BOJ Bundle in Math Vol. 1

공식 해설

문제		의도한 난이도	출제자
B	NL-Queen (mod N)	Diamond	kiwiyou
C	위잉위잉	Gold	jh05013, kiwiyou
E	적은 메모리로 빠르게 많은 소수 찾기	Diamond	jh05013
F	Queens' Rye Cafe	Platinum	kiwiyou
G	Enigmatic Device 2023	Diamond	bubbler
H	A+B - 10 (제1편)	Bronze	jh05013
J	짧은 코드로 빠르게 많은 소수 세기	Diamond	jh05013
K	수열 선물받기	Gold	hibye1217
L	N-Queen (Area)	Platinum	wider93
M	커피우유	Diamond	kiwiyou
O	A+B - 10 (제2편)	Silver	jh05013

B. NL-Queen (mod N)

`ad_hoc, constructive, number_theory, pigeonhole_theorem`

출제진 의도 — **Diamond**

- 제출 67번, 정답 4명 (정답률 5.97%)
- 처음 푼 참가자: **sshs29hanul**, 17시간 36분
- 출제자: `kiwiyou`

B. NL-Queen (mod N)

- 해가 없는 경우부터 살펴봅시다.
- N이 작은 경우에 완전탐색을 통해 $\gcd(N, 6) = 1$이어야 한다고 추측할 수 있습니다.
- 이를 증명할 수 있을까요?[1]

B. NL-Queen (mod N)

- 편의상 $\{A_N\} = A_0, A_1, \cdots, A_{N-1}$이고, $0 \le A_i < N$으로 표현하겠습니다.
- 각 퀸의 대각선 방향 위치에 대응되는 수열 $B_i = (i + A_i) \bmod N$, $C_i = (i - A_i) \bmod N$ 모두 0부터 $N-1$까지의 순열입니다.
- 따라서 $\sum A_i$, $\sum B_i$, $\sum C_i$은 모두 $\frac{N(N-1)}{2}$입니다.
- $\sum C_i \equiv (\sum i) - (\sum A_i) \equiv 0 \pmod N$에서, $\frac{N(N-1)}{2} \equiv 0 \pmod N$입니다.
- 이를 만족하는 N은 2의 배수가 아닙니다.

B. NL-Queen (mod N)

- 다음으로, $\sum A_i^2$, $\sum B_i^2$, $\sum C_i^2$은 모두 $\frac{N(N-1)(2N-1)}{6}$입니다.
- $\sum (B_i^2 - C_i^2) \equiv 4\sum iA_i \equiv 0 \pmod N$에서 $\sum iA_i \equiv 0 \pmod N$입니다.
- $\sum i^2 = \sum B_i^2 \equiv \sum (i^2 + A_i^2) \equiv 0 \pmod N$에서 $\frac{N(N-1)(2N-1)}{6} \equiv 0 \pmod N$입니다.
- 이를 만족하는 N은 2, 3의 배수가 아닙니다.

B. NL-Queen (mod N)

- $\gcd(N, 6) = 1$인 경우에 2-선형 배치는 쉽게 만들 수 있습니다.
- $A_i = 2i \bmod N$으로 두면 됩니다.
- 2-선형 배치는 답이 될 수 없지만, 2-선형 배치와 2-선형 배치의 크로네커 곱은 k-선형 배치가 아닙니다.
- 따라서 $N = pq$이면서 $p, q > 1$이라면 $N = p$와 $N = q$에 대한 2-선형 배치의 크로네커 곱을 구하면 답이 됩니다.
- N을 $\mathcal{O}(N)$ 시간 안에만 인수분해한다면 전체 시간복잡도 $\mathcal{O}(N)$에 문제를 해결할 수 있습니다.

