2024 SCSC 프로그래밍 경시대회 풀이

Official Solution

by

2024 SCPC 출제진 및 검수진

2024년 05월 25일

문제		의도한 난이도	출제자
2A	정육면체의 네 꼭짓점	**Easy**	sjhi00
2B/1A	로그프레소 마에스트로	**Easy**	sungjae0506
2C	죽음의 등굣길	**Easy**	ksi4495
2D	사다리 게임 만들기	**Medium**	sjh1224
2E	문자열 지우기	**Medium**	ksi4495
2F/1B	현대모비스 트럭 군집주행	**Medium**	sungjae0506
2G/1E	1D 게임	**Hard**	ohwphil
2H/1G	삼진논리 OR과 쿼리	**Hard**	sjhi00
1C	멀티버스를 여행하는 성재를 위한 안내서	**Challenging**	sungjae0506
1D	농지 나누어 갖기	**Hard**	sjhi00
1F	멋진 연결 요소와 쿼리	**Hard**	sungjae0506
1H	스시스시 아일랜드	**Hard**	mujigae

2A. 정육면체의 네 꼭짓점

`implementation`
출제진 의도 – **Easy**

- 제출 172번, 정답 94명 (정답률 55.233%)
- 처음 푼 사람: **양창석**, 1분
- 출제자: `sjhi00`

2A. 정육면체의 네 꼭짓점

- 풀이 1: 입력으로 주어진 네 정수를 오름차순으로 정렬하고 순서대로 이어 붙인 문자열이 0123, 4567, 0246, 1357, 0145, 2367 중 하나이면 답이 YES 이고, 그렇지 않으면 답이 NO 입니다.
- 풀이 2: 입력으로 주어진 네 정수 a, b, c, d에 대하여, a, b, c, d를 bitwise AND한 값이 0이 아니거나 a, b, c, d를 bitwise OR한 값이 7이 아니면 답이 YES 이고, 그렇지 않으면 답이 NO 입니다.
  - a, b, c, d를 bitwise AND한 값이 0이 아닐 필요충분조건은, 네 꼭짓점이 모두 평면 $x = 1$ 위에 있거나 모두 평면 $y = 1$ 위에 있거나 모두 평면 $z = 1$ 위에 있다는 것입니다.
  - a, b, c, d를 bitwise OR한 값이 7이 아닐 필요충분조건은, 네 꼭짓점이 모두 평면 $x = 0$ 위에 있거나 모두 평면 $y = 0$ 위에 있거나 모두 평면 $z = 0$ 위에 있다는 것입니다.
- 시간복잡도는 $O(T)$ 입니다. (T: 테스트 케이스의 개수)

2B/1A. 로그프레소 마에스트로

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

- 제출 441번, 정답 56명 (정답률 12.698%) (Div.2)
- 제출 54번, 정답 30명 (정답률 55.556%) (Div.1)
- 처음 푼 사람: **양창석**, 3분 (Div.2) / **김동현**, 2분 (Div.1)
- 출제자: `sungjae0506`

2B/1A. 로그프레소 마에스트로

SNU Computer Study Club

- N, M 의 제한이 각각 $1000, 10000$ 이므로, 1 번부터 N 번 컴퓨터가 감염된 경우를 각각 시뮬레이션 해볼 수 있습니다.
- 먼저 모든 로그를 시간 순으로 정렬합니다.
- 정렬된 로그를 시뮬레이션 한 결과가 입력으로 받은 감염된 컴퓨터 목록과 같게 만드는 컴퓨터 번호를 찾습니다.
- 시간복잡도는 정렬에서 $O(M \log M)$, 시뮬레이션에서 $O(NM)$ 입니다.

2C. 죽음의 등굣길

`bfs, implementation`
출제진 의도 – **Easy**

- 제출 238번, 정답 82명 (정답률 34.454%)
- 처음 푼 사람: **박종경**, 11분
- 출제자: `ksi4495`

SNU Computer Study Club

- 각 보도블록을 노드로 하는 그래프에서 맨해튼 거리가 X 이하이고 색이 동일한 보드블럭들을 간선으로 연결합니다.
- $(1, 1)$ 에서 출발하여 (N, M) 으로 이동할 수 있는지 판별하면 됩니다.
- BFS로 구현했을 때 시간복잡도는 $O(NMX^2)$ 입니다.

- 별해: 출발 지점과 색이 동일한 노드를 모두 저장한 뒤, 노드 사이의 가능한 모든 간선을 연결합니다.
- 출발 지점과 도착 지점이 간선으로 연결되는지 판별하면 됩니다.
- 시간복잡도는 $O(N^2M^2)$ 입니다.
- 별해도 통과할 수 있도록 제한을 설정했습니다.
- 보너스: 원래 제한 조건은 $2 \leq N, M \leq 2\,000$ 이었습니다.

2D. 사다리 게임 만들기

`dynamic programming`
출제진 의도 – **Medium**

- 제출 193번, 정답 41명 (정답률 22.798%)
- 처음 푼 사람: **양창석**, 22분
- 출제자: `sjh1224`

2D. 사다리 게임 만들기

- (WLOG) 시작점에 가까운 가로선부터 순서대로 $1, \ldots, M$ 번째 가로선이라고 합시다.
- i 개의 가로선이 추가된 후에 x 번째 세로선이 1등일 될 확률을 $f(x, i)$ 라고 합시다.
- i 번째 가로선이 $x-1$, x 번째 세로선 사이에 생길 경우 $f(x, i) \leftarrow f(x-1, i-1)$
- i 번째 가로선이 x, $x+1$ 번째 세로선 사이에 생길 경우 $f(x, i) \leftarrow f(x+1, i-1)$
- 그 외의 경우 $f(x, i) \leftarrow f(x, i-1)$

SNU Computer Study Club

- 가로선이 어느 세로선 사이에 생길지 확률을 고려하여 점화식을 세우면 다음과 같습니다.
- $f(x,i) = \frac{1}{N-1} f(x-1, i-1) + \frac{1}{N-1} f(x+1, i-1) + \frac{N-3}{N-1} f(x, i-1)$
  (양쪽 끝의 세로선의 경우 점화식이 달라지는 것에 주의합시다.)
- 점화식을 통해 지환이가 1등이 될 확률을 동적 계획법으로 구할 수 있습니다.
- 시간복잡도는 $O(NM)$ 입니다.

2E. 문자열 지우기

`game`, `dynamic programming`
출제진 의도 – **Medium**

- 제출 168번, 정답 11명 (정답률 6.548%)
- 처음 푼 사람: **김준원**, 43분
- 출제자: `ksi4495`

2E. 문자열 지우기

- 문자열을 노드로 하는 그래프를 생각해봅시다.
- 플레이어가 행동을 할 때마다 문자열의 길이가 줄어들거나 문자열에 존재하는 ?의 개수가 줄어드므로 그래프는 DAG가 됩니다.
- 그래프가 DAG이므로 동적 계획법을 적용할 수 있습니다.

