Hello, BOJ 2024!

Official Solutions

온사이트 콘테스트에

84명 참가

제출 **1088**회

AC **334**회, 해결된 문제 **7**개

총괄 및 운영

- ✓ 김영현 `kipa00` 서울대학교 컴퓨터공학부
- ✓ 김준겸 `ryute` 고려대학교 컴퓨터학과
- ✓ 정우경 `man_of_learning` 전북대학교 컴퓨터공학부

출제

✓ 곽우석	bubbler	SK 하이닉스
✓ 김기범	ibm2006	경기과학고등학교
✓ 김동현	kdh9949	Moloco
✓ 나정휘	jhnah917	숭실대학교 컴퓨터학부
✓ 오주원	kyo20111	숭실대학교 소프트웨어학부
✓ 이종서	leejseo	KAIST 전산학부, Moloco
✓ 이한길	wapas	중앙대학교 예술공학부

검수

✓ 김영현	kipa00	서울대학교 컴퓨터공학부
✓ 김준서	junseo	한양대학교 컴퓨터소프트웨어학부
✓ 모현	ahgus89	가톨릭대학교 의예과
✓ 박선재	cs71107	서울대학교 컴퓨터공학부
✓ 윤시우	cgiosy	서울사이버대학교 컴퓨터공학과
✓ 정현서	jhwest2	서울대학교 컴퓨터공학부
✓ 최준석	stonejjun03	고려대학교

대회 진행

- ✓ 김영현 kipa00 — 서울대학교 컴퓨터공학부
- ✓ 김준겸 ryute — 고려대학교 컴퓨터학과
- ✓ 윤창기 TAMREF — 서울대학교 수리과학부

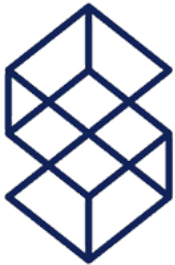
사진작가

- ✓ 박한나 crescent_h — 서강대학교 컴퓨터공학과
- ✓ 정은채 celina324 — 이화여자대학교 컴퓨터공학과

Sponsors

Sponsors

Sponsors



SAMSUNG
SOFTWARE
MEMBERSHIP

FURIOSA


HYUNDAI
AutoEver


Presto


LG전자


LIG넥스원

문제		의도한 난이도	출제
A	모바일 광고 입찰	**Easy**	leejseo
B	2024는 무엇이 특별할까?	**Easy**	jhnah917
C	3+1 하노이 탑	**Medium**	bubbler
D	주행시험장	**Medium**	bubbler
E	신제품 개발	**Medium**	kyo20111
F	깃발 꽂기	**Hard**	kdh9949
G	가상 검증	**Hard**	ibm2006
H	Transformer Knight's Tour	**Hard**	bubbler

문제		처음 푼 사람	해결 시각
A	모바일 광고 입찰	ainta	1분
B	2024는 무엇이 특별할까?	cozyyg	4분
C	3+1 하노이 탑	dlalswp25	12분
D	주행시험장	molamola	31분
E	신제품 개발	molamola	66분
F	깃발 꽂기	molamola	148분
G	가상 검증	Sait2000	164분
H	Transformer Knight's Tour	?	?분

A. 모바일 광고 입찰

`sorting`
출제진 의도 – **Easy**

- ✓ 문제 아이디어: `leejseo`
- ✓ 문제 세팅: `wapas`

- K 개 이상의 광고 지면에서 $A_i + X \geq B_i$ 이 성립하도록 만들어야 합니다.
- 즉, $X \geq \max(0, B_i - A_i)$ 이 성립하는 i 가 K 개 이상 존재해야 합니다.
- 따라서 정답은 가능한 $\max(0, B_i - A_i)$ 중에서 K 번째로 작은 값입니다.
- 정렬을 활용하여 $\mathcal{O}(N \log N)$ 에 해결할 수 있습니다.

