S213 - Chinese Checkers

Vincent Chiu (VCLH)

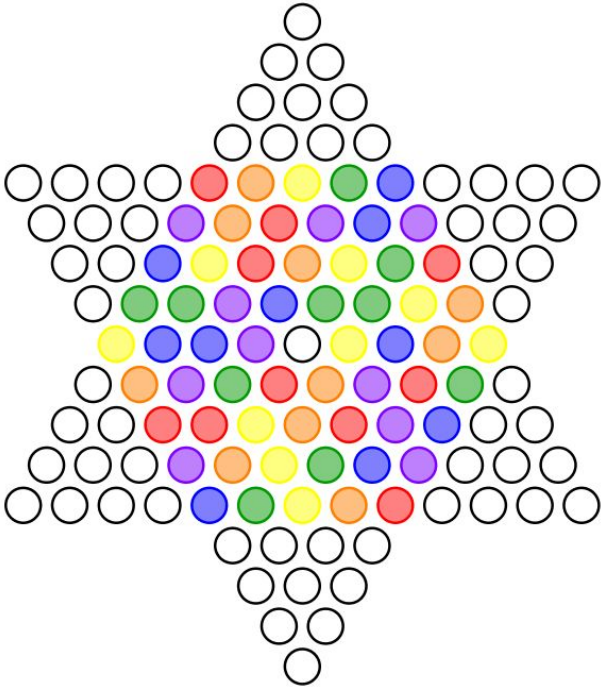
香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Author: VCLH

Setter: VCLH, yaufung

Simulator: nhho (Did you enjoy?)



Background

Source: Wikipedia

歷史

中國跳棋的前身是正方跳棋（Halma），是由美國人George Howard Monks於1883年到1884年發明的[1]，也說法認為是1885年發明[2]。這種跳棋可供2人或4人進行遊戲，棋盤為正方形，共有256格，開始時棋子分佈在角落，以最快跳到對角為目標，規則和現在的中國跳棋相似。Halma原文為希臘文的 ἅλμα，為跳躍之意，遊戲的靈感則來自一個於1854年發明的英國遊戲Hoppity[3]。

正方跳棋誕生後，很快又出現了使用六角星形棋盤的變種，在1892就由德國著名的遊戲公司Ravensburger取得專利，被命名為Sternhalma，意為星形跳棋[1]，也就是後來所稱的中國跳棋。與正方跳棋相比，遊戲的變化和所需的技巧更加複雜。這個遊戲在20世紀初期逐漸在各國開始流行，其較早的英文名為Hop Ching Checker Game，但隨後被改為Chinese Checkers，但事實上和中國沒有關係[1]，只是為了從營銷角度上增加神秘感。中國跳棋的稱法來自英語，而在粵語中稱作波子棋，因為彈珠也被廣泛用作棋子，彈珠在粵語中的說法即為波子。

日本、韓國有種稱為鑽石跳棋的中國跳棋變體，棋子有王兵之分。

History and nomenclature

Boys playing Hop Ching Checkers, Montreal, 1942

Despite its name, the game is not a variation of checkers, nor did it originate in China or any part of Asia. The game was invented in Germany in 1892 under the name "Stern-Halma" as a variation of the older American game Halma.[6] The "Stern" (German for *star*) refers to the board's star shape (in contrast to the square board used in Halma).

The name "Chinese Checkers" originated in the United States as a marketing scheme by Bill and Jack Pressman in 1928. The Pressman company's game was originally called "Hop Ching Checkers".[7]

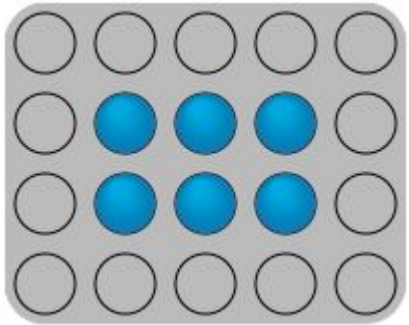
In Japan, the game is known as "Diamond Game" (ダイヤモンドゲーム). The game was introduced to Chinese-speaking regions mostly by the Japanese,[6] where it is known as *Tiaoqi* (Chinese: 跳棋, "jump chess").

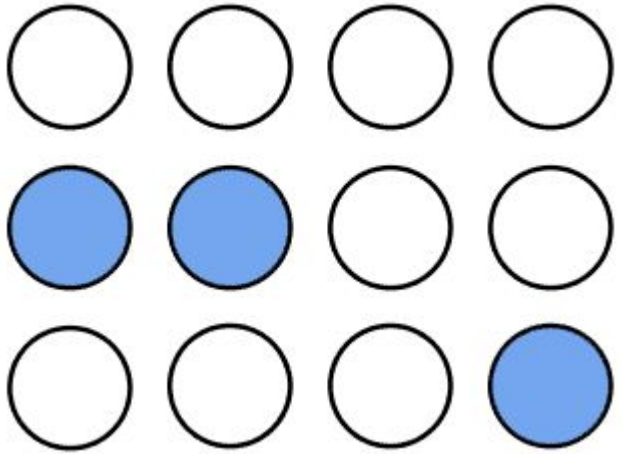
The Problem

Given **R x C** marbles on **(R + 2) x (C + 2)** checkerboard

Remove all marbles except **1**

E.g. **R = 2, C = 3**:

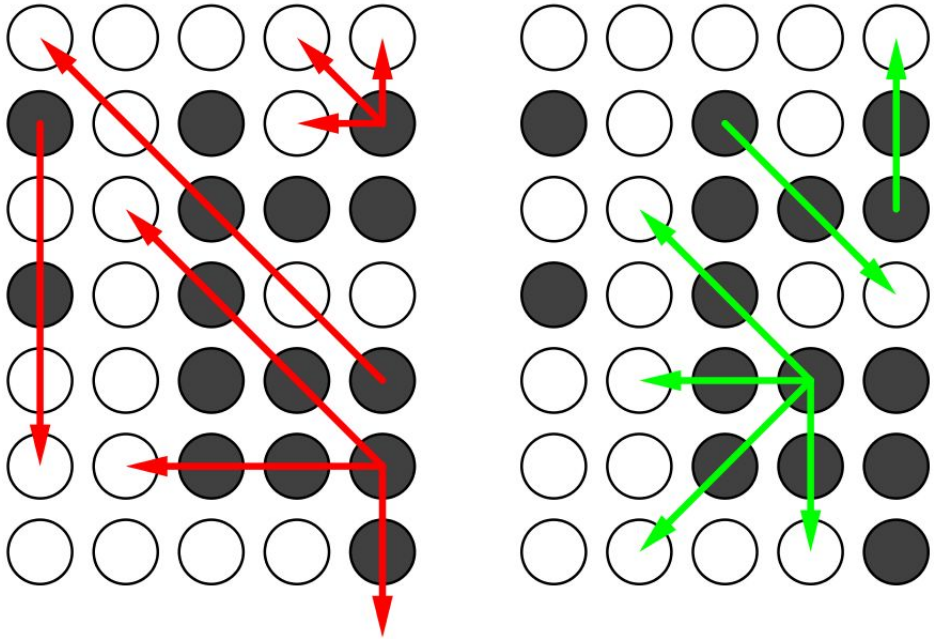




The Problem

Given **R x C** marbles on **(R + 2) x (C + 2)** checkerboard

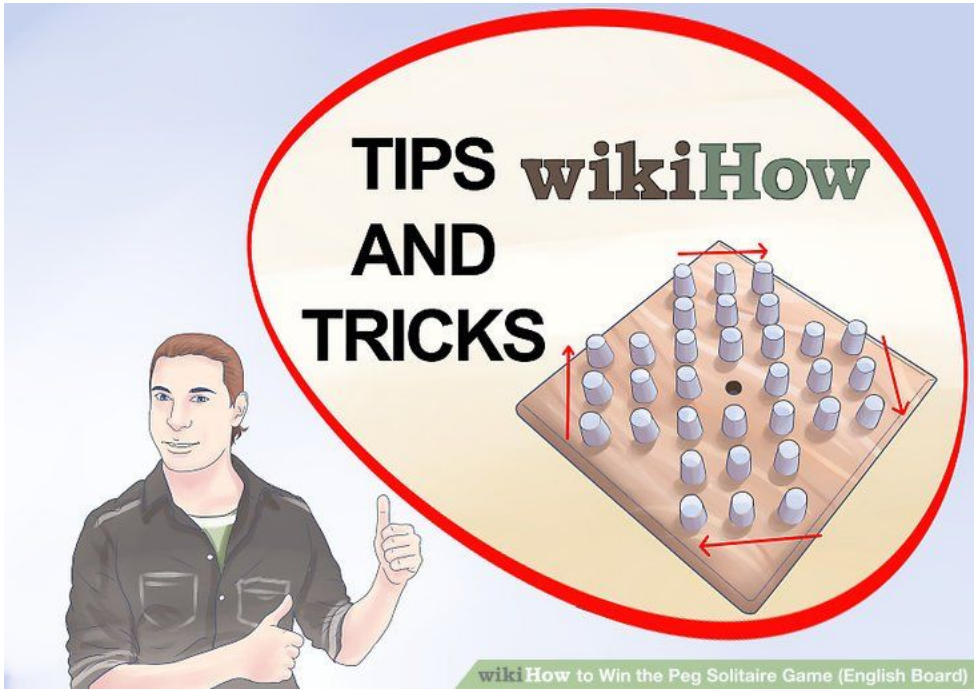
Remove all marbles except **1**



Background

Peg Solitaire

- Only orthogonal jumps
- Only one empty hole



Background

What Cherry “created” is actually an easier version of peg solitaire

- Allow diagonal jumps: **8-move peg solitaire**
- Many more empty holes: entire rows and columns

Scoring

Output **N** moves → **M = (R x C - N)** marbles remain

Score = $40 \times \frac{1}{\sqrt{M}} + 10^{1-\frac{M-1}{\min(R,C)}} + 50^{1-\frac{M-1}{R \times C}}$

Goal: **M = 1** ← **N = R x C - 1**

Sample 1

Input

2 2

Output

3

Sample 1

Input

2 2

Output

3

1 1 1 2

Sample 1

Input

2 2

Output

3

1 1 1 2

2 1 2 2

Sample 1

Input

2 2

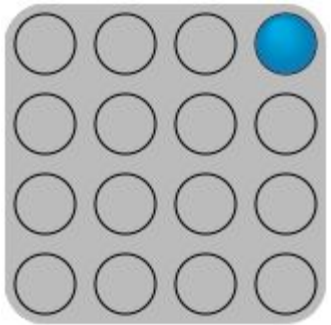
Output

3

1 1 1 2

2 1 2 2

2 3 1 3



Sample 1

Input

2 2

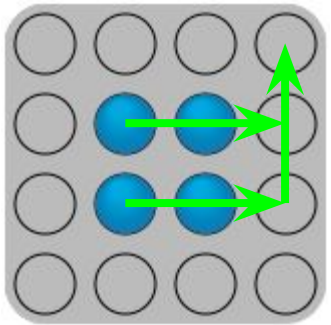
Output

3

1 1 1 2

2 1 2 2

2 3 1 3



Sample 2

Input

2 2

Output

3

Sample 2

Input

2 2

Output

3

1 1 1 2

Sample 2

Input

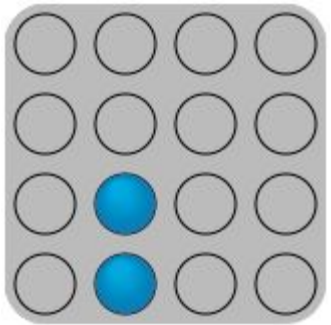
2 2

Output

3

1 1 1 2

1 3 2 2



Sample 2

Input

2 2

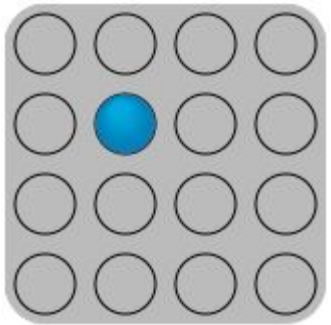
Output

3

1 1 1 2

1 3 2 2

3 1 2 1



Sample 2

Input

2 2

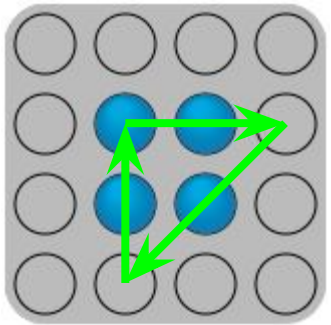
Output

3

1 1 1 2

1 3 2 2

3 1 2 1



Sample 2

Input

2 2

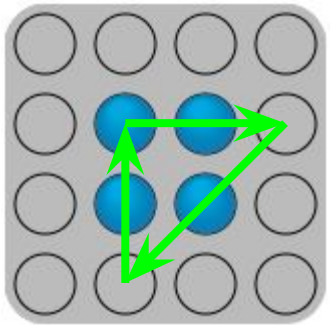
Output

3

1 1 1 2

1 3 2 2

3 1 2 1



Observation: Marble goes back to original position

Subtasks

For all cases: $2 \leq R, C \leq 100$

	Points	Constraints
1	2	$R = 2, C = 2$
2	3	$R = 2, C = 3$
3	4	$R = 3, C = 3$
4	5	$R = 4, C = 3$
5	6	$R = 4, C = 4$
6	14	$2 \leq R, C \leq 5$
7	15	$R = 2$
8	19	$R = 3$
9	32	No additional constraints

Statistics

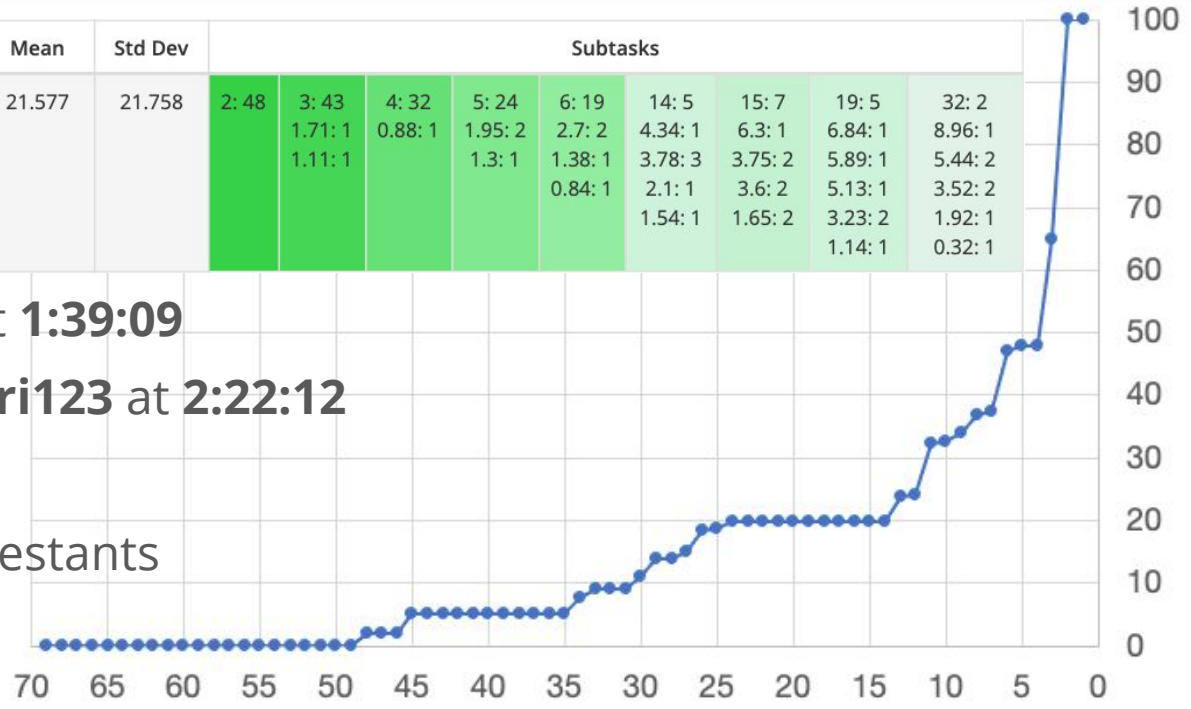
Task	Attempts	Max	Mean	Std Dev	Subtasks								
S213 - Chinese Checkers	48	100	21.577	21.758	2: 48	3: 43 1.71: 1 1.11: 1	4: 32 0.88: 1	5: 24 1.95: 2 1.3: 1	6: 19 2.7: 2 1.38: 1 0.84: 1	14: 5 4.34: 1 3.78: 3 2.1: 1 1.54: 1	15: 7 6.3: 1 3.75: 2 3.6: 2 1.65: 2	19: 5 6.84: 1 5.89: 1 5.13: 1 3.23: 2 1.14: 1	32: 2 8.96: 1 5.44: 2 3.52: 2 1.92: 1 0.32: 1

First solved by **dbsgame** at **1:39:09**

Last solved by **dbstoshinari123** at **2:22:12**

Total solves: **2**

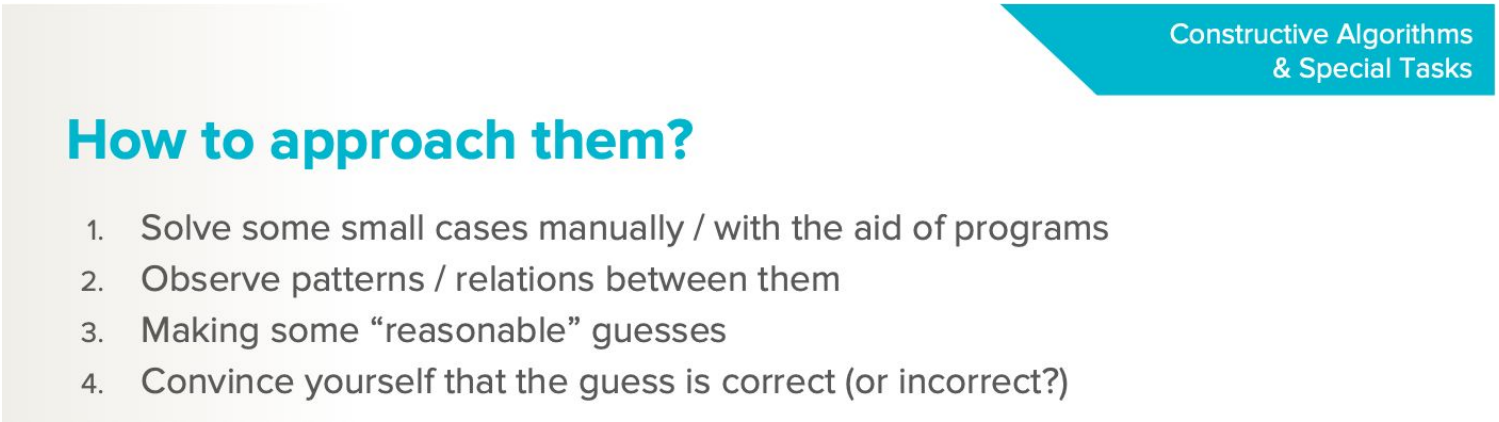
Attempted by: **48 / 69** contestants



Overview

To solve this kind of **ad-hoc** and/or **constructive** problems:

- Usually more interesting (?) and less “standard”
- Usually requires a lot of rough work and/or insight and/or intuition



https://assets.hkoi.org/training2019/cast.pdf

Solutions

Subtask 1 (2 points)

