

DP #4

김현정 Acka1357@gmail.com

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $3 \times N$ 크기의 벽을
- 2×1 , 1×2 크기의 타일로 채우는 경우의 수
- $1 \leq N \leq 30$

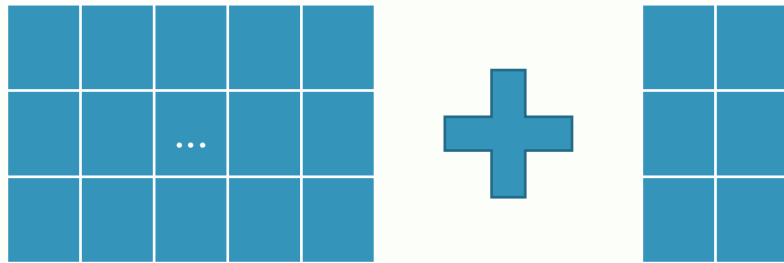
타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i]$: i 번 줄까지 꼭 채우는 경우의 수

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



타일 채우기

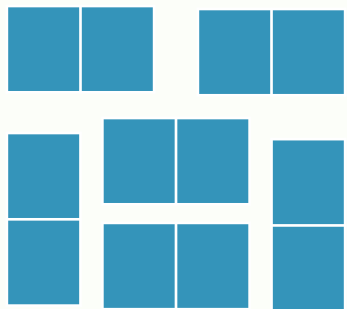
Problem: <https://www.acmicpc.net/problem/2133>



$$D[i]: D[i - 2] + 3$$

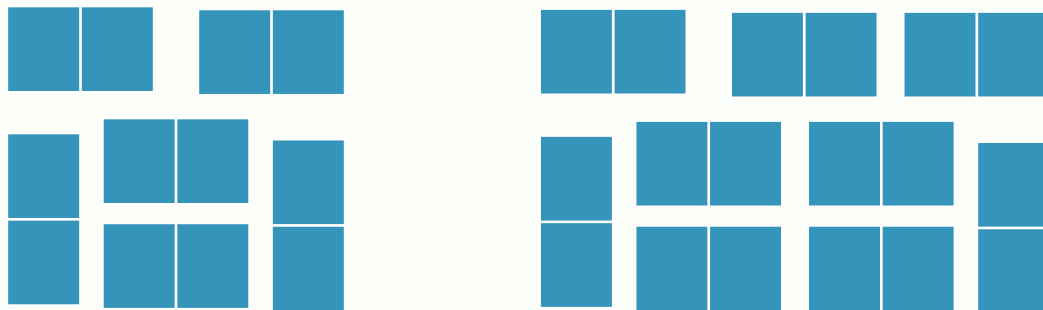
타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



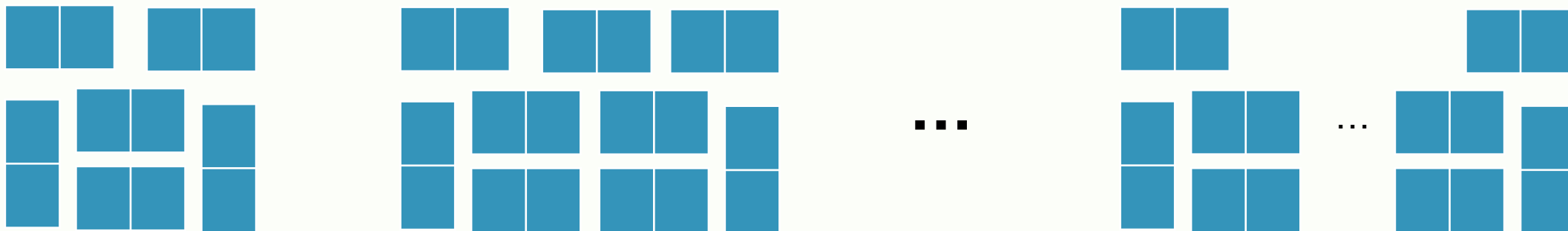
타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