B. 2024는 무엇이 특별할까?

`mathematics`, `number_theory`

출제진 의도 – **Easy**

- ✓ 문제 아이디어: `jhnah917`
- ✓ 문제 세팅: `jhnah917`

- 양의 정수 n을 소인수분해한 결과를 $n = 2^t p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ 라고 합시다.
- n의 약수는 $d = 2^x p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$ 꼴로 나타낼 수 있습니다. (단, $0 \le x \le t, 0 \le f_i \le e_i$)
- 이때 $x = 0$이면 홀수 약수, $1 \le x \le t$이면 짝수 약수이므로 $\tau_e(N) = t\tau_o(N)$이 성립합니다.
- n이 K-특별한 수인 것은 n을 이진법으로 나타냈을 때 trailing zero가 K개인 것과 동치입니다.
- 그러므로 K-특별한 수의 개수는 (2^K의 배수 개수) - (2^{K+1}의 배수 개수)와 동일합니다.
- $\lfloor N/2^K \rfloor - \lfloor N/2^{K+1} \rfloor$을 출력하면 됩니다.

C. 3+1 하노이 탑

`mathematics`, `recursion`

출제진 의도 – **Medium**

- ✓ 문제 아이디어: `bubbler`
- ✓ 문제 세팅: `bubbler`

- 편의상 각 원판을 작은 것부터 순서대로 $1, 2, \cdots, N$ 이라고 합시다.
- 원판 N 을 기둥 A에서 D로 옮기려면 바로 위에 있는 원판 $N-1$ 을 먼저 옮겨야 합니다.
- 이때 원판 $N-1$ 을 D로 옮기면 N 을 D로 옮기는 것이 불가능해지므로, B나 C로 옮겨야 합니다.
- 일반성을 잃지 않고 B로 옮긴다고 가정합시다.

- 원판 $N-1$을 A에서 B로 옮기려면 원판 $1, \cdots, N-2$가 모두 C에 쌓여 있어야 합니다.
- 최소한의 이동으로 이 상태를 만들기 위해서는 3-peg Hanoi 문제를 $N-2$개 원판에 대해 풀면 됩니다.

- 앞의 과정을 수행하고 나서 원판 $N-1$을 B로 옮기고 나면, A에는 N, B에는 $N-1$, C에는 $1, \cdots, N-2$가 있는 상태가 됩니다.
- 이제 N을 A에서 D로, $N-1$을 B에서 D로 옮길 수 있고, 그러고 나면 C에 있는 $1, \cdots, N-2$를 D로 옮기는 문제가 됩니다.
- 남은 원판이 2개 이하가 될 때까지 이 과정을 반복합니다.
- 원판이 2개가 남으면 3번, 1개가 남으면 1번에 남은 원판을 D로 옮길 수 있습니다.

D. 주행시험장

`ad_hoc`, `case_work`
출제진 의도 – **Medium**

- 문제 아이디어: `bubbler`
- 문제 세팅: `bubbler`

- 일반성을 잃지 않고 $n \leq m$ 이라고 하고, 편의상 전방 레이더를 장착한 차를 타입 A, 코너 레이더를 타입 B라고 합시다.
- 먼저, 동쪽에서 한 종류만 남기기 위해 옮겨야 하는 차의 수는 적어도 n(A를 모두 옮기고 B만 남김)입니다. 따라서 $k \geq n$ 입니다.
- $n = m$ 인 경우, n 대의 A를 모두 옮기고 다시 돌아와서 m 대의 B를 모두 옮기는 것이 최적입니다. 이때 이동 횟수는 3입니다.

- ✓ $n < m \leq 2n$인 경우에는 다음과 같은 분석이 가능합니다.
  - 먼저 A를 모두 서쪽으로 옮깁니다.
  - 다음 이동부터 B를 옮길 수 있습니다. 하지만 B를 모두 옮기려면 어떤 시점에서는 양쪽 주행시험장에 B가 있어야 합니다. 이때는 A를 다시 모두 동쪽으로 옮겨야 서쪽에 B를 둘 수 있습니다.
  - 서쪽에 $m - n$대 이상의 B를 두었다면, A를 모두 동쪽으로 옮긴 후 A를 동쪽에 남기고 나머지 B를 모두 서쪽으로 옮길 수 있으므로, $k = n$이면 충분합니다.