R = 2, C = 2

→ Sanity Check

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  int R, C;
  cin >> R >> C;
  if(R == 2 && C == 2){
    cout << "3" << endl;
    cout << "1 1 1 2" << endl;
    cout << "2 1 2 2" << endl;
    cout << "2 3 1 3" << endl;
  }
  return 0;
}
```

Subtask 1 (2 points)

R = 2, C = 2

→ Sanity Check

	S211 Skyscraperhenge	S212 Super Chat	S213 Chinese Checkers	Total
	?	?	?	?
	?	?		?

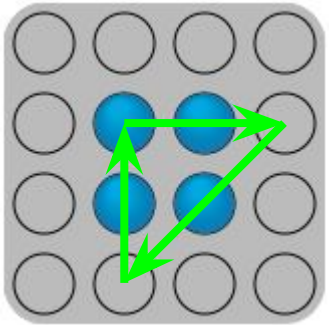
Subtask 1 (2 points)

R = 2, C = 2

→ Sanity Check

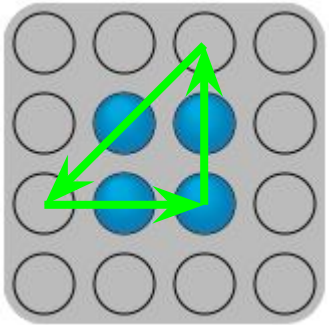
	S211 Skyscraperhenge	S212 Super Chat	S213 Chinese Checkers	Total
	?	?	?	?
	?	?		?

Strategy 0.0 : 2 points



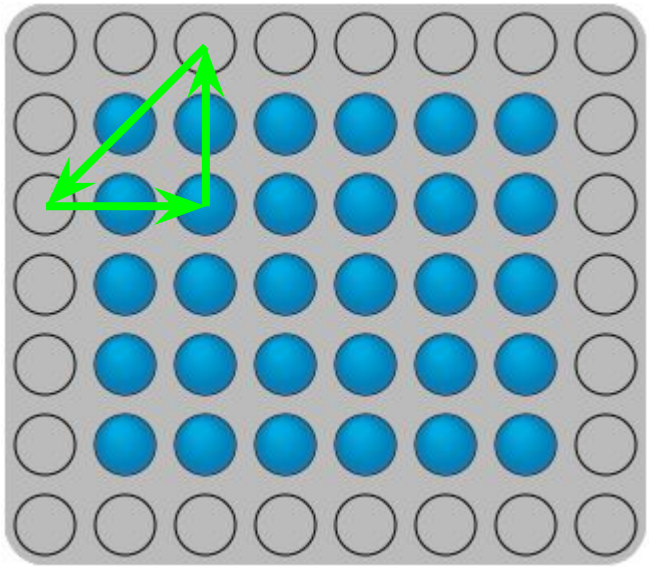
```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  cout << "3" << endl;
  cout << "1 1 1 2" << endl;
  cout << "1 3 2 2" << endl;
  cout << "3 1 2 1" << endl;
  return 0;
}
```

Strategy 0.1



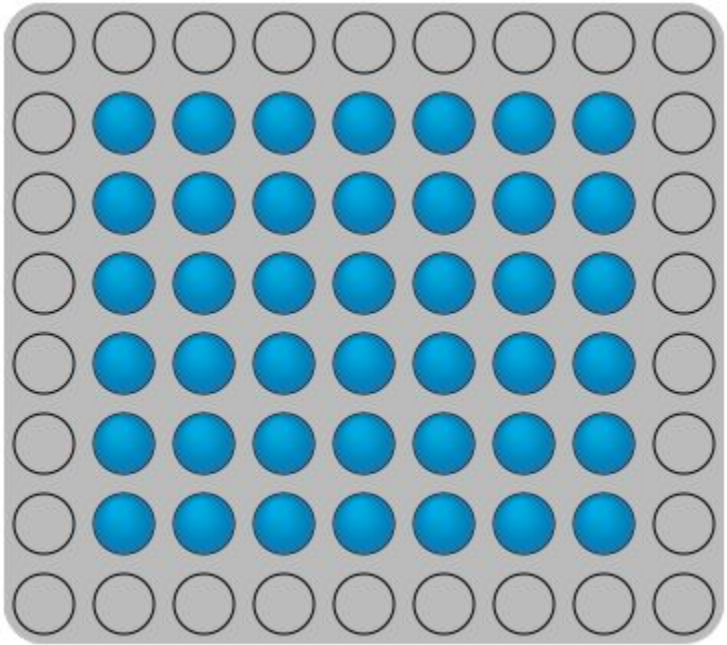
```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  cout << "3" << endl;
  cout << "2 2 1 2" << endl;
  cout << "0 2 1 1" << endl;
  cout << "2 0 2 1" << endl;
  return 0;
}
```

Strategy 0.1 : 8.36 points

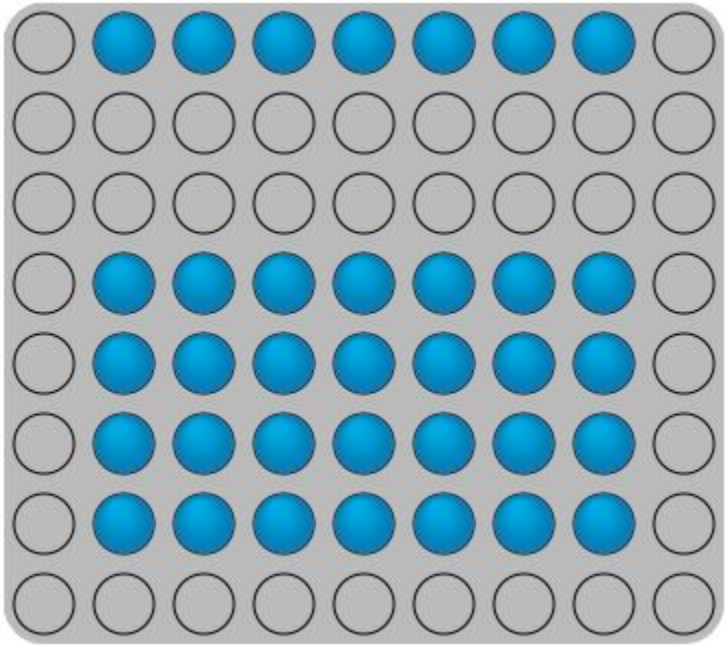


