum 위치에서 0이 짝수 번 연속한 적이 있는지 여부
  - 가장 왼쪽 수와 그 수가 연속한 길이
  - 가장 오른쪽 수와 그 수가 연속한 길이



- **서브태스크 3**에서는 **서브태스크 2**에 아래 정보들을 추가로 저장한 후, Lazy Propagation을 이용해 구간 Update를 수행할 수 있습니다.
  - 가장 큰 Prefix Sum
  - 가장 큰 Prefix Sum 위치에서 1이 짝수 번 연속한 적이 있는지 여부