B. NL-Queen (mod N)

- N이 소수인 경우는 소수만의 특성을 이용해야 합니다.[2]
- 소수 법 N에 대해
  - 두 제곱 잉여의 혹은 두 제곱 비잉여의 곱은 항상 제곱 잉여입니다.
  - 제곱 잉여와 제곱 비잉여의 곱은 항상 제곱 비잉여입니다.
- 제곱 잉여의 집합 R과 서로 다른 원소 $r, s \in R$에 대해, A_i를 다음과 같이 정의합니다.

$$A_i = \begin{cases} ri \bmod N & (i \in R) \\ si \bmod N & (i \notin R) \end{cases}$$

B. NL-Queen (mod N)

- A_i는 순열임을 알 수 있습니다. $r, s \notin R$인 경우에도 성립합니다.
- B_i와 C_i 또한 순열이기 위해서는 $r-1$과 $s-1$, $r+1$과 $s+1$이 잉여이거나 비잉여인 여부가 쌍마다 같아야 합니다.
- 비둘기집 원리에 의해 $N-3 > 2^3$이라면 이러한 r, s가 항상 존재하고, 2^3개의 상태를 저장해 $\mathcal{O}(N)$에 찾을 수 있습니다.
- $N > 11$인 모든 N에 대해서 $\mathcal{O}(N)$에 문제를 해결할 수 있습니다.

B. NL-Queen (mod N)

$N = 5, 7, 11$은 완전탐색을 통해 불가능함을 알 수 있습니다.

C. 위잉위잉

`math, sorting, pythagoras`

출제진 의도 — **Gold**

- 제출 231번, 정답 26명 (정답률 11.688%)
- 처음 푼 참가자: **menborong**, 1시간 49분
- 출제자: `jh05013, kiwiyou`

C. 위잉위잉

- 주어진 사람들의 위치를 (각도, 원점으로부터의 거리) 쌍의 오름차순으로 정렬하면 됩니다.
- 각도를 기준으로 정렬하는 방법으로 다음 세 가지를 사용할 수 있습니다.
  - 64비트 이상의 정밀도를 가지는 atan2 사용
  - CCW를 사용
  - 기울기의 대소 비교

C. 위잉위잉

- 주로 쓰이는 방법은 CCW로, 좌표가 정수면 알고리즘도 정수만 사용한다는 장점이 있습니다.
- 단, 단순히 CCW만 판정하면 추이성($x < y < z \Rightarrow x < z$)이 성립하지 않기 때문에 정렬에 사용할 수 없습니다.
- 공간을 반평면 A와 B로 나눠서, 두 점이 서로 다른 반평면에 있으면 A 쪽이 B 쪽보다 먼저 오도록 판정하고, 아니면 CCW로 판정하면 됩니다.

E. 적은 메모리로 빠르게 많은 소수 찾기

`sieve`

출제진 의도 — **Diamond**

- 제출 51번, 정답 6명 (정답률 19.608%)
- 처음 푼 참가자: **jh01533**, 1일 8시간 37분
- 출제자: `jh05013`

E. 적은 메모리로 빠르게 많은 소수 찾기

- 소수를 찾는 문제는 매우 유명하기 때문에 많은 자료를 찾아볼 수 있습니다.
- 에라토스테네스의 체를 **정말, 정말 효율적으로** 구현하면 됩니다. 주요 최적화 요소는 다음과 같습니다.
  - 비트 배열
  - Segmented sieve
  - Wheel factorization
- 출제자의 풀이는 segmented sieve와 wheel factorization(2의 배수)를 사용하여 12초 정도에 답을 냅니다.

비트 배열

- bool은 1바이트, 즉 8비트입니다. 현대의 컴퓨터가 연산할 때는 딱 1비트만 가져오지 않기 때문입니다.
- bool 배열 대신 비트 배열을 사용하거나, 몇 개 단위로 비트마스크를 만들어 체를 관리하면 메모리를 8배 아낄 수 있습니다.
  - C++의 경우 `vector<bool>` 또는 `bitset`을 사용할 수 있습니다.
- 출제자 경험으로 시간 면에선 별 차이가 없었습니다.

Segmented sieve

- 가장 중요한 최적화 기법입니다. 이게 있어야 “적은 메모리"를 달성할 수 있고, 덤으로 속도 향상도 볼 수 있습니다.
- 기본적인 아이디어는 이 문제(스포 주의) 와 비슷합니다. 한번에 좁은 범위만 체로 거르는 것입니다.

