KCPC 2023 정답과 해설

Official Solutions

by

고려대학교 문제해결 동아리 ALPS,AlKor,MatKor

운영진 소개

- 김주원(sprout429) - 고려대학교
- 최준석(stonejjun03) - 고려대학교

출제진 소개

- 강근호(rootsquare) - 고려대학교
- 김기현(surface_03) - KAIST
- 이종우(yijwo930) - 고려대학교
- 김동우(kidw0124) - 고려대학교

검수진 소개

- 김동현(kdh9949)
- 옥찬호(utilforever)
- 오주원(kyo20111)

후원자 소개

Solved.AC

고려대학교

삼성 소프트웨어 멤버십

스타트링크

utilforever

문제		의도한 난이도	출제자
2A	KOREA 만들기	**Easy**	rootsquare
1A/2B	돌아온 똥게임	**Easy**	rootsquare
1B/2C	국기 색칠하기	**Easy**	rootsquare
2D	온도 맞추기	**Normal**	surface_03
2E	정사각형 연결하기	**Normal**	rootsquare
1C/2F	Endless Rain	**Hard**	surface_03

문제		의도한 난이도	출제자
1D	KCPC 개최하기	**Hard**	`kidw0124`
1E/2G	Spanning Minimum Tree	**Very Hard**	`surface_03`
1F/2H	업&다운	**Very Hard**	`rootsquare`
1G	외판원 순회 로봇	**Challenging**	`yijw0930`
1H	몰래 교환하기	**Challenging**	`surface_03`

2A. KOREA 문자열 만들기

문자열

- 제출 63명, 정답 29명 (정답률 46.032%)
- Div. 2 처음 푼 사람: **박준민**, 1분
- Open Contest 처음 푼 사람: **seawon0808**, 0분
- 출제자: `rootsquare`

2A. KOREA 만들기

- 문자열을 앞에서부터 한 글자씩 읽으며, 필요한 문자가 나오면 남기고 그렇지 않으면 지운다.
- K,O,R,E,A 순으로 알파벳을 남긴 후 다시 K부터 반복해서 글자를 모으면 된다.

1A/2B. 돌아온 똥게임

탐욕법, 구현

- 제출 155번, 정답 23명 (정답률 15.839%)
- Div. 1 처음 푼 사람: **김재우**, 8분
- Div. 2 처음 푼 사람: **박준민**, 8분
- Open Contest 처음 푼 사람: **seawon0808**, 5분
- 출제자: `rootsquare`

1A/2B. 돌아온 똥게임

- 곱셈은 덧셈의 연속이므로, 장비는 본인의 전투력을 최대한 올릴 수 있는 만큼 올린 후 얻는 것이 본인의 전투력을 가장 크게 상승시키는 방법이다.
- 고로 현재 남아있는 적들 중 전투력이 가장 낮은 몬스터를 잡을 수 있으면 잡고, 그렇지 않다면 현재 남아있는 장비들 중 전투력이 가장 낮은 것을 얻는다.
- 이 과정을 통해 모든 방을 돌파하거나, 몬스터도 잡을 수 없고 장비도 더 이상 얻을 수 없다면 게임을 종료한다.
- 단, 게임 중 본인의 전투력이 10^9를 넘어가는 경우 이후 진행을 어떻게 해도 모든 방을 돌파할 수 있게 된다!(몬스터 전투력은 최대 10^9 이다.) -> 여기서 많은 사람들이 틀렸습니다.

1B/2C. 국기 색칠하기

너비 우선 탐색, 깊이 우선 탐색

- 제출 62번, 정답 20명 (정답률 32.258%)
- Div. 1 처음 푼 사람: **김재우**, 19분
- Div. 2 처음 푼 사람: **박준민**, 20분
- Open Contest 처음 푼 사람: **edenooo**, 8분
- 출제자: `rootsquare`

1B/2C. 국기 색칠하기

- 너비 우선 탐색을 통해 특정 칸과 같은 영역에 속한 칸들을 모두 찾자.
- 이때, 찾은 칸들이 국기 B에서 모두 같은 색으로 표시되어 있는 경우 해당 영역의 칸들을 해당 색으로 칠하면 되지만, 두 가지 이상의 색이 나오는 경우 답은 불가능이다.
- 격자판에서 찾을 수 있는 모든 영역이 색을 국기 B에 맞게 바꿀 수 있으면 정답은 YES, 그렇지 않으면 NO이다.