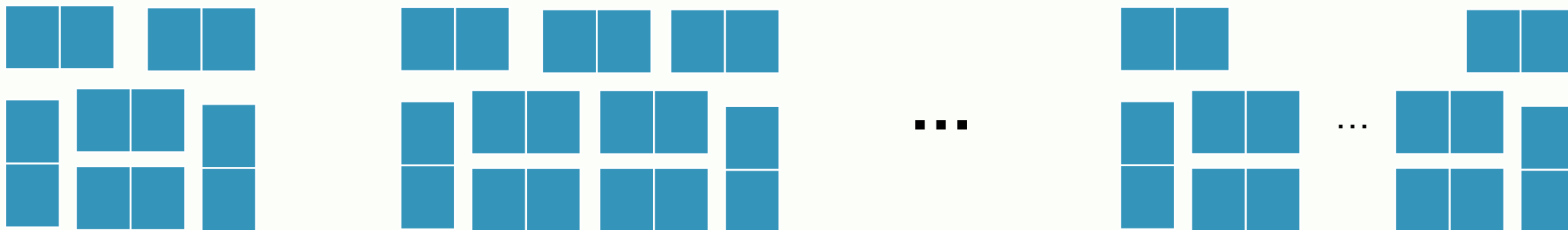
타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>



$$D[i]: \sum_{j=0 \text{ to } i-2} (2 * D[i-j])$$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i]$: i 번 줄까지 꼭 채우는 경우의 수
- $D[i] = D[i - 2] * 3 + \sum_{j = \text{even } 0 \text{ to } i - 4} (2 * D[j])$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i]$: i 번 줄까지 꼭 채우는 경우의 수
- $D[i] = D[i - 2] * 3 + \sum_{j = \text{even } 0 \text{ to } i - 4} (2 * D[j])$
- 답: $D[N]$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

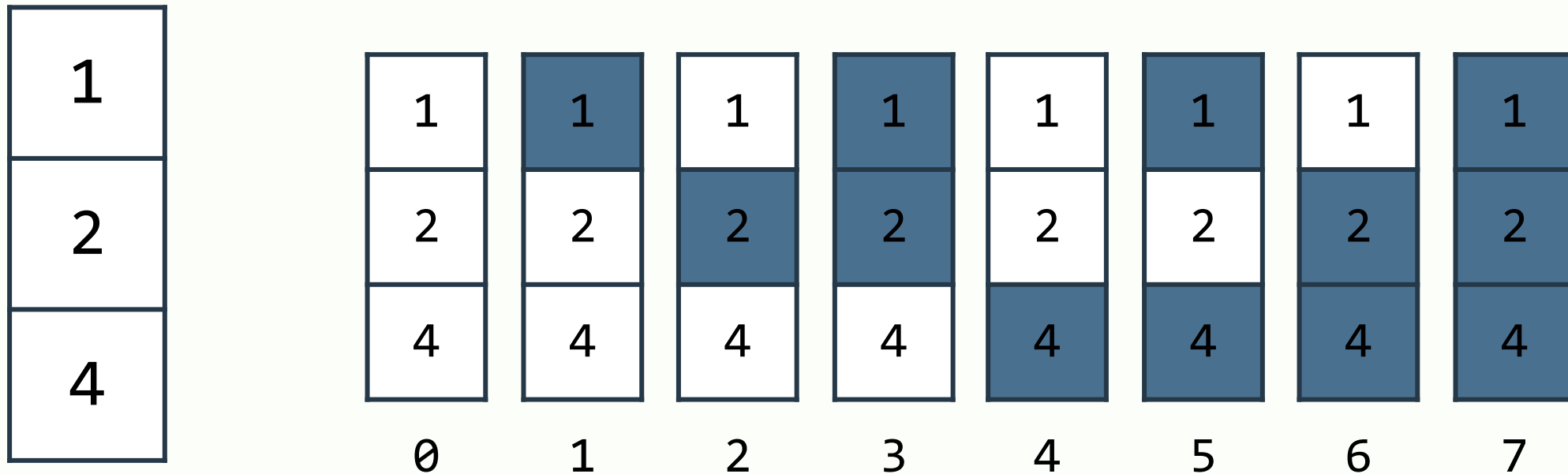
- 마지막 줄의 상태를 아래와 같이 표현해보자.

1
2
4

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- 마지막 줄의 상태를 아래와 같이 표현해보자.



타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i][j]$: $i - 1$ 번 줄까지 꼭 채우고 i 번 줄의 상태가 j 인 경우의 수

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- 각각의 $D[i - 1]$ 상태에서는 다음과 같이 타일을 채울 수 있다.

