제1회 실버컵 풀이

Official Solutions

by

cozyyg (a.k.a. 16silver)

2022년 3월 12일



문제		의도한 난이도	
A	돌무더기 게임 1	**Easy**	`math`
B	돌무더기 게임 2	**Medium**	`ad_hoc`, `game_theory`
C	Silver-16	**Hard**	`ad_hoc`, `constructive`
D	청소기 마술	**Challenging**	`ad_hoc`, `dp`
E	수식 완성 게임	**Medium**	`precomputation`
F	I와 I	**Hard**	`string`, `dp_bitfield`
G	평면을 채우는 오각형	**Challenging**	`ad_hoc`, `math`, `geometry`



A. 돌무더기 게임 1

`ad_hoc`, `math`
출제진 의도 – **Easy**

- 제출 560번, 정답 119명 (정답률 24.107%)
- 처음 푼 사람: **octane**, 7분
- 출제자: `16silver`



- 매번 **모든 돌무더기의 돌 개수의 홀짝성**이 바뀝니다.
- 처음에 돌 개수가 홀수인 돌무더기(이하 홀수 무더기)가 t개였다고 합시다.
- 선공의 차례에는 언제나 홀수 무더기가 t개, 후공의 차례에는 언제나 $3-t$개입니다.
- 더 이상 아무 것도 할 수 없는 상태는 두 개의 돌무더기에 돌이 0개인 상태뿐입니다.
- 이 상태는 홀수 무더기가 1개 이하입니다. 그리고 t와 $3-t$ 중 정확히 하나만이 1 이하입니다.
- 따라서 두 플레이어가 어떤 시행을 하는지에 관계없이 $t \leq 1$이면 선공이 이기고, 아니면 후공이 이깁니다.



B. 돌무더기 게임 2

`ad_hoc`, `math`, `game_theory`
출제진 의도 – **Medium**

- 제출 284번, 정답 16명 (정답률 5.634%)
- 처음 푼 사람: **kclee2172**, 52분
- 출제자: `16silver`

B. 돌무더기 게임 2



- ✓ 버전 1과는 다르게 직관으로 접근하기 어렵습니다.
- ✓ 다이나믹 프로그래밍 등으로 x, y, z가 작을 때의 답을 구한 뒤 규칙을 찾아봅시다.
- ✓ 결론은 다음과 같습니다.

후공이 이기기 위한 조건

$x \le y \le z, w = \min(z, x + y - 2)$라 할 때

- ✓ $x = 0$이고, y가 홀수
- ✓ $x = 1$이고, $y = z$이며 y가 홀수
- ✓ $x > 1$이고, x, y, w가 모두 짝수
- ✓ $x > 1$이고, $(x + y + w)$를 4로 나눈 나머지가 3

중 하나가 성립할 경우에만 후공이 이깁니다. (이하 [조건]이라 합니다.)



증명을 위해서는 다음 세 가지를 보이면 됩니다.

1. 더 이상 아무것도 할 수 없는 상태는 [조건]을 만족하지 **않는다.** (이 문제는 아무 것도 할 수 없으면 이깁니다.)
2. [조건]을 만족하는 상태의 모든 가능한 다음 상태들은 [조건]을 만족하지 않는 상태다.
3. [조건]을 만족하지 않는 상태의 가능한 다음 상태 중 [조건]을 만족하는 상태가 존재하거나, **다음 상태가 없다.**

✓ 더 이상 아무 것도 할 수 없는 상태는 $x = y = 0$일 때로, [조건]을 만족하지 않습니다.

✓ $x \leq 1$인 경우들을 먼저 봅시다.



[조건]을 만족하는 경우들 ($x \leq 1$)

1. $x = 0, y = 2k + 1$: 다음 상태는 반드시 $(0, 2k, z - 1)$로 [조건]을 만족하지 않습니다.
2. $x = 1, y = z = 2k + 1$: 다음 상태는 $(0, 2k, 2k + 1)$ 또는 $(1, 2k, 2k)$로, 모두 [조건]을 만족하지 않습니다.

[조건]을 만족하지 않는 경우들 ($x \leq 1$)

1. $x = 0, y = 0$: 다음 상태가 없습니다.
2. $x = 0, y = 2k$: 다음 상태는 반드시 $(0, 2k - 1, z - 1)$로 [조건]을 만족합니다.
3. $x = 1, y = 2k + 1, z > y$: 다음 상태들 중 $(0, 2k + 1, z - 1)$이 [조건]을 만족합니다.
4. $x = 1, y = 2k$: 다음 상태들 중 $(0, 2k - 1, z)$이 [조건]을 만족합니다.