2D. 온도 맞추기

수학

- 제출 79번, 정답 23명 (정답률 29.114%)
- Div. 2 처음 푼 사람: **박준민**, 29분
- Open Contest 처음 푼 사람: **fermion5**, 6분
- 출제자: `surface_03`

2D. 온도 맞추기

- 생각하기 수월하도록 초기 온도, 목표 온도, 온도 변화 수치를 변형해봅시다. 다음과 같이 변형할 수 있습니다.
  - $A'_i = 0$
  - $B'_i = \dfrac{|A_i - B_i|}{X_i}$
  - $X'_i = 1$
- $|A_i - B_i|$가 X_i의 배수가 아닌 i가 존재한다면 당연히 온도를 맞출 수 없습니다.
- 따라서 B'_i가 모두 정수인 경우만 고려하면 됩니다.

2D. 온도 맞추기

- 매번 온도 조절 장치를 실행할 때마다 각 비커는 온도가 1 오르거나 1 내려가기 때문에, 항상 모든 비커 온도의 홀짝성은 동일합니다.
- 즉, B_i'의 홀짝성이 모두 같지 않으면 온도를 맞출 수 없습니다.

2D. 온도 맞추기

- B'_i의 홀짝성이 모두 같은 경우는 어떨까요?
- B'_i의 최댓값을 M이라고 합시다. M번의 실행을 통해 온도를 맞출 수 있습니다.
- 각 i에 대해, 온도를 $\frac{M + B'_i}{2}$번 올리고, $\frac{M - B'_i}{2}$번 내리면 온도를 전부 맞출 수 있습니다.
- M번 미만의 실행을 통해 온도를 맞출 수 없다는 것은 자명하므로, 답은 M번이 됩니다.

- 전체 시간복잡도는 $\mathcal{O}(N)$입니다.

2E. 정사각형 연결하기

수학

- 제출 59번, 정답 18명 (정답률 30.508%)
- Div. 2 처음 푼 사람: **장한**, 31분
- Open Contest 처음 푼 사람: **nflight11**, 9분
- 출제자: `rootsquare`

2E. 정사각형 연결하기

- 정사각형들을 이어붙여서 만든 도형들의 둘레의 길이는 항상 짝수이다. 둘레가 4,6,8,10... 인 도형들 중 정사각형의 개수가 가장 많은 것을 순차적으로 구하면 다음과 같다.
- 4: 1*1 정사각형, 6: 1*2 직사각형, 8: 2*2 정사각형, 10: 2*3 직사각형...

2E. 정사각형 연결하기

- 정사각형이 T 개 있고, 어떤 양의 정수 n에 대해 $n^2 \leq T < (n+1)^2$라 하자.
- $T = n^2$일 경우, 정사각형을 만들게 되며 정답은 $4n$이다.
- $n^2 < T \leq n^2 + n$일 경우 우선 한 변의 길이가 n인 정사각형을 만들고 그 오른쪽에 일렬로 남는 정사각형들을 순차적으로 붙인다. 가로 길이가 $n+1$, 세로 길이가 n인 직사각형에 모든 사각형이 들어가며, 이 때의 정답은 $4n+2$이다.
- $n^2 + n < T < (n+1)^2$인 경우, 위와 같은 방법으로 한 변의 길이가 $n+1$인 정사각형에 모든 사각형이 들어가며, 이 때의 정답은 $4n+4$이다.