1	1
2	2
4	4

0

1	1
2	2
4	4

1

1	1
2	2
4	4

4

1	1
2	2
4	4

7

$$D[i][1] = D[i - 1][0] \quad D[i][4] = D[i - 1][0] \quad D[i][7] = D[i - 1][0]$$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- 각각의 $D[i - 1]$ 상태에서는 다음과 같이 타일을 채울 수 있다.

1	1
2	2
4	4

1

1	1
2	2
4	4

0

1	1
2	2
4	4

6

$$D[i][0] = D[i - 1][1] \quad D[i][6] = D[i - 1][1]$$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- 각각의 $D[i - 1]$ 상태에서는 다음과 같이 타일을 채울 수 있다.

1	1
2	2
4	4

2

1	1
2	2
4	4

5

$$D[i][5] = D[i - 1][2]$$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- 각각의 $D[i - 1]$ 상태에서는 다음과 같이 타일을 채울 수 있다.

1	1
2	2
4	4

3

1	1
2	2
4	4

4

1	1
2	2
4	4

7

$$D[i][4] = D[i - 1][3] \quad D[i][7] = D[i - 1][3]$$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i][j]$: $i - 1$ 번 줄까지 꼭 채우고 i 번 줄의 상태가 j 인 경우의 수
- $D[i][0] = D[i-1][7]$
- $D[i][1] = D[i-1][6]$
- $D[i][2] = D[i-1][5]$
- $D[i][4] = D[i-1][3]$
- $D[i][3] = D[i-1][4] + D[i-1][7]$
- $D[i][6] = D[i-1][1] + D[i-1][7]$
- $D[i][5] = D[i-1][2]$
- $D[i][7] = D[i-1][0] + D[i-1][3] + D[i-1][6]$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- $D[i][j]$: $i - 1$ 번 줄까지 꼭 채우고 i 번 줄의 상태가 j 인 경우의 수
- 답: $D[N][7]$

타일 채우기

Problem: <https://www.acmicpc.net/problem/2133>

- C/C++: <https://gist.github.com/Acka1357/0755cd75916a7db902d8ec641df048d0>
- Java: <https://gist.github.com/Acka1357/12a43080c9563c37bdbbc1d99cfe37cec>

Bitwise Operator

이진수에 대해 비트 단위로 적용되는 연산

Bitwise Operator

이진수에 대한 비트연산

- $A \mid B \Rightarrow A \text{ or } B$
- $A \& B \Rightarrow A \text{ and } B$
- $A \wedge B \Rightarrow A \text{ xor } B$
 - $A = 1001, B = 0101$
 - $A \mid B = 1101, A \& B = 0001, A \wedge B = 1100$
- $\sim A \Rightarrow \text{ones complement of } A$
- $A \ll x \Rightarrow \text{left shift } A \text{ as } x$
- $A \gg x \Rightarrow \text{right shift } A \text{ as } x$
 - $A = 00110101$
 - $\sim A = 11001010, A \ll 2 = 11010100, A \gg 2 = 00001101$

Bitwise Operator

이진수에 대한 비트연산

- In Java,
- $00111111 \gg 2 = 00001111$
- $10100111 \gg 2 = 11101001$

- use \gg
- $00111111 \gg 2 = 00001111$
- $10100111 \gg 2 = 00101001$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 1번부터 N번 도시가 있다.
- 어느 한 도시에서 출발해 N개의 도시를 모두 거쳐 원래 도시로 돌아오는 순회 여행 경로
- 한번 방문한 도시는 다시 방문할 수 없다. (출발 도시 제외)
- 이러한 여행 경로 중 비용의 최소값

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 갈 수 있는 모든 경로
- 첫번째 도시를 선택하는 경우의 수 * 두번째 도시를 선택하는 경우의 수 * 세번째 도시 ...
- $N * (N - 1) * (N - 2) * \dots * 1 \Rightarrow O(N!)$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 갈 수 있는 모든 경로
- 첫번째 도시를 선택하는 경우의 수 * 두번째 도시를 선택하는 경우의 수 * 세번째 도시 ...
- $N * (N - 1) * (N - 2) * \dots * 1 \Rightarrow O(N!)$
- $2 \leq N \leq 16$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 갈 수 있는 모든 경로
- 첫번째 도시를 선택하는 경우의 수 * 두번째 도시를 선택하는 경우의 수 * 세번째 도시 ...
- $N * (N - 1) * (N - 2) * \dots * 1 \Rightarrow O(N!)$
- $2 \leq N \leq 16$
- $16! \Rightarrow 20,922,789,888,000$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 1 -> 2 -> 3 -> 4 -> ?
- 1 -> 3 -> 2 -> 4 -> ?