```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  cout << "3" << endl;
  cout << "2 2 1 2" << endl;
  cout << "0 2 1 1" << endl;
  cout << "2 0 2 1" << endl;
  return 0;
}
```

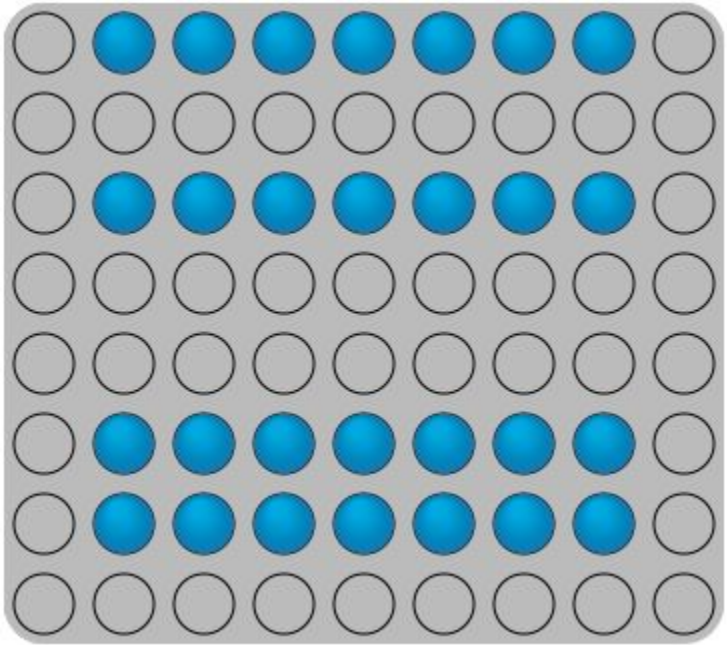
Strategy 1.1



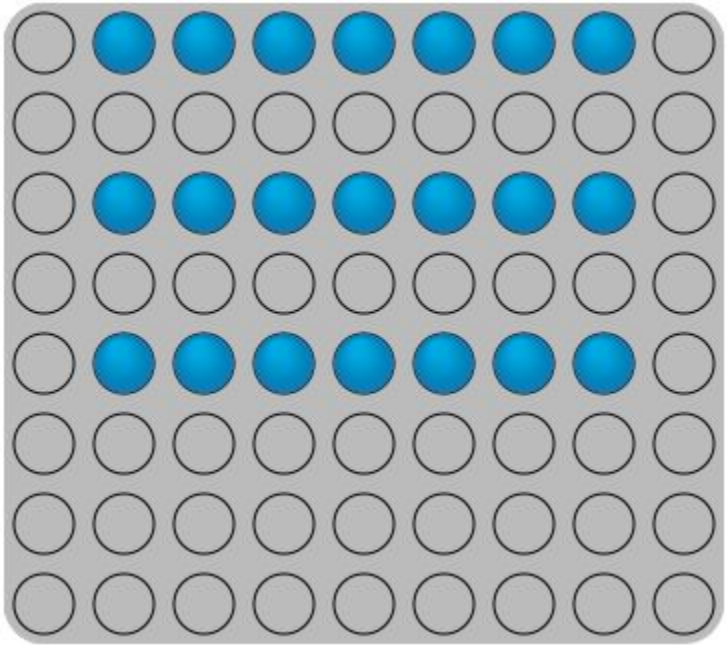
Strategy 1.1 : 9.33 points



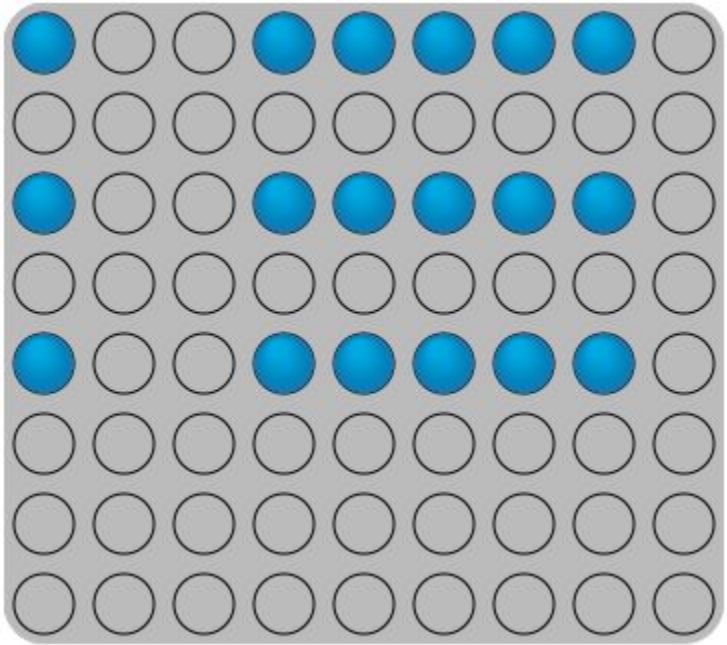
Strategy 1.2



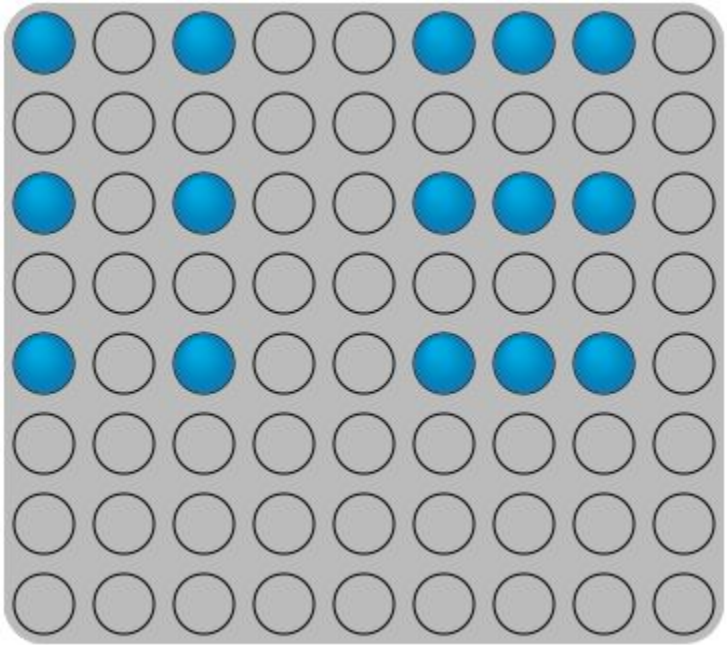
Strategy 1.2 : 12.48 points



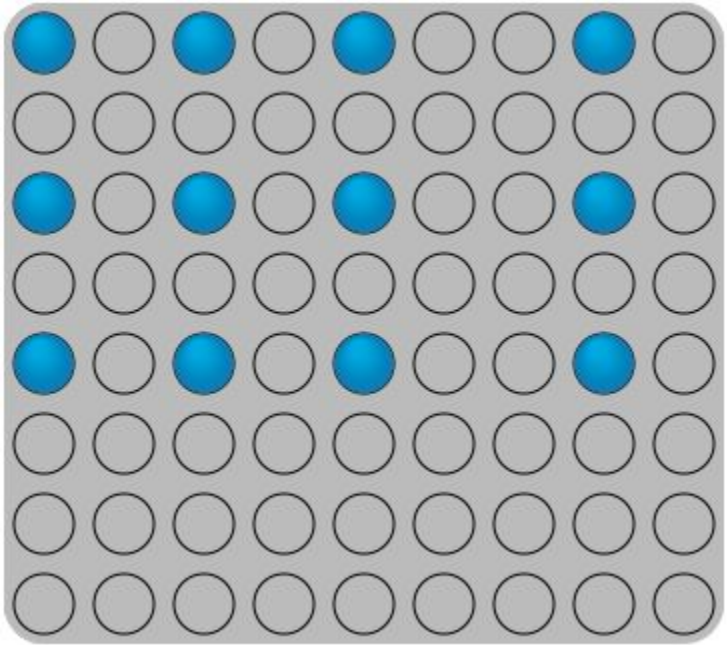
Strategy 1.3



Strategy 1.3



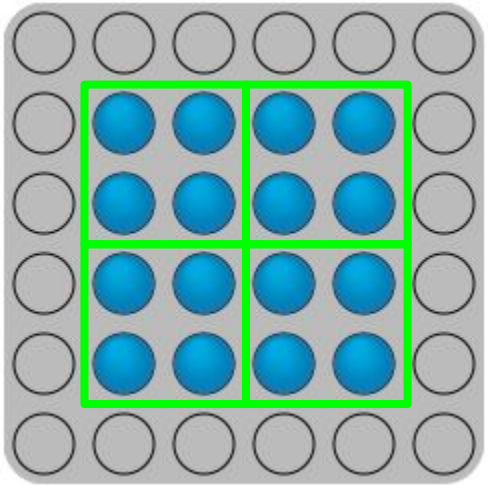
Strategy 1.3 : 25.77 points



Back to subtasks...

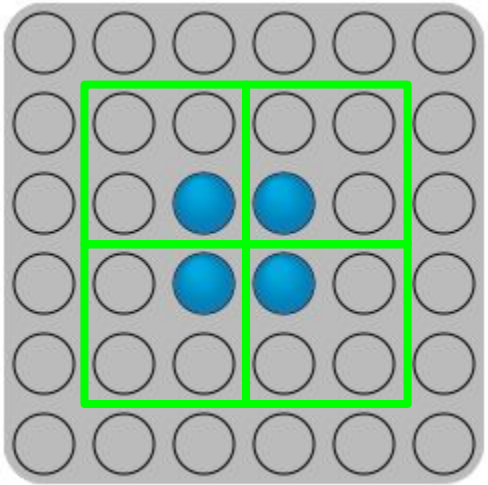
Subtask 5 (6 points)

R = 4, C = 4



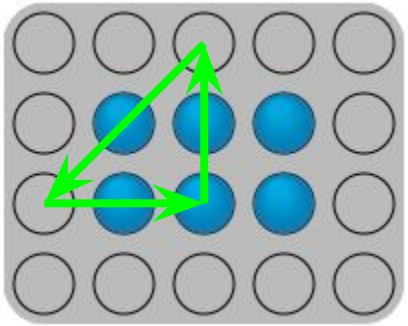
Subtask 5 (6 points)

R = 4, C = 4



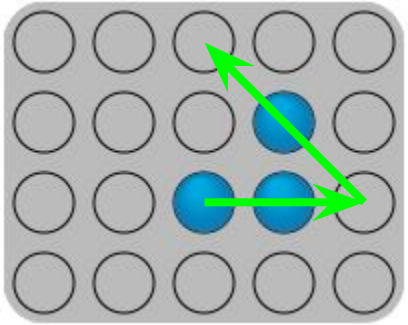
Subtask 2 (3 points)

R = 2, C = 3



Subtask 2 (3 points)

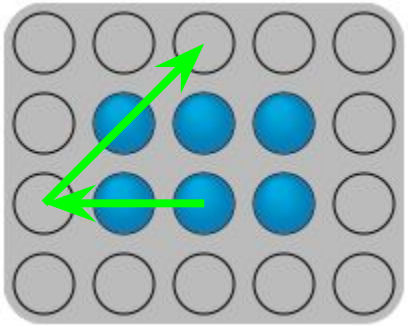
R = 2, C = 3



Subtask 2 (3 points)

R = 2, C = 3

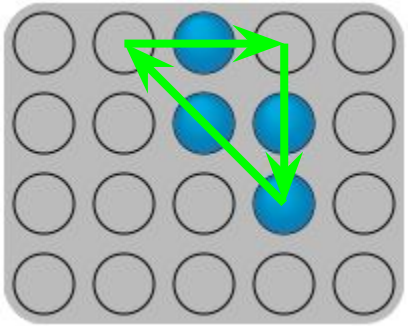
Another way:



Subtask 2 (3 points)

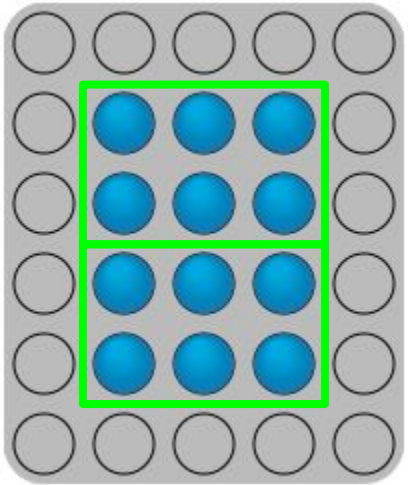
R = 2, C = 3

Another way:



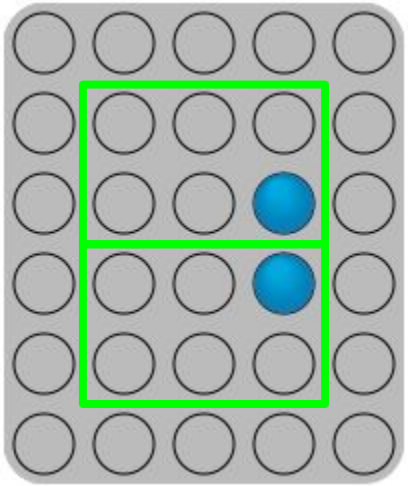
Subtask 4 (5 points)

R = 4, C = 3



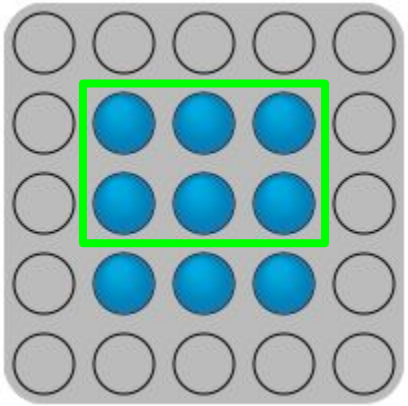
Subtask 4 (5 points)

R = 4, C = 3



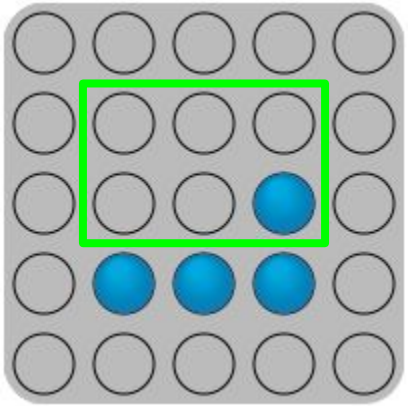
Subtask 3 (4 points)

R = 3, C = 3



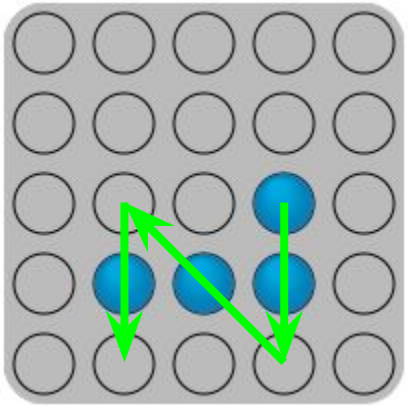
Subtask 3 (4 points)

R = 3, C = 3



Subtask 3 (4 points)

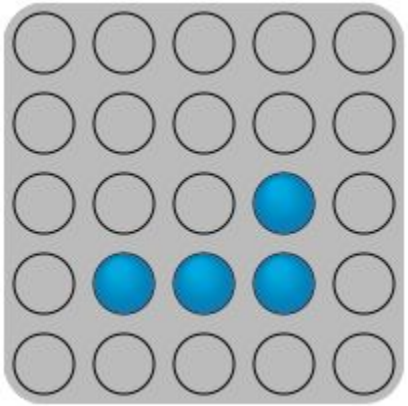
R = 3, C = 3



Subtask 3 (4 points)

R = 3, C = 3

