

ICPC Sinchon
Summer Camp 2021
Final Exam Intermediate

Official Solutions

문제	의도한 난이도	출제자
A permutation making	Easy	강효규서강대 djs100201
B 짝수싫어수	Medium	김도현홍익대 swoon
C 안산 탐지기	Medium	이국렬연세대 lky7674
D 신촌 수열과 쿼리	Hard	강효규서강대 djs100201
E 미팅	Hard	이국렬연세대 lky7674
F 여행사 운영하기	Hard	전해성서강대 seastar105
G 신촌방위본부	Challenge	이국렬연세대 lky7674

중급

A. permutation making

출제자 : djs100201(강효규, Sogang ICPC Team)

가장 먼저 푼 사람 : 00:04 pom0319(연세대학교)

프리즈 정답률 : 53.2%(17정답 32제출)

- 초급 C번과 동일한 문제입니다.
- 쉬운 constructive + greedy 유형입니다.
- 원래 문제는 $n/2$ 개 이상이 되도록 만드는 거였으나 random 풀이를 막고자
지금과 같이 변경되었습니다.

- 핵심적인 아이디어는 합이 n 이 되는 순열의 두 원소를 연속하게 이어 붙이는 것입니다.
- $n, 1, n-1, 2, n-2, 3, n-3 \dots$
- 이런식으로 순열 A 를 배열하게 되면, P 는 $0, 1, 0, 2, 0, 3, 0 \dots$ 으로 조건을 만족하게 됩니다.

중급

B. 짝수 싫어수

출제자 : SWOON(김도현, HI-ARC)

가장 먼저 푼 사람 : 00:60 portal3046(연세대학교)

프리즈 정답률 : 66.67%(2정답 3제출)

B. 짝수싫어수

- 3, 5, 7의 개수가 각각 짝수인지 홀수인지만 구분하면 됩니다.
- 비트마스킹을 이용하여 상태를 bit로 관리합니다. bit = 5 = 101₂ => 3, 7의 개수가 홀수개
- dp[i][bit]는 i자리이며 상태가 bit인 수의 개수를 나타냅니다.
- 반복문으로 dp테이블을 채워줍니다.
- 예제의 맥스데이터를 근거로 dp테이블의 각 값이 long long 범위를 넘어가지 않음을 알 수 있습니다.

B. 짝수싫어수

7	5	3	3	3										
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--



- 현재 위치에 어느 숫자를 뒤야 할 지 계산합니다.
- 앞의 bit값과 현재 bit값과 이후의 bit값을 계산했을 때, 가능한 값만 cnt에 더해줍니다.
- 앞의 bit값이 111_2일 때, 현재 7를 넣으려면
이후에 3이 홀수번, 5가 홀수번, 7이 짝수번 나오면 안 된다.
- 이는 $111_2(\text{앞bit}) \wedge 100_2(\text{현재bit})$ 가 이후bit와 xor해서 0이 나오면 안 된다는 뜻이다.

B. 짝수 싫어수

7	3	3	5	3										
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--



- 따라서 앞 bit값, 현재 bit값, 이후의 bit값을 xor한 것이 0이면 안 된다.
- 0이 아닌 경우에만 cnt에 더해줍니다.
- 만약 cnt가 k보다 크거나 같으면 현재 자리수에 해당 숫자를 적고 다음으로 갑니다.
- k미만인 경우 k에서 cnt를 빼주고 이번 칸에 적을 수가 없을 경우 적지 않고 다음으로 갑니다.

중급

C.안산 탐지기

출제자 : lky7674(이국렬, 모르고리즘)

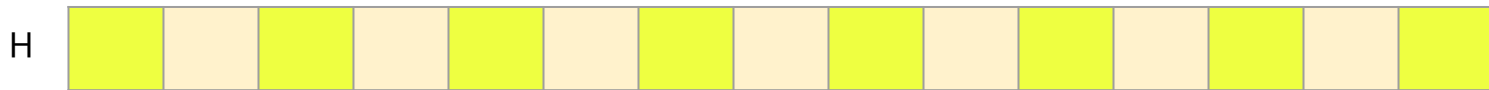
가장 먼저 푼 사람 : ???