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 1 -> 2 -> 3 -> 4 -> ?
- 1 -> 3 -> 2 -> 4 -> ?
- 두 경로 뒤에 올 수 있는 경로는 같다.
- 현재 위치와 남은 도시의 종류가 같기 때문

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 필요한 정보
- 현재 위치
- 남은 도시의 종류

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 필요한 정보
- 현재 위치
- 남은 도시의 종류
 - (1번 도시가 남았거나 아니거나) * (2번 도시가 남았거나 아니거나) * (3번 도시가 남았거나 아니거나) * ... * (N번 도시가 남았거나 아니거나)
 - 2^N 종류

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- $d[i][j]$: j 의 1인 비트에 해당하는 도시들을 방문했고,
현재 i 도시에 있을 때 경로의 최소 비용

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- $d[i][j]$: j 의 1인 비트에 해당하는 도시들을 방문했고,
현재 i 도시에 있을 때 경로의 최소 비용

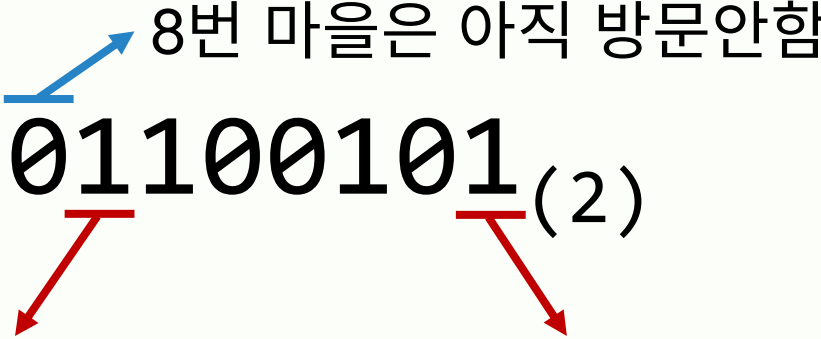
$$j: 101 = 01100101_{(2)}$$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- $d[i][j]$: j 의 1인 비트에 해당하는 도시들을 방문했고,
현재 i 도시에 있을 때 경로의 최소 비용

$j: 101 = 01100101_{(2)}$



8번 마을은 아직 방문안함
 7번 마을은 방문함 1번 마을은 방문함

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- $d[i][j]$: j 의 1인 비트에 해당하는 도시들을 방문했고,
현재 i 도시에 있을 때 경로의 최소 비용
- $d[cur][visited] = \min_{past = 0 \text{ to } N-1} (d[past][visited - 2^{past}] + cost[past][cur])$
 - 2^{past} : $past$ 도시의 상태 비트
 - $past$ 는 cur 가 아니면서, 이미 visited한 도시 중 하나
 - $visited \& 2^{past} \neq 0$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 한 도시에서 출발해 그 도시로 돌아오는 순회문제이므로
- 출발점(끝점)이 어디든 상관 없다.

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- 시간복잡도: $O(N * N * 2^N)$
 - 모든 상태에 대해서: $O(N * 2^N)$
 - 이전 마을을 선택한다: $O(N)$

외판원 순회

Problem: <https://www.acmicpc.net/problem/2098>

- C/C++: <https://gist.github.com/Acka1357/be750aba1f1950f50c93dce7d038752c>
- Java: <https://gist.github.com/Acka1357/25a34b9ea7e49ad148fe4cfe2fe6143b>

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $N \times M$ 격자판을
 - $2 \times 1, 1 \times 2$ 도미노로
 - 빈 칸 없이 채우는 경우의 수
-
- $1 \leq N, M \leq 14$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 3 X N을 채우듯 모든 경우를 직접 구하기 어렵다.
- 각 줄의 상태가 2^M 가지
- $2^{14} = 16,384$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 아래와 같은 격자판이 있을때, 한 줄 단위로 생각해보자.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
							
							

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 모든 칸을 채워야하기 때문에, 1을 채우는 경우는 두 가지가 있다.
- (0, 1) 또는 (0, 10)을 한 도미노로 채우는 경우

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
							
							

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $(0, 1)$ 을 한 도미노로 채우는 경우

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
							
							

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 각 칸의 상태를 도미노의 유무에 따라 0과 1로 구분할 수 있다.
- 첫 칸을 0번 자리, 마지막 칸을 M-1번 자리라고 하면

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
							
							

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $\text{status} = 2^0 + 2^1$

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
				...					
				...					