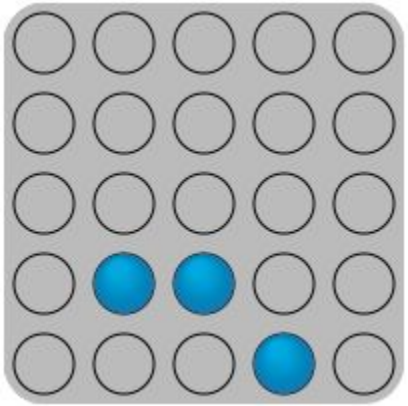
Another way:



Subtask 3 (4 points)

R = 3, C = 3

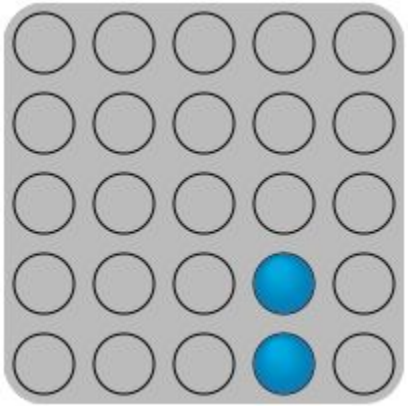
Another way:



Subtask 3 (4 points)

R = 3, C = 3

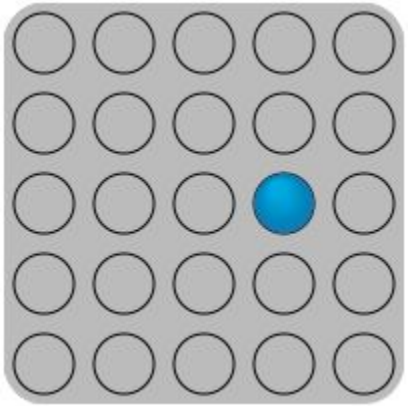
Another way:



Subtask 3 (4 points)

R = 3, C = 3

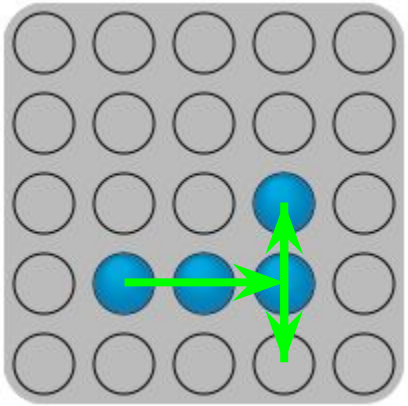
Another way:



Subtask 3 (4 points)

R = 3, C = 3

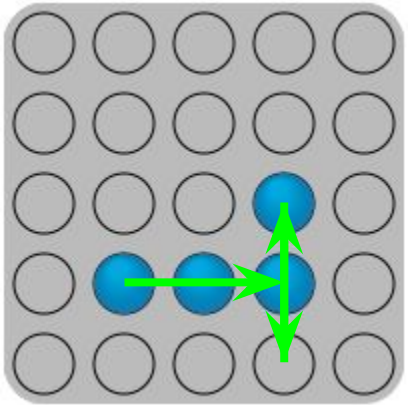
Another way:



Subtask 3 (4 points)

R = 3, C = 3

Another way:



Observation: Marble goes back to original position

Subtask 6 (14 points)

2 <= R, C <= 5

Method 1: Brute Force DFS

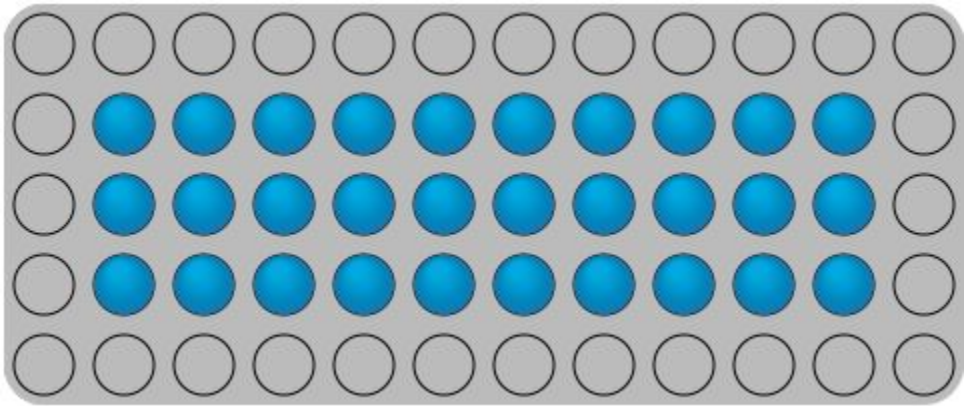
Method 2: Hardcode

	S211 Skyscraperhenge	S212 Super Chat	S213 Chinese Checkers	Total
?			34	34

Subtask 8 (19 points)

R = 3

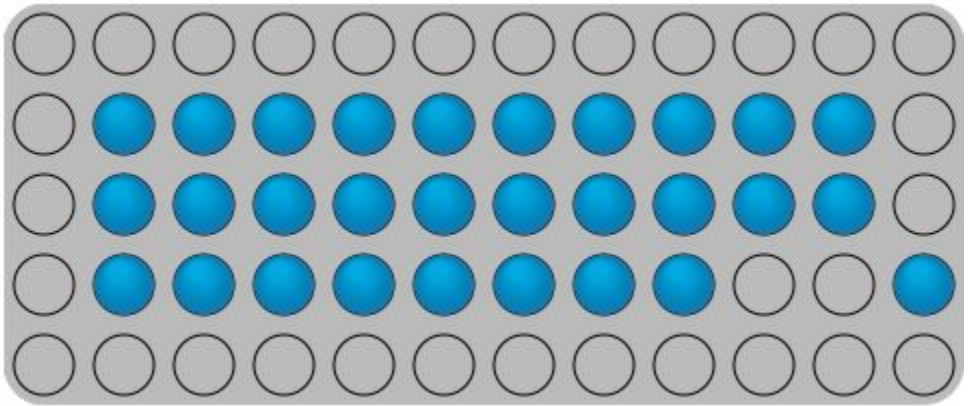
Reduce 1 column:



Subtask 8 (19 points)

R = 3

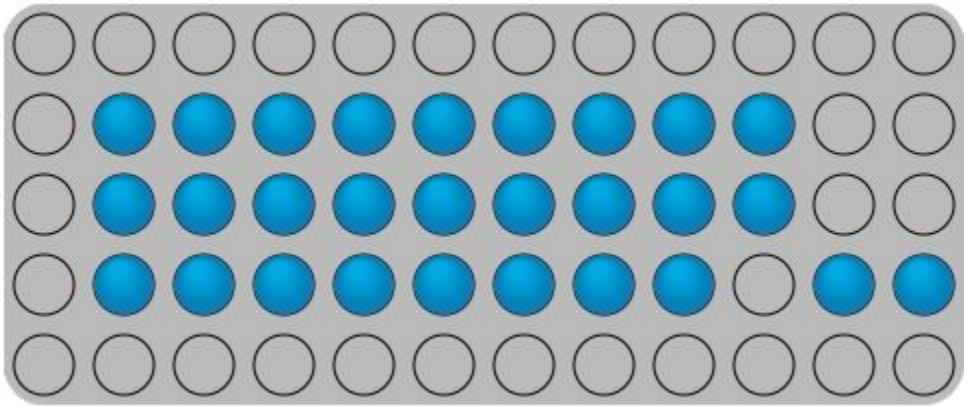
Reduce 1 column:



Subtask 8 (19 points)

R = 3

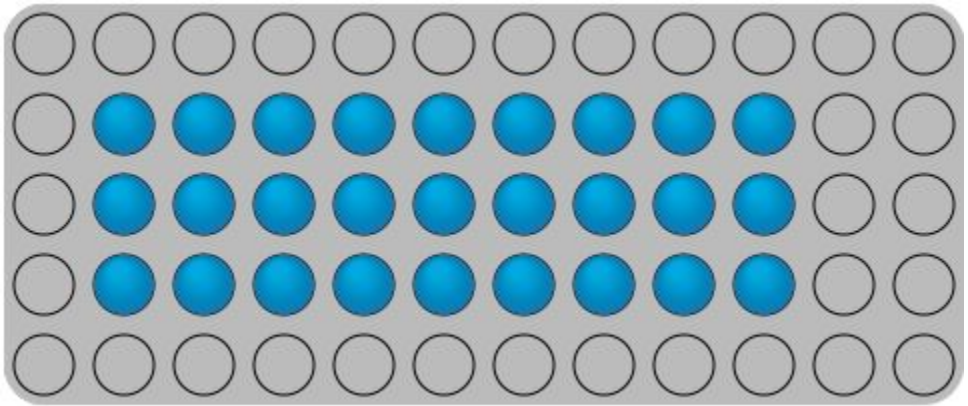
Reduce 1 column:



Subtask 8 (19 points)

R = 3

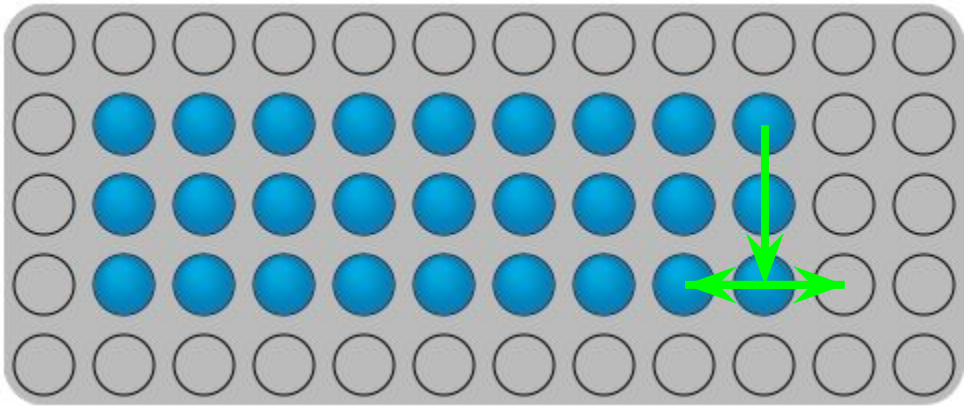
Reduce 1 column:



Subtask 8 (19 points)

R = 3

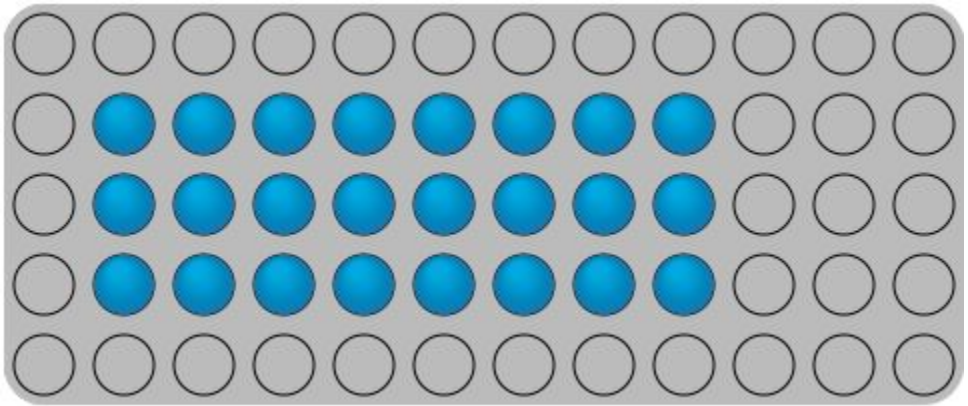
Reduce 1 column:



Subtask 8 (19 points)

R = 3

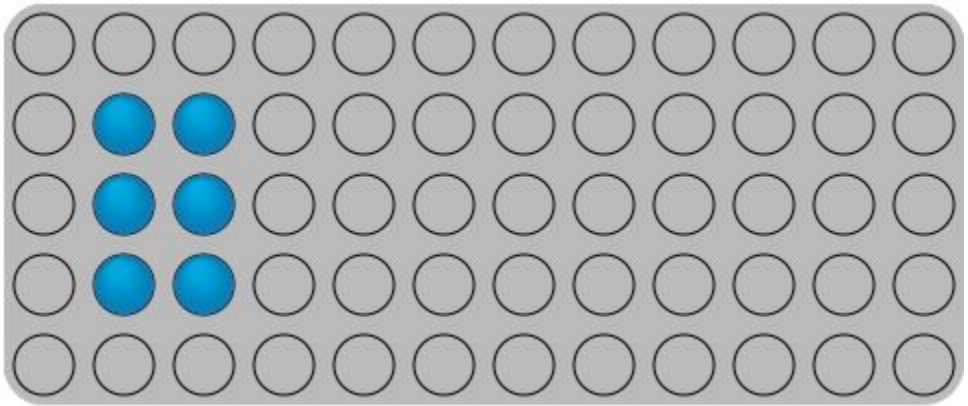
Reduce 1 column:



Subtask 8 (19 points)

R = 3

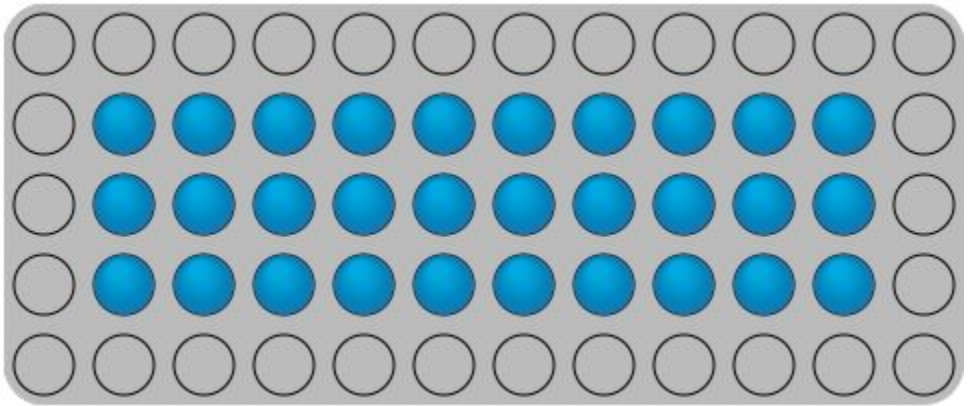
Reduce 1 column:



Subtask 8 (19 points)

R = 3

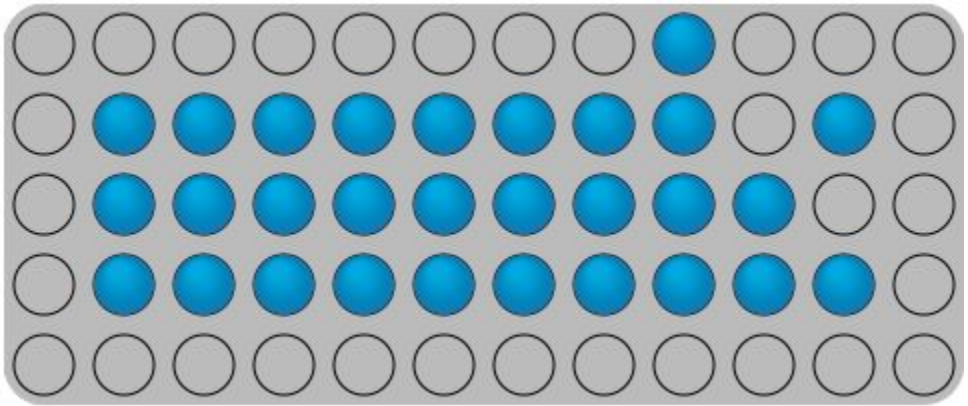
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

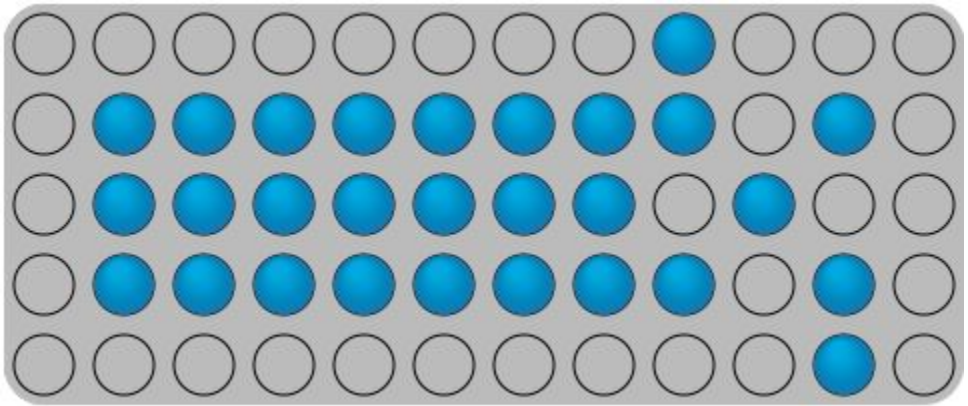
