제2회 피갤컵 풀이

by

Official Solutions

문제		의도한 난이도	출제자
A	피갤컵	**Very Easy**	plast
B	나이트의 이동	**Easy**	plast
C	2^3은?	**Easy**	lunarlity
D	1과 5	**Medium**	bubbler
E	판드랄추	**Medium**	pyb1031
F	치터 잡기	**Hard**	ibm2006
G	내 맘대로 정렬	**Hard**	lunarlity
H	햄부기	**Hard**	pyb1031
I	식당	**Challenging**	pyb1031
J	피돌이 vs 피붕이	**Challenging**	pyb1031

A. 피갤컵

`input, output`
출제진 의도 – **Very Easy**

- 제출 584번, 정답자 362명 (정답률 66.27%)
- 처음 푼 사람: **seawon0808**, 0분
- 출제자: `plast`

A. 피갤컵

- 풀이 1 : n이 5 이하이므로, 각각의 답을 구한 뒤 if문으로 출력할 수 있습니다.
- 풀이 2 : y = 2024, m = 8 이라고 적어둔 뒤, (n - 1)회 m에 7을 더하는 것을 반복합니다. 이때 m이 12를 초과할 때마다 y의 값을 1 증가시키고 m의 값을 12 감소시킵니다.

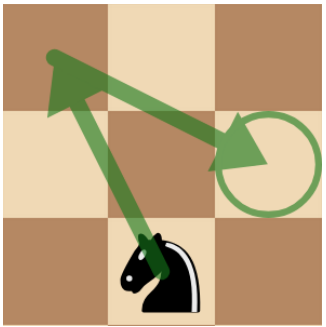
B. 나이트의 이동

ad_hoc
출제진 의도 – **Easy**

- 제출 772번, 정답자 201명 (정답률 27.46%)
- 처음 푼 사람: **xiaowuc1**, 3분
- 출제자: `plast`

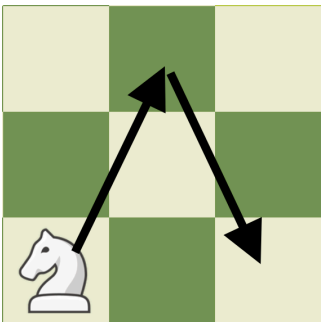
- 관찰 1 : 나이트는 한 번 이동할 때마다, 기존의 체스판의 색과 다른 색으로 이동합니다. 따라서 나이트가 두 번 이동하면 반드시 기존의 체스판의 색과 같은 색으로 이동합니다.

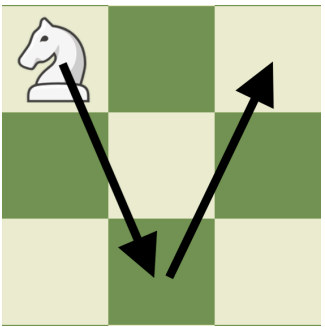
- 관찰 2 : 나이트는 같은 색 안에서 이동할 때에는 꽤 자유롭습니다. 특히, 특정 방향으로 2칸 이동과 대각선 이동이 대부분의 상황에서 가능합니다.



B. 나이트의 이동

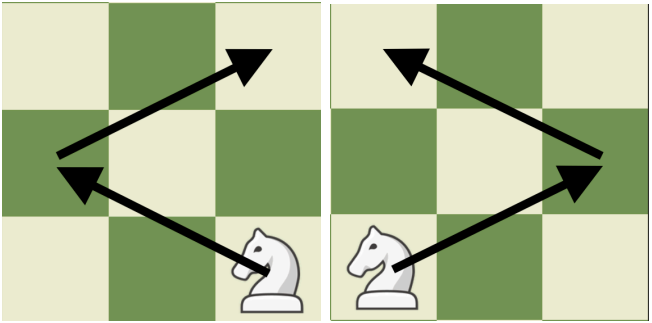
- ✓ 조금 더 엄밀하게 생각해 봅시다. 윗쪽으로 두 칸이 여유가 있거나, 아랫쪽으로 두 칸이 여유가 있다면 반드시 좌우로 두 칸 이동하는게 가능합니다.
- ✓ 그런데, N >= 4인 경우 어느 위치에 있더라도 윗쪽으로 두칸이 여유가 있거나, 아랫쪽으로 두 칸은 여유가 있게 됩니다.





