코드포스 12주차 문제 풀이

문제 난이도

	문제 번호	문제 이름	난이도
A	25166	배고픈 아리의 샌드위치 구매하기	B1
B	27960	사격 내기	B3
C	10728	XOR 삼형제 1	S1
D	4312	3의 제곱	S3
E	25046	사각형 게임 (Small)	G5
F	5175	문제없는 문제	G5
G	19849	문제지 나르기	P2
H	18184	분할하기	P4

A. 배고픈 아리의 샌드위치 구매하기

수학, 비트마스킹

A. 배고픈 아리의 샌드위치 구매하기

- ✓ **아리의 동전으로만 계산**:
  - ✓ 아리가 가진 모든 동전의 합이 1023원이므로, 만약 샌드위치의 가격 s가 1023원 이하라면, 아리의 돈만으로 살 수 있습니다. 이 경우 "No thanks"를 출력하고 프로그램을 종료합니다.

- ✓ **쿠기의 도움이 필요한 경우**:
  - ✓ 샌드위치의 가격 s가 1024원 이상일 경우, 아리의 돈으로는 부족하므로 쿠기의 도움이 필요합니다.
  - ✓ 이때 부족한 금액 change는 s - 1023으로 계산됩니다.

- ✓ **부족한 금액이 쿠기의 동전으로 해결 가능한지 확인**:
  - ✓ change를 비트 단위로 확인하면서, 각 비트가 1인 경우에 해당하는 금액이 쿠기가 가진 동전들(m)에 포함되는지 확인합니다.
  - ✓ 만약 change의 특정 비트가 1인데, 쿠기가 해당 금액의 동전을 가지고 있지 않다면, "Impossible"을 출력하고 프로그램을 종료합니다.
  - ✓ 부족한 금액을 쿠기의 동전으로 충당할 수 있는 경우, "Thanks"를 출력합니다.

B. 사격 내기

수학, 비트마스킹

B. 사격 내기

- 과녁 점수는 2의 거듭제곱 형태로 주어지며, 이는 비트마스크와 아주 잘 맞습니다. 각 과녁의 점수를 비트마스크로 표현하면 다음과 같습니다:
  - 1점: 0000000001
  - 2점: 0000000010
  - 4점: 0000000100
  - 8점: 0000001000
  - 16점: 0000010000
  - 32점: 0000100000
  - 64점: 0001000000
  - 128점: 0010000000
  - 256점: 0100000000
  - 512점: 1000000000

- A와 B의 총합 점수는 각각의 비트마스크를 더한 것입니다. 예를 들어, A가 1점과 2점을 맞혔으면 0000000011이 되고, B가 2점과 4점을 맞혔으면 0000000110이 됩니다.

- C가 맞힌 과녁은 A와 B의 비트 연산을 통해 구할 수 있습니다: XOR 연산 (^)을 사용하여, A와 B 중 한 명만 맞힌 과녁을 찾습니다.

B. 사격 내기

- **XOR 연산의 특징**
  - 1 ^ 0 = 1 (한 명만 맞힌 과녁)
  - 0 ^ 1 = 1 (한 명만 맞힌 과녁)
  - 1 ^ 1 = 0 (둘 다 맞힌 과녁)
  - 0 ^ 0 = 0 (둘 다 못 맞힌 과녁)

- 즉, A와 B의 비트마스크를 XOR하면 C가 맞힌 과녁의 비트마스크가 됩니다.

C. XOR삼형제 1

브루트포스 알고리즘, 해 구성하기, 비트마스킹

C. XOR삼형제 1

- ✓ **모든 부분 집합 탐색:**
  - ✓ 주어진 수 N까지의 수 중에서 가능한 모든 부분 집합을 탐색합니다. 각 수를 포함하거나 포함하지 않는 모든 조합을 고려합니다. 이는 비트마스크를 사용하여 구현할 수 있습니다.
  - ✓ 각 부분 집합은 2^N개가 되며, N이 작다면 이 방법이 가능합니다.
- ✓ **부분 집합 검증:**
  - ✓ 생성된 각 부분 집합에 대해 세 개의 원소를 선택하여 그들의 XOR 값을 계산합니다.
  - ✓ 이 XOR 값이 0이 되는 경우가 있으면 해당 부분 집합은 유효하지 않습니다.
  - ✓ 모든 가능한 세 원소 조합에 대해 XOR 연산을 수행하여 조건을 검증합니다.
- ✓ **최장 부분 집합 찾기:**
  - ✓ 조건을 만족하는 부분 집합 중 가장 긴 부분 집합을 기록합니다.
  - ✓ 부분 집합의 길이를 비교하여 가장 긴 것을 선택합니다.