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

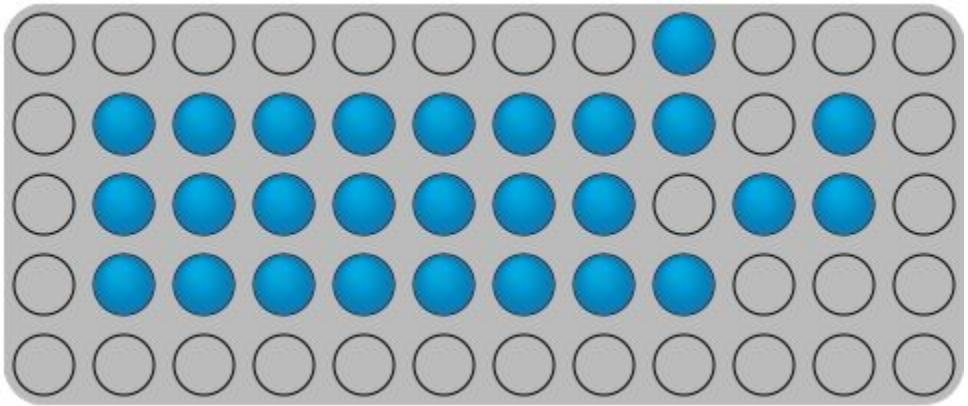
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

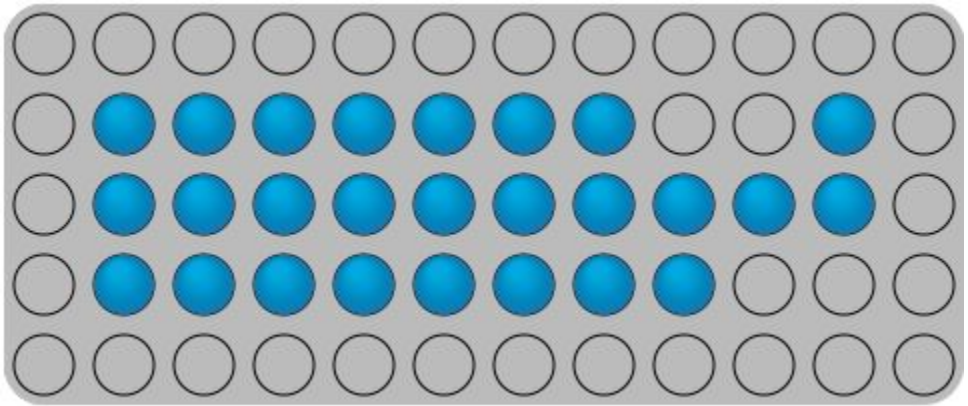
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

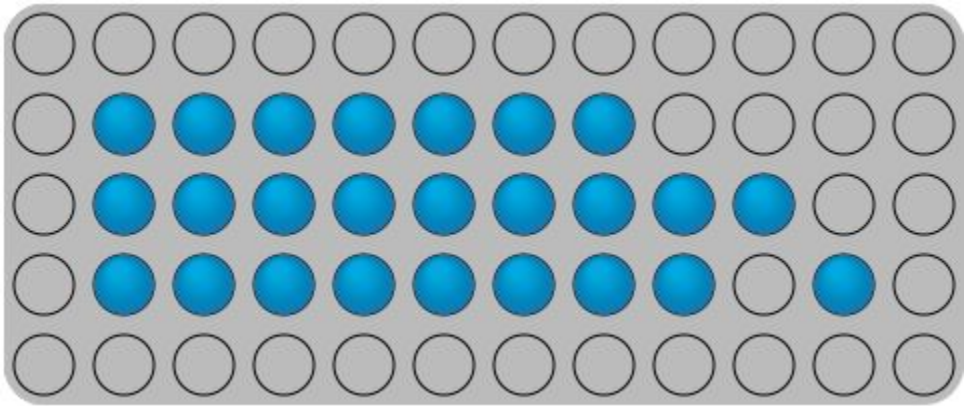
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

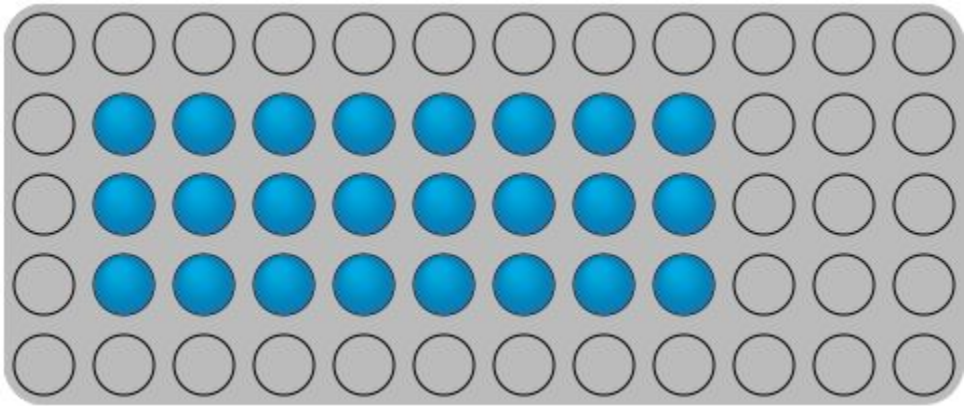
Reduce 2 columns: (by **dbsgame**)



Subtask 8 (19 points)

R = 3

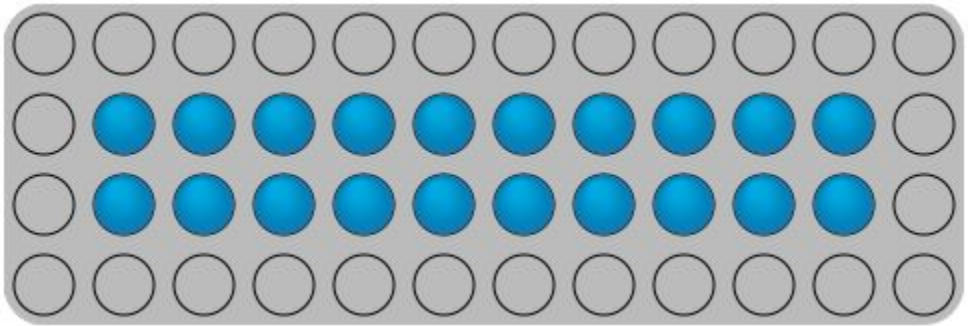
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

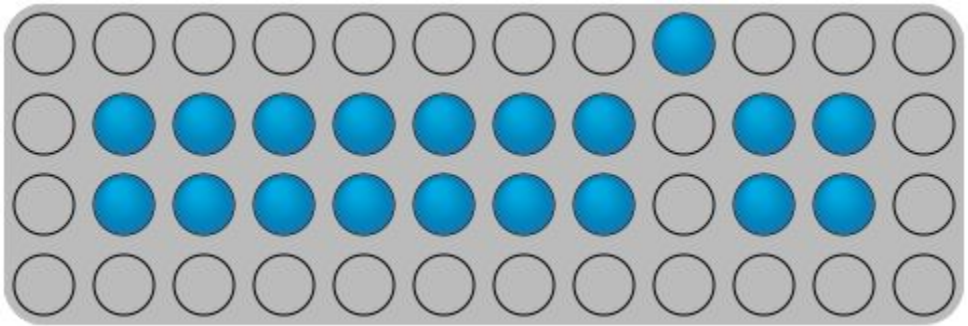
Reduce 3 columns:



Subtask 7 (15 points)

R = 2

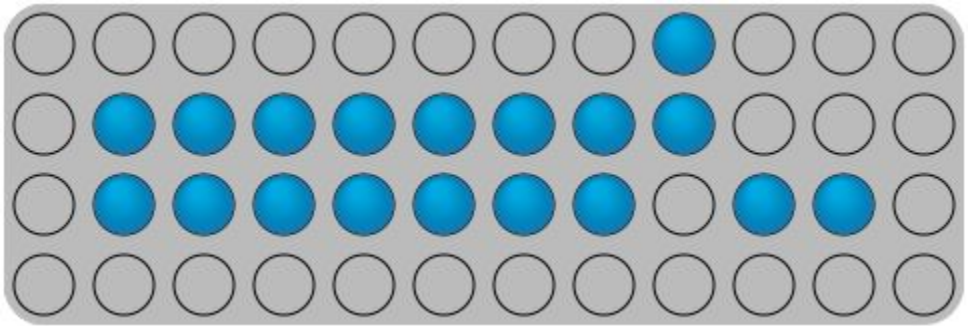
Reduce 3 columns:



Subtask 7 (15 points)

R = 2

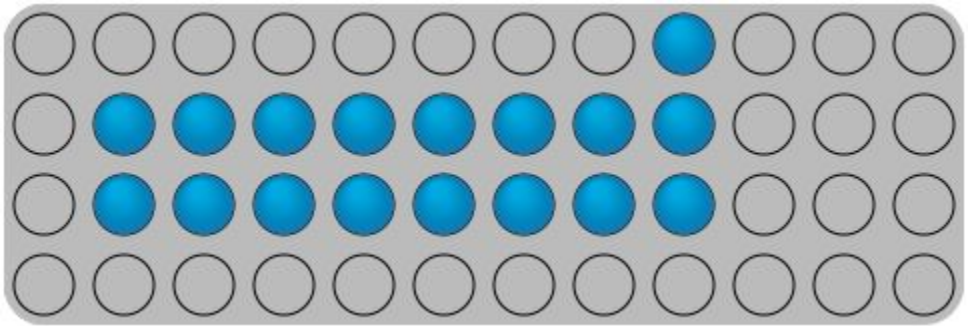
Reduce 3 columns:



Subtask 7 (15 points)

R = 2

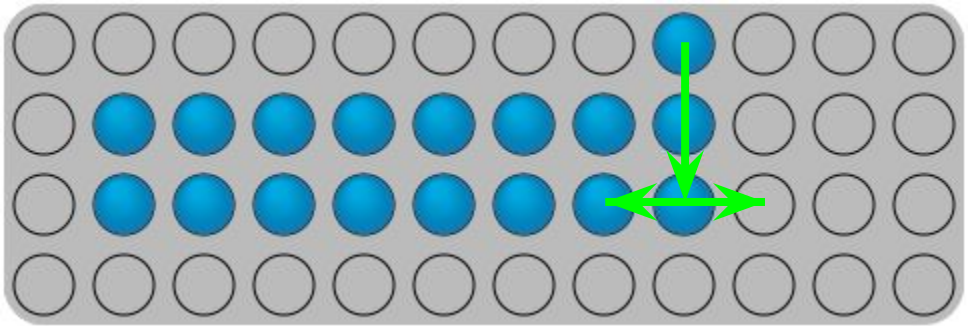
Reduce 3 columns:



Subtask 7 (15 points)

R = 2

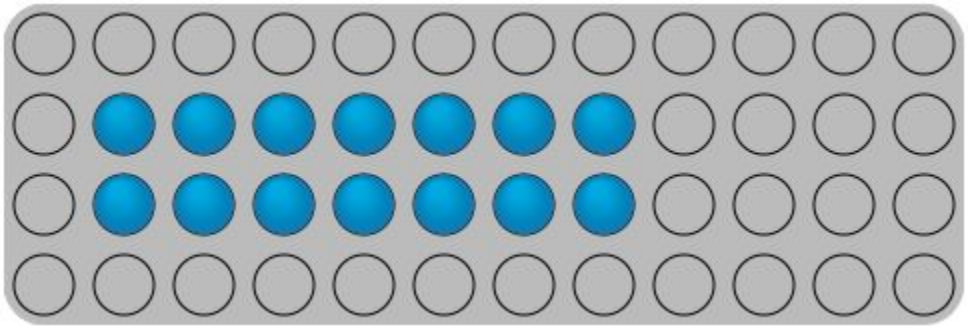
Reduce 3 columns:



Subtask 7 (15 points)

R = 2

Reduce 3 columns:

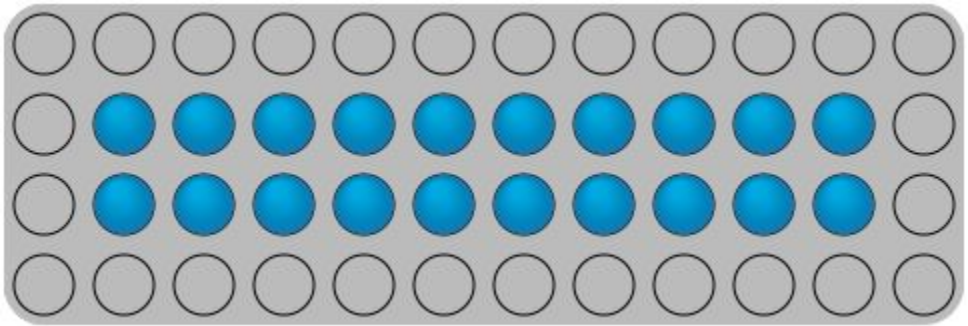


Subtask 7 (15 points)

R = 2

Reduce 2 columns: (by **dbsgame**)

Initialization:

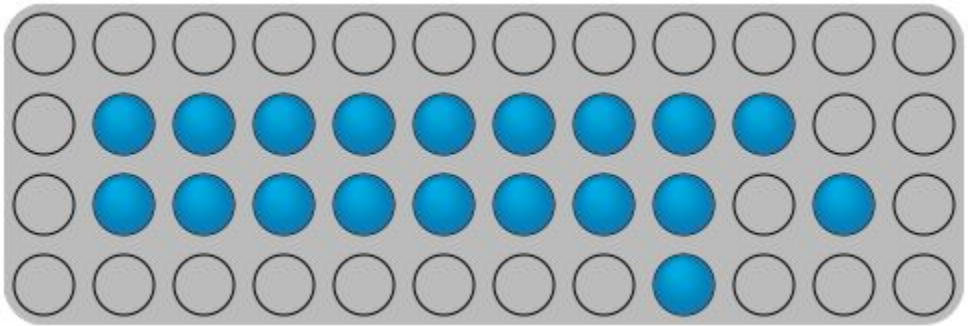


Subtask 7 (15 points)

R = 2

Reduce 2 columns: (by **dbsgame**)

Initialization:

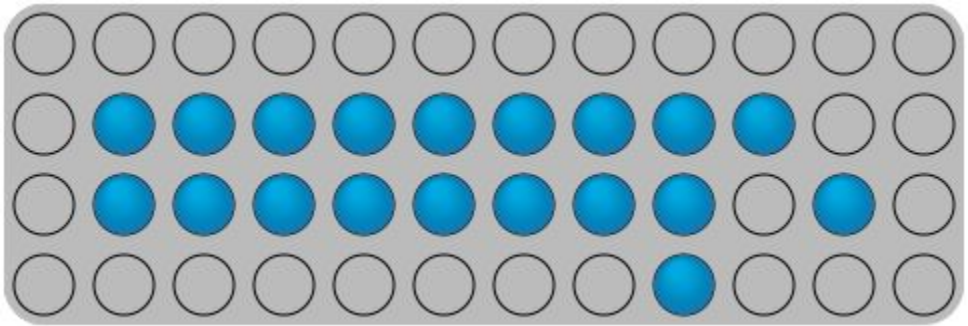


Subtask 7 (15 points)

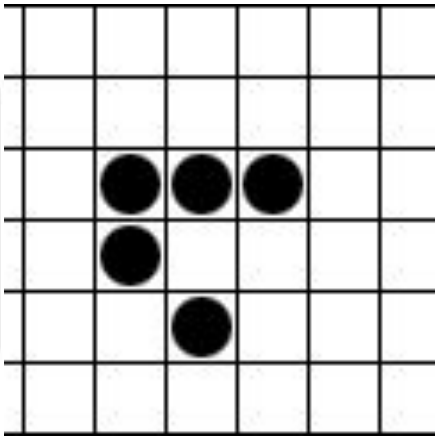
R = 2

Reduce 2 columns: (by **dbsgame**)

Initialization:



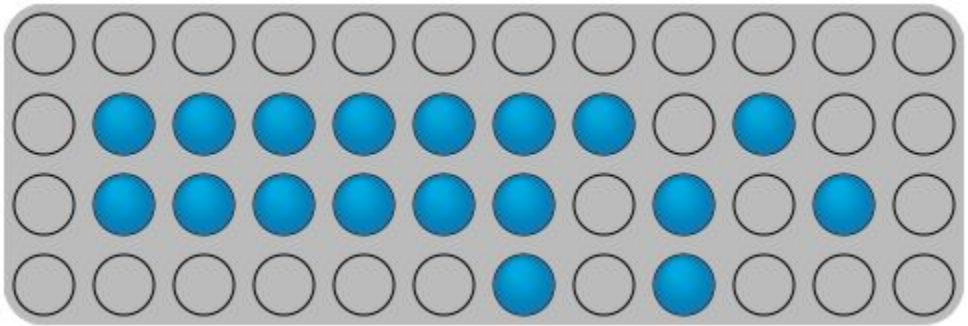
Glider (Conway's Game of Life)



Subtask 7 (15 points)