SNU Computer Study Club

- 노드 g와 게임의 결과를 대응하는 함수를 f라고 합시다.
- $f(g) = \begin{cases} 1, & \text{첫 플레이어가 이기는 경우} \\ 0, & \text{첫 플레이어가 지는 경우} \end{cases}$ 일 때
- $f(g) = max_{g'}(1 - f(g'))$ (g' : 행동 후 가능한 다음 노드)인 것을 알 수 있습니다.

- 문자열을 지울 때 앞이나 뒤에서부터만 지워나가기 때문에 행동 후의 문자열을 원래 문자열의 부분 문자열입니다.
- 따라서 현재 노드를 주어진 문자열에서의 시작점과 끝점, 그리고 $?$의 형태로 표현 할 수 있습니다.
- 각 노드를 $dp[l][r][?\text{의 형태}]$로 나타내면 이를 동적 계획법으로 구할 수 있습니다.
- 시간복잡도는 $O(N^2)$ 입니다.

2F/1B. 현대모비스 트럭 군집주행

`dijkstra`

출제진 의도 – **Medium**

- 제출 219번, 정답 23명 (정답률 10.959%) (Div.2)
- 제출 61번, 정답 25명 (정답률 40.984%) (Div.1)
- 처음 푼 사람: **김준원**, 56분 (Div.2) / **박신욱**, 10분 (Div.1)
- 출제자: sungjae0506

2F/1B. 현대모비스 트럭 군집주행

- 문제에서 다른 트럭을 뒤따르는 트럭들은 10%의 운송비 절감이 된다고 했는데 이를 다시 생각하면 가장 선두의 트럭이 더 많은 비용을 내는 것으로 볼 수 있습니다.
- 또한 모든 트럭은 최단 거리로 이동하기 때문에 최단 거리 합의 90%는 경로와 상관없이 공통적으로 지불해야 하는 비용입니다.

2F/1B. 현대모비스 트럭 군집주행

SNU Computer Study Club

- (WLOG) 트럭이 목적지에 도착하는 순서대로 $1, \cdots, N$ 번 트럭이라고 합시다. (1번 트럭은 출발하자 마자 도착하는 것으로 가정합니다.)
- 또한 간선을 이동할 때 어느 트럭이 가장 선두에 서는지는 비용에 영향을 미치지 않으므로 이동한 후에 도착하는 트럭이 가장 선두에 있는 것으로 가정해도 무방합니다.
- x 번 트럭이 도착했을 때 x 번 트럭의 최단 경로상에서 도착 직전 노드를 목적지로 하는 트럭을 y 번 트럭이라고 합시다.
- 가정으로부터 y 번 트럭이 x 번 트럭보다 먼저 목적지에 도착하기 때문에 x 번 트럭은 y 트럭이 도착할 때까지 뒤따라 가다가 y 에서 x 로 움직일 때만 가장 선두에 있게 됩니다.
- 따라서 x 번 트럭은 최단 경로 상의 마지막 간선의 비용의 10% 만큼 추가 비용을 지불하게 됩니다.

- 따라서 최소 비용 경로는 각 트럭이 추가적으로 지불하는 간선의 비용의 합을 최소화하는 경로이므로 다익스트라 알고리즘을 통해 구할 수 있습니다.
- 시간복잡도는 $O((N+M)\log N)$ 입니다.

2G/1E. 1D 게임

`greedy`, `data structures`, `binary search`
출제진 의도 – **Hard**

- 제출 44번, 정답 3명 (정답률 6.818%) (Div.2)
- 제출 43번, 정답 13명 (정답률 30.233%) (Div.1)
- 처음 푼 사람: **김준원**, 124분 (Div.2) / **이진혁**, 58분 (Div.1)
- 출제자: `ohwphil`

- 이 문제에 대한 논의를 하기 전에 용어들을 정의하고 가겠습니다.
- 만약 p, q 번 칸에 모두 영구 발판이 놓여 있고, 이 둘 사이에 다른 영구 발판이 존재하지 않는다면, 이 둘을 "인접하다"고 표현하겠습니다.
- 어떤 영구 발판의 오른쪽에 있으면서 인접한 영구 발판을 "다음 영구 발판"이라고 표현하겠습니다.
- $p < q$ 라고 한다면 $q - p$ 를 p, q 번 칸에 놓인 "두 영구 발판 사이의 간격"(혹은 "발판 간격")이라고 표현하겠습니다.

- 마찬가지로, u, v 번 턴이 모두 위험한 턴이며, 이 둘 사이에 다른 위험한 턴이 없으면, 이 둘을 "인접하다"고 표현하겠습니다.
- 또한 어떤 위험한 턴보다 턴 번호가 높으면서 인접한 위험한 턴을 "다음 위험한 턴"이라고 표현하겠습니다.
- $u < v$라고 한다면 $v - u$를 턴 번호가 u, v번인 "두 위험한 턴 사이의 간격"(혹은 "턴 간격")이라고 표현하겠습니다.

- 본격적으로 풀이를 설명하기에 앞서, 가능한 관찰들을 살펴봅시다.
- 우선 캐릭터는 위험한 턴에는 반드시 영구 발판에 있어야 합니다.
- 따라서 게임에서 승리하기 위해서는 위험한 턴이 오기 전에 캐릭터가 다음 영구 발판에 도달하는 과정이 반복되어야 합니다.
- 그렇기 때문에 최대한 빠르게 다음 영구 발판으로 그리디하게 이동하는 것이 유리하다는 것을 알 수 있습니다.
- 이로부터 왼쪽으로 이동하는 것은 필요없다는 것도 관찰할 수 있습니다.

- 앞에서의 논의를 수식으로 나타내 봅시다.
- 현재 턴의 번호를 t, 다음 위험한 턴의 번호를 t_i, 현재 칸의 번호를 l, 다음 영구 발판의 번호를 l_j로 나타낸다면 다음 영구 발판으로 이동할 수 있는 필요충분조건은 $t_i - t \geq l_j - l$인 것입니다.
- 캐릭터는 승리하기 위해서 중간에 영구 발판을 거치지 않고 $l_2 - l_1, l_3 - l_2, \cdots, l_K - l_{K-1}$ 칸을 이동할 수 있어야 합니다.