- 실제 이동의 예시는 다음과 같습니다.

$$
\begin{aligned}
(0, m) &\xrightarrow{(n,0)} (0, 0) \\
(0, m) &\xleftarrow{(0,0)} (n, 0) \\
(0, m-n) &\xrightarrow{(0,n)} (n, 0) \\
(0, m-n) &\xleftarrow{(n,0)} (0, n) \\
(n, 0) &\xrightarrow{(0,m-n)} (0, n) \\
(n, 0) &\xleftarrow{(0,0)} (0, m) \\
(0, 0) &\xrightarrow{(n,0)} (0, m)
\end{aligned}
$$

- 이때 이동 횟수는 7입니다.
- 첫 2회 이동 후에는 m대의 B가 모두 동쪽에 남아있을 수밖에 없고, 반대로 마지막 2회 이동 직전에는 m대의 B가 모두 서쪽에 있어야 하므로, 5회 이동으로는 모든 차를 서쪽으로 이동할 수 없습니다.

- ✓ $m > 2n$인 경우, $k = n$이면 A를 모두 가지고 돌아온 시점에서 A를 남기고 B를 모두 가지고 갈 수 없습니다. $k = n + 1$이면 A를 모두 동서로 이동하면서 B를 계속 한 대씩 옮기는 방법이 존재하므로 이 값이 답이 됩니다.
- ✓ 이동 횟수를 최소화하기 위해서는 다음과 같이 분석할 수 있습니다.
  - 맨 처음에는 A를 모두 서쪽으로 옮기고, 그 다음에는 B를 $n + 1$대 옮기고 A를 모두 다시 동쪽으로 가져옵니다.
  - 이제 동쪽에 B가 $n + 1$대 이하로 남을 때까지는 A를 어느 한쪽에 둘 수 없으므로, 그때까지 B를 1대씩 옮기는 것을 반복합니다.
  - 동쪽에 B가 $n + 1$대 이하가 남았다면, A를 동쪽에 두고 남은 B를 모두 서쪽으로 옮기고, 마지막으로 돌아와서 A를 모두 옮깁니다.

- 실제 이동의 예시는 다음과 같습니다.

$$
\begin{array}{ccc}
(0, m) & \xrightarrow{(n,0)} & (0, 0) \\
(0, m) & \xleftarrow{(0,0)} & (n, 0) \\
(0, m-(n+1)) & \xrightarrow{(0,n+1)} & (n, 0) \\
(0, m-(n+1)) & \xleftarrow{(n,0)} & (0, n+1) \\
(0, m-(n+2)) & \xrightarrow{(n,1)} & (0, n+1) \\
(0, m-(n+2)) & \xleftarrow{(n,0)} & (0, n+2) \\
 & \cdots &
\end{array}
$$

- 실제 이동의 예시는 다음과 같습니다. (continued)

$$
\begin{array}{ccc}
 & \cdots & \\
(0, n+1) & \xrightarrow{(n,1)} & (0, m-(n+2)) \\
(0, n+1) & \xleftarrow{(n,0)} & (0, m-(n+1)) \\
(n, 0) & \xrightarrow{(0,n+1)} & (0, m-(n+1)) \\
(n, 0) & \xleftarrow{(0,0)} & (0, m) \\
(0, 0) & \xrightarrow{(n,0)} & (0, m)
\end{array}
$$

- 이때 이동한 횟수는 $n < m \leq 2n$의 경우와 비교했을 때 B를 1대씩 이동한 횟수가 추가된 형태이므로, $m \leq 2n + 2$이면 7회, 그보다 많으면 $7 + 2(m - (2n + 2))$ 회가 됩니다.
- $2n + 1 \leq m \leq 2n + 2$인 경우, 첫 2회 이동 후에 동쪽에 남아있는 B의 대수는 적어도 $m - 1$대이며, 마지막 2회 이동 직전에 서쪽에 있어야 하는 B의 대수 역시 적어도 $m - 1$대입니다.
- 이 경우는 A n대와 B 1대를 가지고 가서 A n대를 가지고 돌아온 상황이므로, 남은 B를 모두 가지고 갈 수 없다면 B를 추가로 1대만 더 가지고 서쪽으로 갈 수 있는 상황이 됩니다.
- 이는 3번째 이동에 B $n + 1$대를 이동할 수 있는 정해와 진행률이 같거나 더 손해입니다.