E. 적은 메모리로 빠르게 많은 소수 찾기

- $X := \left\lfloor \sqrt{N} \right\rfloor$ 이하의 소수를 평범한 에라토스테네스 체로 모두 찾아냅니다.
- 그 다음, 적당한 상수 C를 잡아서
  - $[X+1, X+C]$ 구간에서 X 이하의 소수의 배수를 모두 걸러내면, 남은 수들은 모두 소수입니다.
  - $[X+C+1, X+2C]$ 구간에서 (위와 동일)
  - $[X+2C+1, X+3C]$ 구간에서 (위와 동일)
  - ...
- 출제자는 $C := 500000$을 사용했습니다.

E. 적은 메모리로 빠르게 많은 소수 찾기

- 최대 C 크기의 배열만 쓰기 때문에 메모리를 압도적으로 절약할 수 있습니다.
- 흥미롭게도, 이렇게 하면 캐시 히트가 잘 되어서 시간까지 절약됩니다.
  - 간단히 말하면, 컴퓨터는 서로 가까운 메모리에 접근하는 것을 좋아합니다. 이런 메모리 영역의 정보는 **캐시 메모리** 에 저장되어 빠르게 반복 접근할 수 있습니다.

E. 적은 메모리로 빠르게 많은 소수 찾기

Wheel factorization

- 4 이상의 짝수는 모두 합성수입니다.
- 어차피 짝수는 맨 처음에 다 걸러질 테니, 홀수만 가지고 체를 만들어도 됩니다.
- 시간과 메모리가 대충 2배 정도 절약됩니다.

E. 적은 메모리로 빠르게 많은 소수 찾기

- 한 발 더 나아가서, 3의 배수까지 미리 걸러내면 6으로 나눈 나머지가 1 또는 5인 수만 가지고 체를 만들 수도 있습니다. 메모리가 3배 절약됩니다.
  - 눈금이 6개인 바퀴의 1, 5번째 눈금에 마킹을 하고 바퀴를 굴리는 모습을 연상할 수 있어서 “wheel factorization"이라는 이름이 붙었습니다.
- 5의 배수까지 미리 걸러내면 30으로 나눈 나머지가 1, 7, 11, 13, 17, 19, 23, 29인 수만 가지고 체를 만들 수도 있습니다. 메모리가 3.75배 절약됩니다.
- 미리 걸러내는 범위가 커질수록 무거워지면서 메모리 절약 효과도 덜 늘어나기 때문에, 2, 3, 5, 7 정도가 가장 좋다고 합니다.[3]

F. Queens' Rye Cafe

`extended_euclidean, number_theory`

출제진 의도 — **Platinum**

- 제출 15번, 정답 5명 (정답률 33.333%)
- 처음 푼 참가자: **ychangseok**, 5시간 50분
- 출제자: `kiwiyou`

F. Queens' Rye Cafe

- 수열 A_N은 패리 수열로, 인접한 두 기약분수 $\frac{a}{b} < \frac{c}{d}$에 대해 $ad - bc = 1$입니다.
- 또한 $\frac{a}{b} < \frac{p}{q} < \frac{c}{d}$가 인접한 세 기약분수라면 $\frac{p}{q} = \frac{a+c}{b+d}$입니다.
- a와 b를 아는 경우에 c와 d를 결정하는 문제는 확장 유클리드 호제법을 통해 $\mathcal{O}(\log \max(a, b))$에 구할 수 있습니다.
- 이때 가능한 한 $\frac{c}{d}$를 $\frac{a}{b}$에 가깝게 만들어야 하는데, $d \le N$인 범위 내에서 $c := c + a$, $d := d + b$를 동일 횟수만큼 반복하면 됩니다.

F. Queens' Rye Cafe

- 이전 두 항을 아는 경우 다음 항을 $\mathcal{O}(1)$에 계산할 수 있습니다.
- 인접한 세 기약분수 $\frac{a}{b} < \frac{c}{d} < \frac{p}{q}$에 대해서 $\frac{c}{d} = \frac{a+p}{b+q}$이므로, $p = kc - a$, $q = kd - b$입니다.
- $kd - b$는 N 이하이면서 가장 커야 하므로, $k = \left\lfloor \frac{N+b}{d} \right\rfloor$입니다.
- 전체 시간복잡도 $\mathcal{O}(j + \log N)$에 문제를 풀 수 있습니다.