R = 2

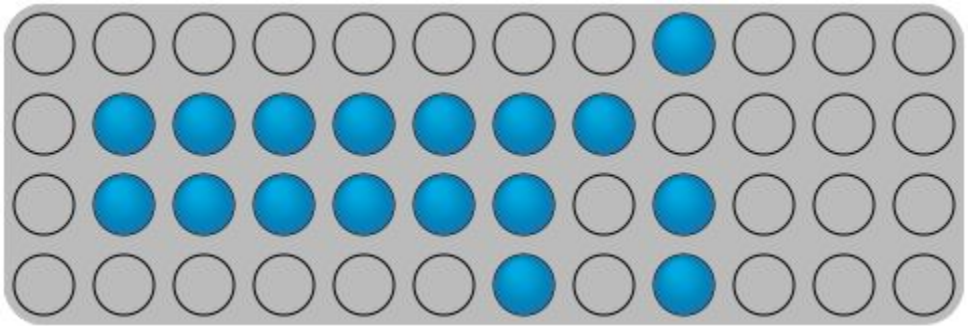
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

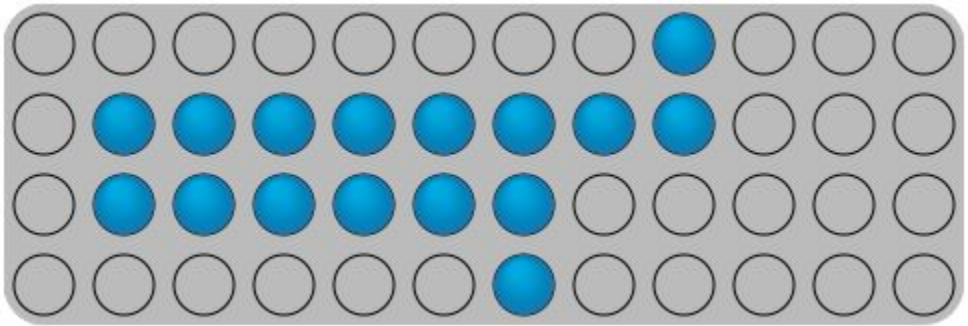
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

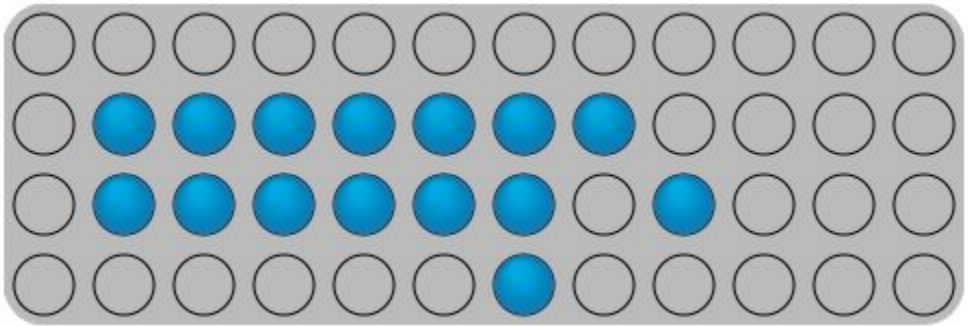
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

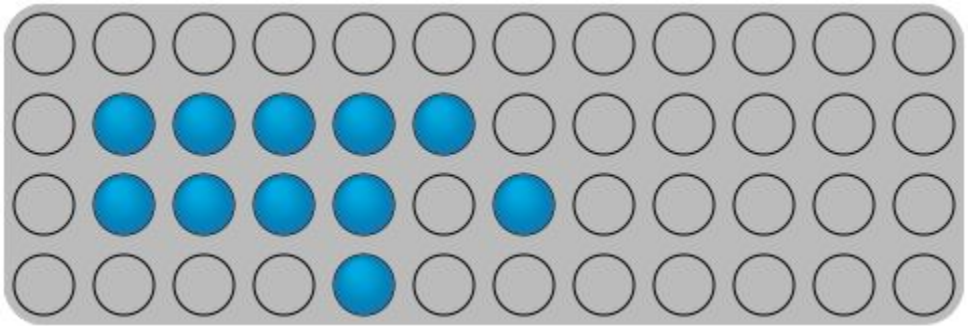
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

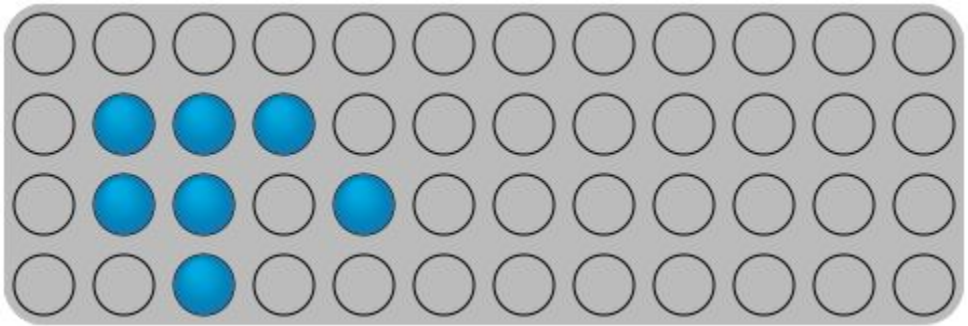
Reduce 2 columns: (by **dbsgame**)



Subtask 7 (15 points)

R = 2

Reduce 2 columns: (by **dbsgame**)

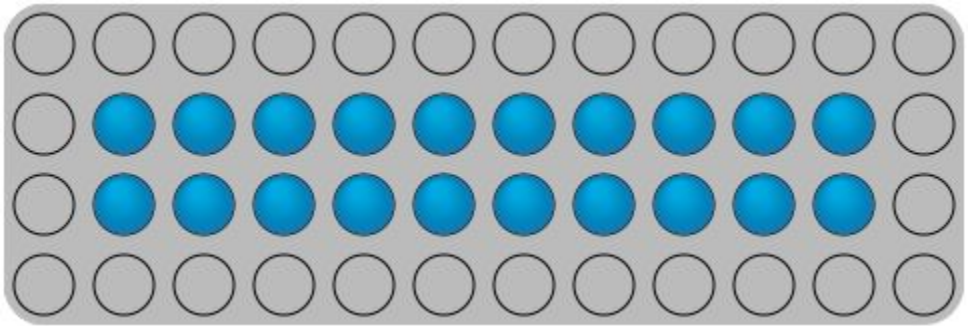


Subtask 7 (15 points)

R = 2

Reduce 1 column: (by **dbstoshinari123**)

Initialization:

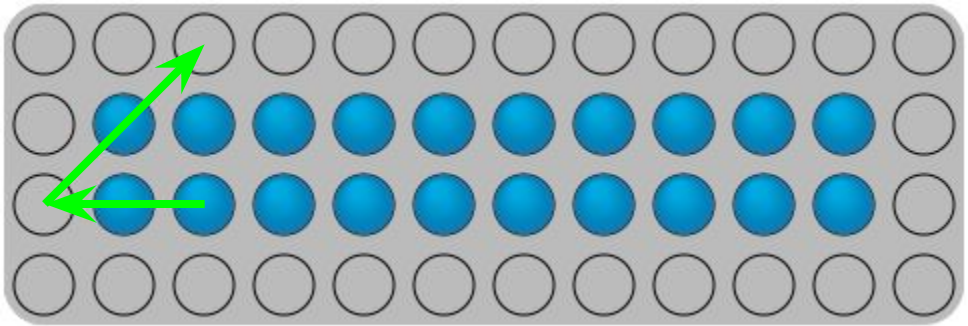


Subtask 7 (15 points)

R = 2

Reduce 1 column: (by **dbstoshinari123**)

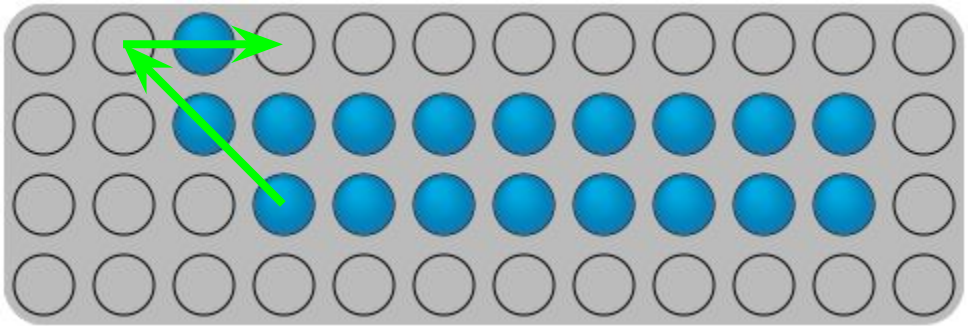
Initialization:



Subtask 7 (15 points)

R = 2

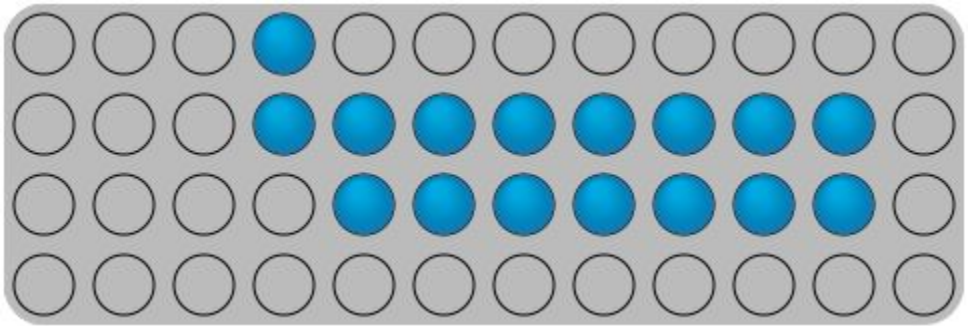
Reduce 1 column: (by **dbstoshinari123**)



Subtask 7 (15 points)

R = 2

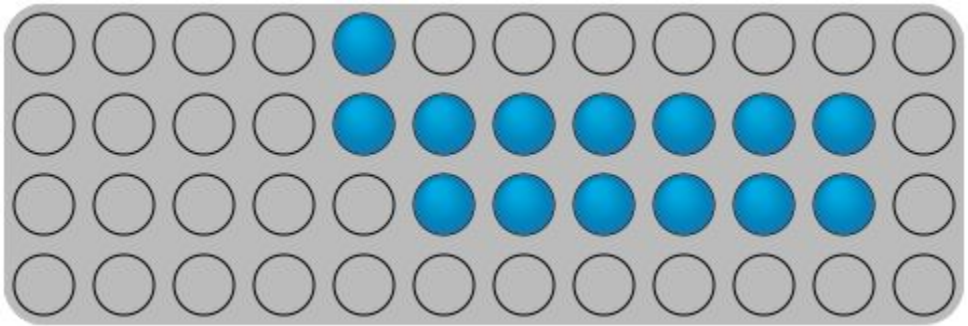
Reduce 1 column: (by **dbstoshinari123**)



Subtask 7 (15 points)

R = 2

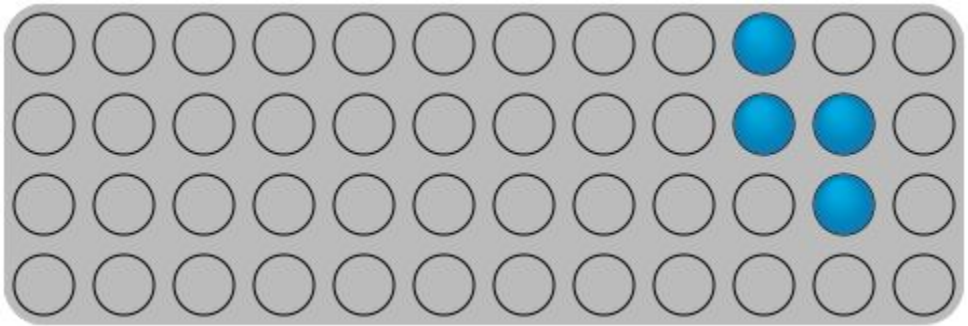
Reduce 1 column: (by **dbstoshinari123**)



Subtask 7 (15 points)

R = 2

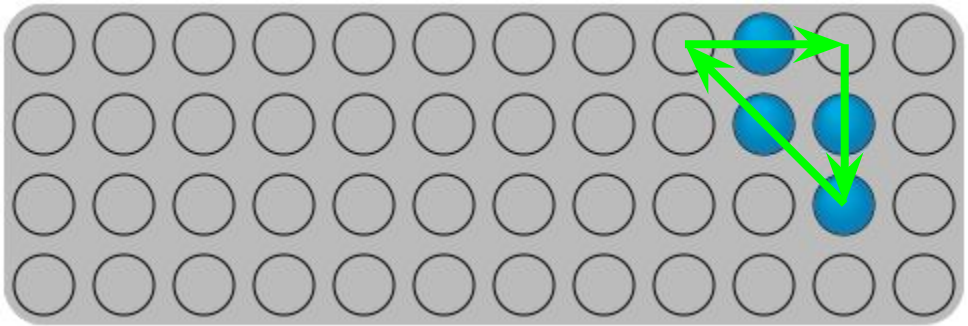
Reduce 1 column: (by **dbstoshinari123**)



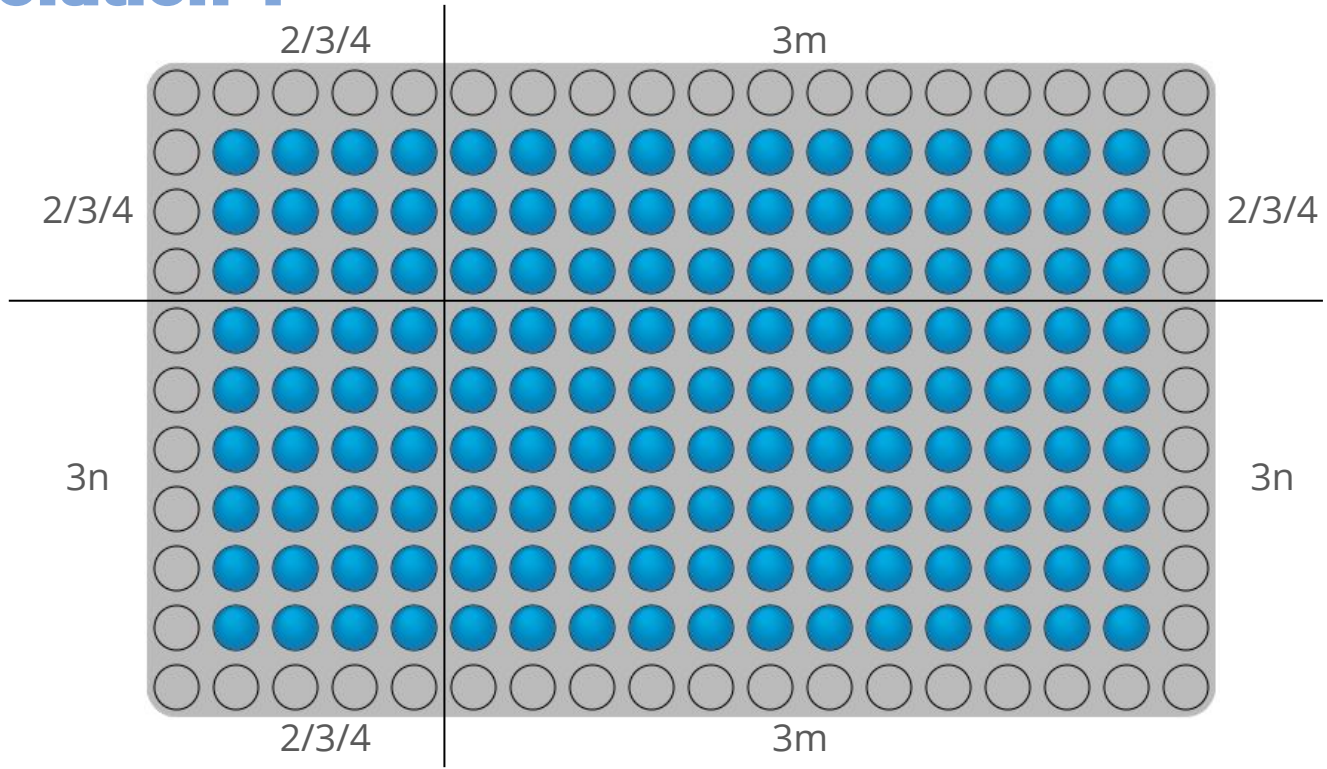
Subtask 7 (15 points)

R = 2

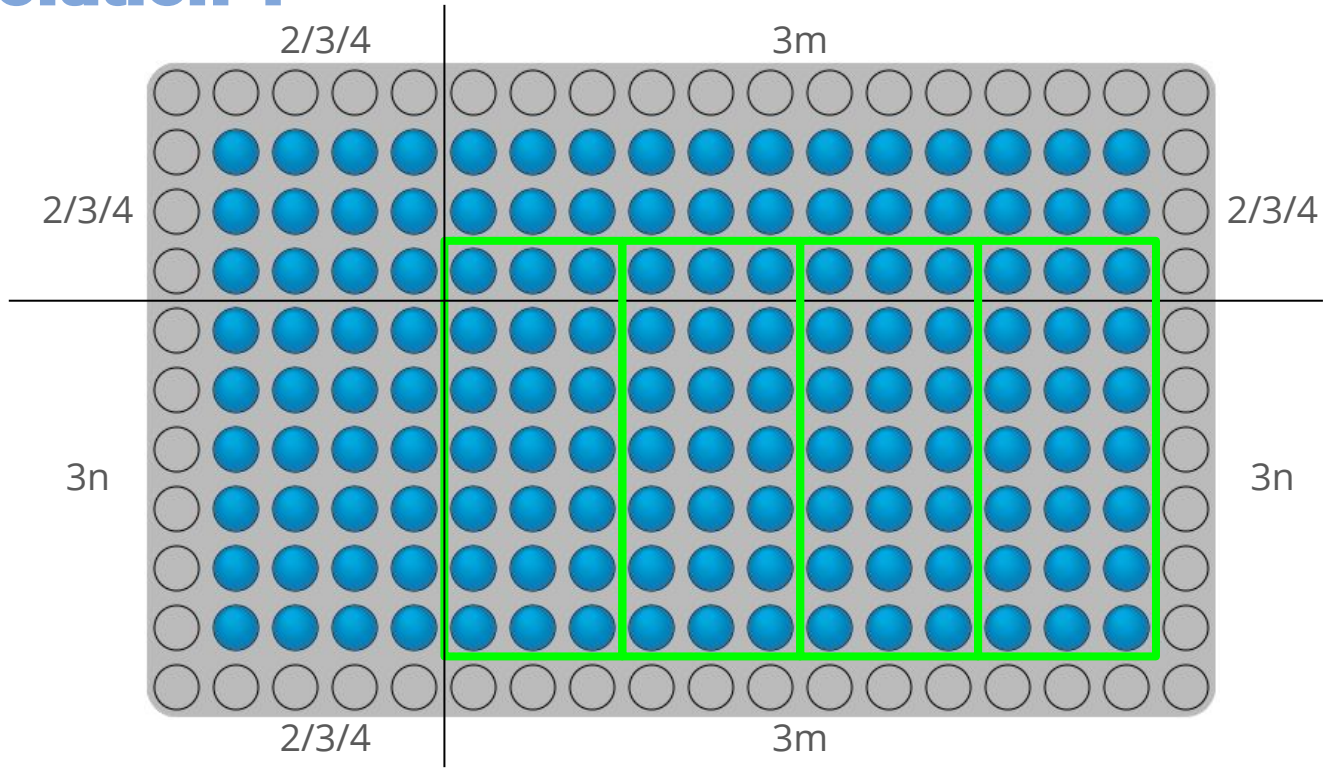
Reduce 1 column: (by **dbstoshinari123**)



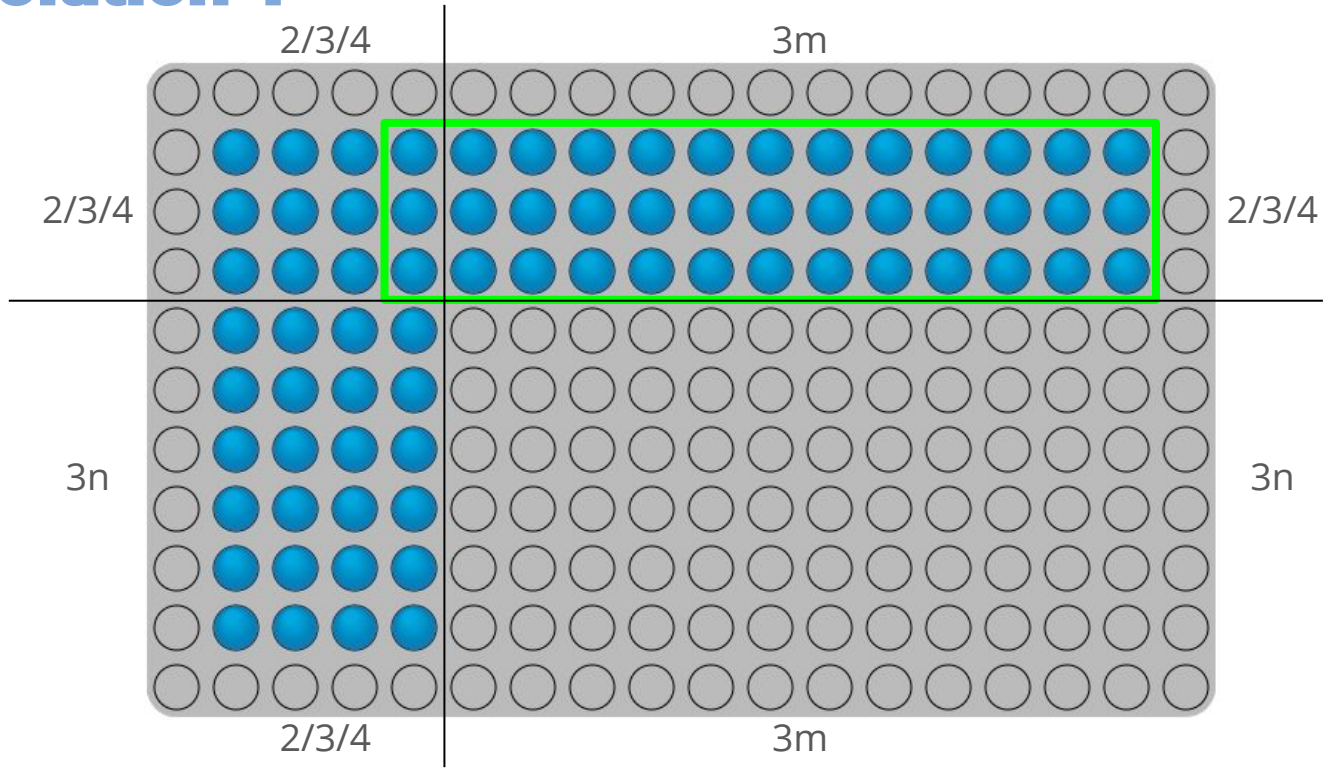
Full Solution 1