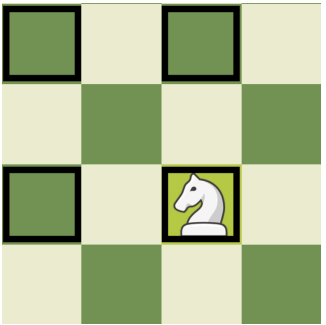
B. 나이트의 이동

- 비슷하게, 왼쪽으로 두 칸이 여유가 있거나, 오른쪽으로 두 칸이 여유가 있다면 반드시 상하로 두 칸 이동하는게 가능합니다.
- 그런데, N >= 4인 경우 어느 위치에 있더라도 왼쪽으로 두칸이 여유가 있거나, 오른쪽으로 두 칸은 여유가 있게 됩니다.



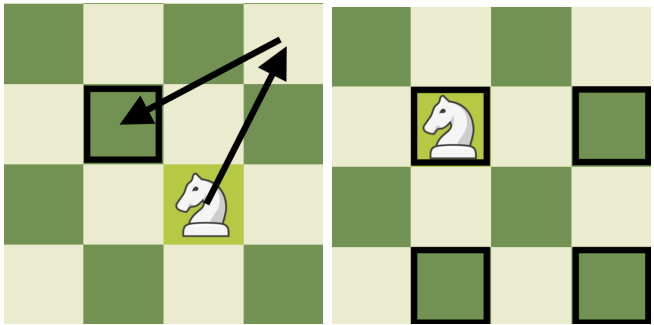
B. 나이트의 이동

- 다시 말하면, N >= 4인 경우에는 어느 위치에서든 상하좌우 두칸 차이에 있는 위치로 이동이 가능합니다.
- 그림으로 예를 들면, 이 나이트는 상하좌우 두 칸 차이에 있는 위치로 이동하는 연산을 통해 네모로 쳐져 있는 칸으로 이동이 가능합니다.



B. 나이트의 이동

- ✓ 나머지 색칠된 칸 역시, 중앙에서 대각선 이동을 하게 될 경우 어렵지 않게 이동이 가능함을 확인할 수 있습니다.
- ✓ 따라서, N >= 4인 경우, 항상 색칠된 모든 칸을 이동할 수 있습니다.



- ✓ N = 3인 경우에만 예외 케이스가 있는데, 이는 상하좌우 공간이 모두 좁기 때문입니다. 이 때에는 직접 나이트를 이동시켜 보며 관찰을 해야만 합니다.

C. 2^3은?

`ad_hoc`, `bitmask`, `math`

출제진 의도 – **Medium**

- 제출 238번, 정답자 137명 (정답률 59.24%)
- 처음 푼 사람: **adhd**, 5분
- 출제자: `lunarlity`

C. 2^3은?

- a, b, c의 범위가 좀 커보입니다. 약간의 케이스워크가 필요하다는 생각을 할 수 있습니다.
- 먼저 $a = 1$인 경우를 생각해 봅시다.
- $1^{b^c} = 1 \oplus b \oplus c$를 만족해야 하고, 이는 $1 = 1 \oplus b \oplus c$와 같습니다.
- 양변에 어떤 수를 XOR 취하여도 등식이 항상 성립하므로, $b \oplus c = 0$이 되어야 합니다.
- 저 수식은 모든 비트가 같음을 의미하므로, $b = c$인 경우만 가능합니다. 즉 답은 $\min(q, r)$입니다.
- 그 다음으론 $a \neq 1, b = 1$인 경우를 생각해 봅시다.
- $a^{1^c} = a \oplus 1 \oplus c$를 만족해야 하고, 이는 $a = a \oplus 1 \oplus c$와 같습니다.
- 마찬가지로 정리하면 $1 \oplus c = 0$을 만족해야 합니다. 이 경우 c는 1만 가능하고 a는 2 이상이면 되므로 답은 $p - 1$입니다.