✓ 다음으로 $x = 2$인 경우입니다. 이 때 항상 $w = \min(y, z) = y$입니다.

[조건]을 만족하는 경우들 ($x = 2$)

1. $x = 2, y = 2k, z = 2k$: 다음 상태인 $(1, 2k-1, 2k), (2, 2k-1, 2k-1)$ 모두 [조건]을 만족하지 않습니다.
2. $x = 2, y = 2k, z > y$: 다음 상태인 $(1, 2k-1, z), (1, 2k, z-1), (2, 2k-1, z-1)$ 모두 [조건]을 만족하지 않습니다.

[조건]을 만족하지 않는 경우들 ($x = 2$)

1. $x = 2, y = 2k+1$: 다음 상태들 중 $(2, 2k, z-1)$이 [조건]을 만족합니다.



✓ $z \geq x + y - 2$인 경우입니다. 이 때 다음 상태 (x', y', z')에서 항상 $w' = x' + y' - 2$입니다.

[조건]을 만족하는 경우들 $(x > 2, z \geq x + y - 2)$

1. $x = 2k, y = 2t$: 다음 상태인 $(2k - 1, 2t - 1, z), (2k - 1, 2t, z - 1), (2k, 2t - 1, z - 1)$ 모두 [조건]을 만족하지 않습니다.

[조건]을 만족하지 않는 경우들 $(x > 2, z \geq x + y - 2)$

1. $x = 2k, y = 2t + 1$: 다음 상태들 중 $(2k, 2t, z - 1)$이 [조건]을 만족합니다.
2. $x = 2k + 1, y = 2t$: 다음 상태들 중 $(2k, 2t, z - 1)$이 [조건]을 만족합니다.
3. $x = 2k + 1, y = 2t + 1$: 다음 상태들 중 $(2k, 2t, z)$이 [조건]을 만족합니다.



✓ $z \leq x + y - 4$인 경우입니다. 이 때 다음 상태 (x', y', z')에서 항상 $w' = z'$입니다.

[조건]을 만족하는 경우들 $(x > 2, z \leq x + y - 4)$

1. x, y, z가 모두 짝수: 다음 상태에서 홀수 2개, 짝수 1개로 [조건]을 만족하지 않습니다.
2. $(x + y + z) \equiv 3 \pmod 4$: 다음 상태에서 $(x' + y' + z') \equiv 1 \pmod 4$로 [조건]을 만족하지 않습니다.

[조건]을 만족하지 않는 경우들 $(x > 2, z \leq x + y - 4)$

1. x, y, z 중 1개만 짝수: 홀수 2개를 골라서 가져가면 모두 짝수가 되어 [조건]을 만족합니다.
2. $(x + y + z) \equiv 1 \pmod 4$: 다음 상태에서 $(x' + y' + z') \equiv 3 \pmod 4$로 [조건]을 만족합니다.



- ✓ $z = x + y - 3$인 경우입니다. 조건을 만족하는 경우는 $(x + y + z) \equiv 3 \pmod 4$ 뿐이며, 조건을 만족하지 않는 경우는 $(x + y + z) \equiv 1 \pmod 4$ 뿐입니다.
- ✓ 다음 상태들 중 $(x - 1, y - 1, z)$는 $(x - 1, y, z - 1), (x, y - 1, z - 1)$과 별개로 처리합니다.

[조건]을 만족하는 경우들 $(x > 2, z = x + y - 3 \Rightarrow x + y \equiv 1 \pmod 2)$

1. $(x - 1, y - 1, z)$: $w' = (x - 1) + (y - 1) - 2$므로 [조건]을 만족하기 위해서는 $x - 1, y - 1$ 모두 짝수여야 합니다. 이는 $x + y \equiv 1 \pmod 2$에 모순이므로 [조건]을 만족하지 않습니다.
2. 이외의 경우: 다음 상태에서 $(x' + y' + z') \equiv 1 \pmod 4$로 [조건]을 만족하지 않습니다.

[조건]을 만족하지 않는 경우들 $(x > 2, z = x + y - 3 \Rightarrow x + y \equiv 0 \pmod 2)$

$(x - 1, y, z - 1), (x, y - 1, z - 1)$ 각각이 [조건]을 만족합니다.