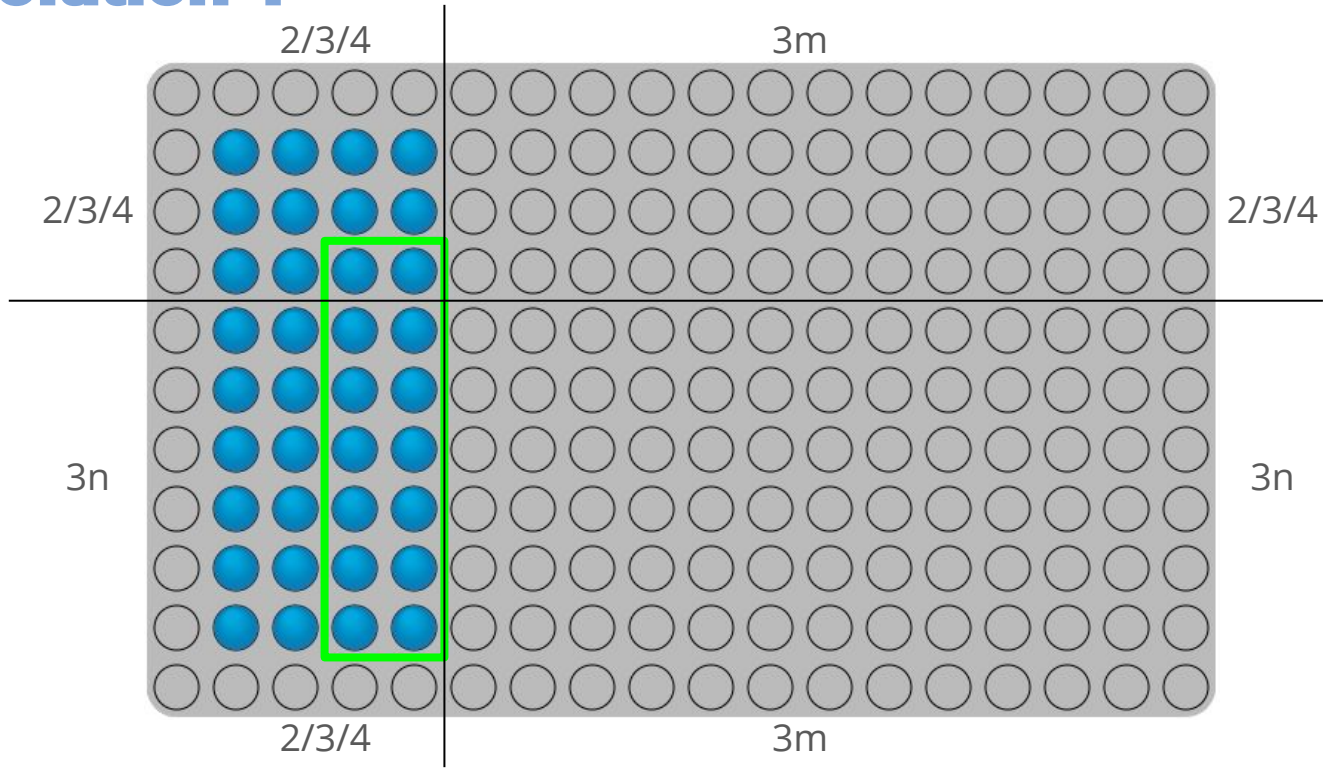
Full Solution 1



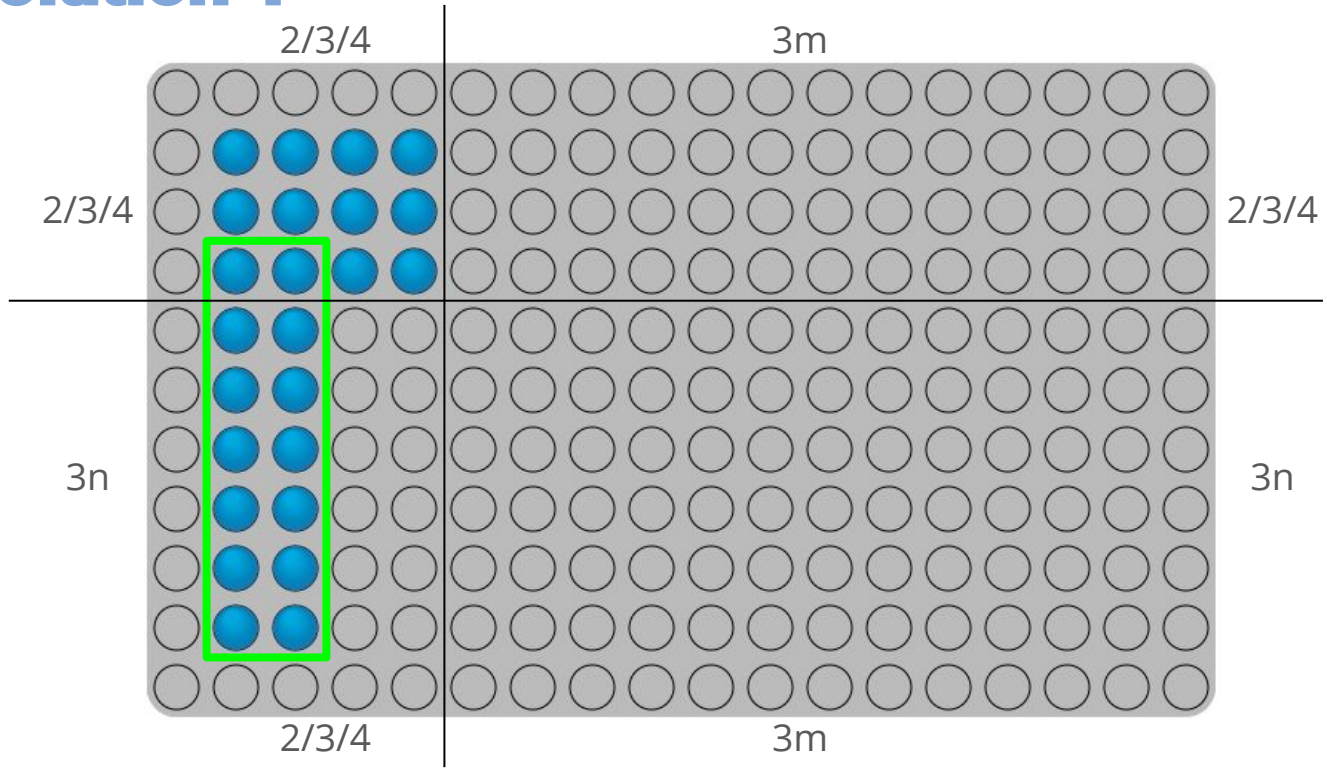
Full Solution 1



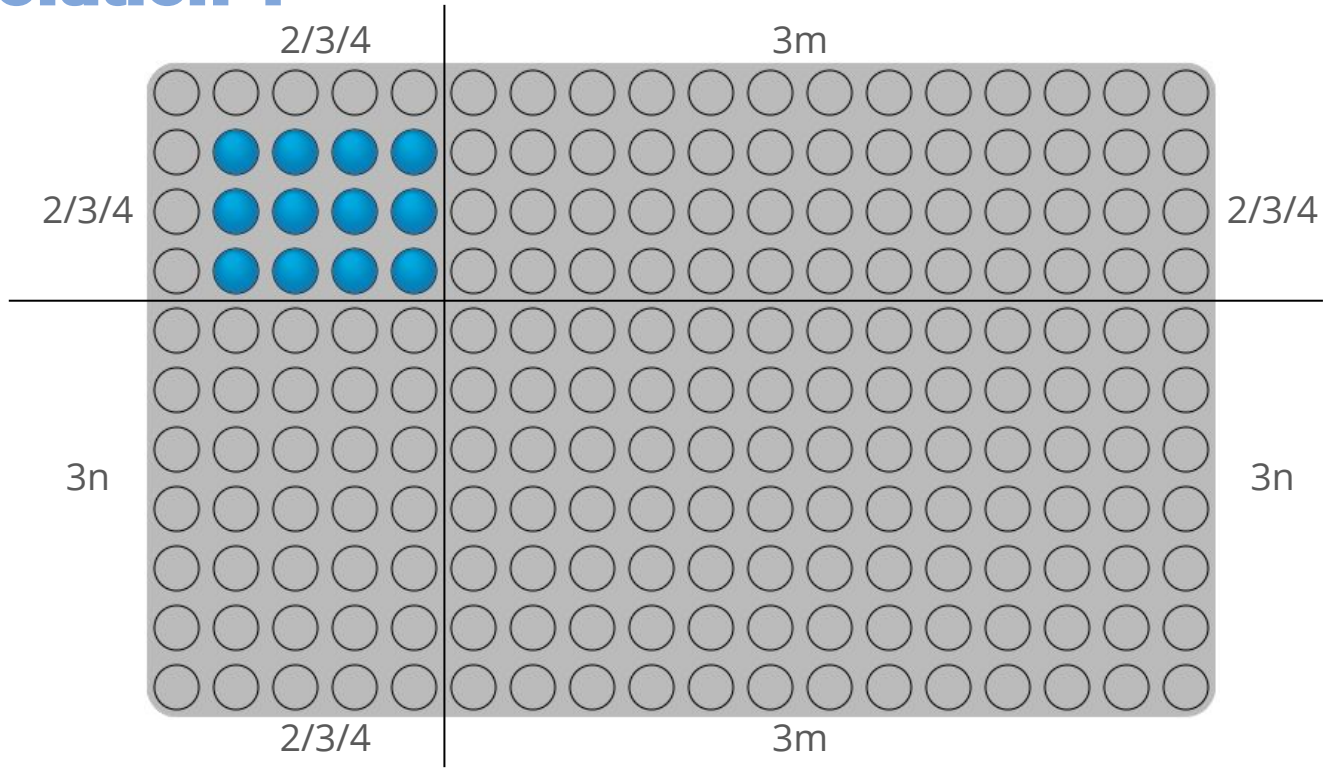
Full Solution 1



Full Solution 1



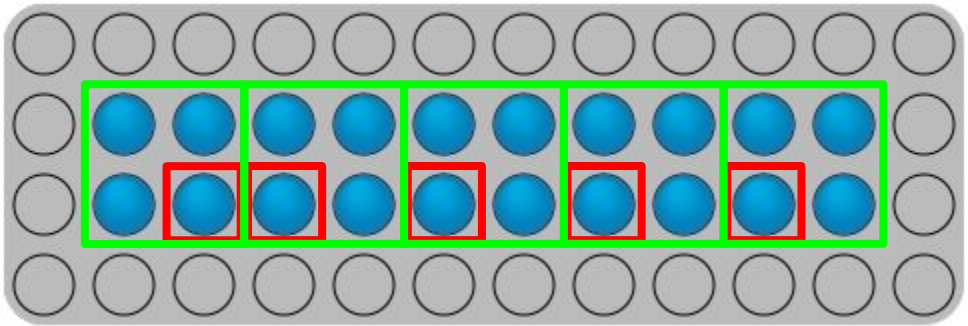
Full Solution 1



Subtask 7 (15 points)

R = 2 (revisited)

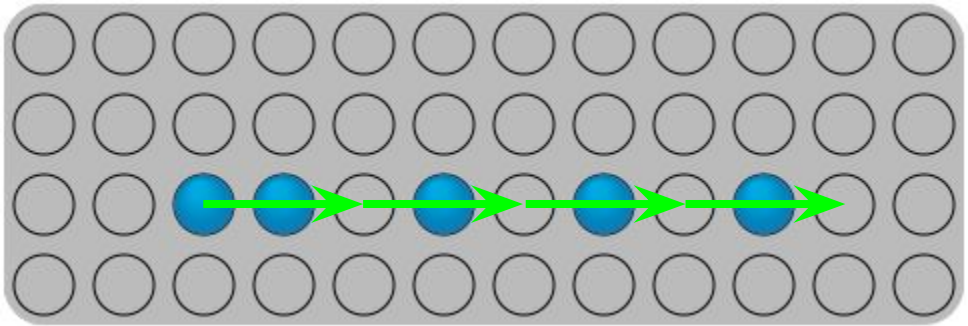
Another strategy:



Subtask 7 (15 points)

R = 2 (revisited)

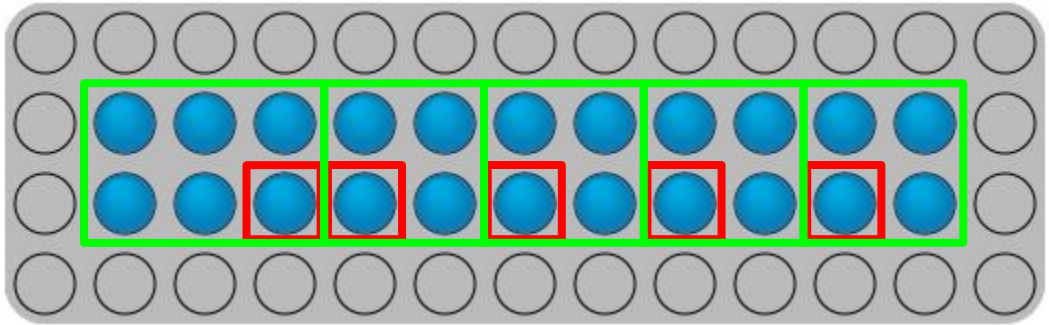
Another strategy:



Subtask 7 (15 points)

R = 2 (revisited)

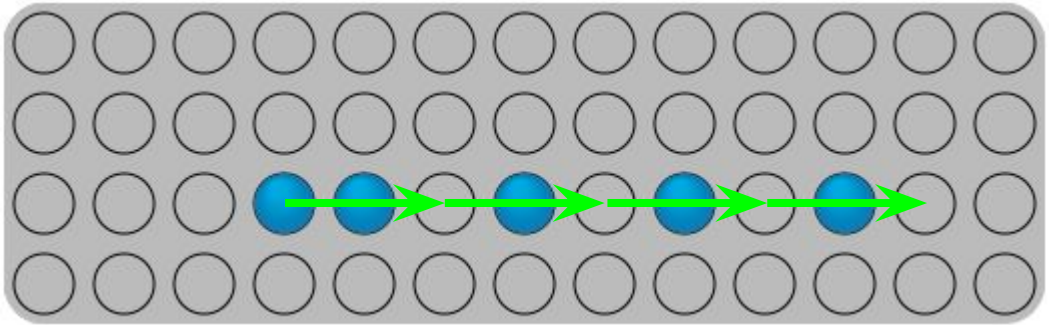
Another strategy:



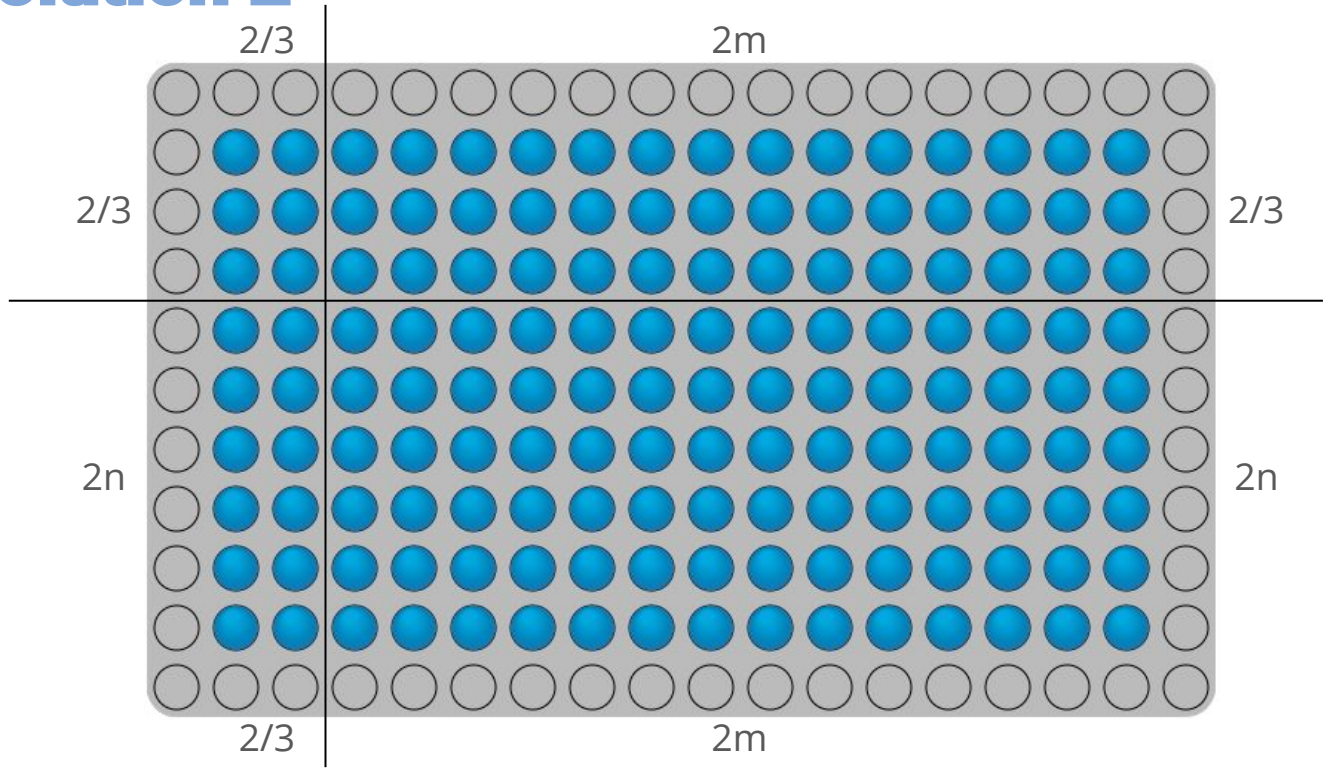
Subtask 7 (15 points)

R = 2 (revisited)

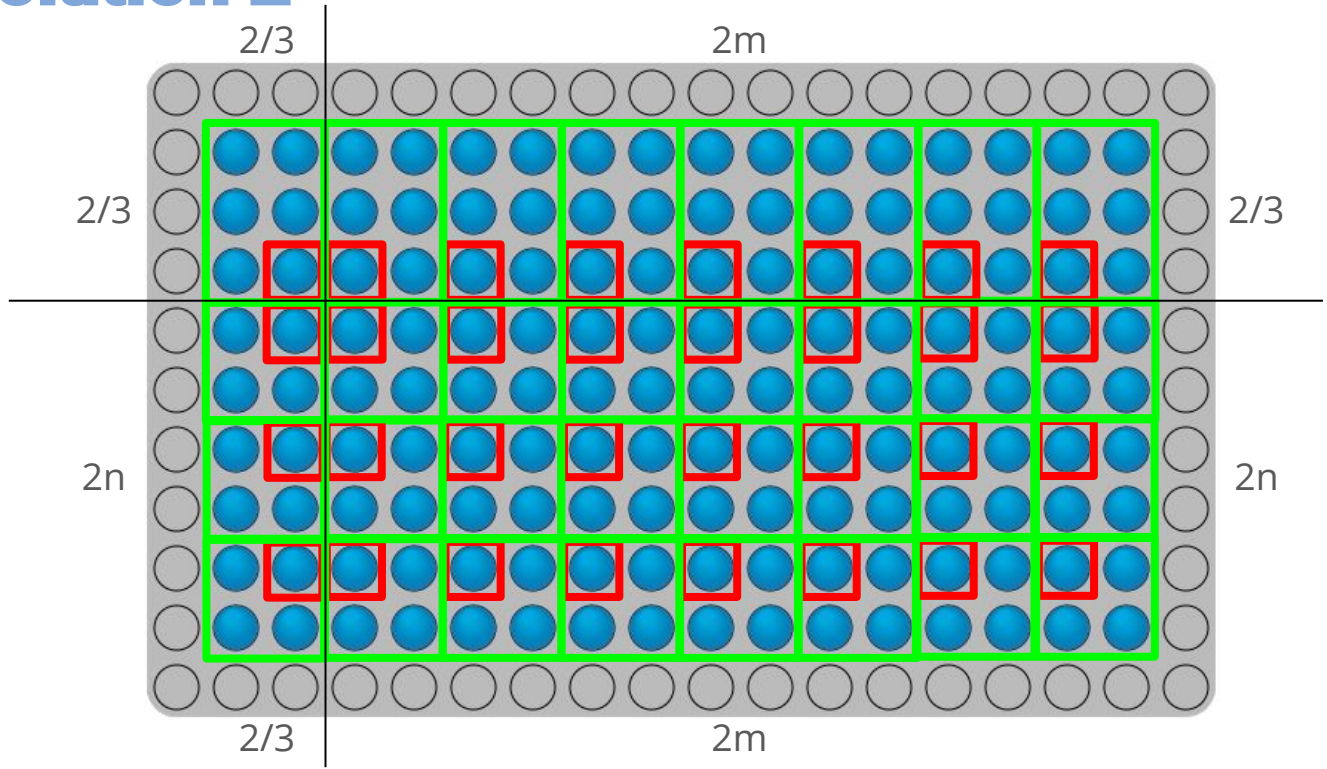
Another strategy:



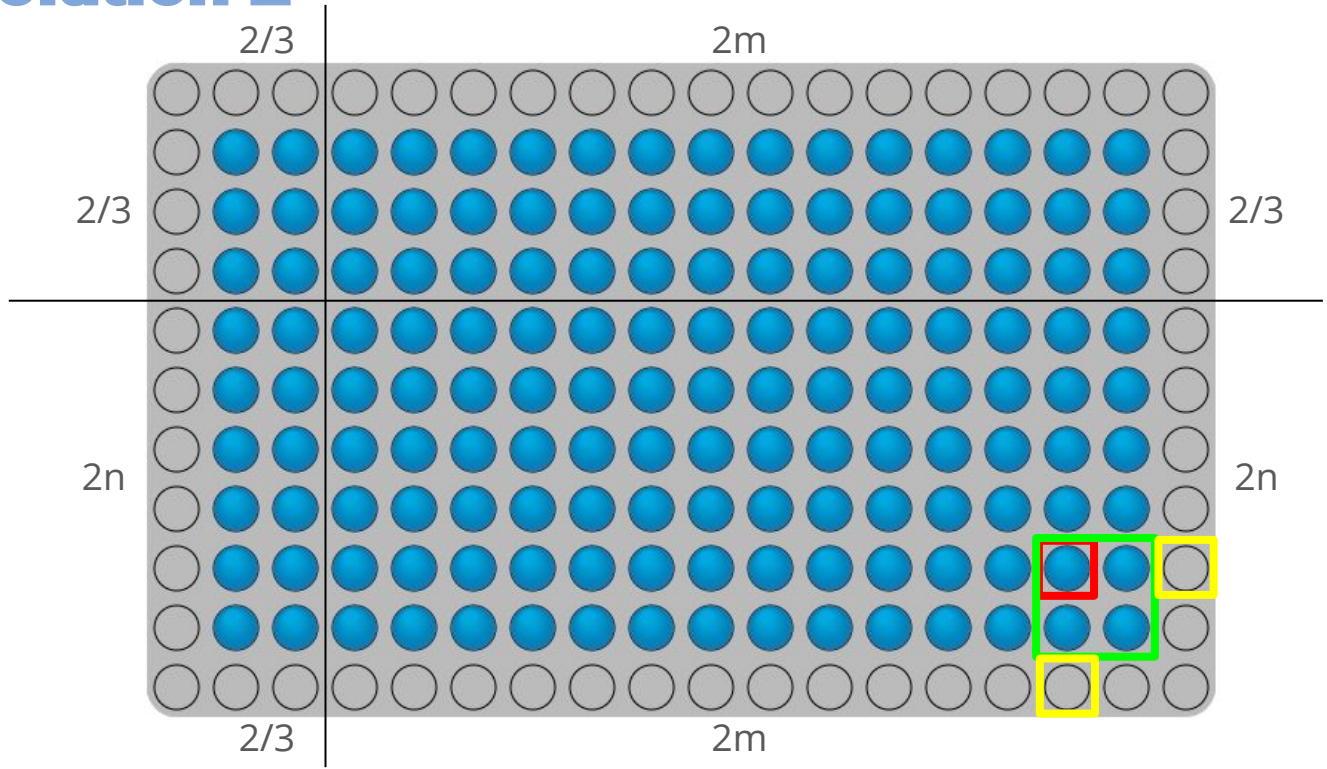
Full Solution 2



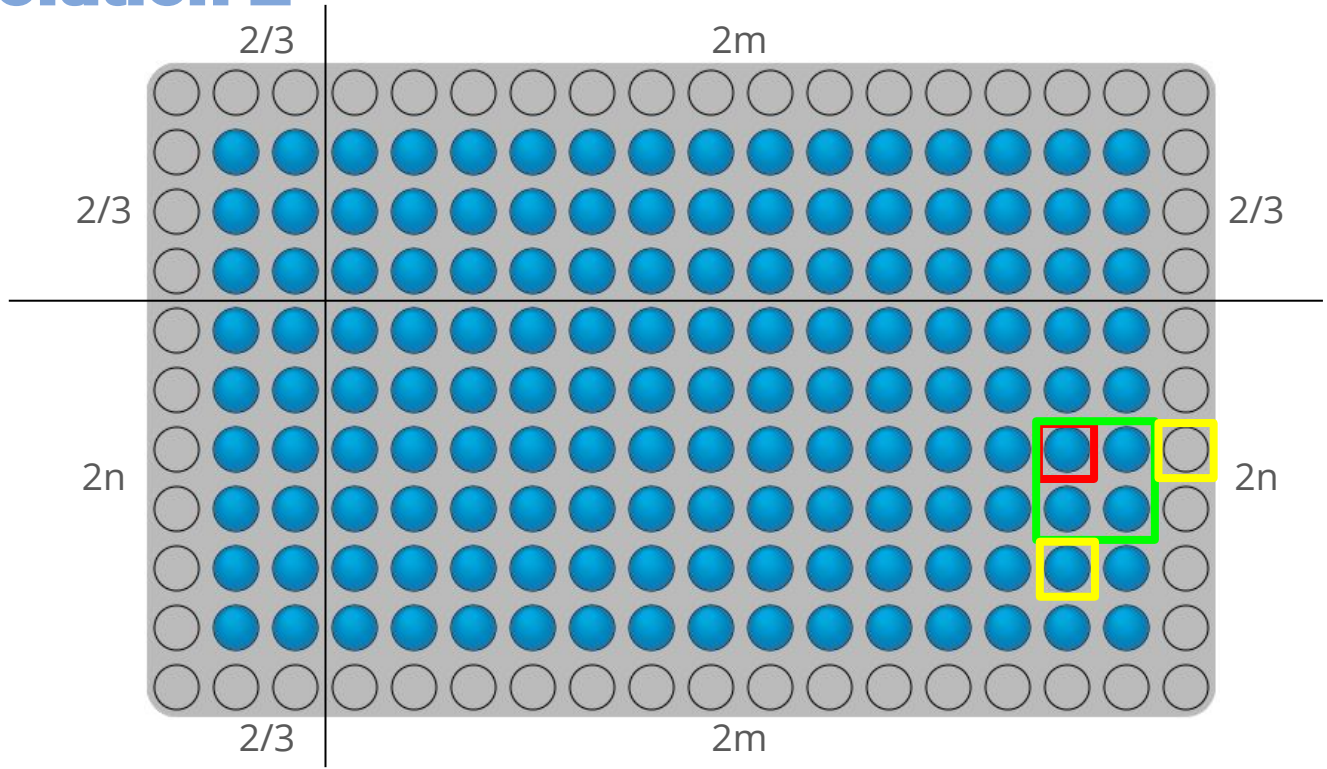
Full Solution 2



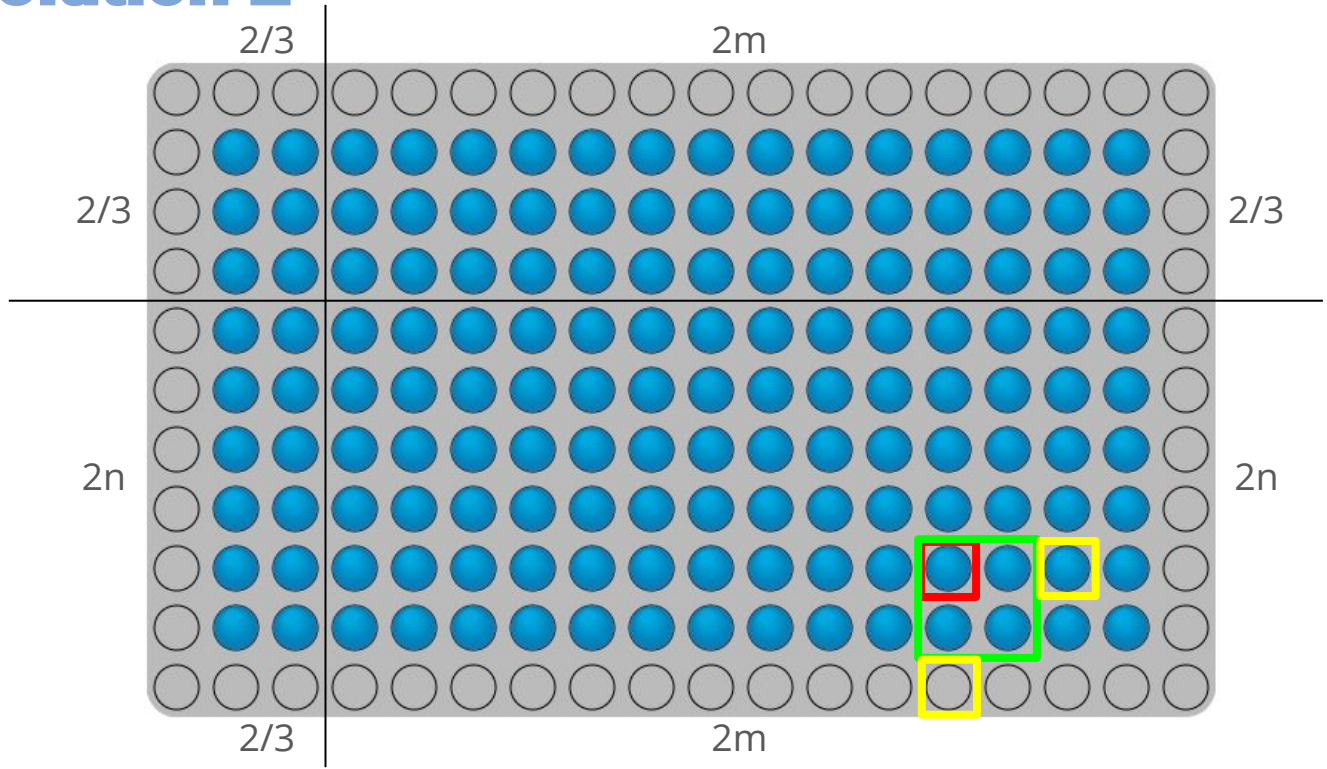
Full Solution 2



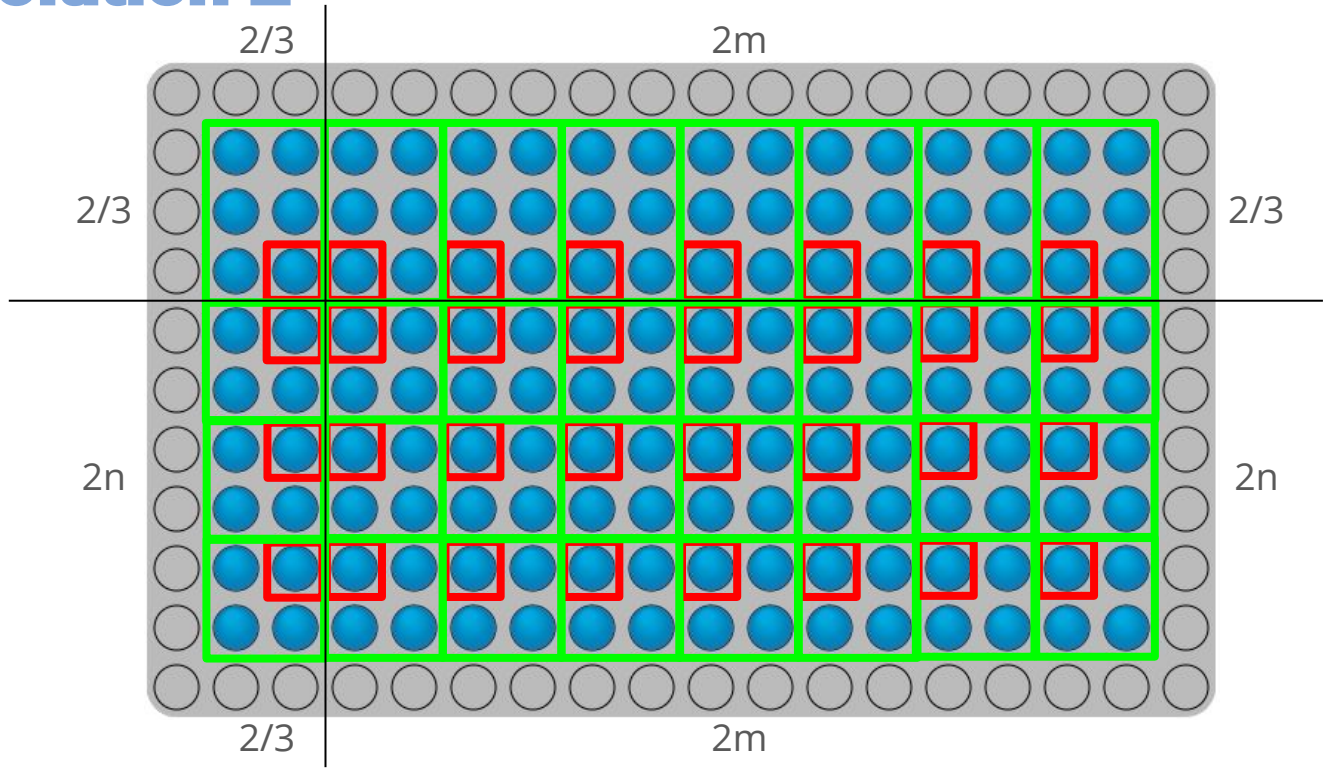
Full Solution 2



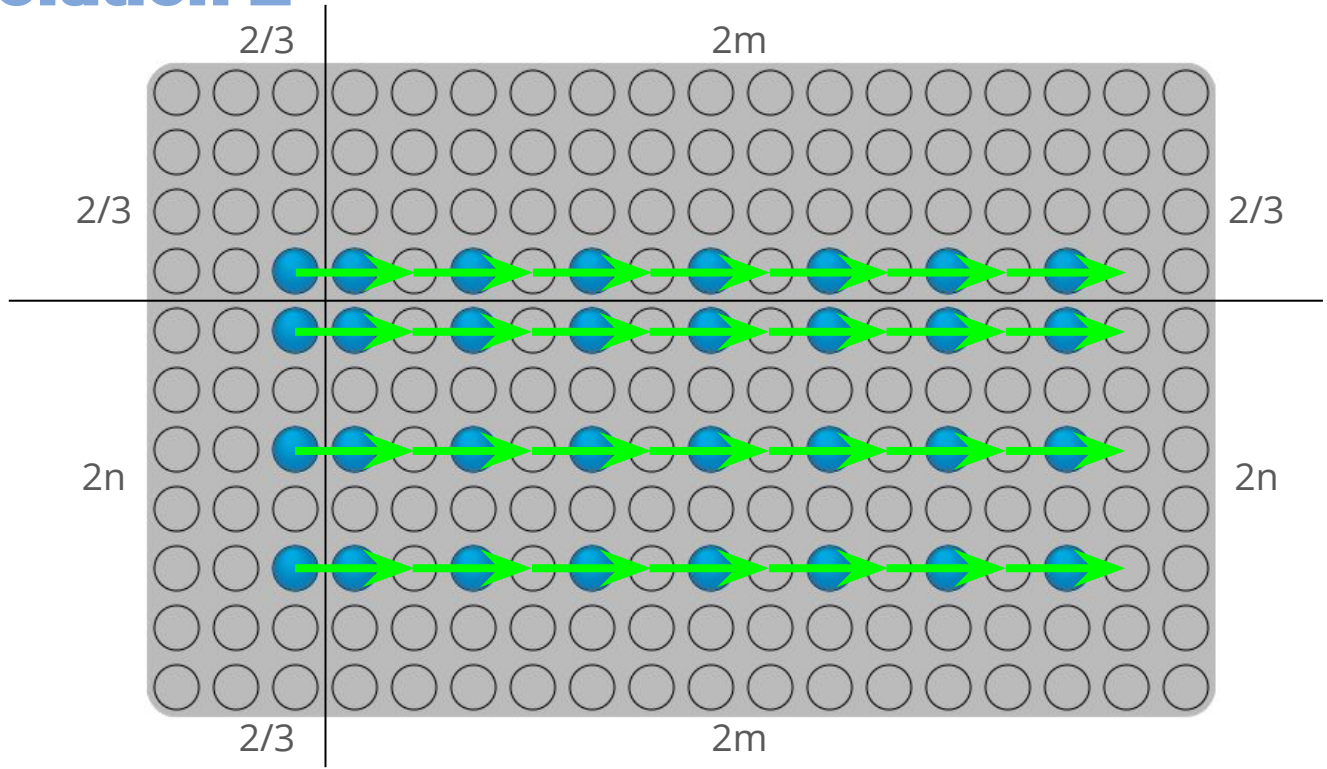
Full Solution 2



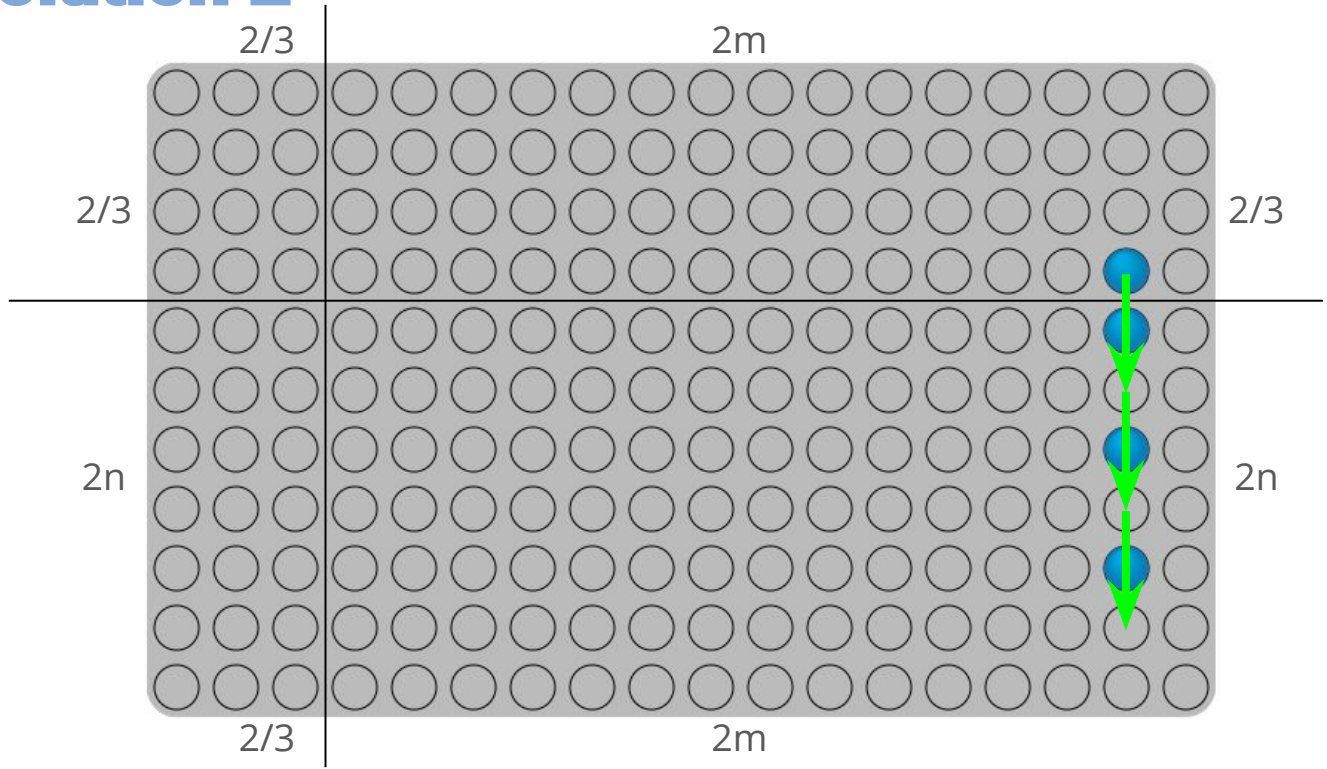
Full Solution 2



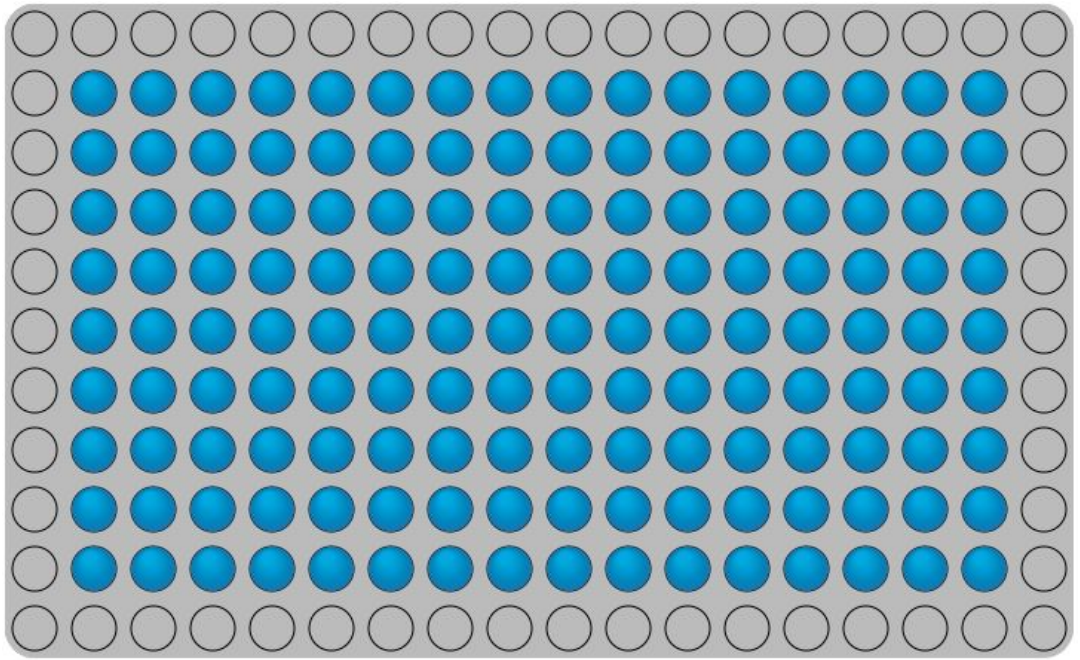
Full Solution 2



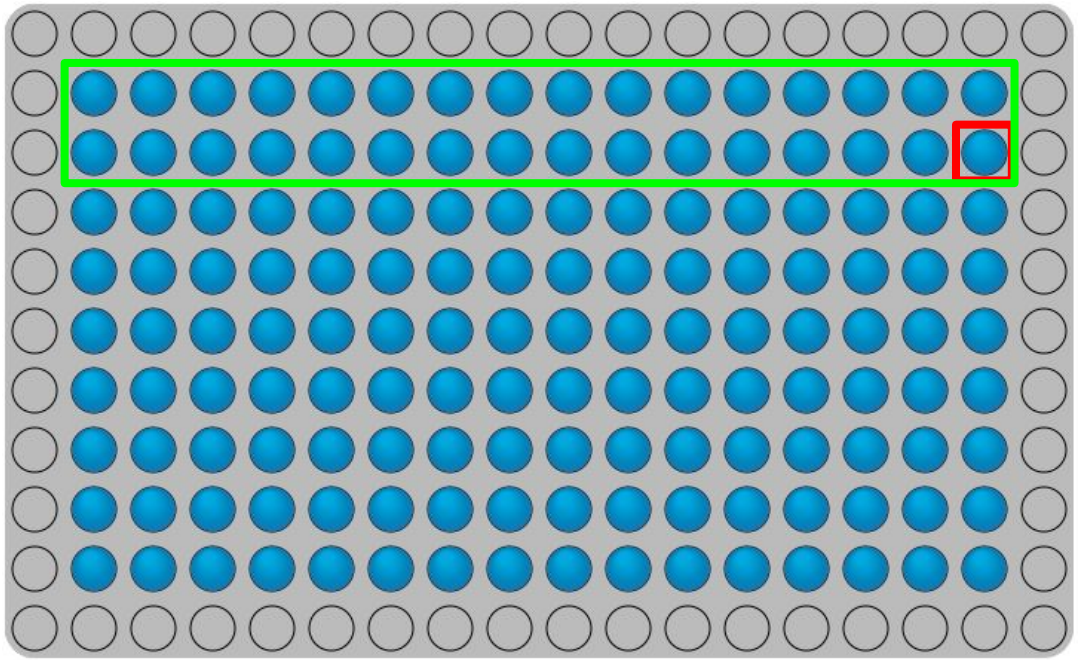
Full Solution 2



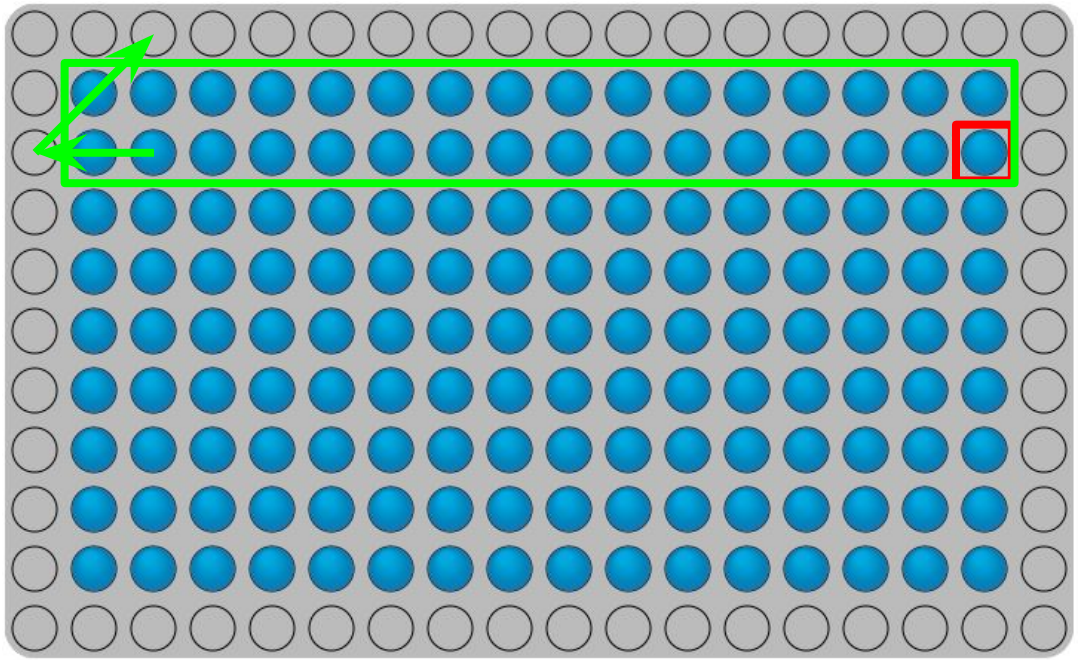
Full Solution 3 - by dbstoshinari123



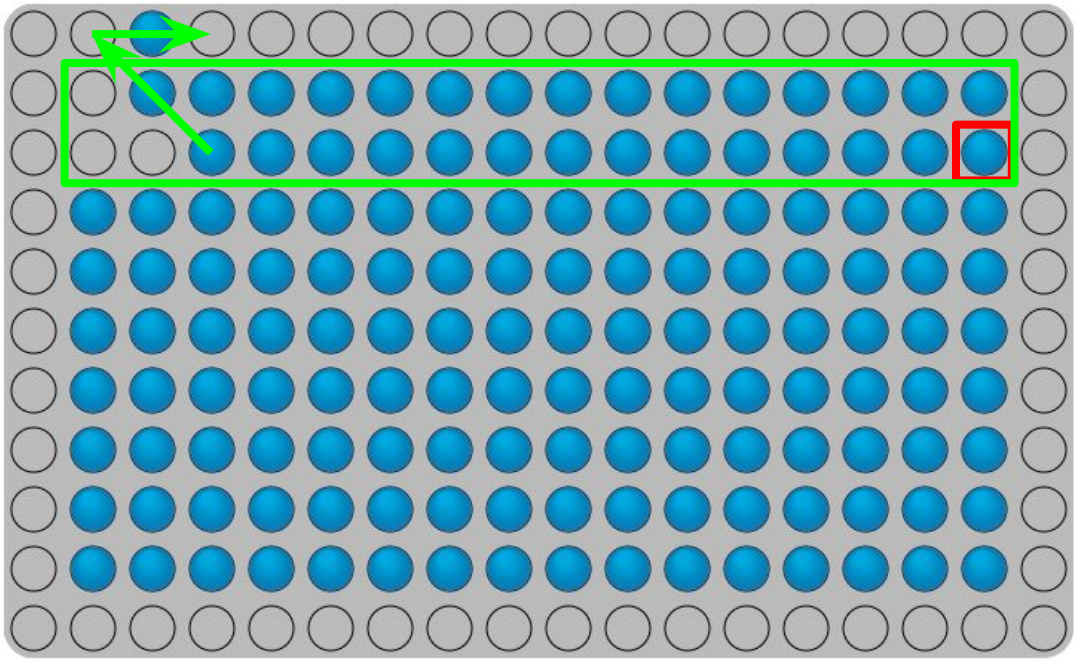
Full Solution 3 - by dbstoshinari123



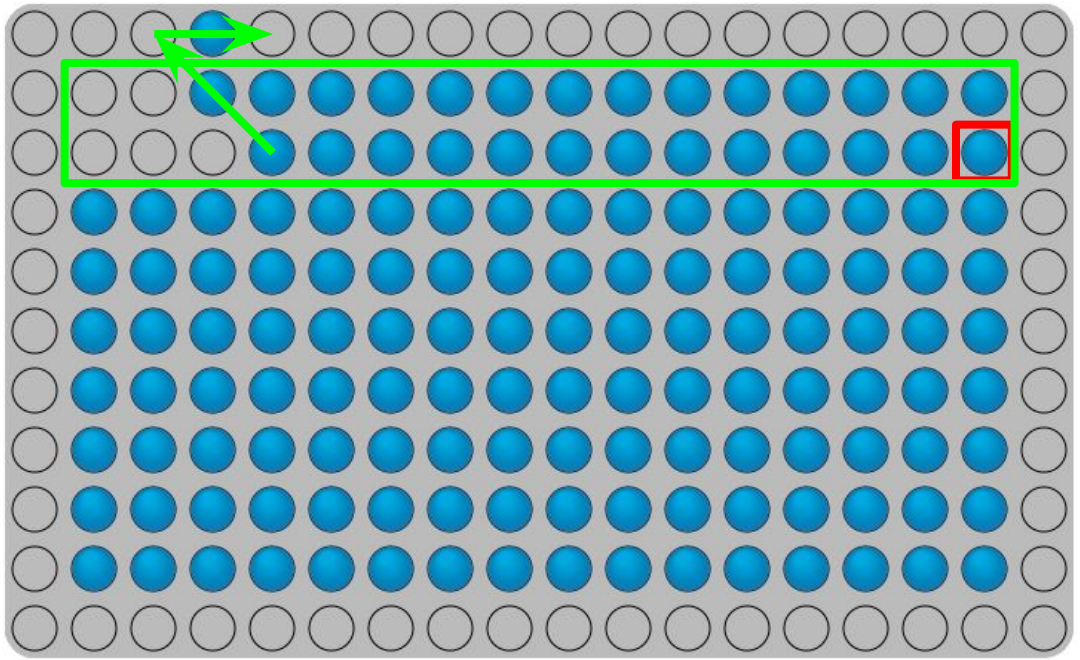
Full Solution 3 - by dbstoshinari123



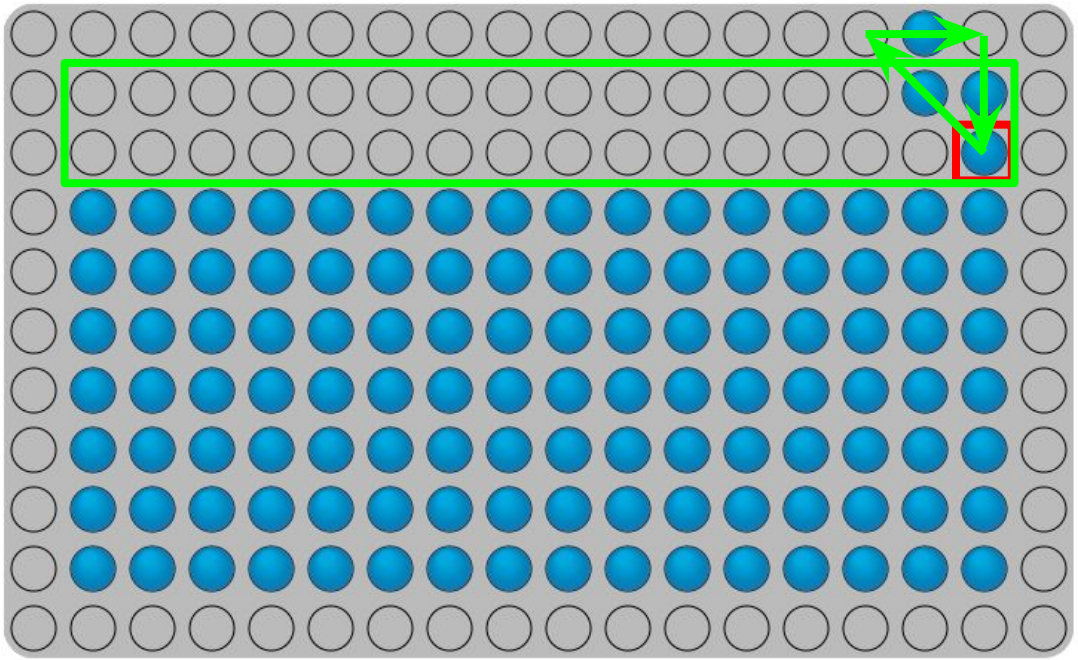
Full Solution 3 - by dbstoshinari123



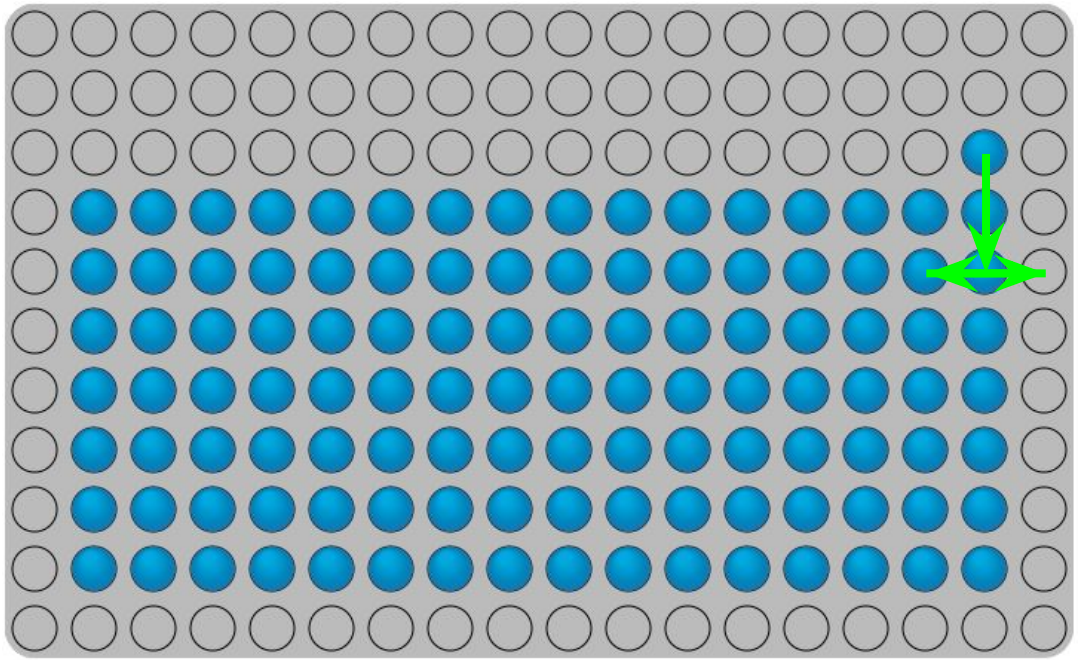