1C/2F. ENDLESS RAIN

분리 집합, 트리를 사용한 집합과 맵

- 제출 73번, 정답 7명 (정답률 9.589%)
- Div. 1 처음 푼 사람: **김재우**, 57분
- Div. 2 처음 푼 사람: **장한**, 114분
- Open Contest 처음 푼 사람: **xiaowuc1**, 15분
- 출제자: `surface_03`

- 풀이의 핵심은, 파라솔을 매일 하나씩 설치하지 않고 몰아서 설치해도 된다는 것입니다.
- 즉, 파라솔을 굳이 설치하지 않아도 되는 날에는 설치하지 않고 넘어갔다가, 파라솔을 여러 개 설치해야 되는 날에 앞에서 설치하지 않았던 파라솔들을 설치하면 됩니다.

- 구체적으로, 다음과 같은 과정을 학기의 모든 날에 대해 반복하면 됩니다. 여기서 cnt는 설치할 수 있는 파라솔의 개수, ans는 학기가 시작하기 전 설치해야 하는 파라솔의 개수입니다.
  1. cnt에 1을 더합니다.
  2. A_i번 건물과 B_i번 건물 사이에 설치해야 하는 파라솔 개수를 구합니다. 이 값을 k라고 합시다.
  3. $cnt \geq k$인 경우, 설치할 수 있는 파라솔이 충분히 많으므로 cnt에 k만 빼주면 됩니다. $cnt < k$인 경우, $k - cnt$개의 파라솔이 부족하므로 ans에 더해주고, cnt를 0으로 초기화합니다.

1C/2F. ENDLESS RAIN

- 이제 설치해야 되는 파라솔 개수를 구하는 것을 빠르게 할 수 있으면, 문제를 해결할 수 있습니다.
- 임의의 건물 번호 i에 대해 i번 건물의 오른쪽에 있는 구간 중 파라솔이 설치되지 않은 가장 가까운 구간을 빠르게 구할 수 있으면, 직접 구간의 개수를 세어주는 것으로 설치해야 되는 파라솔 개수를 구할 수 있습니다.
- 이는 두 가지 방법으로 빠르게 구할 수 있습니다.

1C/2F. ENDLESS RAIN

- 첫 번째 방법은 union find를 이용하는 것입니다.
- x번 건물과 $x+1$번 건물 사이에 파라솔을 설치할 때마다 x번 노드의 부모가 $x+1$이 되도록 합시다.
- 이렇게 하면 i번 노드의 루트 r은 i번 건물의 오른쪽에 있는 구간 중 파라솔이 설치되지 않은 가장 가까운 구간의 왼쪽 건물 번호를 나타내게 됩니다.
- union find를 사용하면 각 노드의 루트를 $\mathcal{O}(\log N)$ 구할 수 있고, 전체 시간복잡도는 $\mathcal{O}((N+M)\log N)$이 됩니다.

1C/2F. ENDLESS RAIN

- 두 번째 방법은 tree set을 이용하는 것입니다.
- 모든 $1 \leq i \leq N-1$에 대해 i를 set에 저장합니다. 각 i가 뜻하는 것은 i번 건물과 $i+1$번 건물 사이에 아직 파라솔이 설치되지 않았다는 것입니다.
- x번 건물과 $x+1$번 건물 사이에 파라솔을 설치할 때마다 set에서 x를 찾아 제거합니다.
- 이렇게 하면 set에서 i 이상의 원소 중 가장 작은 값이 i번 건물의 오른쪽에 있는 구간 중 파라솔이 설치되지 않은 가장 가까운 구간의 왼쪽 건물 번호입니다.
- tree set 자료구조에서 i 이상의 원소 중 가장 작은 값을 찾는 것은 $\mathcal{O}(\log N)$에 가능합니다. 예시로 C++의 경우 `std::set::lower_bound`를 이용할 수 있습니다.
- 따라서 전체 시간복잡도는 $\mathcal{O}((N+M)\log N)$이 됩니다.