C. Silver-16

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

- 제출 86번, 정답 2명 (정답률 2.326%)
- 처음 푼 사람: **octane**, 63분
- 출제자: `16silver`



- 지문에서도 언급했듯이, 프로그램에 있는 F의 개수의 홀짝성과 먼지가 있는 칸의 개수의 홀짝성이 다르면 먼지가 있는 칸이 하나 남게 됩니다.
- 그 칸이 정확히 청소기가 멈춘 칸이 되어야 합니다.
- 하지만 항상 같은 프로그램을 실행하기 때문에, 먼지가 있든 없든 F 실행 시 현재 위치한 칸의 상태를 뒤집게 됩니다.
- 결국 먼지가 있는지 없는지에 따라 달라지도록 할 수 있는 상태는 **현재 청소기의 위치**뿐입니다.
- `udlr` 을 이용하여 위치가 달라지도록 할 수 있습니다.

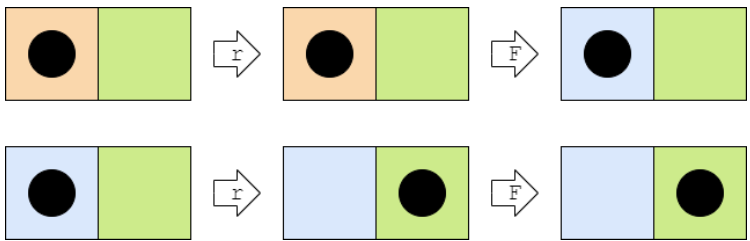




- 다음과 같은 전략을 생각해볼 수 있습니다.

Basic Strategy

- 목표: (x, y)에 먼지가 없도록 합니다.
- 방법: (x, y)에 위치하도록 한 뒤, `rF`를 실행합니다.





- 이 때 청소기의 위치는 (x, y), $(x, y+1)$ 둘 중 하나입니다.
- $(x, y+1)$ 을 뒤집었는지는 알 수 없습니다.
- 현재 칸과 다음 칸 모두가 미지의 칸이어야 이 전략이 의미가 있습니다.
- 청소기의 위치를 확정짓기 위해서는 왼쪽이나 오른쪽으로 끝까지 이동해야 합니다.
- `rFL`, `RrFLL`, `RRrFLLL` 등을 반복하면 됩니다. 이 때 총 프로그램의 길이는 n^3 정도가 됩니다.
- 두 방향 중 가까운 방향을 택하더라도 프로그램의 길이를 $0.5n^3 = 2048$ 이하로 만들기 어렵습니다.



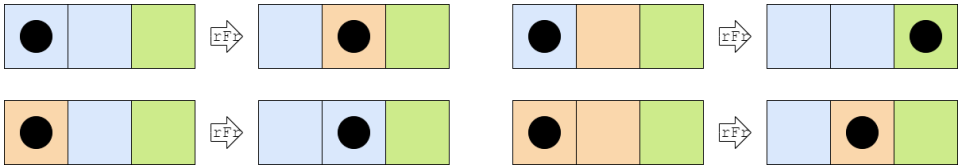
- ✓ $n \times n$ 대신 $1 \times n$ 일 때의 문제를 먼저 풀어봅시다.
- ✓ 프로그램의 가능한 이론상 최소 길이는 얼마일까요?

프로그램의 길이는 최소 $3(n-1)$ 입니다.

- ✓ 모든 칸에 먼지가 있는 경우를 해결하기 위해 F가 최소 $n-1$ 개 있어야 합니다.
- ✓ 조건부 이동 명령이 최소 $n-1$ 개는 있어야 2^{n-1} 가지 서로 다른 위치 sequence가 등장할 수 있습니다.
- ✓ 모든 칸을 방문하기 위해서 최소 $n-1$ 번의 이동이 필요합니다. 한편, 위에서 필요했던 조건부 이동 명령이 등장할 때마다 해당 칸이 먼지가 있도록 초기 상태를 조절할 수 있기 때문에, 총 $2(n-1)$ 번의 이동 명령이 있어야만 합니다.



- ✓ **실제로 길이가 $3(n-1)$ 인 프로그램을 만들 수 있습니다!**
- ✓ 처음 뒤집는 것은 별 의미가 없으므로 `r`를 실행합니다.
- ✓ 이제 `F`를 실행하면, 1번 칸은 먼지가 없습니다.
- ✓ 다음 칸으로 이동합니다. 이 때 처음에 1, 2번 칸에 모두 먼지가 없었다면 현재 2번 칸에 먼지가 있으므로, 이 때는 다음 칸으로 이동하지 말아야 합니다. 따라서 `r`를 실행합니다.
- ✓ 현재까지 가능한 상태를 정리해보면 다음과 같습니다.