0으로부터 한 줄의 상태:

... 0000000011₍₂₎

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- (0, 10)을 한 도미노로 채우는 경우

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
							
							

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- status = 2^0 (한 줄만)

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
				...					
				...					

0으로부터 한 줄의 상태:

... 0000000001₍₂₎

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 다음 1번 칸을 채울때, 0번은 무조건 채워져있으므로,
- 0번 칸의 정보는 필요가 없다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
				...					
				...					

0으로부터 한 줄의 상태:

... 0000000001₍₂₎

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 0번 타일을 채울때 10번 타일을 사용했으므로,
- 10번 타일 까지의 정보가 필요하다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
				...					
				...					

1로부터 ‘한 줄’의 상태:
 ... 1000000000₍₂₎

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- i 번 타일을 채울 때 $(i + M)$ 번 타일을 채울 수도 있다.
- 다음 상태로 갈때에는 $i + 1$ 번 부터 $i + M$ 번 타일의 정보가 필요하다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
				...					
				...	<div>1로부터 M칸의 상태:</div> <div>1000000000₍₂₎</div>				

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 12번 칸에 대해서

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

12부터 M칸의 상태:
 $1111001001_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 12번 칸에 대해서
- 12번 칸은 이미 타일이 놓여져있으므로, 13번 칸을 채워야한다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

12부터 M칸의 상태:
 $1111001001_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 13부터 M칸의 상태는
- 12부터 M칸의 상태를 모두 오른쪽으로 한 칸씩 미는 것과 같다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

12부터 M칸의 상태:
 $1111001001_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $\text{status} \gg 1$ (or $\text{status} \neq 2$)

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 이때 13번 칸을 채우기 위한 경우 역시
- (13, 14)와 (13, 23)으로 나뉜다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- (13, 14)를 놓기 위해서는 (13, 14)가 모두 비어 있어야 한다.
- 0번째자리와 1번째자리 모두 0이어야 한다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- (13, 14)에 도미노를 놓으면,
- 14부터 M칸의 상태는 다음과 같이 표현할 수 있다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $(\text{status} \gg 1) + 2^0$
- 14부터 M칸의 상태에서 14번은 0번자리

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

14부터 M칸의 상태:
 $0011110011_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- (13, 23)을 놓기 위해서는 (13, 23)가 모두 비어 있어야 한다.
- 23번 칸은 당연히 비어있다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- (13, 23)에 도미노를 놓으면,
- 14부터 M칸의 상태는 다음과 같이 표현할 수 있다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

13부터 M칸의 상태:
 $0111100100_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $(\text{status} \gg 1) + 2^{M-1}$
- 14부터 M칸의 상태에서 23번칸은 M-1번 자리이다.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
				...					

14부터 M칸의 상태:
 $1011110010_{(2)}$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $D[i][status]$: $i - 1$ 번 타일까지 다 채워져있고,
 i 번부터 M 개 타일의 상태가 $status$ 인 경우의 수
 - $0 \leq status < 2^M$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- i 번 칸이 차있을 때 ($\text{if } (\text{status} \ \& \ 1) == 1$)
- $d[i + 1][\text{status} \gg 1] += d[i][\text{status}]$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- i 번 칸이 차있을 때 ($\text{if } (\text{status} \& 1) == 1$)
- $d[i + 1][\text{status} \gg 1] += d[i][\text{status}]$
- i 번 칸이 차있지 않을 때
 - 가로로 도미노를 놓을 수 있을 때
 - 세로로 도미노를 놓을 수 있을 때

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 가로로 도미노를 놓을 수 있을 때 ($\text{if } (\text{status} \& 3) == 0$)
 - $d[i + 1][(\text{status} \gg 1) + 1] += d[i][\text{status}]$
- 세로로 도미노를 놓을 수 있을 때 ($\text{if } (\text{status} \& 1) == 0$)
 - $d[i + 1][(\text{status} \gg 1) + 2^{M-1}] += d[i][\text{status}]$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 가로로 도미노를 놓을 수 있을 때 ($\text{if } (\text{status} \& 3) == 0$)
 - $d[i + 1][(\text{status} \gg 1) + 1] += d[i][\text{status}]$
- 세로로 도미노를 놓을 수 있을 때 ($\text{if } (\text{status} \& 1) == 0$)
 - $d[i + 1][(\text{status} \gg 1) + 2^{M-1}] += d[i][\text{status}]$
- 단, i번칸이 오른쪽 끝일 경우 가로로 도미노를 놓을 수 없다.