1D. KCPC 개최하기

탐욕법, 구현

- 제출 3회, 정답 2회 (정답률 66.667%)
- Div. 1 처음 푼 사람: **김재우**, 128분
- Open Contest 처음 푼 사람: **xiaowuc1**, 54분
- 출제자: `kidw0124`

1D. KCPC 개최하기

- ALPS A, AlKor B, MatKor C 건물의 좌표를 (x_A, y_A), (x_B, y_B), (x_C, y_C) 라고 하자.
- 우선 거리 제곱의 합을 생각을 하고, x 와 y 좌표를 나누어 생각하자.
- $\overline{AB}$ 의 선분은 총 MatKor 건물의 개수 (K) 만큼 카운팅 된다.
- 즉, x 좌표를 생각하면 모든 $(x_A - x_B)^2 = x_A^2 + x_B^2 - 2x_Ax_B$ 가 K 번 카운팅 된다.
- 여기서 x_A^2 항을 보면 B 로 가능한 개수가 AlKor 건물의 개수 (M) 개가 가능하므로, x_A^2 항은 총 MK 번 나온다. 마찬가지로 생각하면 $(x_A - x_C)^2$ 에도 나오므로 총 $2MK$ 번 나온다.
- 또한 모든 x_Ax_B 의 합을 생각하면, a_ib_j 의 합은 S_AS_B 이므로, A 의 x 좌표의 합과 B 의 y 좌표의 합과 같다. 즉, 정리하면 다음과 같다.
- $2MKS_{A_{xx}} + 2KNS_{B_{xx}} + 2NMS_{C_{xx}} - 2KS_{A_x}S_{B_x} - 2NS_{B_x}S_{C_x} - 2MS_{C_x}S_{A_x}$
- y 좌표도 마찬가지로 구해주면 된다.

- 이제 넓이를 생각하면, 신발끈 공식에 의해
- $S_{\triangle ABC} = \begin{vmatrix} x_A & x_B & x_C & x_A \\ y_A & y_B & y_C & y_A \end{vmatrix} = |(x_A y_B + x_B y_C + x_C y_A) - (y_A x_B + y_B x_C + y_C x_A)|$ 이다.
- 조금 더 생각해보면, 절댓값을 제외한 $(x_A y_B + x_B y_C + x_C y_A) - (y_A x_B + y_B x_C + y_C x_A)$ 값의 부호가 반시계일 때 양수, 시계방향일 때 음수이다.
- 신발끈 없이 벡터 외적으로 생각할 수도 있다.
- 즉, 여기서 $(x_A y_B - y_A x_B) + (x_B y_C - y_B x_C) + (x_C y_A - y_C x_A)$ 라고 생각하면, 각 항이 K, N, M 번 나온다는 것을 알 수 있다. 위와 마찬가지로 생각하면,
- $KS_{A_x}S_{B_y} - KS_{A_y}S_{B_x} + NS_{B_x}S_{C_y} - NS_{B_y}S_{C_x} + MS_{C_x}S_{A_y} - NS_{C_y}S_{A_x}$ 가 된다.

- 최종적으로 이를 더해주면 되므로, 다음과 같다.
- $2MKS_{A_{xx}} + 2KNS_{B_{xx}} + 2NMS_{C_{xx}} - 2KS_{A_x}S_{B_x} - 2NS_{B_x}S_{C_x} - 2MS_{C_x}S_{A_x}$
  $+2MKS_{A_{yy}} + 2KNS_{B_{yy}} + 2NMS_{C_{yy}} - 2KS_{A_y}S_{B_y} - 2NS_{B_y}S_{C_y} - 2MS_{C_y}S_{A_y}$
  $+KS_{A_x}S_{B_y} - KS_{A_y}S_{B_x} + NS_{B_x}S_{C_y} - NS_{B_y}S_{C_x} + MS_{C_x}S_{A_y} - NS_{C_y}S_{A_x}$
- 모듈러 연산에서 음수가 나오지 않게 조심하여 $\mathcal{O}(N+M+K)$에 구하면 된다.