프리즈 정답률 : 00.0%(0정답 7제출)

- 저번 캠프 콘테스트에 이어서 나온 인터랙티브 문제입니다.
- 마찬가지로 분할 정복 문제로 가져왔습니다.

- 가장 높은 봉우리의 높이가 얼마인지 모른다.
- 1번째 탐지기로 배터리 N 개를 사용해서 우선 가장 높은 봉우리의 높이를 구해보자.

C. 안산 탐지기



- 이후 2번째 탐지기를 통해서 가장 높은 봉우리가 홀수 번째에 위치한지, 짝수 번째에 위치한지를 판단해보자.
- 만약에 홀수 번째의 칸 수가 더 많은 경우, 짝수 번째 칸을 탐지해야 사용하는 배터리의 개수를 줄일 수 있다.

C. 안산 탐지기



- 가장 높은 봉우리가 없는 칸은 무시하자.
- 무시하고 나면 칸 수는 절반씩 줄어든다.
- 2번째 탐지기 이후에 4번째, 8번째, ...를 계속 사용해서 가장 높은 봉우리의 위치를 알아내면 된다.

C. 안산 탐지기



- 탐지기 사용 횟수 : $1(\text{1번째 탐지기 사용}) + \log N \leq 20$
- 배터리 최대 사용 횟수 : $N(\text{1번째 탐지기 사용}) + (N - 1) = 2N - 1$

- i 번째 탐지기를 사용할 때, i 가 N 을 넘어가지 않도록 주의해야 한다.
- 마찬가지로 탐지기의 범위가 가장 오른쪽 산 너머로 넘어가지 않도록 주의해야 한다.

중급

D. 신촌 수열과 쿼리

출제자 : djs100201(강효규, Sogang ICPC Team)

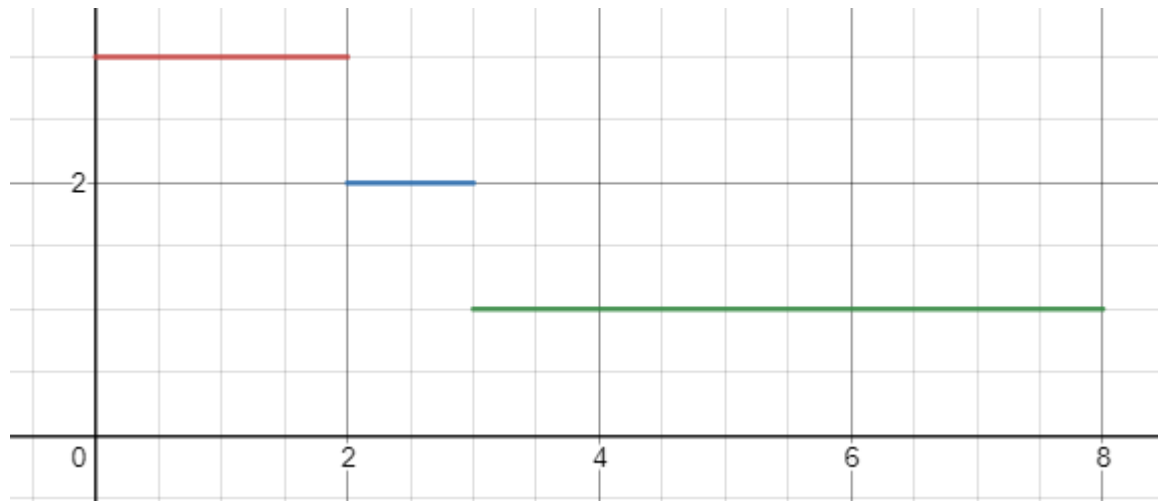
가장 먼저 푼 사람 : 00:37 youx(연세대학교)

프리즈 정답률 : 21.7%(5정답 23제출)

- segment tree 로 해결할 수 있는 수 많은 문제중 parametric search와 결합하여 해결할 수 있는 문제들이 있습니다.
- 그런 문제들의 기본형으로, 세그트리를 대놓고 활용하는 educational 한 문제입니다.
- segment tree 와 parametric search로 $O(M \log^2 N)$ 에 해결할 수 있습니다.