✓ 예외적으로 $n = 1, m = 3$인 경우에는 5회만에 차를 모두 옮기는 방법이 존재합니다.

$$
\begin{aligned}
(0,2) &\xrightarrow{(1,1)} (0,0)\\
(0,2) &\xleftarrow{(1,0)} (0,1)\\
(0,1) &\xrightarrow{(1,1)} (0,1)\\
(0,1) &\xleftarrow{(1,0)} (0,2)\\
(0,0) &\xrightarrow{(1,1)} (0,2)
\end{aligned}
$$

- $2n + 2 < m$ 이라면, 동쪽의 상태가 $(n, m - (n + 1))$ 인 시점과 서쪽의 상태가 $(n, m - (n + 1))$ 인 시점을 반드시 거쳐야 합니다.
- 첫 번째 이동에서는 B를 최대 1대만 이동할 수 있으므로 전자에 도달하려면 적어도 4번의 이동이 필요합니다.
- 마찬가지로 후자의 상태에서 시작해서 모든 차를 서쪽으로 옮기려면 역시 적어도 4번의 이동이 필요합니다.
- 전자에서 후자까지 가는 과정에서는 첫 이동에 B를 1대, 그 다음부터는 매 2회 이동에 1대를 서쪽으로 보낼 수 있으므로 $2(m - (2n + 2)) - 1$ 회 이동이 소요됩니다.

E. 신제품 개발

functional_graph
출제진 의도 – **Medium**

- ✓ 문제 아이디어: kyo20111
- ✓ 문제 세팅: stonejjun03

- 임의의 c에서 각 정점에서 이동하게 될 다음 정점은 유일하므로 어떤 순간의 그래프는 함수 그래프입니다.
- $B_i = 0$인 나가는 간선이 있는 정점을 제외하면, 아무런 행동을 하지 않은 $c = 0$일 때 각 정점은 자기 자신을 가리키는 나가는 간선을 가지고 있다고 생각할 수 있습니다.
- 사용하지 않은 간선 중 가장 작은 B_i를 가진 간선의 번호를 mn번이라고 합시다.
- c가 증가할 때마다 $B_{mn} \leq c$일때까지 mn번 간선을 사용하여 u_{mn}에서 나가는 간선을 지우고 새롭게 v_{mn}를 가리키는 간선을 갖도록 변경하는 것으로 c가 해당 값을 가질 때의 함수 그래프를 찾을 수 있습니다.

- 우리는 $c < B_{mn}$ 이면서 이동한 횟수가 총 K 번이 되기 전까지 고정된 함수 그래프에서 이동할 수 있습니다.
- 함수 그래프에서 이동하는 것을 관찰해 보면 어떤 경로를 따라가다가 사이클에 진입하게 되고, 사이클에 진입한 이후에는 사이클 내부에서만 이동한다는 것을 알 수 있습니다.

- 사이클의 길이를 L, 사이클에 속한 A_i의 합을 S라고 합시다.
- 사이클에 막 진입한 이후, 사이클 안에서 한 바퀴를 돌아 같은 정점으로 돌아오는 것을 총 $\left\lfloor \min\left(\frac{\text{남은 이동 횟수}}{L}, \frac{B_{mn} - c}{S} \right) \right\rfloor$ 번 한다는 것을 알 수 있습니다.
  - $S = 0$인 경우는 사이클 내부에서 무한히 이동하게 되므로 남은 이동 횟수를 L로 나눈 나머지로 두면 됩니다.
- 이를 수식으로 처리하고, 사이클의 내부에서 L보다 작은 횟수의 이동을 한 뒤에 mn번 간선이 사용되거나, 이동을 종료합니다.
- 사이클까지 이동한 뒤에 L과 S를 구하는 것까지 최대 N번의 이동을 해서 알아낼 수 있고, 이후 최대 $L(\leq N)$번의 이동을 하게 되므로 $\mathcal{O}(N)$에 이 단계를 해결할 수 있습니다.