1E/2G. 스패닝 최소 트리

수학, 최소 스패닝 트리

- 제출 27번, 정답 1명 (정답률 3.704%)
- Div. 1 처음 푼 사람: **김재우**, 206분
- Div. 2 처음 푼 사람: **없음**, —
- Open Contest 처음 푼 사람: **xiaowuc1**, 32분
- 출제자: `surface_03`

1E/2G. 스패닝 최소 트리

- 조건을 만족하는 그래프 G에서 크루스칼 알고리즘을 적용했을 때 구해진 최소 스패닝 트리의 간선의 가중치를 $B_1, B_2, \cdots, B_{N-1}$이라고 합시다($B_1 < B_2 < \cdots < B_{N-1}$).
- n개의 정점으로 이루어진 완전그래프의 간선의 개수는 $\frac{n(n-1)}{2}$입니다.
- 즉 $1, 2, \cdots, (\frac{i(i-1)}{2}+1)$의 가중치를 가진 간선들로 만들어진 그래프의 정점은 최소 $i+1$개가 됩니다. 크루스칼 알고리즘의 작동 방식에 의해 B_i는 $1, 2, \cdots, (\frac{i(i-1)}{2}+1)$ 중 하나의 값을 가지게 됩니다.
- 따라서 $B_i \leq \frac{i(i-1)}{2}+1$를 만족합니다.

- 반대로 $B_i \leq \frac{i(i-1)}{2} + 1$를 만족한다면 항상 $B_1, B_2, \cdots, B_{N-1}$을 최소 스패닝 트리의 간선의 가중치로 가지는 그래프를 만들 수 있습니다.
- $x < i + 1$을 만족하는 임의의 정수 x에 대해 x번 정점과 $i + 1$번 정점을 연결하는 간선의 가중치가 B_i가 되게 합니다.
- 나머지 가중치의 간선들에 대해, 간선의 가중치가 c라면 $\frac{(n-1)(n-2)}{2} < c \leq \frac{n(n-1)}{2}$인 n을 찾고, 정점 번호가 n이하인 두 정점을 연결하게 하면 됩니다.

- 따라서 다음과 같은 조건을 만족하는 수열 B를 찾으면 문제를 풀 수 있습니다.
  - $B_1 < B_2 < \cdots < B_{N-1} \leq M$
  - $B_1 + B_2 + \cdots + B_{N-1} = S$
  - $B_i \leq \dfrac{i(i-1)}{2} + 1$

1E/2G. 스패닝 최소 트리

- 초기항이 $\frac{i(i-1)}{2}+1$인 수열에서 총 $\sum_{i=1}^{N-1}(\frac{i(i-1)}{2}+1)-S$만큼 빼서 조건을 만족하는 B를 찾아봅시다.
- $B_{N-1} \leq M$을 만족해야 하므로, 수열에서 가능한 인덱스가 큰 항을 빼줘야 합니다.
- 또한 $B_1 < B_2 < \cdots < B_{N-1}$을 만족하도록 수를 빼줘야 합니다.

1E/2G. 스패닝 최소 트리

- 예시로 $N = 7, S = 26$인 경우를 생각해봅시다. 수열에서 빼야 하는 수는 $\sum_{i=1}^{6} (\frac{i(i-1)}{2} + 1) - 26 = 15$입니다.
- 다음과 같은 과정으로 빼면 됩니다.

$$
\begin{aligned}
&[1,\ 2,\ 4,\ 7,\ 11,\ 16] \\
\Rightarrow &[1,\ 2,\ 4,\ 7,\ 11,\ 12] \\
\Rightarrow &[1,\ 2,\ 4,\ 7,\ 8,\ 9] \\
\Rightarrow &[1,\ 2,\ 4,\ 6,\ 7,\ 8] \\
\Rightarrow &[1,\ 2,\ 4,\ 5,\ 7,\ 8] \\
\Rightarrow &[1,\ 2,\ 4,\ 5,\ 6,\ 8]
\end{aligned}
$$

1E/2G. 스패닝 최소 트리

- 수열의 앞쪽 항 중 수를 빼지 않아도 되는 항들을 S의 범위에 따라 미리 계산해놓으면, $\mathcal{O}(N)$의 시간복잡도로 수열 B를 구할 수 있습니다.
- 구한 수열 B가 $B_{N-1} > M$을 만족하면, 조건을 만족하는 그래프 G는 없습니다.
- $B_{N-1} \leq M$을 만족하면, 구한 B_i를 이용해 그래프를 구성해서 출력하면 됩니다.