- 우리는 구간안의 모든 원소가 j 이상인 구간합의 최댓값을 구해야 합니다.
- parametric search를 이용하면 해결할 수 있습니다.
- 구간 $[i, R]$ 의 최솟값이라는 함수는 i 가 고정되어 있을때, R 의 증가에 대해 단조 감소 하는 함수입니다. 마찬가지로 $[L, i]$ 라는 함수 또한 L 의 감소에 따라 단조 감소합니다.
- 또 구간 $[i, R]$ 에서의 합이라는 함수도 R 의 증가에 따라 단조 증가 합니다. 마찬가지로 $[L, i]$ 라는 구간 합이라는 함수도 L 의 감소에 따라 단조 증가 합니다.

● 구간 최소 함수 예시



- 따라서 점 i 를 고정하면 구간의 최솟값이 j 이상인지에 따라 오른쪽 방향에 대해 parametric search, 왼쪽 방향에 대해 parameteric search를 각각 진행하면서, 가능한 구간의 최댓값을 구할 수 있습니다.
- 업데이트도 segment tree이기 때문에 쉽게 해결 가능합니다.

- 검수진 몇 분에서 $O(M \log N)$ 세그먼트 트리 풀이를 성공하셨습니다.
- parametric search를 하지 않고, min 세그의 노드를 순회하면서 구간의 양 끝을 탐색할 수 있습니다.
- $O(M \log^2 N)$ 풀이를 막으면 많은 분들이 고통이 예상되어 제한을 줄이지는 않았습니다.
- 평방분할을 이용한 $O(M \sqrt{N})$ 풀이도 존재합니다.

중급

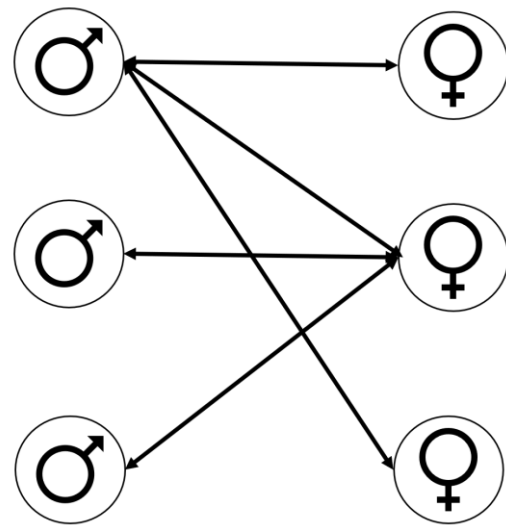
E.미팅

출제자 : lky7674 (이국렬,모르고리즘)

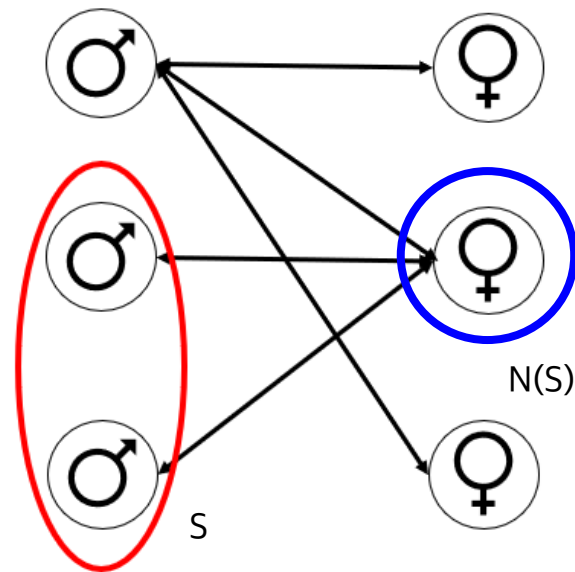
가장 먼저 푼 사람 : 01:45 luciaholic(연세대학교)

프리즈 정답률 : 11.11%(1정답 9제출)

- 지문에 주어진 그림과 같이 남성을 왼쪽 정점, 여성을 오른쪽 정점, 서로 호감이 있는 관계쌍을 간선으로 표현하면 이분 그래프가 나온다.



- 왼쪽 정점의 부분 집합 S 에 이웃하는 오른쪽 정점의 집합을 $N(S)$ 라고 해보자.
- 그렇다면 $|S| > N(S)$ 를 만족하는 집합 S 를 찾으면 된다.