- ✓ 간선을 하나씩 사용하는 행동을 M 번 해야하고, 한 개의 간선을 사용하는 데 $\mathcal{O}(N)$ 의 시간이 걸리므로 $\mathcal{O}(NM)$ 에 변하지 않는 최종 상태의 그래프를 찾을 수 있습니다.
  - 간선을 모두 사용하지 않은 상태여도 중간에 K 번의 이동을 한 이후에는 즉시 종료해야 함을 유의하며 구현해야 합니다.
- ✓ 최종 상태의 그래프에서 간선을 지울 때처럼 남은 이동 횟수만큼 이동하면 이후 $\mathcal{O}(N)$ 에 문제의 정답을 찾을 수 있습니다.

- A_i를 K번 더하는 연산은 64비트 정수 자료형의 범위를 넘어가므로 따로 처리해야 합니다.
- c의 값이 B_i 중 최대값에 도달하게 된다면 그 이후부터는 c와 B_i를 비교할 필요가 없어지므로 c와 A_i의 값을 모두 $10^9 + 7$로 mod한 값으로 관리하며 답을 찾거나,
- 128비트 정수 자료형의 범위를 넘어가지 않으므로 이를 사용하면 따로 처리하지 않아도 문제를 해결할 수 있습니다.

F. 깃발 꽂기

`two_pointers`

출제진 의도 – **Hard**

- ✓ 문제 아이디어: kdh9949
- ✓ 문제 세팅: kdh9949

- 깃발 M 개를 각각 $a_1 < a_2 < \cdots < a_M$ 좌표에 꽂기로 했다고 합시다.
- 좌표 p에서 출발할 때, 이동해야 하는 최소 거리는 $a_M - a_1 + \min(|p - a_M|, |p - a_1|)$ 입니다.
  - a_1 과 a_M 좌표를 최초로 모두 방문하게 되는 순간 이동을 종료하게 됩니다. 그 전까지는 양 끝 중 하나에 깃발을 아직 못 꽂은 상태이며, 양 끝을 모두 방문했다면 사이에 있는 점들 역시 무조건 방문했을 것입니다.
  - a_1 과 a_M 둘 중 한 곳을 먼저 찍고 반대쪽으로 가는 두 가지 경우를 고려하면 됩니다.
- 즉, 가장 왼쪽/오른쪽 깃발의 위치만 중요합니다. 이제 깃발을 꽂는 위치들을 구간으로 나타내도 됩니다.

- 구간 $[s, e]$ $(s \leq e)$에 대해 $d(s, e, p) = e - s + \min(|s - p|, |e - p|)$ 라고 합시다.
- $d(s, e, p)$는 p에서 출발하여 $[s, e]$ 구간에 깃발을 꽂는 데 필요한 최소 이동 거리입니다.
- 생각을 좀 해 보면, 두 구간이 포함 관계에 있을 경우 포함되는 쪽이 무조건 더 좋습니다.
- 즉, $s_1 \leq s_2, e_2 \leq e_1$ 이면 임의의 p에 대해 $d(s_1, e_1, p) \geq d(s_2, e_2, p)$ 입니다.
- 따라서, '더 이상 줄일 수 없는 구간'들만 고려해도 충분합니다. 이들을 **중요한 구간**이라 합시다.

- ✓ 이미 꽂힌 N 개의 깃발들로 나뉘는 영역들을 **블록**이라 정의합시다.
- ✓ 편의상 $X_0 = -K$, $X_{N+1} = L + K$ 라 하고, 영토의 $[X_i, X_{i+1}]$ 에 해당하는 영역을 i 번 블록이라 합시다. $(0 \le i \le N)$
- ✓ 중요한 구간은 아래와 같은 두 가지 경우로 나눌 수 있습니다.
  1. s 와 e 가 같은 블록에 속하는 경우
  2. s 와 e 가 다른 블록에 속하는 경우

1. s와 e가 같은 블록에 속하는 경우

- ✓ 중요한 구간은 깃발 M개를 연속해서 다닥다닥 붙여놓은 형태가 될 것입니다.
- ✓ 이 때 구간 길이 $e - s = (M-1)K$ 입니다.
- ✓ $X_i + K \le s \le X_{i+1} - MK$ 인 정수 s에 대해 $[s, s + (M-1)K]$ 가 i번 블록에 속하는 중요한 구간들입니다.