D. 3의 제곱

수학, 비트마스킹, 임의 정밀도 / 큰 수 연산

D. 3의 제곱

- **부분집합의 인덱스 계산:**
  - n이 1부터 시작하므로 실제 인덱스는 $n-1$입니다. 즉, n번째 부분집합을 찾기 위해서 $n-1$ 번째 부분집합을 계산합니다.
- **이진수로 부분집합 결정:**
  - n−1을 이진수로 변환하여 해당 비트가 1인 위치를 찾습니다. 이 비트가 1인 각 위치는 부분집합에 포함된 3의 제곱수의 지수를 나타냅니다.
  - 비트가 1인 각 위치에 대해 해당 지수의 3의 제곱수를 결과 리스트에 추가합니다.
- 예를 들어, n = 5인 경우:
  - n−1 = 4
  - 4를 이진수로 표현하면 100입니다. 이는 3의 제곱수 중 $3^2(9)$만 포함된다는 의미입니다.
  - 따라서 5번째 부분집합은 { 9 } 입니다.

E. 사각형 게임 (SMALL)

브루트포스 알고리즘, 비트마스킹

E. 사각형 게임 (Small)

- **비트마스크를 이용한 행 선택:**
  - N개의 행 중에서 어떤 행을 선택할지 결정하기 위해 비트마스크를 사용합니다.
  - 0부터 $2^N - 1$까지의 숫자를 비트마스크로 사용하여, 각각의 숫자가 어떤 행을 선택할지 나타냅니다. 예를 들어, 011은 2, 3번째 행을 선택하는 경우를 나타냅니다.

- **민우의 점수 계산:**
  - 각 비트마스크에 대해 민우가 선택한 행을 결정하고, 해당 행에서 얻는 점수를 계산합니다.
  - 선택한 행에 있는 모든 칸의 점수를 더하여 초기 점수를 계산합니다.

E. 대표 선수

- **종진의 점수 계산:**
  - 각 열에 대해 민우가 선택한 행과 종진이 선택할 수 있는 열에 따라 점수를 계산합니다.
  - 각 열에 대해 민우가 색칠하지 않은 행에 있는 칸의 점수를 종진이 얻을 수 있는 점수로 계산합니다.
  - 종진이의 열 선택에 따라 민우의 점수가 빠지기도 하고 더해지기도 할 것입니다. 이때 종진이가 얻을 수 있는 점수가 민우의 빠지는 점수보다 크면 선택합니다.
  - 각 열에 대해 민우가 선택한 행의 점수를 종진이 선택한 열의 점수에서 빼주어 민우가 더 얻는 점수를 계산합니다.

- **최대 점수 갱신:**
  - 모든 비트마스크에 대해 민우가 얻을 수 있는 점수를 계산하고, 이 중에서 가장 큰 점수를 저장합니다.
  - 최종적으로 계산된 민우의 최대 점수를 출력합니다.

F. 문제없는 문제

브루트포스 알고리즘, 비트마스킹

F. 문제없는 문제

- **비트마스크를 이용한 알고리즘 표현:**
  - 각 문제에 대해 사용되는 알고리즘을 비트마스크로 표현합니다.
  - 예를 들어, 문제 i 가 알고리즘 1, 3, 4를 사용하면, 비트마스크는 1011 (2진수)로 표현됩니다.
- **최소 문제 세트 찾기:**
  - 모든 문제에 대해 하나씩 선택하며 전체탐색 합니다.(비트마스크 or 백트래킹)
  - 각 문제를 선택할 때마다 현재의 비트마스크 상태를 갱신하고, 이를 재귀적으로 호출하여 다음 문제를 선택합니다.
  - 모든 알고리즘을 포함하는 최소 문제 세트를 찾으면 이를 갱신합니다.
- **결과 출력:**
  - 각 테스트 케이스마다 최소 문제 세트를 사전순으로 출력합니다.
  - 여러 답이 있는 경우, 사전순으로 가장 앞선 것을 선택합니다.