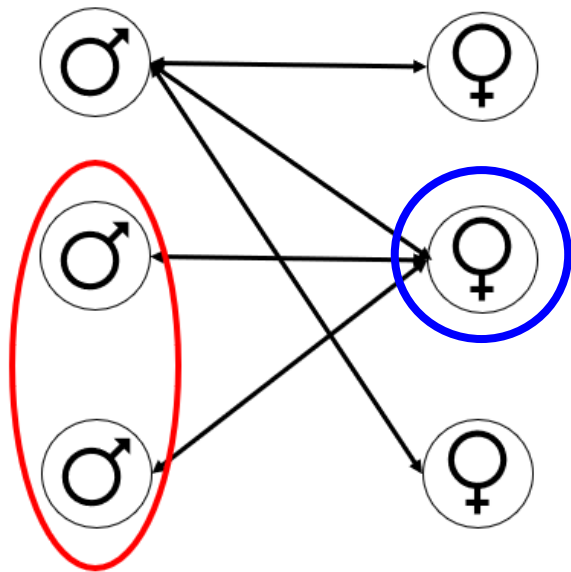
- 11회차 강의 내용 중 홀의 결혼 정리를 보자.

Perfect Matching

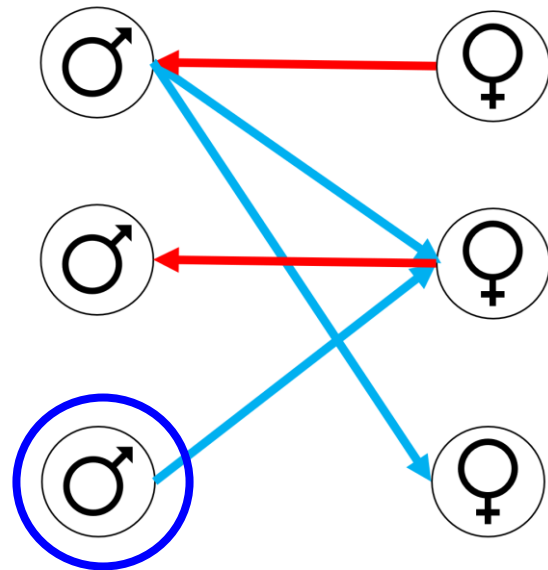
홀의 결혼 정리 : 이분 그래프 $G = (V = L \cup R, E)$ 가 있고, $S \subseteq L$ 에 대해서 S 에 속한 정점 중 최소 1개와 이웃한 정점의 집합을 $N(S)$ 라고 하자. 이 때, 다음 2개의 조건이 동치다.

- 모든 $S \subseteq L$ 에 대해서 $|S| \leq |N(S)|$ 를 만족한다.
- G 의 Perfect Matching이 존재한다.

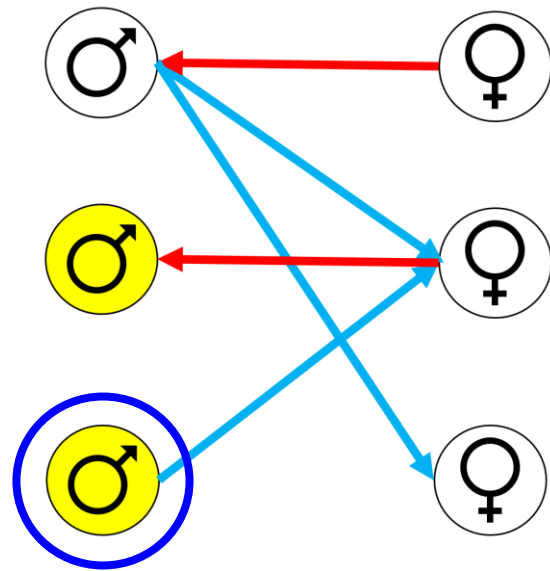
- 홀의 결혼 정리에 의해서, Perfect Matching이 없는 경우 $|S| > |N(S)|$ 를 만족하는 왼쪽 정점의 부분 집합 S 는 무조건 존재한다.
- 이제 S 를 직접적으로 construct 하는 과정만 남았다.