- 현재 위치 기준으로 왼쪽에는 먼지가 없습니다.
- 현재 위치는 먼지가 있을 수도 있고 없을 수도 있습니다.
- 따라서 처음과 같은 상황이며 길이가 1 이상 줄어들었습니다.
- `rFr`를 $n-1$번 반복하면 우리가 원하는 길이 $3(n-1)$짜리 프로그램이 완성되었습니다!
- $n \times n$에서도 이 전략을 사용하여 각 가로줄에 맨 끝에만 먼지가 남도록 할 수 있습니다.
- 오른쪽 끝 열은 세로로 다시 한 번 같은 전략을 사용합니다.
- 각 가로줄이 끝난 뒤 $(n-1)$번 왼쪽으로 이동해야 하며, 프로그램의 길이는 $(n-1)(3(n-1)+(n-1)+1)+3(n-1)+3(n-1) = 4n^2-n-3 = 1005$입니다.

C. Silver-16



- 각 가로줄이 끝난 뒤 $(n-1)$ 번 이동하는 것은 비효율적입니다.
- 하지만 번갈아가며 오른쪽, 왼쪽으로 이동한다면 남은 칸을 한 번에 처리할 수 없게 됩니다.
- 오른쪽 그림과 같이 모든 가장자리를 마지막에 방문하면, 프로그램의 길이는 $3(n^2-1)+2(n-2)=3n^2+2n-7=793$ 이며, 이는 문제 제한인 **800**에 들어맞습니다.

1	2	3	4	5	
6	7	8	9	10	35
11	12	13	14	15	34
16	17	18	19	20	33
21	22	23	24	25	32
26	27	28	29	30	31

1	2	3	4	5	
	10	9	8	7	6
11	12	13	14	15	
	20	19	18	17	16
21	22	23	24	25	
	30	29	28	27	26

21	22	23	24	25	26
20	1	2	3	4	27
19	8	7	6	5	28
18	9	10	11	12	29
17	16	15	14	13	30
	35	34	33	32	31



D. 청소기 마술

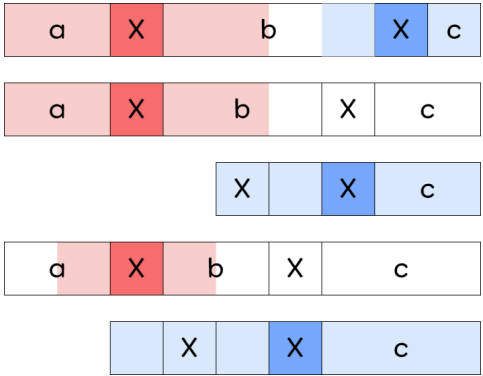
`ad_hoc`, `dp`
출제진 의도 – **Challenging**

- 제출 13번, 정답 0명 (정답률 0.000%)
- 처음 푼 사람: -, -분
- 출제자: `16silver`



- 입출력 기준으로 다시 문제를 써보면, ?X? $\rightarrow$ X, X $\rightarrow \epsilon$의 변환이 가능할 때, 원래 문자열을 빈 문자열로 만들어야 합니다.
- 초기 문자열을 S, 첫 번째 변환을 $f(idx)$, 두 번째 변환을 $g(idx)$로 표기합니다.
- (idx는 처음 위치 기준 1-based입니다)
- 가능한 케이스와 불가능한 케이스를 분류해야 합니다.
- 문자열의 길이 $n = 2k + 1$인 경우를 먼저 봅니다.
- $S[k+1] =$ X면, $f(k+1)$ k번, $g(k+1)$ 1번으로 가능합니다.
- 전체 문자열에서 X가 단 하나일 때는 그 X가 $k+1$번째여야만 가능합니다.

- 다음으로 X가 두 개일 때입니다.
- 두 X에 의해 나눠지는 세 구역의 . 개수를 각각 a, b, c라 합니다.
- f 한 번마다 b는 1씩 줄어듭니다.
- 두 위치에서 시행할 수 있는 f 변환의 횟수는 각각 a, c입니다.
- 그러므로 $b > a + c$면 불가능합니다.
- 반대로 $b < a + c$면 가능합니다.