2. s와 e가 다른 블록에 속하는 경우

- ✓ s가 i번 블록, e가 j번 블록에 속한다고 합시다. $(i < j)$
- ✓ $(i+1), \ldots, (j-1)$번 블록에는 깃발을 가능한 많 꽂았을 것입니다.
- ✓ c_k를 k번 블록에 최대로 꽂을 수 있는 깃발의 수라 합시다.
- ✓ $C = M - \sum_{k=i+1}^{j-1} c_k$ 이라 하면, i번 블록과 j번 블록에 깃발을 합쳐서 C개 더 꽂아야 합니다.
- ✓ 이 상황이 가능하려면 $2 \le C \le c_i + c_j, 1 \le c_i, 1 \le c_j$ 를 우선 만족해야 합니다.

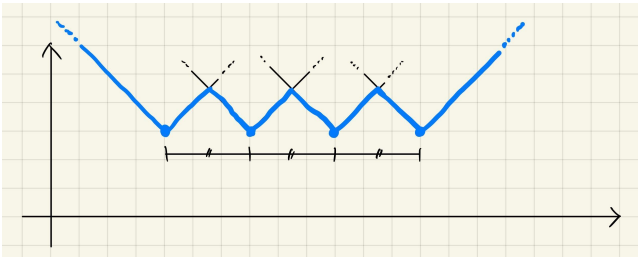
2. s와 e가 다른 블록에 속하는 경우

- ✓ C개 중 l개를 i번 블록에 꽂는다 하면, $C-l$개는 j번 블록에 꽂을 것입니다.
- ✓ $1 \le l \le c_i, 1 \le C-l \le c_j$를 모두 만족해야 합니다.
- ✓ i번 블록에 꽂은 깃발들은 최대한 오른쪽부터, j번 블록은 최대한 왼쪽부터 다닥다닥 붙어 있어야 합니다.
- ✓ $\max(1, C-c_j) \le l \le \min(C_i, C-1)$인 정수 l에 대해 $[X_{i+1} - lK, X_j + (C-l)K]$가 중요한 구간들입니다.

- $d(s, e, p) = e - s + \min(|s - p|, |e - p|)$ 라는 식을 다시 살펴 봅시다.
- 구간 s, e를 고정하면 이는 p에 대한 함수입니다.
- $\min(|s - p| + e - s, |e - p| + e - s)$ 와 같이 식을 바꾸어 봅시다.
- 꼭짓점이 각각 $(s, e - s)$와 $(e, e - s)$에 있는 두 개의 절댓값 (V자 형태) 그래프의 최솟값이 됩니다.
- 결국 각 쿼리는 절댓값 그래프들이 여러 개 있을 때 특정 좌표에서 최솟값을 구하는 문제가 됩니다.

- ✓ 앞서 보았던 중요한 구간의 두 가지 경우에 대해 몇 가지 관찰이 필요합니다.
  1. i 고정 $\rightarrow$ 특정 구간의 정수 s에 대해 $[s, s + (M-1)K]$
  2. (i, j) 고정 $\rightarrow$ 특정 구간의 정수 l에 대해 $[X_{i+1} - lK, X_j + (C-l)K]$
- ✓ 두 경우 모두 특정한 Parameter를 고정하면 길이가 모두 같고 일정한 간격으로 나열된 구간을 얻습니다.

- 구간의 시작점끼리 (또는 끝점끼리) 모아보면, 아래 그림과 같은 절댓값 그래프의 **덩어리**들을 얻을 수 있습니다.



- 덩어리의 총 개수는 $\mathcal{O}(N)$ 입니다.
- 1.의 경우는 블록 개수가 $N+1$개 이므로 자명합니다.
- 2.의 경우는, $2 \leq C \leq c_i + c_j, 1 \leq c_i, 1 \leq c_j$ 인 (i, j) 가 $\mathcal{O}(N)$ 개임을 증명해야 합니다.
- (i, j) 가 조건을 만족하면 $a, b \geq 1$에 대해 $(i+a, j-b)$과 $(i-a, j+b)$은 조건을 만족하지 않는다는 사실을 증명할 수 있습니다.
- 2.의 경우에 대해 실제로 덩어리를 구하는 것은 Two pointers 기법으로 가능합니다.