- 이제 승리하는 것이 가능할 필요충분조건을 도출해 봅시다.
- 캐릭터는 게임 시작 직후 t_1 턴 동안 다음 영구 발판으로 이동할 기회가 있습니다.
- 그 후 t_1 번 턴부터 $t_1 + T$ 번 턴까지 $t_2 - t_1, t_3 - t_2, \cdots, t_N - t_{N-1}, t_1 + T - t_N$ 턴 동안 다음 영구 발판으로 이동할 기회가 있으며, 이러한 턴 간격이 반복됩니다.
- 만약 어떤 인접한 두 영구 발판 사이의 간격이 모든 인접한 두 위험한 턴 사이의 간격보다 크다면, 캐릭터는 승리할 수 없습니다.
- 그렇지 않다면, 캐릭터는 승리할 수 있습니다.
- 노트: $t_1 < t_1 + T - t_N$ 이 무조건 성립하기 때문에 위에서 t_1 에 대한 언급을 하지 않아도 되는 것입니다.

- $O(NK)$ 풀이: 시뮬레이션
- 그렇다면 처음 t_1 은 예외처리해 주고, 턴 간격인 $t_2 - t_1, t_3 - t_2, \cdots, t_N - t_{N-1}, t_1 + T - t_N$ 를 순환하면서 각 턴 간격마다 도달할 수 있는 가장 오른쪽의 영구 발판으로 이동하는 시뮬레이션 풀이를 생각해볼 수 있습니다.
- 하지만, 이 풀이는 문제를 해결하기에는 너무 느립니다.
- 만약 $t_2 - t_1, t_3 - t_2, \cdots, t_N - t_{N-1}, t_1 + T - t_N$ 중 하나만 매우 크고 나머지가 매우 작으며, $l_2 - l_1, l_3 - l_2, \cdots, l_N - l_{N-1}$ 이 모두 크다면, 하나의 발판 간격을 이동하는 동안 N 번의 턴 간격을 순회하게 되어 총 시간복잡도가 $O(NK)$ 가 되기 때문입니다.

- 이제 시간복잡도를 줄이기 위한 아이디어를 생각해 봅시다.
- 턴 간격을 순회하는 것이 아니라, 현재 지나가야 하는 두 영구 발판 사이의 간격보다 크거나 같으면서, 가장 가까운 미래에 오는 턴 간격을 찾는 것으로 방식을 바꾸어 봅시다.
- 즉, 어떤 배열에서 특정 값보다 크거나 같으면서 탐색하는 범위에서 가장 왼쪽에 있는 원소의 인덱스를 구하면 됩니다.
- 이러한 쿼리당 시간복잡도가 $O(\log N)$ 만큼 걸리는 적절한 자료구조를 이용하면 총 시간복잡도를 $O(K \log N)$ 로 줄일 수 있습니다.
- 여기에서는 세그먼트 트리와, 스파스 테이블(희소 배열)을 이용한 풀이를 소개합니다.

- $O(K \log N)$ 풀이 1: 세그먼트 트리
- 세그먼트 트리에서 $t_2 - t_1, t_3 - t_2, \cdots, t_N - t_{N-1}, t_1 + T - t_N$ 가 담겨 있는 배열의 구간 최댓값을 저장하고, 세그먼트 트리 상에서의 이분 탐색을 이용하면 됩니다.

- $O(K \log N)$ 풀이 2: 스파스 테이블
- 스파스 테이블을 이용해서 문제를 풀 수도 있습니다.
- 현재 턴 간격을 기준으로 최대 2^i 번 이내에 찾아오는 턴 간격들 중 최댓값을 스파스 테이블에 저장해 놓는다면 역시 이분 탐색을 이용하여 문제를 해결할 수 있습니다.

SNU Computer Study Club

- 구현 시 유의사항

- 어떤 턴 간격이 매우 크다면 다음 영구 발판을 지나쳐 그 다음 영구 발판까지 갈 수 있는 가능성이 있기 때문에 이에 유의해야 합니다.
- 턴 간격이 다 지나가기 전에 L 번 칸에 도달하는 경우 경과한 턴 수를 실제보다 크게 계산하지 않도록 유의해야 합니다.

2H/1G. 삼진논리 OR과 쿼리

`dynamic programming`
출제진 의도 – **Hard**

- 제출 48번, 정답 0명 (정답률 0.000%) (Div.2)
- 제출 20번, 정답 2명 (정답률 10.000%) (Div.1)
- 처음 푼 사람: - (Div.2) / **조승현**, 106분 (Div.1)
- 출제자: `sjhi00`

- 2번 쿼리를 선형 시간으로 처리하면 총 $O(M^2)$의 시간복잡도로 시간 초과를 받게 됩니다.
- 따라서 2번 쿼리를 선형 시간보다 빠르게 처리해주어야 합니다.
- 이제 OR_3 연산의 성질에 대해 살펴봅시다.
- OR_3 연산의 결과는 비트 단위로 생각했을 때 상위 비트의 결과가 더 크면 하위 비트 값에 상관없이 최종 결과 값이 더 큽니다.

- OR_3의 연산 결과는 상위 비트에 의해 우선적으로 결정되므로 2번 쿼리에서 주어진 x를 삼진법으로 변환하여 최상위 비트부터 고려해봅시다.
- x의 최상위 비트가 2일 경우 어떤 원소와 OR_3를 하든지 해당 비트는 영향을 받지 않습니다.
- x의 최상위 비트가 1일 경우 해당 비트가 2인 원소가 최댓값의 후보이고, 그런 원소가 없을 경우 해당 비트는 영향을 받지 않습니다.
- x의 한 비트가 0일 경우 해당 비트가 가장 큰 원소가 최댓값의 후보입니다.

- $K = 15$(즉, $0 \le x < 3^K$)라고 하면, 앞에서의 관찰로부터 해당 비트가 $0, 1, 2$인 원소가 집합 S에 존재하는지 $O(1)$에 찾을 수 있다면 2번 쿼리를 $O(K)$에 처리할 수 있게 됩니다.
- $x < 3^K$ 이므로 해당 원소가 존재하는지 배열에 저장해두는 것으로 $O(M)$개의 2번 쿼리를 시간복잡도 $O(KM)$ 안에 처리할 수 있습니다.
- 이제 1번 쿼리를 처리하는 일이 필요합니다.
- 1번 쿼리에서 x의 비트필드의 모든 부분 집합을 배열에 업데이트하면 $O(3^K)$ 또는 $O(2^K)$ 만큼 업데이트 해야 하지만, 이미 배열에 들어있는 원소는 업데이트를 할 필요가 없으므로, DFS(깊이 우선 탐색) 또는 BFS(너비 우선 탐색)의 원리를 이용하면 $O(M)$개의 1번 쿼리를 총 $O(K \cdot 3^K)$ 만큼의 시간으로 처리할 수 있습니다.