G. Enigmatic Device 2023

`math, lazy_segment_tree, sqrt_decomposition`

출제진 의도 — **Diamond**

- 제출 20번, 정답 6명 (정답률 35%)
- 처음 푼 참가자: **cozyyg**, 2시간 4분
- 출제자: `bubbler`

G. Enigmatic Device 2023

- $f(x) = x^2 \bmod 2023$ 이라고 정의합시다.
- 그러면 $f^{12}(x) = f^4(x)$가 성립합니다.
- 이는 $0 \le x < 2023$인 모든 정수 x에 대해 f를 반복 적용하여 사이클을 분석해 보면 알 수 있습니다.

G. Enigmatic Device 2023

- 이를 이용하면 느리게 갱신되는 세그먼트 트리에 필요한 연산을 정의할 수 있습니다.
- 먼저, 세그먼트 트리의 리프에는 배열 $[a_i, f(a_i), f^2(a_i), \cdots, f^{11}(a_i)]$를 저장합니다.
- 그리고 두 노드 $[x_0, \cdots, x_{11}]$과 $[y_0, \cdots, y_{11}]$을 합친 상위 노드에는 $[x_0 + y_0, \cdots, x_{11} + y_{11}]$을 저장합니다.

G. Enigmatic Device 2023

- 갱신 함수에는 1번 쿼리가 적용된 횟수를 저장합니다. 1번 쿼리가 적용될 때마다 1씩 증가시키고, 적용 횟수가 12가 되면 4로 되돌려줍니다.
- 어떤 세그먼트 트리 노드 $[x_0, \cdots, x_{11}]$에 갱신 함수 1을 적용하면 $[x_1, \cdots, x_{11}, x_4]$로 바뀝니다. 갱신 함수의 값이 k이면 이를 k번 적용한 결과로 업데이트하면 됩니다.

G. Enigmatic Device 2023

- 비슷한 방법으로 평방분할을 이용해서도 풀 수 있습니다.
- 길이 n의 배열을 길이 m의 구간으로 나누고, 배열 X, Y, Z를 다음과 같이 정의합니다.
  - $X[i][j] = f^j(a_i)$
  - $Y[k][j] = \sum_{i \in S_k} X[i][j]$ (S_k는 k번째 구간의 배열의 인덱스의 집합)
  - $Z[k] = k$번째 구간 전체가 업데이트된 횟수

G. Enigmatic Device 2023

- k번째 구간의 일부분만 업데이트하려면 k번째 구간에 해당하는 X 배열을 새로 계산하고, 그에 따라 Y도 새로 계산해줍니다.
- k번째 구간 전체를 업데이트하려면 $Z[k]$를 1 증가시키면 됩니다. 역시 $Z[k]$가 12가 되면 4로 바꿔줍니다.
- 구간 합 쿼리를 처리하려면 구간의 일부에 대해서는 해당 구간의 $X[i][Z[k]]$들을 더해주고, 구간 전체에 대해서는 $Y[k][Z[k]]$를 더하면 됩니다.

H. A+B - 10 (제1편)

`brute_force, arithmetic`

출제진 의도 — **Bronze**

- 제출 438번, 정답 235명 (정답률 57.306%)
- 처음 푼 참가자: **kdg870918**, 15분
- 출제자: `jh05013`

H. A+B - 10 (제1편)

- 예시 코드에 A를 구하는 과정이 나와 있습니다. 1부터 9까지 차례대로 A가 그 수인지 물어보고 답을 얻는 것을 확인할 수 있습니다.
- 같은 방법으로 1부터 9까지 차례대로 물어보면서 B를 구하면 됩니다.

H. A+B - 10 (제1편)

이 문제를 만든 이유는 크게 세 가지입니다.

- 상호작용 문제 입문을 더 쉽게 만들기: 상호작용 문제를 풀려면 기본적으로 알아야 하는 정보가 꽤 있습니다. 그런 정보를 이 문제를 통해 제공하고자 했습니다.
- 외래어 순화: “interactive"와 “adaptive"의 번역어를 제안하였고, 특히 “적응적"은 존재하지 않는 단어이기 때문에 “적응하는"을 사용하였습니다. `flush`는 함수 이름으로도 쓰이기 때문에 번역하지 않았습니다.
- 더 쉬운 `flush` 방법 소개: 사실 C++, Python 등의 여러 주요 언어에서는 `flush()`를 직접 호출할 필요가 없습니다. 예시 코드를 참조하세요.