- ✓ 각 덩어리는 크게 세 부분으로 나눌 수 있습니다.
  1. 왼쪽으로 뻗은 기울기 -1 반직선
  2. 가운데 지그재그 모양
  3. 오른쪽으로 뻗은 기울기 1 반직선
- ✓ 좌표 p에서 기울기 -1 인 반직선의 최솟값을 구하기 위해서는, x 좌표가 p 이상인 시작점들에 대해 (x 좌표 $+ y$ 좌표)의 최솟값을 빠르게 구할 수 있으면 됩니다.
- ✓ 기울기 1 인 반직선은 뒤집어서 비슷하게 하면 됩니다.
- ✓ 전처리 배열 + 이분탐색으로 $\mathcal{O}\left((N+Q)\log N\right)$ 에 해결됩니다.

- 가운데 지그재그 모양을 처리하기 위해서는 한 가지 관찰이 필요합니다.
- 구간의 시작점끼리 모은 덩어리들만 봤을 때, 덩어리의 가운데 부분에 해당하는 구간들은 겹치지 않습니다.
- 구간의 끝점끼리 모은 덩어리 역시 마찬가지입니다.
- 이 말인 즉슨, 임의의 좌표 p에 대해 가운데 부분이 p를 포함하는 덩어리는 많아야 2개라는 것입니다.
- p를 포함하는 모든 덩어리 가운데 부분에 대해 최솟값을 직접 구해 주면 됩니다.
- 이 역시 덩어리를 정렬한 배열을 만들면 이분 탐색으로 $\mathcal{O}\left((N+Q)\log N\right)$에 할 수 있습니다.

G. 가상 검증

ad_hoc,combinatorics

출제진 의도 – **Hard**

- ✓ 문제 아이디어: ibm2006
- ✓ 문제 세팅: ibm2006

- 시계 순서가 섞일 수 있다는 조건 때문에, 각 시계의 시침이 아닌 특정 시각을 가리키는 시계의 수만 의미가 있습니다.
- 시계들의 모임에 대해, i 시를 가리키는 시계의 수를 모아놓은 수열을 **시각 분포**라고 부르겠습니다.

- Type 1 시계 중 1시에 있던 시계가 주행 이후 가리키는 시각을 알 수 있다면, 시계에 가해진 자기장의 세기를 알 수 있습니다.
- 이를 위해 1시에 **과반수 이상**인 12개의 Type 1 시계를 배정하는 방법을 생각할 수 있습니다.
- 이 아이디어에 더하여, 추가로
  - 주행 이후의 시각 분포에서 원래의 두 시각 분포를 복구하는 방법을 찾아야 합니다.
  - 주행 이후 12개 이상의 시계가 가리키는 시각이 둘 이상이면 안됩니다.

- ✓ 수열의 유일한 복구 방법을 제공하기 위해,
  - Type 1 시계의 2-24 시에는 0개 또는 1개의 시계만 배정합니다.
  - Type 2 시계의 1-24 시에는 **짝수 개**의 시계만 배정합니다.
- ✓ 12개 이상의 시계가 두 개 이상의 시점에 모이지 않도록,
  - Type 2 시계의 1-24 시에는 **10 이하의 짝수 개**의 시계만 배정합니다.

편의상 i 시를 가리키는 시계의 수를 c_i 라고 합시다. $(1 \le i \le 24)$

- ✓ 우선 $c_x \ge 12$ 인 유일한 x 를 찾습니다.
- ✓ 주행 이후에 x 시를 가리키는 Type 1 시계는 12 개, Type 2 시계는 $c_x - 12$ 개입니다.
- ✓ $y \ne x$ 에 대해 y 시를 가리키는 Type 1 시계는 $c_y \bmod 2$ 개, Type 2 시계는 $c_y - (c_y \bmod 2)$ 개입니다.
- ✓ 이제 $-(x-1)$ 만큼의 자기장을 걸어 주행 이전의 시각 분포를 복원할 수 있습니다.