C. 2^3은?

- 마지막으로 $a \neq 1, b \neq 1$인 경우를 생각해 봅시다.
- 직관적으론 이 경우에는 가능한 경우가 없음을 알 수 있습니다. 자세한 증명은 다음과 같습니다:
- $a \oplus b \oplus c < 2\max(a, b, c)$는 항상 만족하는 수식입니다.
- 우리는 $a > 1, b > 1$에서 $a^{b^c} = a \oplus b \oplus c$ 대신 $a^{b^c} > 2\max(a, b, c) = M > a \oplus b \oplus c$가 항상 성립함을 보일 것입니다.
- Case 1: $M = a$일 때, $a^{b^c} \geq a^2 \geq 2a = 2M > a \oplus b \oplus c$이므로 불가능합니다.
- Case 2: $M = b$일 때, $a^{b^c} \geq a^{b^1} \geq a^b \geq 2^b \geq 2b = 2M > a \oplus b \oplus c$이므로 불가능합니다.
- Case 3: $M = c$일 때, $a^{b^c} \geq 2^{2^c} > 2c = 2M > a \oplus b \oplus c$이므로 불가능합니다.
- 즉 모든 경우에서 불가능하므로, 최종적인 답은 $\min(q, r) + (p - 1)$ 입니다.

D. 1과 5

`case_work`, `constructive`, `math`, `number_theory`

출제진 의도 – **Medium**

- 제출 307번, 정답자 106명 (정답률 35.18%)
- 처음 푼 사람: **sorohue**, 4분
- 출제자: `bubbler`

D. 1과 5

- 먼저 3을 소인수로 만들 수 있는지를 생각해 봅시다.
- 모든 자리 숫자의 합을 3으로 나눈 나머지는 원래의 수를 3으로 나눈 나머지와 같습니다.
- 따라서, 원래 수가 3의 배수이면 아무 숫자도 제거하지 않으면 되고, 나머지가 1이면 1을 하나, 나머지가 2면 5를 하나 제거하는 방법을 생각할 수 있습니다.

D. 1과 5

- 그런데 이것이 불가능한 경우가 두 가지 있습니다. 모든 자리가 1이면서 나머지가 2인 경우, 그리고 모든 자리가 5이면서 나머지가 1인 경우입니다.
- 후자의 경우 항상 5로 끝나므로 5의 배수입니다.
- 전자를 처리하려면 11의 배수로 만드는 방법을 생각할 수 있습니다.
- 모든 자리의 숫자가 같으므로, 그 수의 길이가 짝수이면 11의 배수가 됩니다.
- 길이가 홀수라면 아무 자리를 하나 제거하여 짝수 길이로 만들어주면 됩니다.
- 이러한 판정들의 순서를 잘못 짜면 111이 입력으로 들어올 때 하나를 지우고 11을 출력할 수 있어 주의가 필요합니다.

- 이러한 과정을 거치지 않고 알고리즘적으로 푸는 방법도 있습니다.
- 적당한 소수 p에 대해, N의 모든 prefix와 suffix를 p로 나눈 나머지를 구하고, N의 길이 이하인 음 아닌 정수 n에 대해 $10^n \bmod p$도 구합니다.
- 이들을 잘 조합하여 계산하면, 주어진 수의 모든 위치에 대해 그 숫자를 지운 결과를 p로 나눈 나머지를 N의 길이에 비례하는 시간에 구할 수 있습니다.
- 이 과정을 적당히 작은 p들에 대해 반복하면 AC를 받을 수 있습니다. 앞의 풀이에 의해, 시도해 보는 소수의 목록에 3, 5, 11이 포함되면 올바른 풀이임이 보장됩니다.