1F/2H. 업&다운

창의력, 해 구성하기

- 제출 39명, 정답 4명 (정답률 10.256%)
- Div.1 처음 푼 사람: **김재우**, 186분
- Div.2 처음 푼 사람: **박예영**, 196분
- Open 처음 푼 사람: **aeren**, 218분
- 출제자: `rootsquare`

1F/2H. 업&다운

- 우선 손에 있는 카드들 중 홀수가 적힌 것은 검정색으로, 짝수가 적힌 것은 흰색으로 칠하자.
- 카드를 놓는 규칙에 의해, 탁자에는 항상 검정색 카드와 흰색 카드가 번갈아 가며 놓여야 한다.(같은 색의 카드를 연속해서 놓을 수 없다.)
- 고로 검정색 카드의 수와 흰색 카드의 장수 차이가 2 이상일 경우 카드를 모두 놓는 것이 불가능하다.

1F/2H. 업&다운

- 우선 탁자에 검정색 카드들을 카드에 적힌 수 기준 오름차순으로 일렬 배치하고, 두 검정색 카드 사이에는 카드 한 장이 들어가는 정도의 간격을 두자. 이제 검정색 카드 사이사이에 흰색 카드를 한 장씩 배치해야 한다.
- 어떤 이웃한 두 검정색 카드에 적힌 수가 $(2t-1, 2t+1)$과 같은 형태라면, 이들 사이에는 반드시 $2t$가 적힌 흰색 카드 한 장을 우선적으로 놓아야 한다.(t는 양의 정수) 만일 해당 카드가 없다면 모든 카드를 탁자에 놓는 것이 불가능하다.
- 예시: 탁자에 놓인 검정색 카드가 1 1 3 3 3 5 7 7과 같다면, 1과 3 사이에 2, 3과 5 사이에 4, 5와 7 사이에 6을 각각 우선적으로 놓아야 한다. 배치를 하고 난 후에는 1 1 2 3 3 3 4 5 6 7 7이 된다.

1F/2H. 업&다운

- 이제 남은 흰색 카드들을 남아있는 빈틈에 각각 한 장씩 놓아 검정색 카드와 흰색 카드가 교차로 배열되게 하자. 앞에서 서로 다른 수가 적힌 이웃한 검정색 카드 사이에는 카드를 이미 넣었으므로, 이제 서로 같은 수가 적힌 이웃한 검정색 카드 사이에 카드를 넣어야 한다.
- 어떤 이웃한 두 검정색 카드에 적힌 수가 모두 t라면, 그들 사이에는 $(t+1)$ 한 장 또는 $(t-1)$ 한 장을 놓을 수 있다.
- 가장 왼쪽에 있는 빈틈부터 순서대로 카드를 놓으며, 둘 수 있는 카드가 여러 장일 경우 그 중 작은 수가 적힌 것을 먼저 배치한다. 만일 어떤 빈틈에 놓을 수 있는 카드가 없다면 카드를 모두 놓는 것이 불가능하다.
- 여기까지 왔다면, 탁자에는 가장 왼쪽과 오른쪽에는 검정색 카드가 놓여있고, 검정색과 흰색이 교대로 배열된 모습을 볼 수 있을 것이다. 이제 손에는 흰색 카드 몇 장이 남아있을 것이다.

1F/2H. 업&다운

- 손에 흰색 카드가 몇 장 남아있다면, 탁자의 카드배열 양 끝에 추가로 카드를 더 놓아보자.
- 가장 왼쪽의 검정색 카드에 적힌 수가 a일 경우, 그 왼쪽에는 $(a+1)$ 한 장을 더 놓을 수 있다.
- 가장 오른쪽의 검정색 카드에 적힌 수가 b일 경우, 그 오른쪽에는 $(b-1)$ 한 장을 더 놓을 수 있다.
- 여기까지의 모든 과정을 진행 후, 손에 흰색 카드가 남아있지 않으면 만들어진 배열이 곧 정답이며, 카드가 남아있을 경우 모든 카드를 탁자에 놓는 것이 불가능하다.