H. A+B - 10 (제1편)

예시 코드의 “이 부분은 직접 채워보자.” 뒤에 이모지를 넣고 싶었으나, 아쉽게도 코드포스가 이모지를 허용하지 않습니다. 🦐🦀🍲

J. 짧은 코드로 빠르게 많은 소수 세기

`dp, sieve`

출제진 의도 — **Diamond**

- 제출 92번, 정답 16명 (정답률 17.391%)
- 처음 푼 참가자: **p_ce1052**, 3시간 1분
- 출제자: `jh05013`

J. 짧은 코드로 빠르게 많은 소수 세기

- 소수를 세는 문제는 매우 유명하기 때문에 많은 자료를 찾아볼 수 있습니다.
- 한국어 자료는 https://rkm0959.tistory.com/189 에서 볼 수 있습니다. $O\left(n^{\frac{2}{3}}\right)$까지 갈 필요는 없고, Lucy_Hedgehog 알고리즘을 map이 아닌 배열로 구현하면 충분합니다.
  - Lucy와 Hedgehog 두 명의 이름을 딴 게 아니라, 프로젝트 오일러 유저 Lucy_Hedgehog의 아이디를 딴 것입니다. 하이픈이 아니라 언더스코어입니다.
- 같은 방법으로 소수의 k제곱의 합, $4k+1$ 꼴의 소수의 합 등도 계산할 수 있습니다.

J. 짧은 코드로 빠르게 많은 소수 세기

- 코드포스에 있는 4.4초짜리 코드를 찾으셨다면, 그 코드는 비효율적인 부분이 꽤 있기 때문에 2.4초 정도까지 줄여보시는 것을 권합니다.
- 코드 길이 제한이 있는 이유는 DB 풀이를 막기 위함입니다.
  - Lucy_Hedgehog 알고리즘은 구현이 매우 간단하기 때문에, 3,000 바이트는 널널한 제한입니다. 출제자의 구현은 29줄입니다.

K. 수열 선물받기

`permutation_cycle_decomposition, ad_hoc, constructive`

출제진 의도 — **Gold**

- 제출 53번, 정답 19명 (정답률 39.623%)
- 처음 푼 참가자: **p_ce1052**, 1시간 53분
- 출제자: `hibye1217`

K. 수열 선물받기

- 주어진 순열을 사이클로 분할해봅시다.
- 즉, 사이클 $C = C_1 \to C_2 \to C_3 \to \cdots \to C_N \to C_1$에 대해서 생각해봅시다.
  - C_1은 P_{C_2}에 들어가야 하고, C_2는 P_{C_3}에 들어가야 하고, ...를 의미합니다.
- 그럼, 하나의 사이클 C를 빠르게 처리할 수 있을까요?

K. 수열 선물받기

- 사이클의 크기가 1이라면 아무런 처리가 필요없습니다.
- 사이클의 크기가 2 이상이라면 아래와 같이 처리할 수 있습니다.
  - C_1 1 N: P_{C_1}의 값을 $\operatorname{mex}(P) = 0$으로 바꿉니다.
  - C_2 1 N: P_{C_2}의 값을 $\operatorname{mex}(P) = P_{C_1}$로 바꿉니다.
  - C_3 1 N: P_{C_3}의 값을 $\operatorname{mex}(P) = P_{C_2}$로 바꿉니다.
  - $\vdots$
  - C_k 1 N: P_{C_k}의 값을 $\operatorname{mex}(P) = P_{C_{k-1}}$로 바꿉니다.
  - C_1 1 N: P_{C_1}의 값을 $\operatorname{mex}(P) = P_{C_k}$로 바꿉니다.

K. 수열 선물받기

- 사이클의 처리가 끝난 뒤의 P는 다시 순열이 되므로, 위 연산은 항상 제대로 작동합니다.
- 연산 횟수는 사이클당 $|C| + 1$이고, 연산이 일어나는 최소 사이클의 크기가 2이므로 총 연산 횟수는 최대 $\left\lfloor \frac{3}{2}N \right\rfloor$회가 됩니다.