2H/1G. 삼진논리 OR과 쿼리

- 각각의 정점이 0부터 $3^K - 1$까지의 서로 다른 정수로 번호가 붙여진, 정점이 3^K개인 그래프를 생각합시다.
- $0 \le x < 3^K$인 x에 대하여, x의 3진법 표기가 $x = \overline{x_m x_{m-1} \cdots x_0}_{(3)}$일 때, $x_i > 0$을 만족하는 모든 $0 \le i \le m$에 대하여 x에서 $\overline{x_m \cdots x_{i+1} 0 x_{i-1} \cdots x_0}_{(3)}$으로 향하는 유향 간선(directed edge)을 만듭니다.
- 이렇게 하면 정점의 개수가 3^K이고 간선의 개수가 $O(K \cdot 3^K)$인 DAG(유향 비순환 그래프, Directed Acyclic Graph)가 만들어집니다.
- 정점 x를 방문할 때, 정점 x에서 출발하여 방문할 수 있는 모든 정점을 DFS 또는 BFS로 탐색합니다. 이때, 각 정점마다 방문 여부를 0으로 초기화된 배열 b에 기록합니다. x번 정점을 방문했을 때 $b[x]$를 1로 갱신합시다.
- 이미 방문한 정점은 더 이상 탐색할 필요가 없습니다.

- 1번 쿼리가 주어질 때마다 정점 x에서 출발하면서 탐색하면, 총 정점의 개수가 3^K이고 간선의 개수가 $O(K \cdot 3^K)$이므로, $O(M)$개의 1번 쿼리를 $O\left(K \cdot 3^K\right)$로 처리할 수 있습니다.
- 이제 2번 쿼리의 처리 과정을 다시 살펴봅시다.
- $x = \overline{x_{K-1} \cdots x_1 x_0}_{(3)}$이라고 합시다. (leading zero가 존재할 수 있습니다.)
- 0번 정점에서 출발하여 이동해봅시다.
- $i = K-1, K-2, \cdots, 1, 0$에 대하여 x의 i번째 비트 x_i를 확인해봅시다. 이때 현재 머물러 있는 정점을 y_{i+1}라고 합시다. 즉, $y_K = 0$입니다.

SNU Computer Study Club

- $x_i = 2$인 경우, 현재 y_{i+1} 번 정점 그대로 머무릅니다. ($y_i = y_{i+1}$)
- $x_i = 1$이고 $b[y_{i+1} + 2 \cdot 3^i] = 1$ 이면, $y_{i+1} + 2 \cdot 3^i$ 번 정점으로 이동합니다. ($y_i = y_{i+1} + 2 \cdot 3^i$)
- $x_i = 1$이고 $b[y_{i+1} + 2 \cdot 3^i] = 0$ 이면, 현재 정점 그대로 머무릅니다. ($y_{i+1} = y_i$)
- $x_i = 0$이고 $b[y_{i+1} + 2 \cdot 3^i] = 1$ 이면, $y_{i+1} + 2 \cdot 3^i$ 번 정점으로 이동합니다. ($y_i = y_{i+1} + 2 \cdot 3^i$)
- $x_i = 0$이고 $b[y_{i+1} + 2 \cdot 3^i] = 0, b[y_{i+1} + 3^i] = 1$ 이면, $y_{i+1} + 3^i$ 번 정점으로 이동합니다. ($y_i = y_{i+1} + 3^i$)
- $x_i = 0$이고 $b[y_{i+1} + 2 \cdot 3^i] = b[y_{i+1} + 3^i] = 0$ 이면, 현재 정점 그대로 머무릅니다. ($y_i = y_{i+1}$)

- 앞에서의 과정을 거친 후, $OR_3\left(x, y_0\right)$ 가 구하려고 하는 답이 됩니다.
- 2번 쿼리는 쿼리 하나 당 $O(K)$에 처리할 수 있으므로, 최종 시간복잡도는 $O(K \cdot 3^K + KM)$ 입니다.

2H/1G. 삼진논리 OR과 쿼리

- 별해: 15자리의 3진수 x를 앞 9자리 x_1과 뒤 6자리 x_2로 끊어서 생각할 수 있습니다.
- 앞 9자리는 상위 비트부터 탐색하는 용도로 사용하고, 뒤 6자리는 브루트포스로 탐색하기 위한 용도로 사용합니다.
- $3^6 \times 3^9$ 크기의 2차원 배열 b와 3^9 크기의 1차원 배열 c가 0으로 초기화되어 있다고 하겠습니다.
- 1번 쿼리에 의하여 업데이트를 할 때, $OR_3(x_1, i) = x_1$을 만족하는 모든 $0 \le i < 3^9$에 대하여, $b[x_2][i]$와 $c[i]$의 값을 1로 갱신합니다.
- 2번 쿼리를 처리할 때, 최상위 비트부터 차례대로 상위 9개의 비트를 확인하여 배열 c를 이용해서 OR_3의 최댓값의 상위 9개의 비트를 구할 수 있습니다.
- 하위 6개의 비트는 2차원 배열 b를 이용하여 시간복잡도가 $O(3^6)$인 브루트포스 알고리즘으로 계산할 수 있습니다.

- OR_3 값을 $O(1)$에 구할 수 있다고 가정할 때, 이 과정을 종합하면 1번 쿼리는 시간복잡도가 쿼리 당 $O(3^9)$ 이고, 2번 쿼리는 시간복잡도가 쿼리 당 $O(3^6)$ 입니다.
- 하지만 $M \leq 500\ 000$ 이므로 시간복잡도의 개선이 필요합니다.
- 1번 쿼리에 대한 연산을 bitset으로 처리하거나, 앞 9자리를 상위 6자리와 하위 3자리로 다시 분리해서 이진수의 bitwise 연산을 사용하면 1번 쿼리의 시간복잡도를 쿼리 당 $O(3^6)$ 이하로 줄일 수 있습니다.
- 3^6 미만의 음이 아닌 두 정수에 대한 OR_3 값은 전처리 및 배열 저장을 통해서 $O(1)$에 계산할 수 있습니다.
- 전체 시간복잡도는 $O(3^6 \cdot M)$ 입니다.

1C. 멀티버스를 여행하는 성재를 위한 안내서

`math`, `calculus`, `fft`
출제진 의도 – **Challenging**

- 제출 9번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: -
- 출제자: `sungjae0506`

- k 번째 천체의 좌표 (x, y) 를 복소평면 상에 $p_k = x + yi$ 로 표현하고,
  기댓값을 구하는 식을 쓰면 다음과 같습니다.
- $$\frac{1}{\pi R^2} \int_0^R \int_0^{2\pi} \prod_{k=1}^{N} |re^{i\theta} - p_k|^2 \, rd\theta dr$$
- $$= \frac{1}{\pi R^2} \int_0^R \int_0^{2\pi} \prod_{k=1}^{N} \{(re^{i\theta} - p_k)\overline{(re^{i\theta} - p_k)}\} \, rd\theta dr$$
- $$= \frac{1}{\pi R^2} \int_0^R \int_0^{2\pi} \prod_{k=1}^{N} (re^{i\theta} - p_k) \prod_{k=1}^{N} \overline{(re^{i\theta} - p_k)} \, rd\theta dr$$