- 우선 주어진 이분 그래프에서 최대 매칭을 찾고, 이에 대한 residual graph를 만든다.
- S 가 존재한다면, 매칭에 속하지 않은 왼쪽 정점은 존재한다.



- 매칭에 속하지 않은 왼쪽 정점을 아무거나 선택한다.
- 해당 정점으로부터 도달할 수 있는 왼쪽 정점들의 집합을 S 로 선택하면 된다.
- 만약에 $|S| \leq |N(S)|$ 라면, 매칭에 속하지 않은 왼쪽 정점은 augment path를 통해서 매칭에 속할 수 있기 때문.
- 이분 매칭 알고리즘 : $O(NM) = O(N^3)$



중급

F.여행사 운영하기

출제자 : seastar105 (전해성, Sogang ICPC Team)

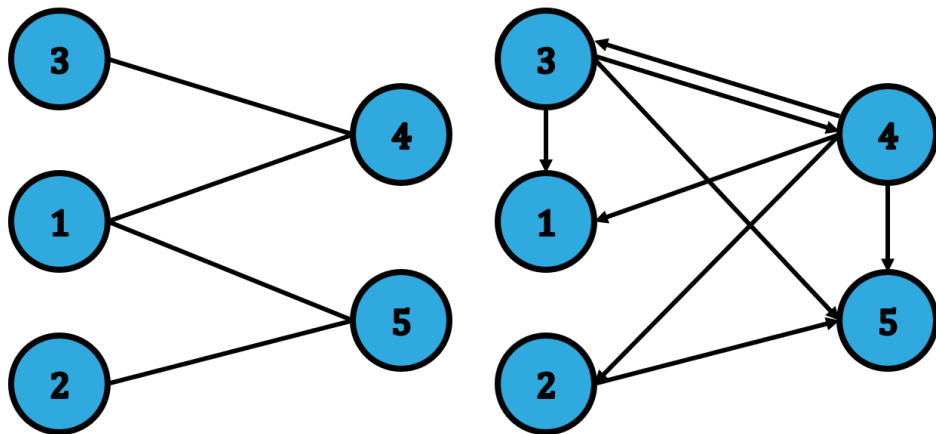
가장 먼저 푼 사람 : ???

프리즈 정답률 : 0.0%(0정답 0제출)

문제 요약 :

- 도시는 정점, 도로는 간선으로 보면 주어지는 도시들은 크기가 N 인 가중치 트리입니다.
- 각 도시 별로 즐거움 c_i 가 있으며 해당 도시에서 출발했을 때 d_i 만큼만 멀어질 수 있습니다. 이 때, 각 도시별로 도달 가능한 도시들의 c_i 의 최댓값과 최솟값의 차이를 구해야 합니다.

- 한 도시에서 출발한 뒤에 다른 도시에 내려 그 도시에서 출발하는 것은 가능합니다.
- 모든 도시 별로 자기 도시와의 거리가 d_i 이하인 도시로 유행 간선을 이어준 그래프를 G' 라고 합시다.
- 도시 i 에서 도달 가능한 도시는 G' 에서 i 에서 도달 가능한 도시와 동일합니다.



- G'의 정점의 수는 N개로 동일하지만 간선의 수는 최대 N^2 개로 모든 정점에서 DFS/BFS를 수행하여 도달 가능한 도시를 찾는 것은 시간초과를 받습니다.
- 이를 줄여봅시다.

- G' 에서 도시 u 와 도시 v 가 같은 SCC에 속한다면 두 도시가 도달 가능한 도시의 집합은 동일합니다.
- G' 의 SCC를 구하고 SCC들을 노드로 하고 서로 다른 SCC에 속하는 정점간에 간선이 있었다면 두 정점이 속하는 SCC 사이에 간선을 이어주어 새로운 그래프 G'' 를 만듭니다.

- 이렇게 만들어진 G'' 는 DAG(Directed Acyclic Graph)이기 때문에 해당 그래프 내에서 DP를 생각해볼 수 있습니다.
- G'' 의 각 노드 v 에서 도달가능한 도시의 최대, 최소 즐거움을 각각 $c_{\max}(v)$, $c_{\min}(v)$ 라고 합시다.
- G'' 의 위상정렬의 역순으로 위 값들을 계산하면 모든 값을 구할 수 있습니다.
- 따라서 문제를 $O(N^2)$ 에 해결할 수 있습니다.