K. 수열 선물받기

- ♻ 여담으로, 원래 처음 문제를 만들었을 때는 연산 횟수 제한이 $2N$이었습니다.
- ♻ 그러다가 $\left\lceil \frac{3}{2}N \right\rceil$회를 거쳐 지금의 제한이 되었습니다.

🧩. 🧩 N-Queen (Area)

`geometry, math`

출제진 의도 — **Platinum**

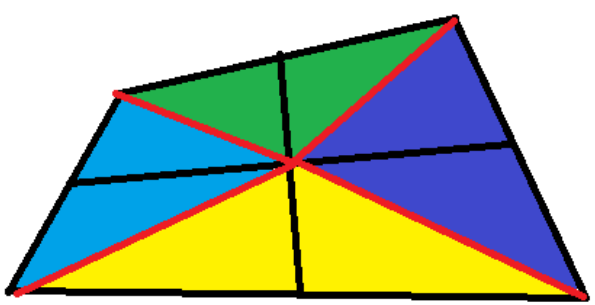
- 제출 25번, 정답 12명 (정답률 52%)
- 처음 푼 참가자: **menborong**, 3시간 12분
- 출제자: `wider93`

N-Queen (Area)

- 사각형 $PQRS$의 사각형의 넓이는 $\frac{|(P-R) \bigotimes (Q-S)|}{2}$로 구할 수 있습니다.
- 사각형을 문제에 주어진 대로 나누어서 선분들의 교점을 구하고 위 식을 사용해서 더하는 방식으로 작은 n에 대해 브루트포스 탐색을 돌려보면 **모든 배치에서 넓이의 합이 같음을** 관찰할 수 있습니다.
- 이대로 믿음의 제출을 하면 맞았습니다를 받습니다.
- 증명을 시도해 봅시다.

N-Queen (Area)

- 코드를 돌려보면, 사실 한 행이나 열에서 두 사각형을 택하지 않는다는 조건만 만족해도 항상 넓이가 같다는 것을 관찰할 수 있습니다.
- 먼저 작은 경우를 증명해 봅시다. $n = 2$인 경우에 방법이 정확히 두 가지가 있습니다. 좌상+우하와 좌하+우상, 두 방법이 넓이가 같음을 보이면 됩니다.
- 그림으로 이에 대한 증명을 대신합니다.



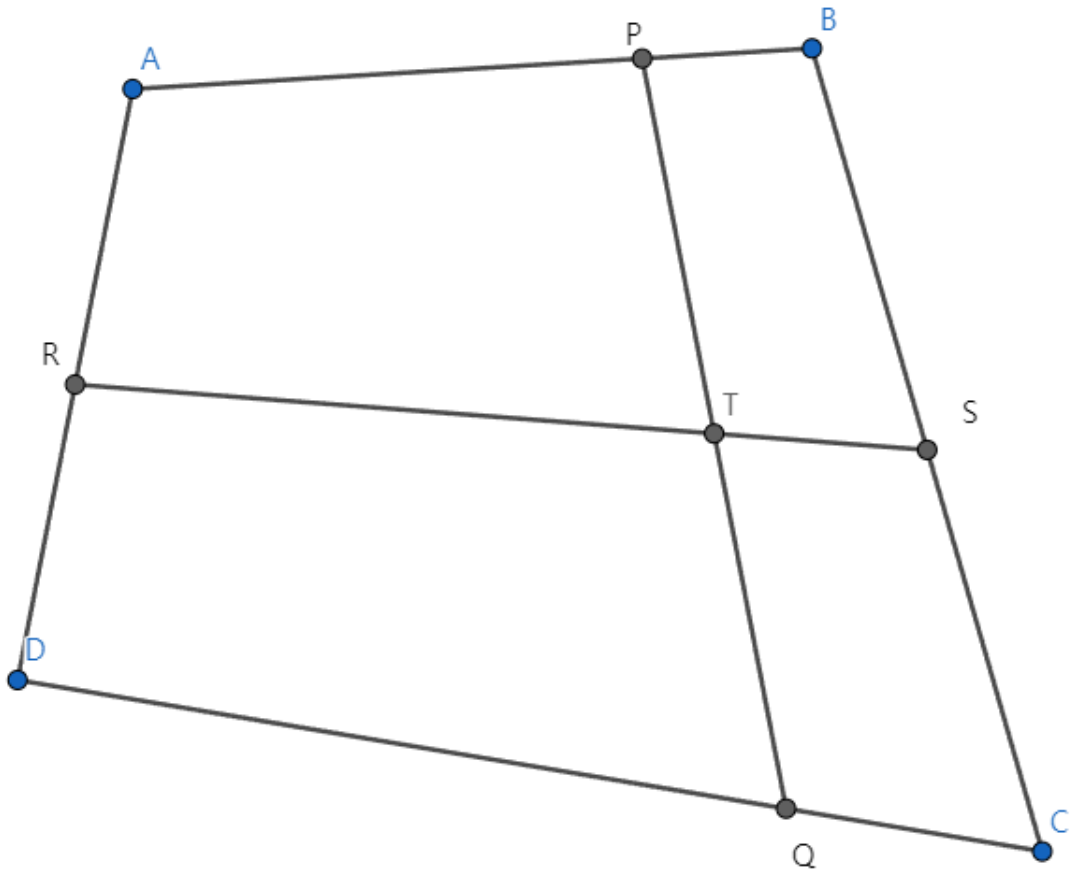
⁂. ⁂ N-Queen (Area)