- 오른쪽 X가 가운데에 오도록 하는 것이 목표입니다.
- 오른쪽 X 전까지의 문자열의 길이가 c가 되어야 합니다.
- 왼쪽 X에 f를 $(a+b+1-c)/2$번 적용하면 됩니다.
- 이것이 가능하기 위해서는 $0 \le (a+b+1-c)/2 \le \min(a,b)$여야 합니다.
- 정리하면 $a+b+1-c \ge 0, a+c \ge b+1, b+c \ge a+1$입니다.
- $n = a+b+c+2$이므로 $n-3 \ge 2b, n-3 \ge 2a, n-1 \ge 2c$이어야 합니다.
- 왼쪽 X가 가운데에 오려면 $n-3 \ge 2b, n-3 \ge 2c, n-1 \ge 2a$이면 됩니다.
- 이에 해당하지 않는 경우는 $n-3 < 2b, n-1 < 2a, n-1 < 2c$ 중 하나입니다.



- ✓ 위 결론을 일반적인 경우로 확장할 수 있습니다.

  n 이 홀수일 때, 빈 문자열로 만들 수 있을 필요충분조건

  S 를 빈 문자열로 만들 수 있음은, S 의 연속한 . 의 개수의 최댓값 M 에 대해

- ✓ $M \leq (n-3)/2$
- ✓ $M = (n-1)/2$ 고, 그 구간의 양쪽 끝 중 하나가 문자열의 끝 중 하나를 만족하는 것과 동치입니다.
- ✓ 가능 여부 판단, construction 모두 $\mathcal{O}(n)$ 에 가능합니다.



- n이 짝수인 경우, g 변환만이 문자열의 길이의 홀짝성을 바꿉니다.
- g 변환이 없으면 빈 문자열이 될 수 없으므로, g 변환은 최소 두 번 있어야 합니다.
- 변환의 순서를 잘 바꾸면, 어떤 홀수 t에 대해 $S[1..t]$와 $S[t+1..n]$을 각각 빈 문자열로 바꾼 sequence가 됨을 증명할 수 있습니다.
- 그러므로 각 홀수 t에 대해 $S[1..t]$와 $S[t+1..n]$가 모두 조건을 만족하는지 확인하면 됩니다.
- $\mathcal{O}(n^2)$이지만, DP를 이용하면 $\mathcal{O}(n)$에 가능합니다.



E. 수식 완성 게임

`implementation`, `bruteforcing`, `game_theory`, `precomputation`
출제진 의도 – **Medium**

- 제출 36번, 정답 1명 (정답률 2.778%)
- 처음 푼 사람: **cologne**, 153분
- 출제자: `16silver`



- 문제 조건 그대로 구현하면 됩니다.
- 칠판에 현재 쓰여진 수로 목표 수를 만들 수 있는지는, 연산의 종류와 순서로 가능한 모든 경우를 직접 계산해보면 됩니다.
- 나눗셈의 값을 유리수로 저장해야 함에 유의합니다.
- 게임의 상태는 현재 칠판에 쓰여 있는 수가 되며, 이기는 상태는 재귀적으로 정의됩니다.
- 현재 상태에서 가능한 모든 다음 상태가 이기는 상태라면 지는 상태, 하나라도 지는 상태라면 이기는 상태입니다.
- 상태 계산이 오래 걸리고, 상태의 개수도 생각보다 많기 때문에 전처리가 필요합니다.
- 1-5 중 어떤 수를 먼저 시도해볼지도 프로그램의 실행 시간에 영향을 줍니다.



F. I와 I

`string`, `dp_bitfield`
출제진 의도 – **Hard**

- 제출 3번, 정답 1명 (정답률 33.333%)
- 처음 푼 사람: **koosaga**, 75분
- 출제자: `16silver`



- LCS를 다이나믹 프로그래밍으로 구할 때 사용하는 점화식은 다음과 같습니다.

$$dp(i,j) = \begin{cases} dp(i-1,j-1)+1 & S[i]=T[j] \\ \max(dp(i-1,j), dp(i,j-1)) & else \end{cases}$$

- $dp(i,j)(1 \le j \le n)$은 $dp(i-1,j)(1 \le j \le n)$로부터 결정됩니다.
- 그러므로 $dp(i,j)(1 \le j \le n)$를 하나의 상태로 두고, 각 상태에서 두 문자가 추가될 때 다음 상태를 알아낼 수 있다면 다이나믹 프로그래밍으로 문제를 풀 수 있습니다.