중급

G.신촌방위본부

출제자 : lky7674 (이국렬, 모르고리즘)

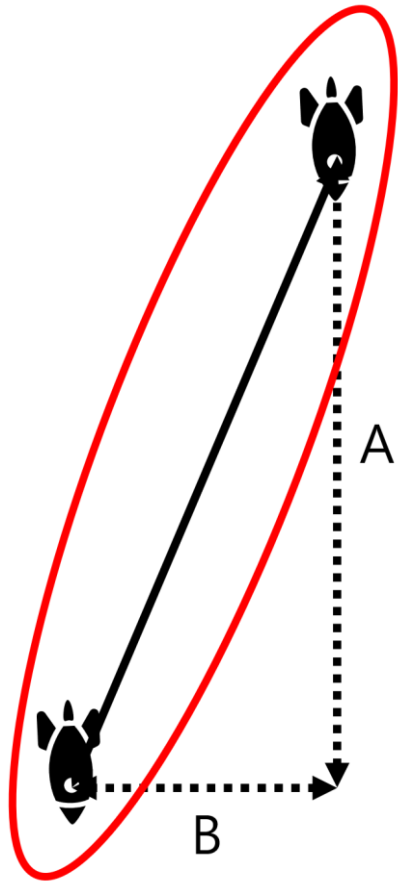
가장 먼저 푼 사람 : ???

프리즈 정답률 : 0.0%(0정답 1제출)

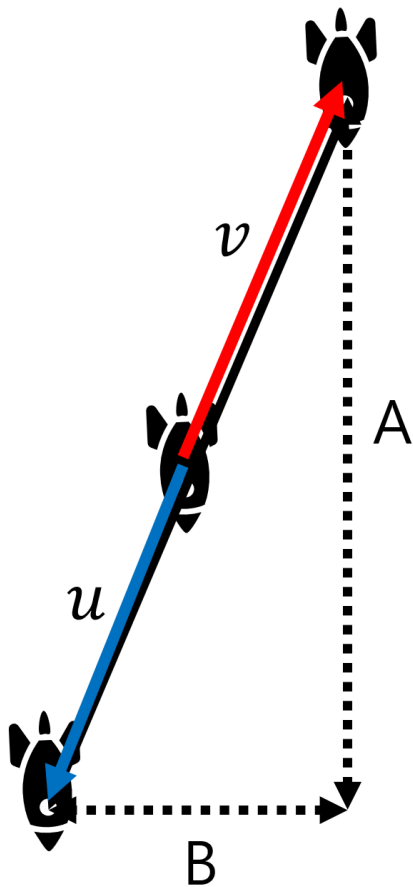
- 문제 요약
- N 개의 점으로 구성된 convexhull이 주어짐.
- M 개의 점이 주어짐.
- M 개의 점을 제외한 convexhull에 속한 격자점의 개수를 구하는 문제.

- convexhull의 꼭짓점의 개수가 2개 이하인 경우는 따로 빼줘야 한다.
- 꼭짓점의 개수가 1개인 경우는 그 꼭짓점과 같은지, 다른지만 판별하면 된다.