- 큰 n에 대해 서술하기 위해 사각형에 좌표를 부여합시다. 가령 사각형 $ABCD$의 선분 AB와 DC를 m등분하고 AD와 BC를 n등분한 경우 $(1,1),(m,1),(m,n)$ 등의 좌표를 가진 작은 사각형들이 존재합니다.
- (i,j)의 좌표에 있는 사각형의 넓이를 $S(i,j)$로 표기합시다.
- 임의의 순열은 인접호환들의 곱으로 나타낼 수 있습니다.
- 따라서, $S(i,a)+S(i+1,b)$와 $S(i,b)+S(i+1,a)$가 임의의 $1 \leq i < n$에 대해서 같다는 것을 보이면, 이는 인접한 열의 선택을 넓이 변화 없이 바꾸어줄 수 있다는 뜻이고, 모든 순열의 넓이의 합이 같다는 것이 증명됩니다.

N-Queen (Area)

- 임의의 실수 $0 < t, s < 1$에 대해,
  - 사각형 $ABCD$의 변 AB를 $t : 1 - t$로 등분하는 점을 P,
  - 변 DC를 $t : 1 - t$로 등분하는 점을 Q,
  - 변 AD를 $s : 1 - s$로 등분하는 점을 R,
  - 변 BC를 $s : 1 - s$로 등분하는 점을 S라고 합시다.
- 선분 PQ와 RS의 교점을 T라고 합시다. 그러면 T는 선분 PQ의 $s : 1 - s$ 내분점이자 RS의 $t : 1 - t$ 내분점임을 증명할 수 있습니다.

N-Queen (Area)



- 이는 다음이 성립하기 때문입니다:
  - $P = tB + (1-t)A, Q = tC + (1-t)D$
  - $R = sD + (1-s)A, S = sC + (1-s)B$
  - $T_1 = tS + (1-t)R$은 RS의 $t : 1-t$ 내분점
  - $T_2 = sQ + (1-s)P$은 PQ의 $s : 1-s$ 내분점
  - $T_1 = (1-t)(1-s)A + t(1-s)B + tsC + (1-t)sD = T_2$
- 따라서 마주보는 변을 이은 모든 변은 동일하게 n등분되어 있습니다.

7. N-Queen (Area)

- 이제 $n \times 2$ 사각형의 각 2×2짜리 부분 사각형이 문제의 세팅과 같으므로 기저 케이스의 사실을 사용할 수 있습니다. 즉 $S(i,j) + S(i+1,j+1) = S(i+1,j) + S(i,j+1)$이 성립합니다.
- $j = a, \cdots, b-1$에 대해 위 식을 더하고 같은 값을 소거하면 원하는 식인 $S(i,a) + S(i+1,b) = S(i,b) + S(i+1,a)$를 얻을 수 있습니다.
- 따라서 각 순열에 대해 넓이의 합이 같음이 증명되었습니다.
- 따라서 답은 사각형 $ABCD$의 넓이의 $\frac{1}{n}$입니다.