Full Solution 3 - by dbstoshinari123



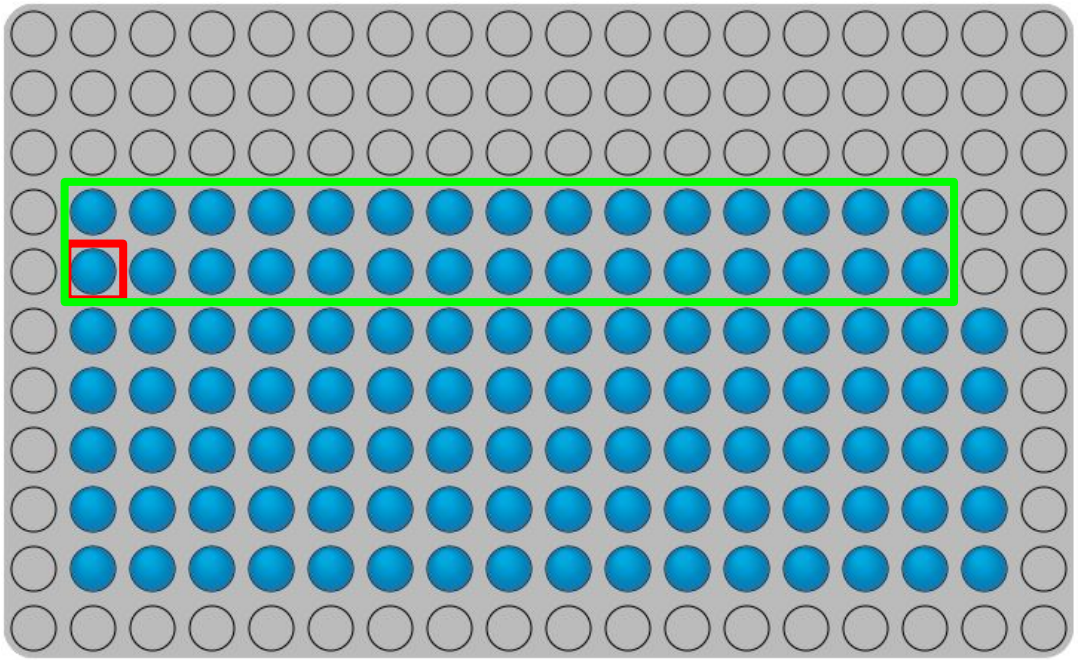
Full Solution 3 - by dbstoshinari123



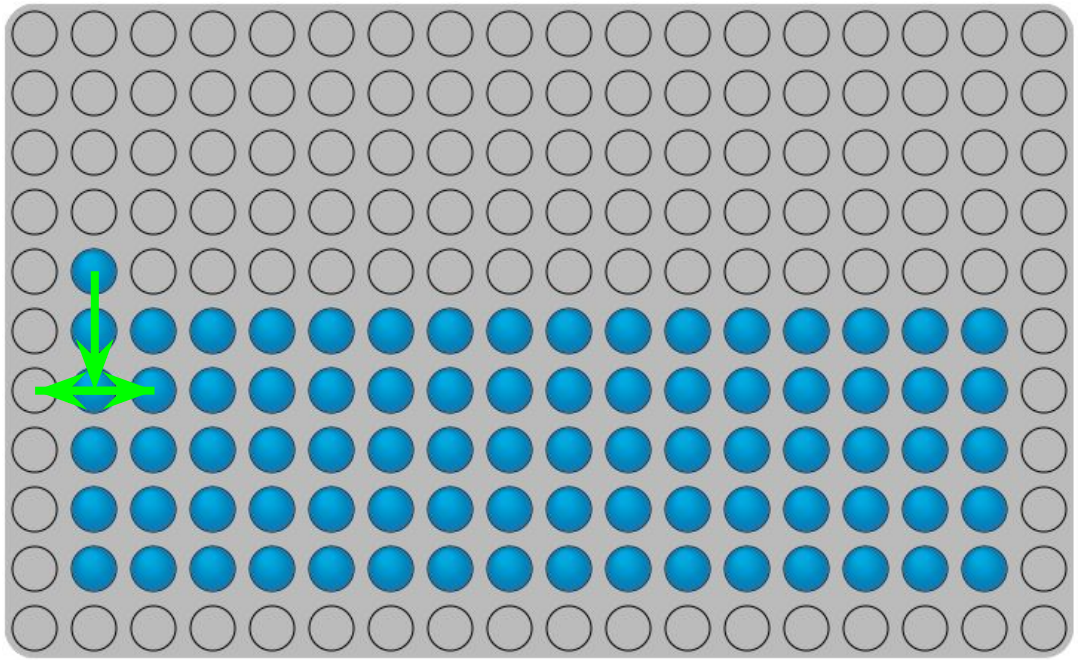
Full Solution 3 - by dbstoshinari123



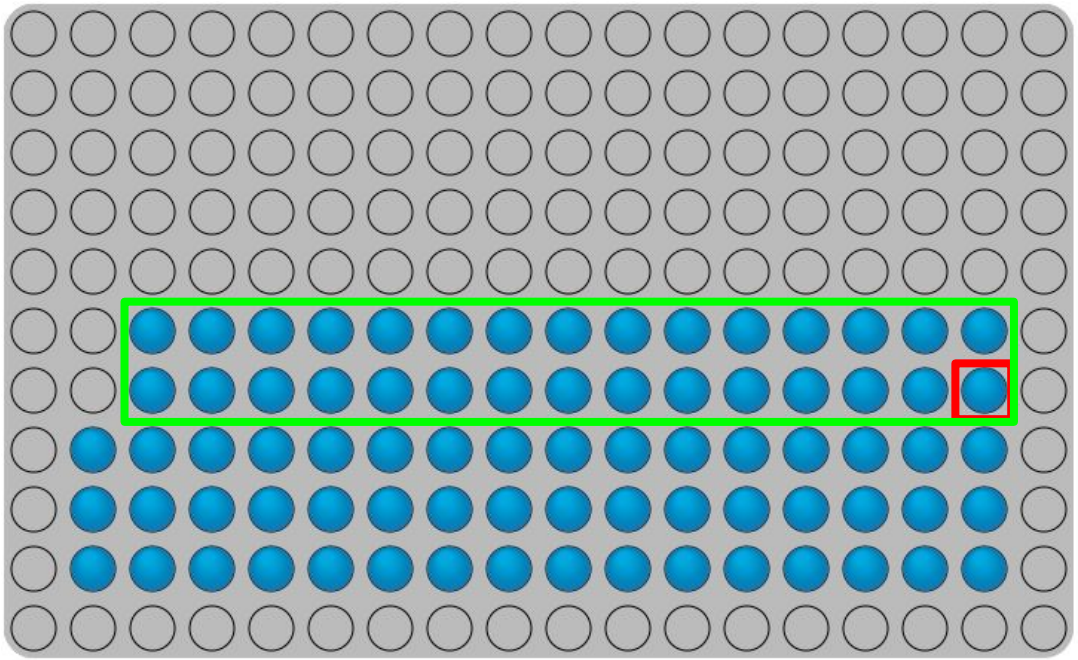
Full Solution 3 - by dbstoshinari123



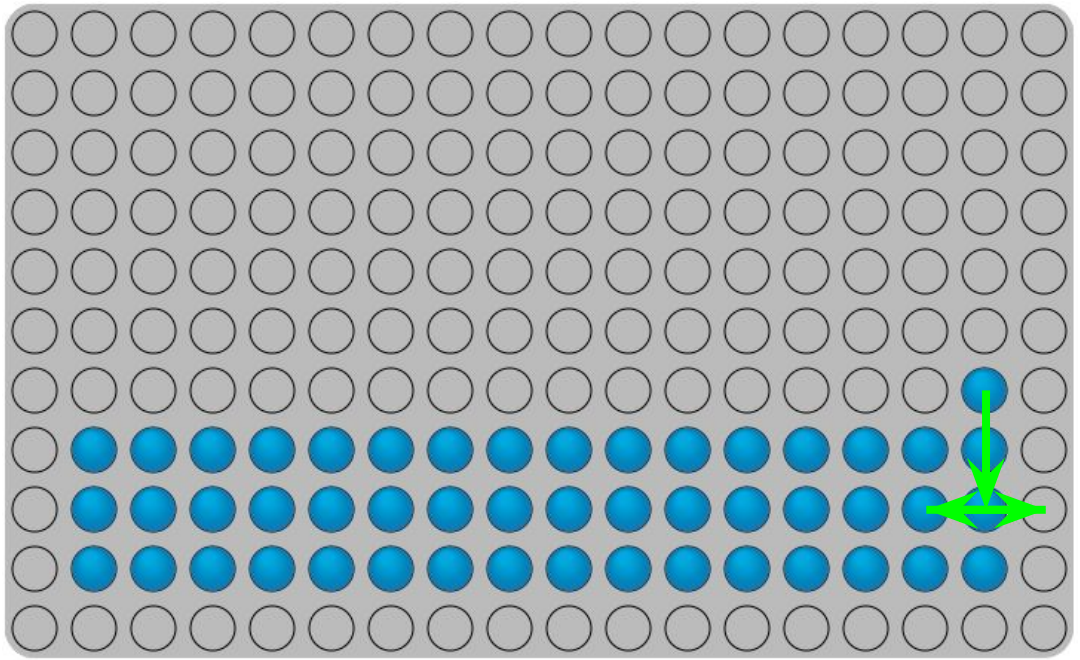
Full Solution 3 - by dbstoshinari123



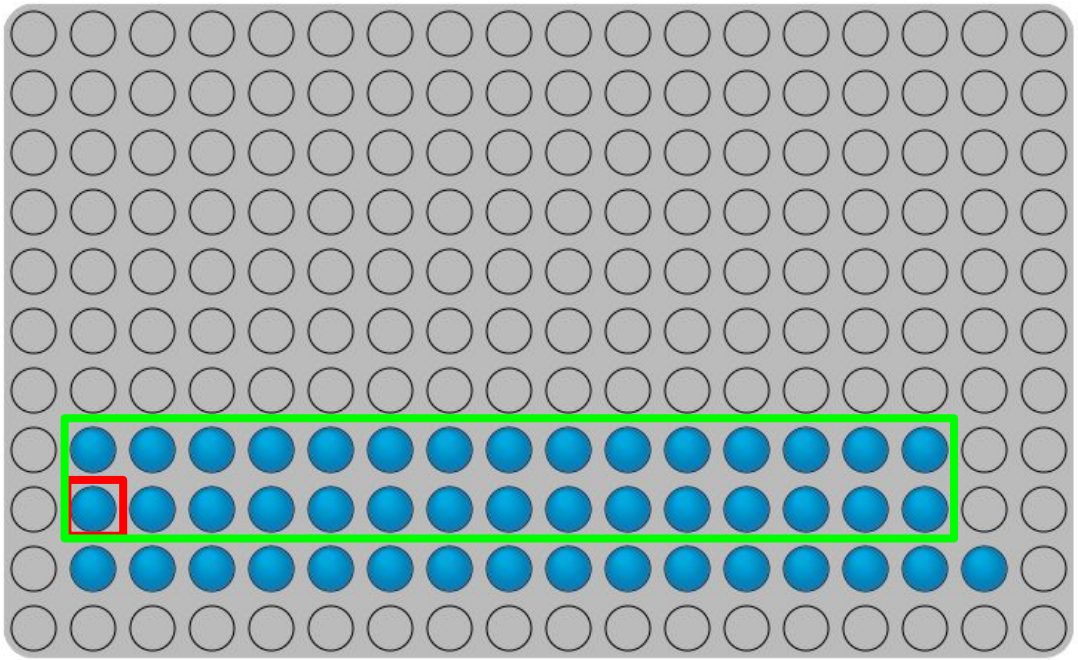
Full Solution 3 - by dbstoshinari123



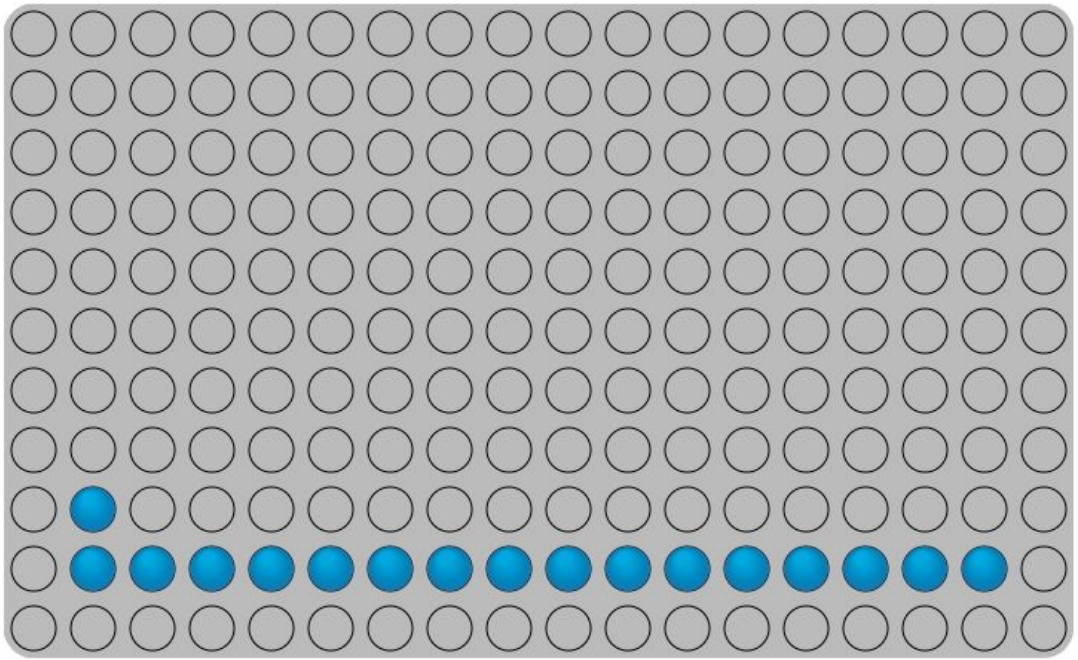
Full Solution 3 - by dbstoshinari123



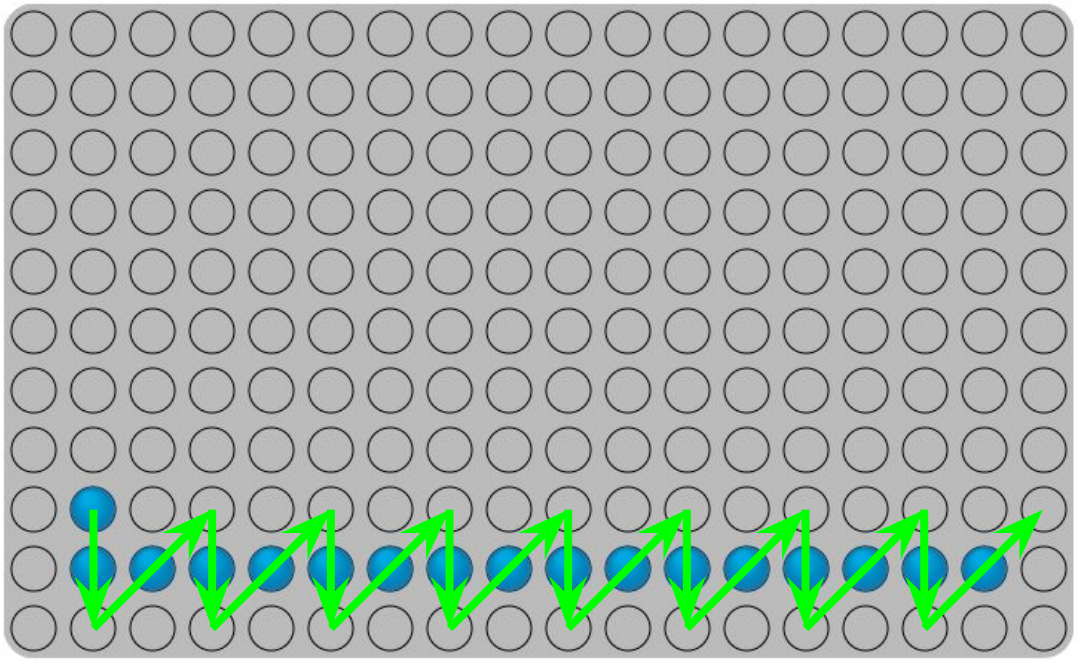
Full Solution 3 - by dbstoshinari123



Full Solution 3 - by dbstoshinari123



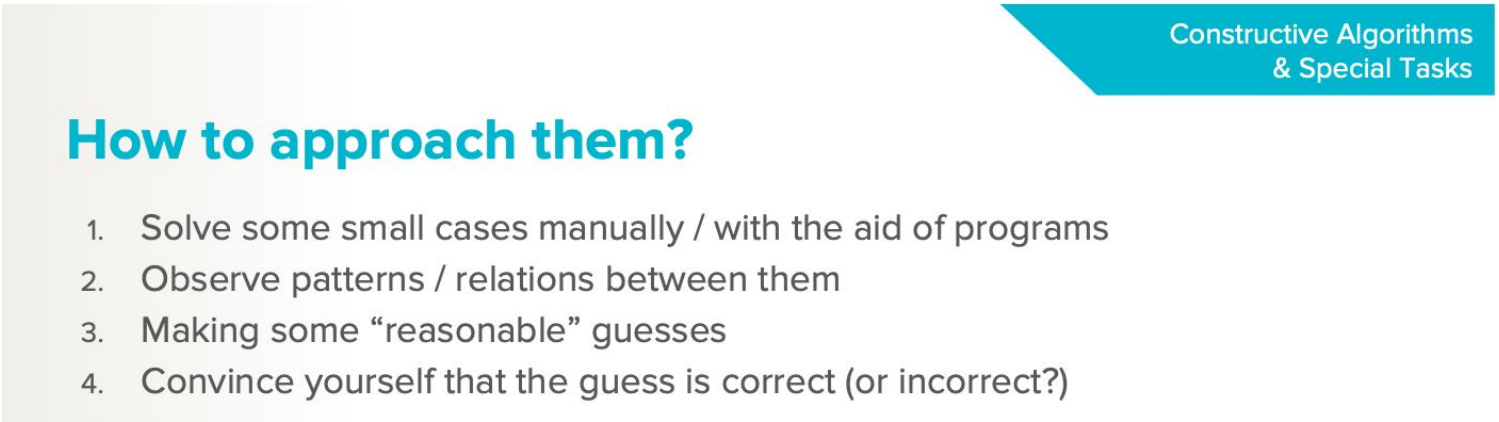
Full Solution 3 - by dbstoshinari123



Overview

To solve this kind of **ad-hoc** and/or **constructive** problems:

- Usually more interesting (?) and less “standard”
- Usually requires a lot of rough work and/or insight and/or intuition



https://assets.hkoi.org/training2019/cast.pdf

Implementation Tricks

- Reduce the problem into smaller cases / lower dimensions
  - Solve recursively
- Divide your code into sections
  - Clear comments stating which section does what
  - Wrap them into functions with *meaningful* names
- A lot of repetitive set of moves
  - e.g. 2x2, T-moves
  - Wrap them into helper functions
- Row and column operations are symmetric
  - Store outputs with array/vector and print at once
  - Use bool to indicate if you have to swap coordinates (r,c) / (c,r)

Questions?

Have fun with the simulator :)