1G. 외판원 순회 로봇

그래프 이론, 비트필드를 이용한 다이나믹 프로그래밍

- 제출 0회, 정답 0명 (정답률 0.000%)
- Div. 1 처음 푼 사람: **없음**
- Open Contest 처음 푼 사람: **xiaowuc1**, 100분
- 출제자: `yijw0930`

1G. 외판원 순회 로봇

- 먼저, 평범한 TSP 문제처럼 다음 DP 식을 채웁니다.
  - $D[i][j][S] =$ (정점 i로부터 j까지 S에 속하는 정점들을 거쳐 가는 경로의 최소길이)
- 다음으로 다음 DP식을 채워야 합니다.
  - $E[i][j][S] =$ (로봇을 가지고 있는 상태의 외판원이 정점 i에서 시작해서 S에 속하는 정점들을 모두 방문한 후 정점 j에서 로봇을 가지고 있는 상태로 종료할 수 있는 최소 시간)
- 이를 위한 외판원과 로봇의 경로를 합쳐 E-경로라고 하자

1G. 외판원 순회 로봇

- $E[i][j][S]$는 다음 세 가지 경우 중 최소입니다.
  - 외판원이 로봇과 분리되지 않은 상태로 D에서 계산한 경로를 그대로 따라 이동
  - 외판원과 로봇이 i에서 분리되고, 독립적으로 D에서 계산된 (S의 두 부분집합을 순회하는) 경로를 따라 이동한 후 j에서 합류함
  - 어떤 중간 정점 k를 기점으로 두 (S의 두 부분집합을 순회하는) E-경로의 합

- 이때 문제가 있는데, 합집합이 S가 되는 두 부분집합을 고르는 경우의 수는 $3^{|S|}$ 이므로, 전부 탐색하기에는 시간이 부족하다.
- 따라서 두 독립적인 부분집합만을 탐색할 것이다. 이때 경우의 수는 $2^{|S|}$ 이다.
- 모든 가능한 S에 대한 가능한 경우의 수를 합한 값은 이항정리를 통해 3^N 임을 알 수 있다.
- 따라서, 일단 이를 통해 $O(N^3 3^N)$ 속도를 얻을 수 있지만 아직 정확한 알고리즘은 아니다.

1G. 외판원 순회 로봇

- 두 독립적인 부분집합을 골랐기 때문에 오히려 더 많은 정점을 순회하여 더 빨라지는 경우가 배제되므로, 이를 해결하기 위해 $E[i][j][S]$를 구할 때마다 이를 S의 부분집합에 전파해주어야 합니다. (D에서는 이미 충분히 반영되어 있어 필요 없음)
- 이를 통해 두 부분집합이 모두 S의 진부분집합인 경우는 해결했지만 둘 중 하나 이상이 S와 동일한 경우를 해결하기 위해 하나의 S에 대한 모든 E값을 고른 후, 플로이드-워셜 알고리즘을 통해 S 내의 모든 (S의 일부를 지나는) 최단 E-경로를 구하여 이를 이용해 S에 대한 모든 $E[i][j][S]$ 값을 다시 한번 업데이트합니다.
- 따라서 최종 시간복잡도는 $O(N^3 3^N)$입니다. 단, 실제 출제자가 작성한 코드는 구현의 편의를 위해 $O(4^N)$의 전처리 과정을 거쳤습니다.

1H. 몰래 교환하기

수학, 제곱근 분할법, 정렬, 모듈러 곱 역원

- 제출 0번, 정답 0명 (정답률 0.000%)
- Div.1 처음 푼 사람: **없음**
- Open 처음 푼 사람: **aeren**, 218분
- 출제자: `surface_03`