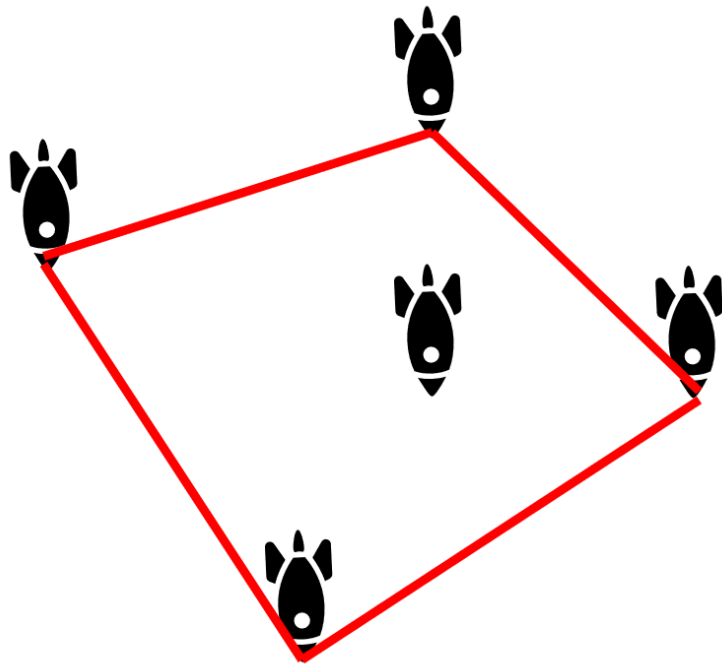
- convexhull의 꼭짓점의 개수 = 2
 - 선분을 의미하는 것
- x축 방향의 변화량 = A, y축 방향의 변화량 = B
- 한 선분 위의 격자점의 개수 = $\gcd(|A|, |B|) + 1$



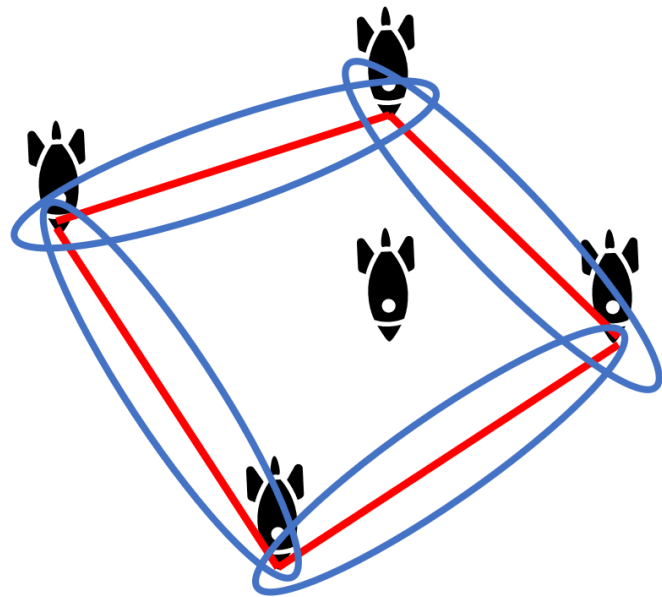
- M개의 점 별로, 선분 위에 있는지 판별하면 된다.
- $|u \times v| = 0$ 이고, 해당 점의 x좌표랑 y좌표가 사이에 있는지 확인하면 된다.



- convexhull의 꼭짓점의 개수가 3개 이상인 경우
- convexhull에 속한 격자점 개수를 세야 한다.
- 우선 convexhull을 $O(N \log N)$ 에 만든다.



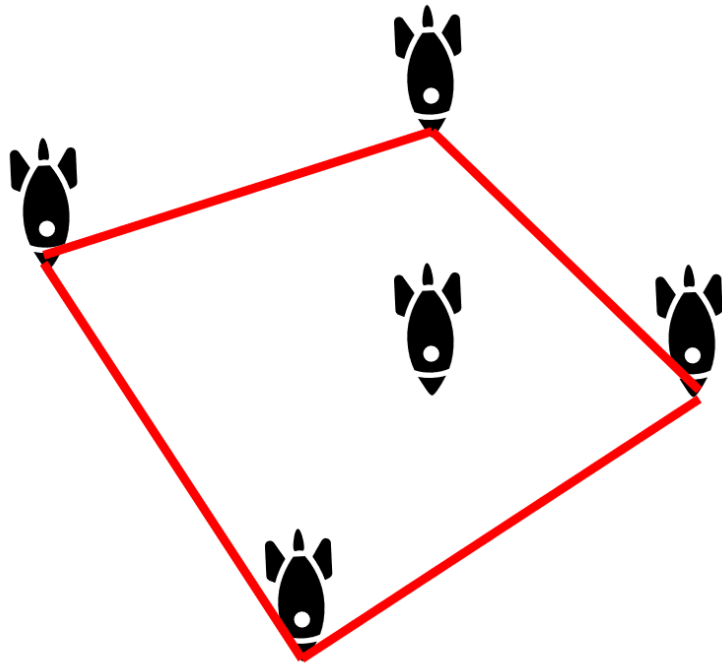
- 테두리의 격자점의 개수는 선분 위의 격자점의 개수를 세는 것과 똑같이 하면 된다.
- 중복되는 끝점을 2번 세지 않게 잘 처리하자.