E. 판드랄추

`ad_hoc`, `constructive`, `bitmasking`
출제진 의도 – **Medium**

- 제출 162번, 정답자 44명 (정답률 27.78%)
- 처음 푼 사람: **ainta**, 16분
- 출제자: pyb1031

- 일반성을 잃지 않고 $a < b$라고 가정합시다.
- 먼저 a와 b의 홀짝이 다르다면 자명하게 불가능합니다.
- 또한 $d = (a + b)/2$인 d에 대해 $a + d + d = b, b \oplus d \oplus d = b$이므로 2번 이하의 시행으로 처리할 수 있습니다.
- 이제 남은것은 단 1번의 시행으로 a와 b를 같게 할 수 있는지 판별하는 것입니다.

E. 판드랄추

- ✓ $a < b$이므로 a에 더하기 연산, b에 xor연산을 해야합니다.
- ✓ $a + x, b \oplus x$에서 x의 각 비트별로 계산한다고 생각해봅시다.
- ✓ i번째 비트에서 계산할 때 x의 i번째 비트가 1이라면 a의 값은 2^i 만큼 증가합니다.
- ✓ b의 값은 b의 i번째 비트가 1이라면 2^i 만큼 감소하고 아니라면 2^i 만큼 증가하게 됩니다.
- ✓ $b - a$의 값을 0으로 만든다고 생각해보면 b의 i번째 비트가 0인 경우 $b - a$의 값에 변화가 없으므로 x의 i번째 비트의 값을 적당히 0으로 설정해도 됩니다.
- ✓ 이런식으로 하다보면 $b \oplus x = b - x$가 될것이라는 것을 알 수 있습니다.
- ✓ 따라서 1번의 시행으로 충분한지 판별하기 위해 $d = (a + b)/2$인 d에 대해 $a + d = b \oplus d$인지 확인해주는것으로 충분합니다.

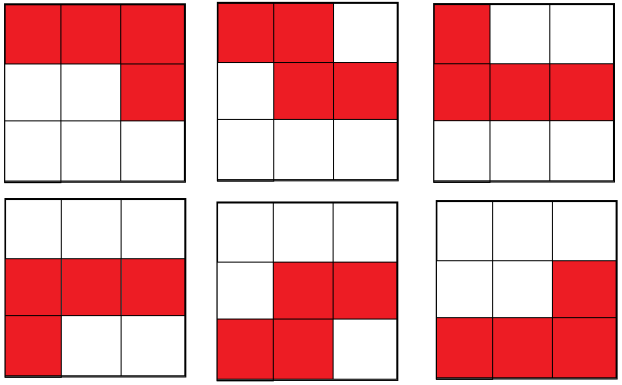
F. 치터 잡기

`ad_hoc, constructive`
출제진 의도 – **Hard**

- ✓ 제출 78번, 정답자 24명 (정답률 30.77%)
- ✓ 처음 푼 사람: **eytns**, 54분
- ✓ 출제자: `ibm2006`

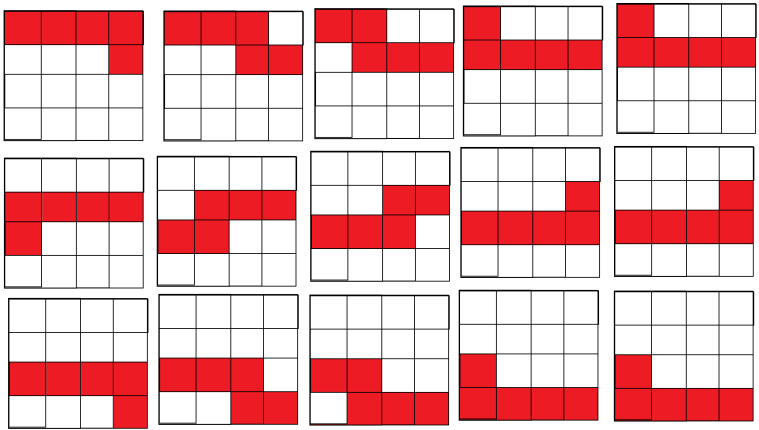
F. 치터 잡기

- ✓ n이 홀수라면, 다음과 같이 구성하면 됩니다:



F. 치터 잡기

✓ n이 짝수라면, 다음과 같이 구성하면 됩니다:



G. 내 맘대로 정렬

`combinatorics`, `dp`, `math`
출제진 의도 – **Hard**

- 제출 46번, 정답자 17명 (정답률 36.96%)
- 처음 푼 사람: **urd05**, 35분
- 출제자: `lunarlity`

G. 내 맘대로 정렬

- 적당히 소원을 만들어보다 보면, 불가능한 경우가 있음을 알 수 있습니다.
- a_{p_i} 가 a_{q_i} 위치로 갈려면 $a_{p_i} > a_{p_i+1},\ a_{p_i} > a_{p_i+2},\ \cdots,\ a_{p_i} > a_{q_i}$ 를 만족해야 합니다. 또한 a_{q_i} 보다 더 오른쪽으로 가면 안되므로 $a_{p_i} < a_{q_i+1}$ 도 만족해야 합니다. 이전 원소가 넘어와도 안되므로 $a_{q_{i-1}} < a_{p_i},\ a_{q_{i-1}+1} < a_{p_i},\ \cdots,\ a_{p_i-1} < a_{p_i}$, 등도 만족해야 합니다.
- $a_{q_i} = N$ 거나 a_{p_0} 인 경우는, 앞으로 적당히 예외를 처리해준다고 생각하겠습니다.
- 이제 주어진 부등호 관계들을 보면, 구간 $[p_i, q_i]$ 중 끝 부분이라도 서로 겹치는 게 있을 경우 부등호 방향이 서로 반대를 가리켜 가능한 경우가 없다는 걸 알 수 있습니다.
- 소원을 정렬해서 편하게 처리할 수 있고, 이때는 0을 출력해주면 됩니다.

G. 내 맘대로 정렬

- 이제 가능한 경우일 때 개수를 세봅시다.
- 부등호 관계들을 모으면 트리를 만들 수 있음을 알 수 있습니다.
- 어떤 수보다 커야 하는 원소는 최대 한 개만 존재하므로, 자식 노드가 부모 노드보다 항상 값이 작아지게 부등호 관계를 모델링하면 될 것 같습니다.
- 만약 아무런 부등호에 영향을 받지 않는 경우, 루트를 $N+1$ 번째 가상 정점으로 바꾸어 $N+1$ 번째 정점에 원소를 넣는다고 생각하면 됩니다.
- 적당한 조합론이나 dp등의 지식을 활용하면, 해당 트리의 원소 개수는 $\frac{N!}{\prod_{i=1}^{N} sz[i]}$ 임을 알 수 있습니다. $sz[i]$ 는 i 번째 정점의 서브트리의 크기를 의미합니다.
- 하지만 N 의 합에는 제한이 없으므로 진짜로 트리를 만들면 시간 초과를 받는데, $sz[i]=1$ 즉 자식 노드가 없는 원소가 상당히 많다는 걸 이용해 최적화를 할 수 있습니다. 이 경우 $2Q$ 정도에 나눠야 할 수가 bound됩니다.

H. 햄부기

`constructive, dp, knapsack`

출제진 의도 – **Hard**

- 제출 19번, 정답자 7명 (정답률 36.84%)
- 처음 푼 사람: **ainta**, 57분
- 출제자: pyb1031

H. 햄부기

- 화난 정도가 0이 된 피돌이들이 밖으로 나간다고 생각해보면, 인접한 두 수를 합쳐 두 수의 차로 바꾸는 문제로 생각할 수 있습니다.
- 따라서 결과 값은 $-a_1 + a_2 + a_3 - \cdots + a_N$ 과 같이 a_1 부터 a_N 까지 적당히 더하고 뺀 값이 나올것임을 알 수 있습니다.
- 위 식의 최솟값이 답의 하한이 되고 이를 실제로 구성할 수 있습니다.