M. 커피우유

`geometry_3d, polynomial_interpolation`

출제진 의도 — **Diamond**

- 제출 6번, 정답 6명 (정답률 100%)
- 처음 푼 참가자: **qwerasdfzxcl**, 1일 14시간 24분
- 출제자: `kiwiyou`

M. 커피우유

- 어떤 격자 다면체 P의 각 꼭짓점 좌표를 k배한 다면체 내부의 격자점 개수 $f_P(k)$는 삼차 Ehrhart 다항식으로 나타납니다.
- 다항식의 계수는 $k = 0, 1, 2, 3$일 때 $f_P(k)$의 값을 직접 계산하여 구할 수 있습니다.
  - 범위 내 모든 격자점에 대해 다면체에 포함되는지 여부를 검사하면 됩니다.
- 계수를 구한 뒤에는 쿼리를 $\mathcal{O}(1)$에 처리할 수 있습니다.
- $k = 0, 1, 2, 3$을 대입했기에 계수의 분모는 최대 12이고, 법 1003003001에 대해 역수가 존재합니다.

O. A+B - 10 (제2편)

`brute_force, randomization`

출제진 의도 — **Silver**

- 제출 703번, 정답 137명 (정답률 21.764%)
- 처음 푼 참가자: **kdg870918**, 34분
- 출제자: `jh05013`

O. A+B - 10 (제2편)

- 일단 20,000번이 아니라 19,998번만 질문해도 되는 이유는 무엇일까요?
- A에 대해 1, 2, ...을 차례대로 물어봤을 때, 9,999까지 "아니요"라는 답변을 받았다면 A는 10,000일 것이고, 10,000까지 물어볼 필요가 없기 때문입니다. 각 변수를 9,999번의 질문으로 알아낼 수 있으므로 총 19,998번 질문하면 됩니다.
- 하지만 이 전략은 A와 B가 모두 10,000이라면 통하지 않습니다.

O. A+B - 10 (제2편)

- A가 10,000인지 먼저 물어보고, 아니라면 1부터 차례대로 질문하면 어떨까요? 이 경우에는 A가 9,999, B가 10,000이라면 통하지 않습니다.
- 물어보는 순서를 어떻게 하든 이 전략이 통하지 않는 (A, B) 쌍이 하나 있습니다.
- 하지만 테스트케이스가 총 10개라고 했으니, 실제로 막히는 (A, B) 쌍은 겨우 10개입니다. 따라서 순서를 아무렇게나 정하고 "운이 좋으면" 통과합니다.

O. A+B - 10 (제2편)

- 구체적으로, 각 A와 B에 대해 마지막으로 시도할 수를 랜덤으로 정합시다.
- A의 값을 알려고 할 때, “마지막으로 시도할 수"만 제외하고 차례대로 물어봅니다. 모든 대답이 “아니요"였다면 A는 그 “마지막으로 시도할 수"와 같습니다. B도 마찬가지입니다.
- 이 전략이 막히려면 기막히게도 실제 A와 B의 값을 “마지막으로 시도할 수"로 정했어야 합니다. 각 수가 랜덤으로 뽑힐 확률이 같다고 가정하면, 그럴 확률은 $\left(\frac{1}{10000}\right)^2$ 밖에 되지 않습니다.

O. A+B - 10 (제2편)

♻ 이렇게 상호작용 문제의 채점기가 **적응하지 않는다면** 랜덤을 시도해볼 수 있습니다. 자주 나오는 패턴이니 기억해 둡시다.

참고문헌

[1] T. Kløve, “The modular n-queen problem”, Discrete Mathematics, vol 19, 호 3, pp 289–291, 1977, doi: https://doi.org/10.1016/0012-365X(77)90110-8.

[2] A. Bruen 와/과 R. Dixon, “The n-queens problem”, Discrete Mathematics, vol 12, 호 4, pp 393–395, 1975, doi: https://doi.org/10.1016/0012-365X(75)90079-5.

[3] kimwalisch, “primesieve”. [Online]. Available at: https://github.com/kimwalisch/primesieve