G. 문제지 나르기

수학, 기하학, 비트마스킹, 백트래킹, 4차원 이상의 기하학

G. 문제지 나르기

- ✓ **비트마스크를 이용한 좌표 계산:**
  - ✓ 각 참가자의 좌표에서 가능한 모든 조합의 합과 차를 계산하여 저장합니다. 이는 비트마스크를 활용하여 구현됩니다.
  - ✓ 비트마스크는 $2^{11} = 2048$개의 조합을 나타낼 수 있으며, 각 조합에 대해 최대 거리를 계산합니다.
  - ✓ 참가자들의 좌표 $x = (x_1, x_2, \cdots, x_{11})$ 에 대해 가능한 모든 조합의 합과 차를 계산하여 저장합니다. 이를 위해 비트마스크를 사용합니다.
  - ✓ 예를 들어, $x = (x_1, x_2, x_3)$ 인 3차원 벡터를 생각해봅시다. 가능한 모든 조합은 다음과 같습니다.
  - ✓ $111 \rightarrow +x_1 +x_2 +x_3$
  - ✓ $110 \rightarrow +x_1 +x_2 -x_3$
  - ✓ $101 \rightarrow +x_1 -x_2 +x_3$
  - ✓ $100 \rightarrow +x_1 -x_2 -x_3$
  - ✓ $011 \rightarrow -x_1 +x_2 +x_3$
  - ✓ $010 \rightarrow -x_1 +x_2 -x_3$
  - ✓ $001 \rightarrow -x_1 -x_2 +x_3$
  - ✓ $000 \rightarrow -x_1 -x_2 -x_3$
  - ✓ 이를 11차원으로 확장하여 각각의 경우의 최댓값을 구해놓습니다.

G. 문제지 나르기

- ✓ **주차장에서 가장 멀리 떨어진 참가자 집 계산:**
  - ✓ 주차장의 좌표에 대해서도 가능한 모든 조합의 합과 차를 계산합니다.
  - ✓ 이 때 참가자 좌표의 경우에서 계산한 것과 반대로 연산해야 합니다.
  - ✓ 예를 들어, $101 \rightarrow +x_1 -x_2 +x_3$로 계산했다면 $101 \rightarrow -y_1 +y_2 -y_3$ 이렇게 계산해야 합니다.
  - ✓ 계산한 값을 미리 계산한 참가자 좌표의 최댓값과 더하여 모든 경우의 대해 최댓값을 구하면 정답입니다.
- ✓ **최종 결과 출력:**
  - ✓ 각 주차장에서 가장 멀리 떨어진 참가자의 집까지의 거리를 계산하여 출력합니다.

H. 분할하기

수학, 해 구성하기, 비트마스킹

H. 분할하기

- ✓ **초기 설정**:
  - ✓ K를 1 감소시켜 $k = K - 1$ 로 설정합니다.
  - ✓ A 의 초기 원소로 0을 추가합니다. 이는 모든 좋은 집합에 포함될 수 있는 최소 원소입니다.
  - ✓ gap(간격) 과 $start$(시작점) 변수를 각각 2와 1로 초기화합니다.

- ✓ **가장 큰 비트를 찾기**:
  - ✓ k의 가장 큰 비트를 찾습니다. 이는 k 보다 작거나 같은 가장 큰 2의 거듭제곱을 찾는 과정입니다.
  - ✓ 예를 들어, $k = 14$ 일 경우 $14 \leq 2^3 = 8$, $14 > 2^4 = 16$, 따라서 가장 큰 비트는 8 입니다.

- ✓ **비트에 따른 원소 추가**:
  - ✓ k의 각 비트를 차례대로 검사하면서, 비트가 1인 경우에 해당 비트 위치의 값을 A 에 추가합니다.
  - ✓ 이 과정에서 시작점은 $start$, 간격은 gap, 추가할 원소 개수는 해당 비트에 해당하는 수입니다.

H. 분할하기

- **예시 : N = 4, K = 14인 경우**
  - 0을 push하고 K = 13이 됩니다.
  - 13은 이진수로 1101 이므로 처음 탐색할 비트는 제일 위 비트인 8입니다.
  - Start = 1, gap = 2, 비트 = 8이므로 1, 3, 5, 7, 9, 11, 13, 15를 push합니다. Push해야 할 원소의 개수는 5이 되었습니다.
  - 그 비트는 4입니다. 이 때 start, gap은 두 배가 되어 2, 4가 됩니다.
  - 2, 6, 10, 14를 push합니다. Push해야 할 원소의 개수는 2가 되었습니다. Start, gap은 4, 8이 됩니다.
  - 그 다음 비트는 1입니다. 4를 push합니다. Push해야 할 원소의 개수는 0이므로 종료합니다.
  - 정렬하고 출력하면 정답입니다.