- $\prod_{i=1}^{N}(x-p_k)=\sum_{k=0}^{N}a_kx^k$ 라고 하면 식은 다음과 같이 정리됩니다.
- $\frac{1}{\pi R^2}\int_0^R\int_0^{2\pi}\sum_{k=0}^{N}a_kr^ke^{ik\theta}\sum_{k=0}^{N}\overline{a_kr^ke^{ik\theta}}\,rd\theta dr$
- $=\frac{1}{\pi R^2}\int_0^R\int_0^{2\pi}\sum_{k=0}^{N}a_kr^ke^{ik\theta}\sum_{k=0}^{N}\overline{a_k}r^ke^{-ik\theta}\,rd\theta dr$
- $=\frac{1}{\pi R^2}\int_0^R\int_0^{2\pi}\sum_{k=0}^{N}\sum_{l=0}^{N}a_k\overline{a_l}r^{k+l}e^{i(k-l)\theta}\,rd\theta dr$

- $\int_0^{2\pi} e^{ik\theta} d\theta = \begin{cases} 2\pi, & k = 0 \\ 0, & \text{Otherwise} \end{cases}$ 이므로 식은 다음과 같이 정리됩니다.
- $\frac{1}{\pi R^2} \int_0^R 2\pi \sum_{k=0}^{N} a_k \overline{a_k} r^{2k} \, r dr$
- $= \frac{2}{R^2} \int_0^R \sum_{k=0}^{N} a_k \overline{a_k} r^{2k+1} \, dr$
- $= \sum_{k=0}^{N} \frac{|a_k|^2 R^{2k}}{k+1}$

- NTT를 이용해 a_k 를 $O(N \log^2 N)$ 시간 안에 구할 수 있습니다.
- 최종 시간복잡도는 $O(N \log^2 N)$ 입니다.

1D. 농지 나누어 갖기

`segtree`, `sweeping`, `coordinate compression`
출제진 의도 – **Hard**

- 제출 48번, 정답 15명 (정답률 31.250%)
- 처음 푼 사람: **이종영**, 29분
- 출제자: `sjhi00`

1D. 농지 나누어 갖기

- 회장과 부회장이 나누어 갖는 땅의 경계선이 x 축에 수직인 경우를 먼저 살펴봅시다.
- 좌표 압축 후에, 직사각형의 각 변의 가능한 경우의 수는 $O(N)$ 이므로, 브루트 포스 알고리즘으로 $O(N^5)$ 또는 $O(N^4)$ 으로 문제를 풀 수 있습니다.
- 윗변과 아랫변을 $O(N^2)$ 가지의 방법으로 고정하고 x 축 방향으로 스위핑을 하면 $O(N^3)$ 으로 시간복잡도를 줄일 수 있습니다.
- 하지만 이렇게 하더라도 아직도 느립니다 :(

SNU Computer Study Club

- 세그먼트 트리(segment tree)를 편의상 세그라고 하겠습니다.
- "잘 알려진" 금광 세그(최대 구간 합 세그)를 응용하여 시간복잡도를 줄일 수 있습니다.
- 구조체 및 연산을 잘 정의해서 구조체 세그를 만들어서 풀면 $O(N^2 \log N)$의 시간복잡도로 문제를 해결할 수 있습니다.

1D. 농지 나누어 갖기

- 길이가 N 인 배열에 해당하는 세그를 다음과 같이 구성합니다.
- 세그의 각 노드에 14 개의 변수 $sum_1, sum_2, lmax_1, lmax_2, rmax_1, rmax_2, lrmax_{12}, lrmax_{21}, lmax_{12}, lmax_{21}, rmax_{12}, rmax_{21}, max_{12}, max_{21}$ 를 만듭니다.
- sum_1 은 해당 구간 안에 속해 있는 a_i 의 합입니다. 마찬가지로, sum_2 는 해당 구간 안에 속해 있는 b_i 의 합입니다.
- $lmax_1$ 은 해당 구간의 (left) prefix subarray(접두사 부분배열)의 a_i 의 합의 최댓값입니다. (앞으로 언급하는 모든 부분배열은 길이가 0 인 부분배열도 허용합니다.)
- $rmax_1$ 은 해당 구간의 (right) suffix subarray(접미사 부분배열)의 a_i 의 합의 최댓값입니다.
- $lrmax_{12}$ 은 해당 구간의 중간에 칸막이를 놓았을 때, 칸막이의 왼쪽의 a_i 의 합과 칸막이의 오른쪽의 b_i 의 합의 최댓값입니다. (칸막이의 왼쪽 또는 오른쪽에 아무 것도 없어도 됩니다.)

- $lmax_{12}$ 는 해당 구간의 (left) prefix subarray에 대한 $lrmax_{12}$ 의 최댓값입니다.
- $rmax_{12}$ 는 해당 구간의 (right) suffix subarray에 대한 $lrmax_{12}$ 의 최댓값입니다.
- max_{12} 는 해당 구간의 continuous subarray에 대한 $lrmax_{12}$ 의 최댓값입니다.
- 나머지 변수도 비슷한 방식으로 정의할 수 있습니다.
- 리프 노드가 아닌 어떤 노드 C 의 자식 노드가 순서대로 A, B 라고 하면, A 과 B 의 변수 값을 통해서 C 의 변수 값을 구할 수 있습니다. 이 과정을 병합(merge)이라고 하겠습니다.
- 병합을 하는 과정은 금광 세그(최대 부분 합 세그)와 유사하지만, 노드에 있는 변수가 많은 만큼 식의 개수도 많고 다소 복잡합니다.

- $C.sum_1 = A.sum_1 + B.sum_1$ 으로 구할 수 있습니다.
- $C.lmax_1 = \max(A.lmax_1, A.sum_1 + B.lmax_1)$ 으로 구할 수 있습니다.
- $C.lrmax_{12} = \max(A.lrmax_{12} + B.sum_2, A.sum_1 + B.lrmax_{12})$ 입니다.
- $C.lmax_{12} = \max(A.lmax_{12}, A.lrmax_{12} + B.lmax_2, A.sum_1 + B.lmax_{12})$ 입니다.
- $C.max_{12} = \max(A.max_{12}, B.max_{12}, A.rmax_{12} + B.lmax_2, A.rmax_1 + B.lmax_{12})$ 입니다.
- 나머지 변수도 비슷한 방법으로 계산할 수 있습니다.
- 이 과정을 통해서 병합 과정을 $O(\log N)$ 으로 구현할 수 있습니다.