- 먼저 dp를 이용해서 식의 값이 최소가 될 때 어떤 수를 더하고 어떤 수를 빼야하는지 계산합니다.
- 더해야 하는 수에 빨간색, 빼야 하는 수에 파란색을 칠했다고 생각해봅시다.
- 잘 생각해보면 어딘가에 빨간색 수와 파란색 수가 인접한 곳이 존재합니다.
- 여기에 시행을 한 후 새롭게 생긴 수에 더 컸던 수의 색을 칠합니다.
- 이렇게 한다면 (빨간색 수의 합) - (파란색 수의 합)에 변화가 없음을 알 수 있습니다.
- 이를 불가능할 때 까지 계속 하면 되므로 총 $O(N \cdot \sum a)$에 해결할 수 있습니다.
- 비트셋을 쓰지않거나 상수 최적화를 하지 않는다면 제한이 다소 빡빡할 수 있습니다.

I. 식당

`data_structure`, `segtree`, `lazy_prop`
출제진 의도 – **Challenging**

- 제출 10번, 정답자 4명 (정답률 40.00%)
- 처음 푼 사람: **jeoffrey0522**, 56분
- 출제자: pyb1031

- $2\,x$ 쿼리를 빠르게 처리하는 방법을 생각해봅시다.
- 먼저 C를 1, P를 -1로 생각해 누적합 배열 s(0-based)을 만들어봅시다.
- 일반성을 잃지않고 x 번째 피돌이가 C++를 쓰고있다고 가정합니다.
- $s[i] = s[x], i < x$를 만족하는 i의 최댓값을 p라고 합시다. 그런 i가 없다면 $p = 0$이라고 합시다.

- 잘 생각해보면 1초가 지날 때마다 구간 $[p, x]$에서 s의 최댓값과 최솟값의 차가 1씩 작아지는것을 알 수 있습니다. 또한 구간 내의 피돌이들이 다 나가기 전까지 x번째 피돌이도 줄에서 나가지 않습니다.
- 따라서 정답은 $s[x] - \min(s[p], s[p+1], \cdots, s[x-1], s[x])$가 됩니다. x번째 피돌이가 파이썬을 쓰는 경우에는 거꾸로 해주면 됩니다.
- 레이지 세그를 이용해 $O((Q+N)\log N)$에 해결할 수 있습니다. 로그가 하나 더 붙어도 돌도록 제한을 잡았습니다.

J. 피돌이 vs 피붕이

`game_thoery`, `dag`, `sprague_grundy`
출제진 의도 – **Challenging**

- ✓ 제출 9번, 정답자 1명 (정답률 11.11%)
- ✓ 처음 푼 사람: **jh01533**, 170분
- ✓ 출제자: pyb1031

J. 피돌이 vs 피붕이

- stair nim 게임의 변형입니다.
- 먼저 각 노드를 다음 규칙에 따라 흰색과 검정색으로 색칠합니다.
- 1. 이 노드에서 하얀 노드로 향하는 간선이 있다면 검정색으로 칠합니다.
- 2. 아니라면 흰색으로 칠합니다.
- 정의에 의해, 모든 검은 노드에는 하얀 노드로 향하는 간선이 있고 하얀 노드에서는 검은 노드로 밖에 가지 못합니다.
- 따라서 상대가 흰색 노드에서 돌멩이를 옮겼을 때 다시 흰색 노드로 옮기는 전략을 쓴다고 생각하면 남은 검은색 노드에서 그런디수를 구하면 될것입니다.

J. 피돌이 vs 피붕이

- 검은색 노드에서 다른 검은색 노드로 향하는 간선은 최대 1개이므로 포레스트 비슷한 모양이 나옵니다.
- 각 트리에서 할 수 있는 시행은 돌멩이 1개 또는 2개를 골라 없애거나 루트쪽으로 한칸 옮기는 시행이라고 생각할 수 있습니다.
- 이 때 그런디 수는 각 노드에서 ((루트노드와의 거리) mod 2+1)×(노드 위의 돌멩이 개수)의 합을 3으로 나눈 나머지와 같습니다.
- 증명도 어렵지 않게 할 수 있습니다.
- 이 그런디 수들을 xor한 값이 0이면 후공, 아니라면 선공이 이기게 됩니다.