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- $d[i + 1][(\text{status} \gg 1)] += d[i][\text{status}]$
 - if $(\text{status} \& 1) == 1$
- $d[i + 1][(\text{status} \gg 1) + 1] += d[i][\text{status}]$
 - if $(\text{status} \& 3) == 0$, and $(i \% M) \neq M - 1$
- $d[i + 1][(\text{status} \gg 1) + 2^{M-1}] += d[i][\text{status}]$
 - if $(\text{status} \& 1) == 0$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 답: $d[N * M][0]$
 - $N * M$ 번칸 직전이자 마지막 칸인 $(N * M - 1)$ 번칸까지 다 차있으며,
 - $N * M$ 번칸부터는 모두 비어있는 경우의 수

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- 시간복잡도: $O(N * M * 2^M)$
 - 모든 칸에 대해서: $O(N * M)$
 - 모든 상태를 전이: $O(2^M)$

격자판 채우기

Problem: <https://www.acmicpc.net/problem/1648>

- C/C++:
<https://gist.github.com/Acka1357/473f67f678c484b19e5d5daabde7522c>
- Java:
<https://gist.github.com/Acka1357/3406495ff3597161c4245d9989dc3fa7>

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- $N \times M$ 두부판
 - $2 \times 1, 1 \times 2$ 로 두부를 자른다
 - 위치마다 두부의 등급이 다르다.
 - 등급 조합에 따라 가격이 결정된다.
-
- 자를 수 있는 두부가격 합의 최대값
 - $1 \leq N, M \leq 14$

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- 두부를 자르는 것 == 도미노를 놓는 것
- 단, 모든 칸을 다 사용하지 않아도 된다.

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- 두부를 자르는 것 == 도미노를 놓는 것
- 단, 모든 칸을 다 사용하지 않아도 된다.
- 칸이 비어 있어도, 다음 칸으로 넘어갈 수 있다.

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- $D[i][status]$: $i - 1$ 번 두부까지 (안)잘랐고,
i번부터 M개 두부의 상태가 status일 때 가격합의 최대값
 - $0 \leq status < 2^M$

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- i 번 두부가 잘렸을 때 ($\text{if } (\text{status} \ \& \ 1) == 1$)
- $d[i + 1][\text{status} \gg 1] \leftarrow d[i][\text{status}]$

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- i 번 두부가 잘렸을 때 ($\text{if } (\text{status} \& 1) == 1$)
- $d[i + 1][\text{status} \gg 1] \leftarrow d[i][\text{status}]$
- i 번 두부가 잘리지 않았을 때
- $d[i + 1][\text{status} \gg 1] \leftarrow d[i][\text{status}]$
- $d[i + 1][(\text{status} \gg 1) + 1] += d[i][\text{status}]$ (가로)
- $d[i + 1][(\text{status} \gg 1) + 2^{M-1}] += d[i][\text{status}]$ (세로)

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- $d[i + 1][(status \gg 1)] \leftarrow d[i][status]$
 - always possible
- $d[i + 1][(status \gg 1) + 1] += d[i][status]$
 - if $(status \& 3) == 0$, and $(i \% M) \neq M - 1$
- $d[i + 1][(status \gg 1) + 2^{M-1}] += d[i][status]$
 - if $(status \& 1) == 0$

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- 답: $d[N * M][0]$
 - $N * M$ 번칸 직전이자 마지막 칸인 $(N * M - 1)$ 번칸까지 다 (안)잘랐으며,
 - $N * M$ 번부터는 모두 사용하지 않았을 때 가격합의 최대값

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- 시간복잡도: $O(N * M * 2^M)$
 - 모든 칸에 대해서: $O(N * M)$
 - 모든 상태를 전이: $O(2^M)$

두부장수 장홍준

Problem: <https://www.acmicpc.net/problem/1657>

- C/C++:
<https://gist.github.com/Acka1357/99cd03e1d98611f02840ab9708052c99>
- Java:
<https://gist.github.com/Acka1357/6d5dbdcc1528957a2896a703f6fa796c>