1D. 농지 나누어 갖기

- 회장과 부회장의 영역을 공유하는 변이 x 축에 수직인 경우부터 고려합시다.
- 윗변과 아랫변을 고정하는 $O(N^2)$ 가지의 방법 각각에 대하여, 각 x 좌표에 대해 a_i 의 합과 b_i 의 합을 구한 뒤, 이 배열의 전체에 해당하는 구간 노드에 대해서 max_{12} 와 max_{21} 값을 계산하면 됩니다.
- 아랫변을 고정시킨 후 윗변을 위로 이동시키는 방법을 사용하면, 아랫변을 고정시키는 $O(N)$ 가지의 방법 각각에 대하여 $O(N)$ 번의 업데이트를 함으로써 금광과 마찬가지로 $O(N^2 \log N)$ 에 구현할 수 있습니다.
- 회장과 부회장의 영역을 공유하는 변이 y 축에 수직인 경우도 비슷한 방법으로 계산하면, 전체 시간복잡도는 $O(N^2 \log N)$ 이 됩니다.

1F. 멋진 연결 요소와 쿼리

`disjoint set`, `tree map`
출제진 의도 – **Hard**

- 제출 130번, 정답 19명 (정답률 14.615%)
- 처음 푼 사람: **김동현**, 30분
- 출제자: `sungjae0506`

1F. 멋진 연결 요소와 쿼리

- ✓ N, Q 제한이 각각 10^5 이므로, $O(NQ)$ 시간복잡도를 가지는 풀이는 통과할 수 없습니다.
- ✓ $1, 2, 3$ 번 쿼리를 각각 적어도 선형시간보다 작은 시간에 처리해야 합니다.

SNU Computer Study Club

- 먼저 1번 쿼리에서 고른 두 정점을 연결하는 것은 두 연결요소를 하나로 합치는 과정이기에 union-find로 해결 가능합니다.
- 만약 고른 두 정점이 각각 서로 다른 “멋진 연결 요소”에 속했다면, 합친 연결 요소도 “멋진 연결 요소”로 바꿀 수 있습니다. 또는 두 정점이 같은 연결 요소에 속했지만 다른 색깔인 경우에는 이미 “멋진 연결 요소”를 만족합니다.
- 이외의 경우에는 “멋진 연결 요소”로 만들 수 없습니다.

SNU Computer Study Club

- 두 정점이 서로 다른 “멋진 연결 요소”에 속하고 같은 색깔인 경우에 “멋진 연결 요소”로 바꾸기 위해서는 더 작은 연결 요소의 색깔을 모두 변경해야 합니다.
- 색깔을 0 또는 1 로 기록하고 union-find에서 merge 함수를 구현할 때 루트부터 자신까지의 경로에 있는 모든 정점을 xor 하면 자신의 색깔이 나오도록 합니다.
- 이렇게 하면 루트의 색깔을 변경할 때 나머지 정점의 색깔도 모두 변경됩니다.
- 2 번 쿼리의 경우도 위와 같이 루트의 색깔을 바꾸는 방법으로 수행할 수 있습니다.

- $1, 2$번 쿼리를 수행할 때마다 빨간색, 파란색 정점의 개수를 기준으로 “멋진 연결 요소”들을 정렬하는 RB tree를 구성합니다.
- 3번 쿼리를 수행할 때마다 빨간색 또는 파란색이 가장 많은 연결요소를 RB tree에서 찾고 최소 정점 번호를 출력합니다.

- 시간복잡도는 $O(QlogN)$ 입니다.
- 이외에도 small to large trick을 이용하거나 segment tree를 이용하는 풀이도 있습니다.

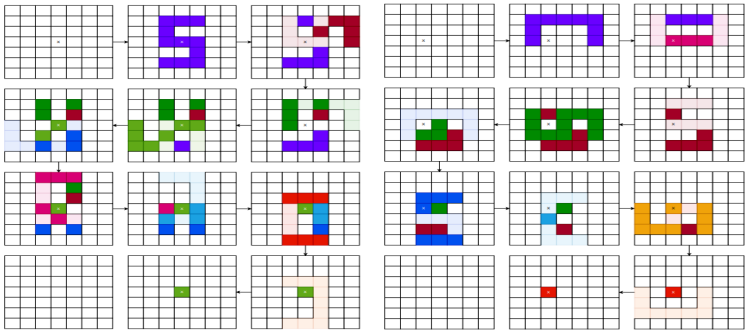
1H. 스시스시 아일랜드

`ad-hoc, implementation`
출제진 의도 – **Hard**

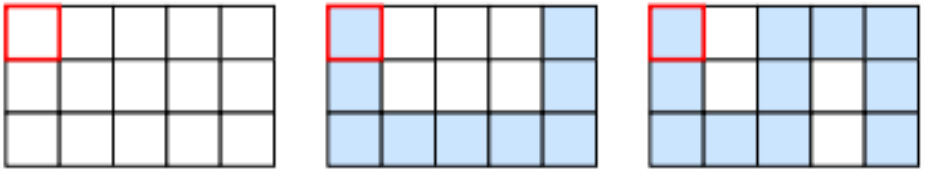
- 제출 17번, 정답 1명 (정답률 5.882%)
- 처음 푼 사람: **김동현**, 147분
- 출제자: `mujigae, sungjae0506`

1H. 스시스시 아일랜드

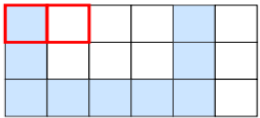
- 문제를 해결하기에 앞서 S와 C를 이용해 어떻게 항상 돌을 원하는 대로 뒤집을 수 있을지 생각해 봅시다.
- 1×1 크기의 단 한 칸만을 S와 C로 뒤집을 수 있다면 가능하지 않을까요?
- 다음과 같은 과정을 통해 정확히 하나의 돌만 뒤집힌 모양을 9번만에 만들 수 있습니다.

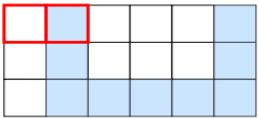


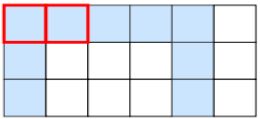
- 석판에 놓인 돌의 개수는 N^2 이고 돌을 뒤집는 횟수의 제한도 N^2 입니다. 돌 하나를 원하는 상태로 만들기 위해 9 번씩 돌을 뒤집는다면, 제한 횟수를 아득히 뛰어 넘게 됩니다.
- 돌 하나당 돌을 최대 한 번 뒤집어서 원하는 상태로 만드는 방법을 생각해볼 수 있습니다.

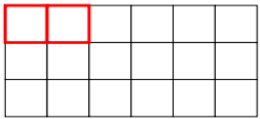


- 이를 더 효율적으로 해결하기 위해서, 적은 횟수를 통해 여러 칸을 원하는 상태로 만드는 방법을 생각해볼 수 있습니다. 정해에서 사용하는 방법은 돌 2개를 최대 한 번에 원하는 상태로 만드는 것입니다.