주행 이전 시각 분포의 수로 가능한 경우의 수를 세어 보면,

- ✓ Type 1 시계를 배정하는 경우의 수 A는 $\binom{23}{12} \simeq 1.3 \times 10^6$ 가지입니다.
- ✓ Type 2 시계를 배정하는 경우의 수 B는 $x_1 + \cdots + x_{24} = 12,\ 0 \le x_1, \cdots, x_{24} \le 5$를 만족하는 정수 수열의 개수와 같고, 이는 8×10^8 개 이상입니다.

두 경우의 수의 곱이 $AB > 1.1 \times 10^{15} > 10^{14}$로 충분히 크고, 각각의 Type에 대해 사전 순 k번째 시각 분포를 찾는 작업도 어렵지 않습니다. 따라서 비밀번호가 x라면 $\left(\left\lfloor \frac{x}{B} \right\rfloor, x \bmod B\right)$를 각 Type의 시각 분포로 인코딩/디코딩해주면 됩니다.

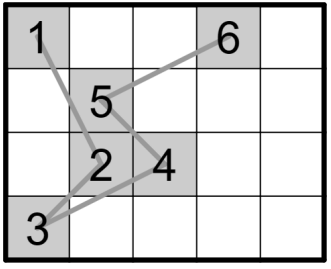
H. Transformer Knight's Tour

`connection_profile_dp`, `exponentiation_by_squaring`
출제진 의도 – **Hard**

- ✓ 문제 아이디어: `bubbler`
- ✓ 문제 세팅: `bubbler`

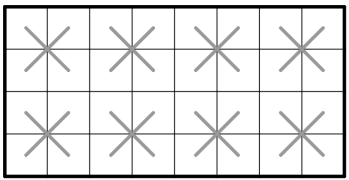
- ✓ Connection Profile DP를 이용하여 풀 수 있는 문제입니다.
- ✓ 그러나 문제 그대로 구현하려고 하면 상태 표현과 전이가 매우 복잡하므로, 이를 단순화시킬 아이디어가 필요합니다.

- 문제에서 요구하는 경로가 존재한다고 가정하고, 격자판의 각 칸에 그 칸을 몇 번째로 밟는지를 그 칸에 적는다고 상상해 봅시다.



- 그러면, 짝수에서 다음 수($4N$ 에서는 다시 1)로 이동할 때에는 모두 퍼즈의 이동 규칙을 사용합니다.
- 이러한 이동은 정확히 $2N$ 번 일어나며, 어떠한 이동도 같은 칸을 공유하지 않습니다.

- ✓ 따라서, 각각의 퍼즈의 이동의 출발점과 도착점을 서로 이으면 격자판의 모든 칸에 대한 완전 매칭이 됩니다.
- ✓ 조금 생각해 보면, 이러한 완전 매칭은 격자판 전체를 2×2로 나누어 각각에 X를 그리는 것 하나뿐임을 알 수 있습니다.



- 이러한 완전 매칭은 N이 짝수일 때만 존재하므로, N이 홀수일 경우 답은 0입니다.
- 이제 2개의 열 단위로 전진하는 Connection Profile DP를 다시 생각해 봅시다.
- 최근 2개의 열의 가능한 상태의 목록은 다음과 같이 구성할 수 있습니다.
  - 먼저, 8개 칸을 2개의 X자로 미리 연결해 놓습니다.
  - 다음으로, 0, 1, 2, 3개의 서로 겹치지 않는 쌍을 연결해 봅니다. 연결 도중에 닫힌 사이클이 등장하는 경우는 제외합니다.
- 이렇게 만들어지는 서로 다른 상태의 수는 풀이에 따라 다를 수 있으나 $k \leq 150$개 이내에서 만들어집니다.
- 따라서, 가능한 상태 전이를 모두 체크하여 전이 행렬을 만들고, 이를 거듭제곱하여 $\mathcal{O}\left(k^3 \log N\right)$ 시간에 문제를 풀 수 있습니다.