- ✓ 위 dp 식의 이웃한 두 항은 최대 1만큼 차이납니다.
- ✓ 따라서 가능한 상태의 개수는 최대 2^n 입니다.
- ✓ 시간복잡도는 $\mathcal{O}(m \cdot 2^n)$ 입니다.
- ✓ 실제로는 상태 개수가 22000을 넘지 않습니다.

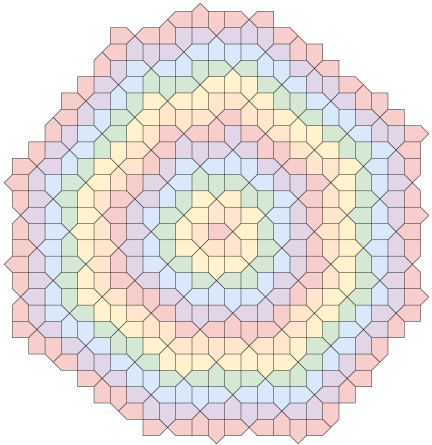


G. 평면을 채우는 오각형

`ad_hoc`, `math`, `geometry`
출제진 의도 – **Challenging**

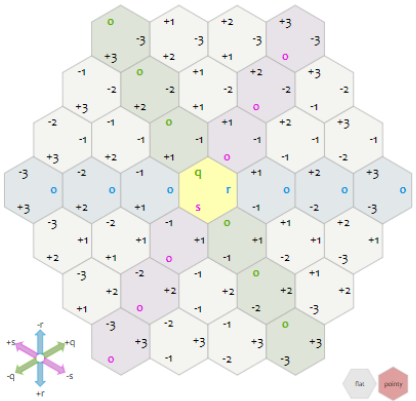
- ✓ 제출 25번, 정답 3명 (정답률 12.000%)
- ✓ 처음 푼 사람: **angello10**, 134분
- ✓ 출제자: `16silver`

G. 평면을 채우는 오각형



- 평면을 채우는 오각형은 총 15가지 타입 중 하나에 속합니다.[1]
- 이 문제에서 등장한 타일링에는 Cairo Pentagonal tiling이라는 이름이 붙어 있습니다.
- 거리를 직접 구해봐도 규칙을 찾기는 쉽지 않습니다.

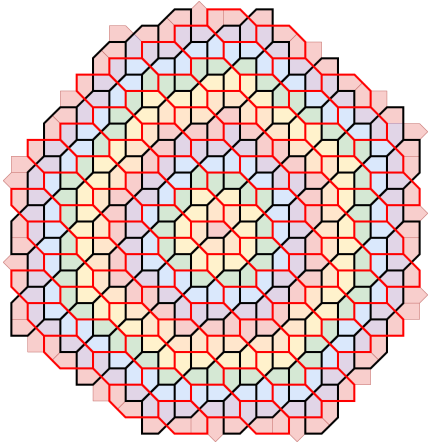
[1] 이 리스트는 2015년에 완성되었고, 2017년에 이 리스트의 completeness가 증명되었습니다.



- 다소 뜬금없지만, 정다각형 타일링 중 정육각형 타일링을 살펴봅시다.
- 좌표공간상의 평면 $x + y + z = 0$의 정수 격자점들이 육각형의 중심이 됩니다.
- 이 때 이웃한 두 점은 x, y, z 좌표 중 두 개가 1 차이 납니다.
- 두 육각형 $(x_1, y_1, z_1), (x_2, y_2, z_2)$ 사이의 거리는 다음과 같습니다.[2]

$$D = \frac{1}{2}\Big(|x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2|\Big)$$

[2]참고: https://www.redblobgames.com/grids/hexagons/#coordinates-cube



- 문제의 오각형 타일링은 육각형 타일링 두 개가 교차하는 것으로 생각할 수 있습니다!
- 이웃한 오각형으로 이동할 때 두 육각형 좌표계 중 한쪽에서만 거리를 1 줄일 수 있습니다.
- 따라서 두 오각형 사이의 거리는 두 육각형 좌표계에서의 거리의 합 이상입니다.
- 그리고 두 육각형 좌표계에서의 거리의 합이 구하려는 답임을 증명할 수 있습니다.[3]

[3] Kovács, G., Nagy, B., & Turgay, N. D. (2021). Distance on the Cairo pattern.