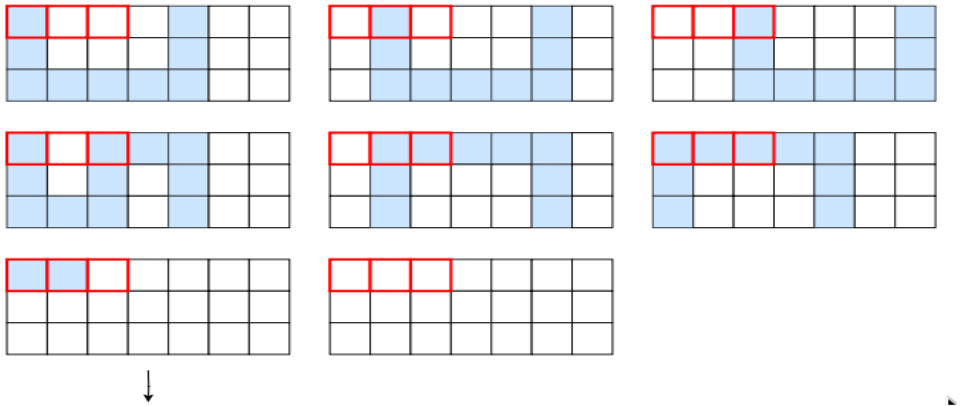






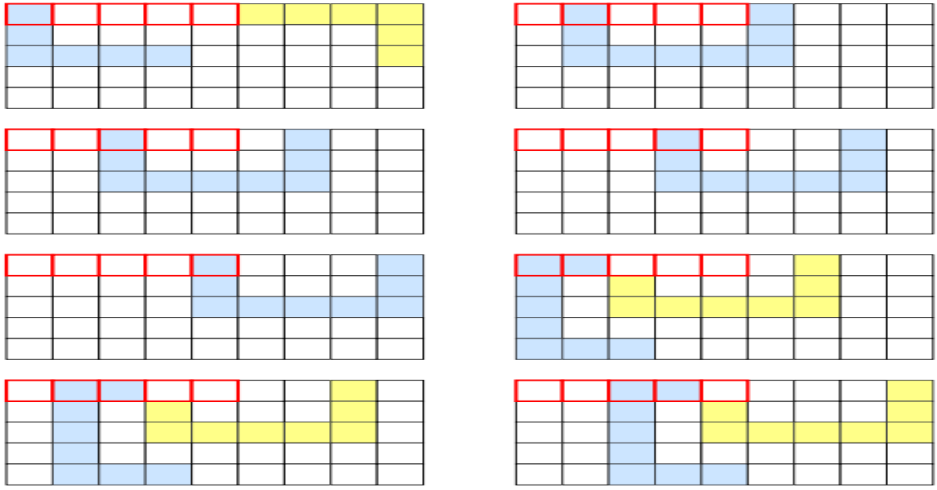
1H. 스시스시 아일랜드

- 이외에도 복잡하지만 돌 3개를 평균적으로 한 번에 원하는 상태로 만들기, 돌 5개를 최대 2번에 원하는 상태로 만들기 등 더욱 효율적인 다양한 방법이 존재합니다.