1H. 몰래 교환하기

- 카드 X 와 카드 Y 가 교환 가능하고, 카드 Y 와 카드 Z 가 교환 가능하다고 해봅시다.
- 카드 X, Y 를 교환하고, Y, Z 를 교환하고, X, Z 를 교환하면 카드 X, Z 를 교환한 것과 같아집니다.
- 일반적으로, 카드 A_1, A_2 간 교환이 가능하고, 카드 A_2, A_3 간 교환이 가능하고, ..., 카드 A_{n-1}, A_n 간 교환이 가능하다면 카드 A_1 과 카드 A_n 는 교환이 가능합니다.
- 따라서 각 카드를 그래프의 정점에 대응하고, 직접 교환이 가능한 두 카드 쌍들을 모두 간선으로 연결하면 같은 컴포넌트에 있는 카드끼리는 직간접적으로 교환이 가능합니다.
- 다른 컴포넌트에 있는 카드끼리는 서로 교환이 불가능함은 자명합니다.

- 따라서 초기 카드 배열을 통해 컴포넌트를 구하고, 각 컴포넌트에서 같은 것이 있는 순열의 개수를 구해 모두 곱하면 답이 됩니다.
- $1!$부터 $N!$까지의 값과 $1!$부터 $N!$까지의 모듈러 곱셈 역원을 미리 구해놓으면, 같은 것이 있는 순열은 쉽게 계산할 수 있습니다.
- 컴포넌트를 어떻게 빠르게 구할 수 있을까요?

1H. 몰래 교환하기

- 우선 식을 간략하게 만들어봅시다.
- $A + B - (A \oplus B) = 2(A\&B)$ 입니다.
- 따라서 $A\&B \leq \frac{K}{2}$ 를 만족하면 교환이 가능합니다.

– 다음과 같은 과정을 통해 정수 X가 적힌 카드가 포함된 컴포넌트를 구할 수 있습니다.

1. S에는 아직 어떠한 컴포넌트에도 포함되지 않은 카드들에 대한 정보가 저장되어 있습니다. T는 아직 아무 정보도 저장되어 있지 않습니다.
2. S에서 X를 찾아 제거하고, T에 X를 저장합니다.
3. 다음 과정을 T에 있는 모든 정수들이 방문 체크가 될 때까지 반복합니다.
   a. T에서 아직 방문 체크를 하지 않은 정수를 고릅니다. 이를 t라고 합시다.
   b. S에서 t와 Bitwise AND 연산을 했을 때 결과가 가장 작아지는 정수 u를 찾습니다.
   c. $t\&u \leq \frac{K}{2}$ 라면 u를 T에서 제거하고 S에 넣고 b번으로 돌아가서 반복합니다.
   d. 정수 t에 방문 체크를 합니다.
4. T에 저장된 카드들은 정수 X가 포함된 컴포넌트와 같습니다.

- 따라서 집합에서 특정 정수와 Bitwise AND 연산을 했을 때 결과가 가장 작아지는 정수를 빠르게 찾을 수 있으면, 문제를 풀 수 있습니다.
- 각 정수의 비트를 절반으로 쪼개서 생각해봅시다.
- 0 부터 $2^9 - 1$ 까지의 각각의 정수들에 대해, 0 부터 $2^9 - 1$ 의 정수들과 Bitwise AND 한 결과가 작아지는 순서대로 정렬한 상태로 관리할 수 있습니다.

- 따라서 상위 9개의 비트가 같은 정수들끼리 따로 관리하게 되면 특정 정수와 Bitwise AND 연산을 했을 때 결과가 가장 작아지는 정수를 빠르게 찾을 수 있습니다.
- 상위 9개의 비트가 같은 정수들끼리 묶게 되면, 집합이 총 512개로 나뉘게 되고, 각 집합별로 특정 원소와 Bitwise AND를 한 결과가 가장 작은 원소는 미리 정렬해놓으면 집합마다 $\mathcal{O}(1)$ 만에 구할 수 있습니다.
- 즉 전체 집합에서 특정 정수와 Bitwise AND 연산을 했을 때 결과가 가장 작아지는 정수는 512번의 연산으로 찾을 수 있습니다.
- 각 정수는 집합에서 한번 빠지면 다시 넣지 않으므로, 총 시간복잡도는 $\mathcal{O}(512N) = \mathcal{O}(N\sqrt{X})$ 가 됩니다.