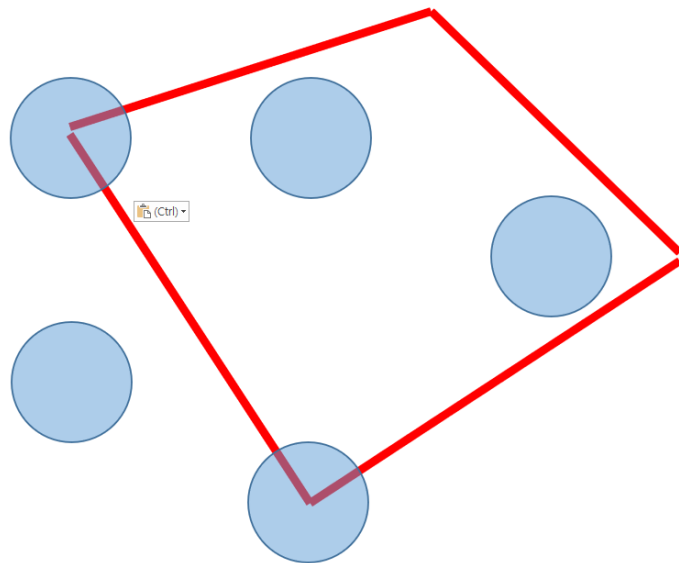
- 다각형 내부의 격자점의 개수는 공식이 있다.
- 픽의 정리 : 다각형의 꼭짓점이 다 격자점인 경우 만족

하는 공식

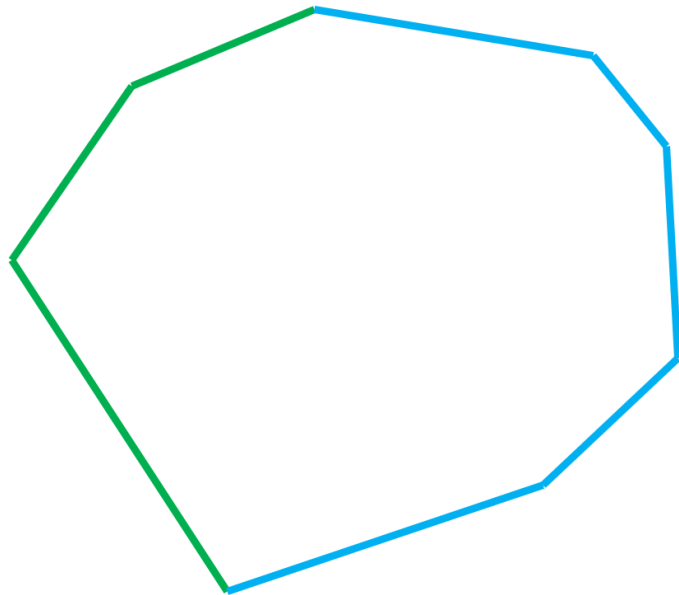
- A : 다각형의 넓이
- L : 다각형의 변 위에 있는 격자점 개수
- I : 다각형 안에 있는 격자점 개수
- $A = I + L/2 - 1$



- 이제 각 M개의 점이 격자점에 포함되어 있는지 확인하면 된다.
- 그러나 단순히 다각형 내의 점인지 확인하면, $O(MN)$ 시간이 걸린다.



- 각 점들이 다각형 내의 점인지 빠르게 확인하기 위해서 우선 다각형의 왼쪽 변과 오른쪽 변을 따로 구분한다.



- 내부에 있는지 확인하려는 점의 y좌표가 어느 변이 커버하는지 확인해보자.
- 이는 이분 탐색 or 정렬 후 two pointer로 처리할 수 있다
- 이제 여기서 해당 격자점이 두 변 사이에 있는지 확인하면
다.
- 확인하는 데 시간 복잡도 : $O(M \log N)$

또는 $O(M \log M + N)$