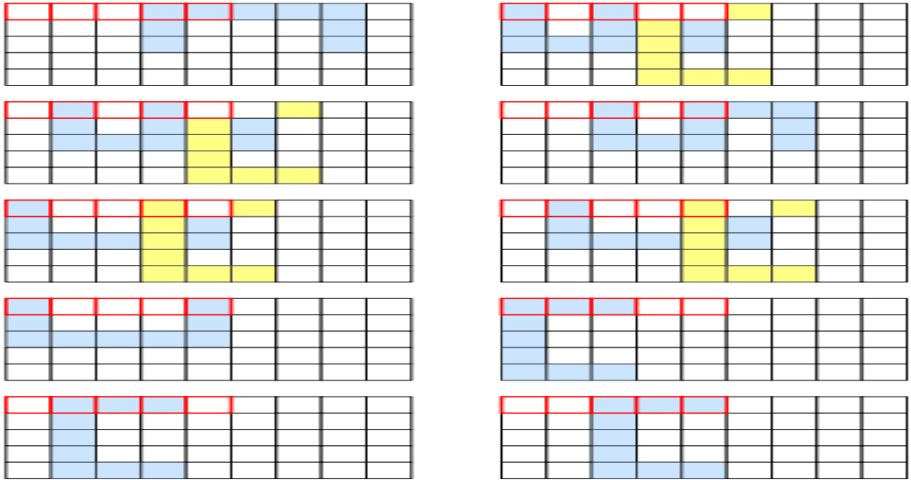


1H. 스시스시 아일랜드

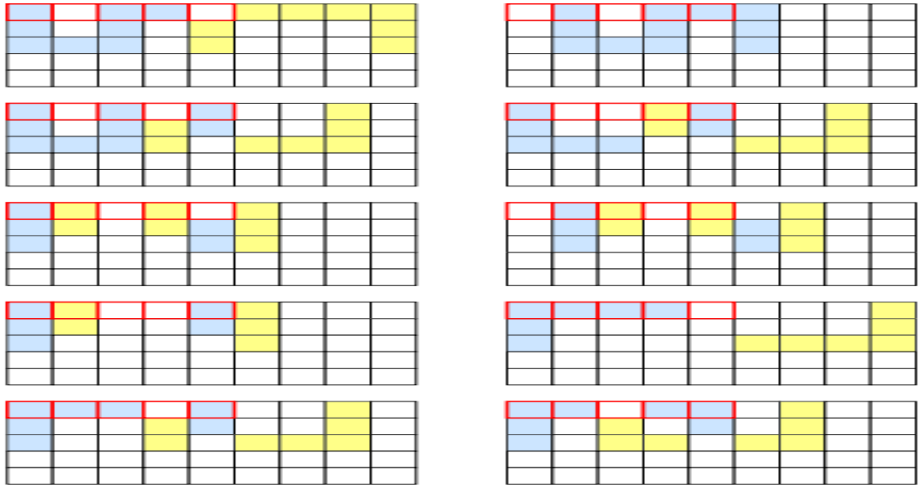
1H. 스시스시 아일랜드



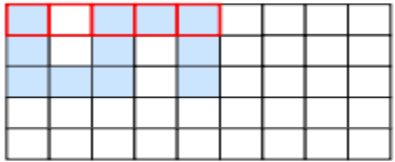
1H. 스시스시 아일랜드

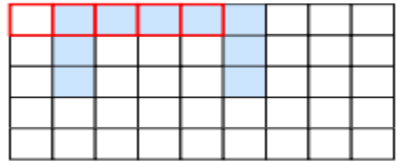


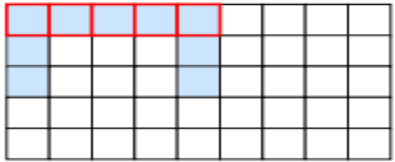
1H. 스시스시 아일랜드

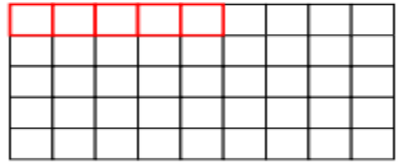


1H. 스시스시 아일랜드



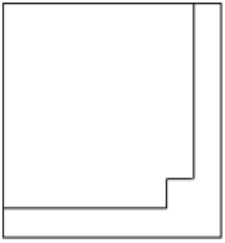






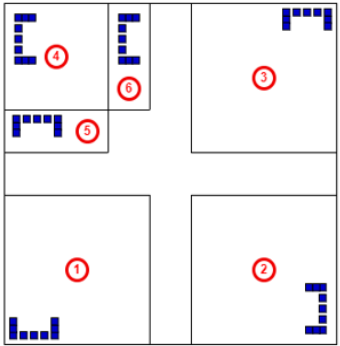
1H. 스시스시 아일랜드

- 이렇게 돌을 한쪽 꼭짓점부터 원하는 상태로 만들어 나간다면 더이상 이 방법으로는 해결할 수 없는 영역이 생기게 됩니다.
- 이러한 영역이 정사각형의 가장자리에 생긴다면 돌을 뒤집을 때 상당히 큰 제약이 생기게 됩니다.



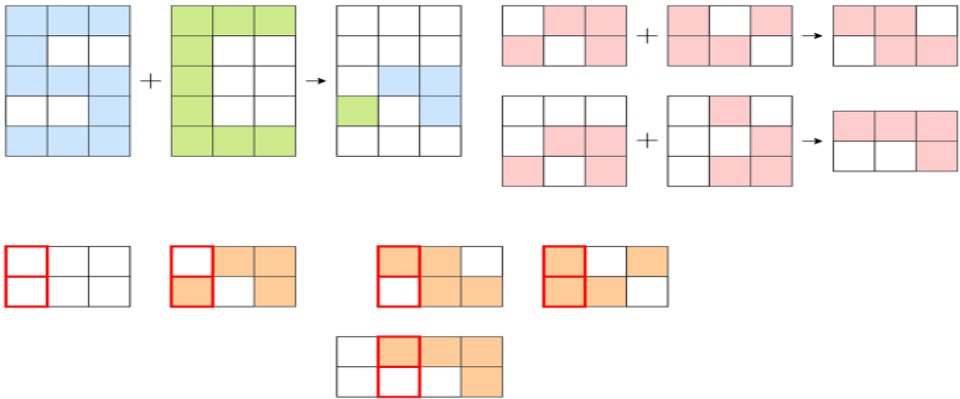
1H. 스시스시 아일랜드

- 모든 꼭짓점에서 돌을 뒤집어 나감으로써 이러한 영역이 정사각형의 중심부에 오도록 만들 수 있습니다. 이 영역의 크기는 위치와 관계없이 항상 일정하게 만들 수 있으므로, 꼭 정확히 중심이 아니더라도 돌을 뒤집는 데 제약이 없는 위치에만 오도록 하면 됩니다.



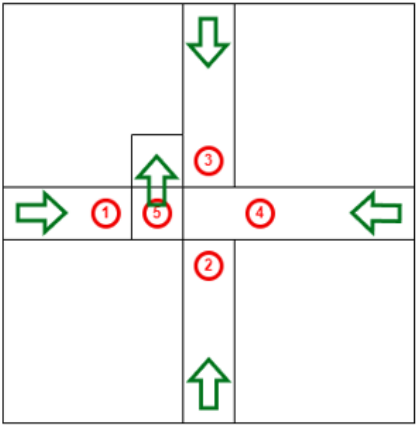
1H. 스시스시 아일랜드

- 비어 있는 십자가 모양 영역의 폭은 2입니다. 다음과 같은 모양을 이용해 1×2 크기의 영역을 최대 4번만에 원하는 대로 뒤집어 나갈 수 있습니다.

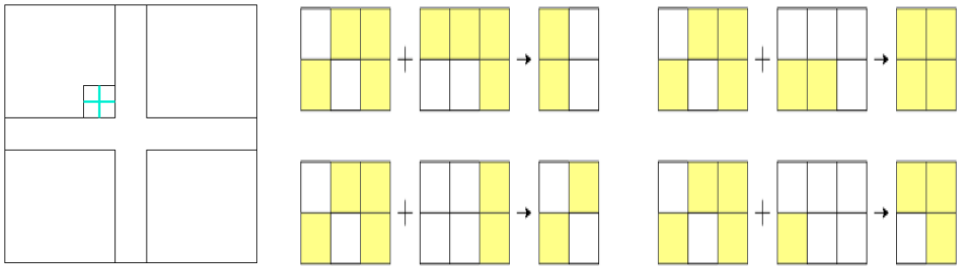


1H. 스시스시 아일랜드

✓ 다음과 같은 순서로 돌을 뒤집어 줍니다.



- 남은 2×2 영역은 가장 먼저 찾았던 1×1 만을 뒤집는 방법을 4 번 적용하여 총 36 번만에 만들어 낼 수 있습니다.
- 36 번 뒤집어도 횟수 제한에 문제는 없지만, 여러 방법을 찾아본다면 다음과 같이 최대 11 번만에 원하는 2×2 영역을 만들 수 있습니다. 더 적은 횟수도 있는데 여러분도 한 번 찾아보세요.



- 폭이 2인 십자가 모양 영역과 2×2 크기의 정사각형 영역을 제외한 영역은 최대 $\frac{(\lceil \frac{N-2}{2} \rceil^2 - 1) \times 4}{2}$ 번 돌을 뒤집어 해결할 수 있습니다.
- 폭이 2인 십자가 모양 영역은 최대 $(4N - 4) \times 2$번 돌을 뒤집어 해결할 수 있습니다.
- 2×2 크기의 정사각형 영역은 최대 36번 돌을 뒤집어 해결할 수 있습니다.
- N이 4의 배수인 경우 최대 $\frac{1}{2}N^2 + 6N + 28$번 돌을 뒤집어 전체 영역을 원하는 모양으로 만들 수 있습니다. $N^2 \geq \frac{1}{2}N^2 + 6N + 28$을 만족하는 자연수 N의 범위는 $N \geq 16$입니다. N이 4의 배수가 아닌 경우도 같은 방식으로 증명할 수 있으며 16 이상의 N에 대해서는 지금까지 소개한 방법으로 N^2번 이내에 석판을 원하는 모양으로 만들 수 있습니다.

SNU Computer Study Club

- $\lfloor \frac{N^2}{2} \rfloor$ 번만에도 문제를 해결할 수 있을까요?
- 가능하다면 조건은 어떻게 될까요?