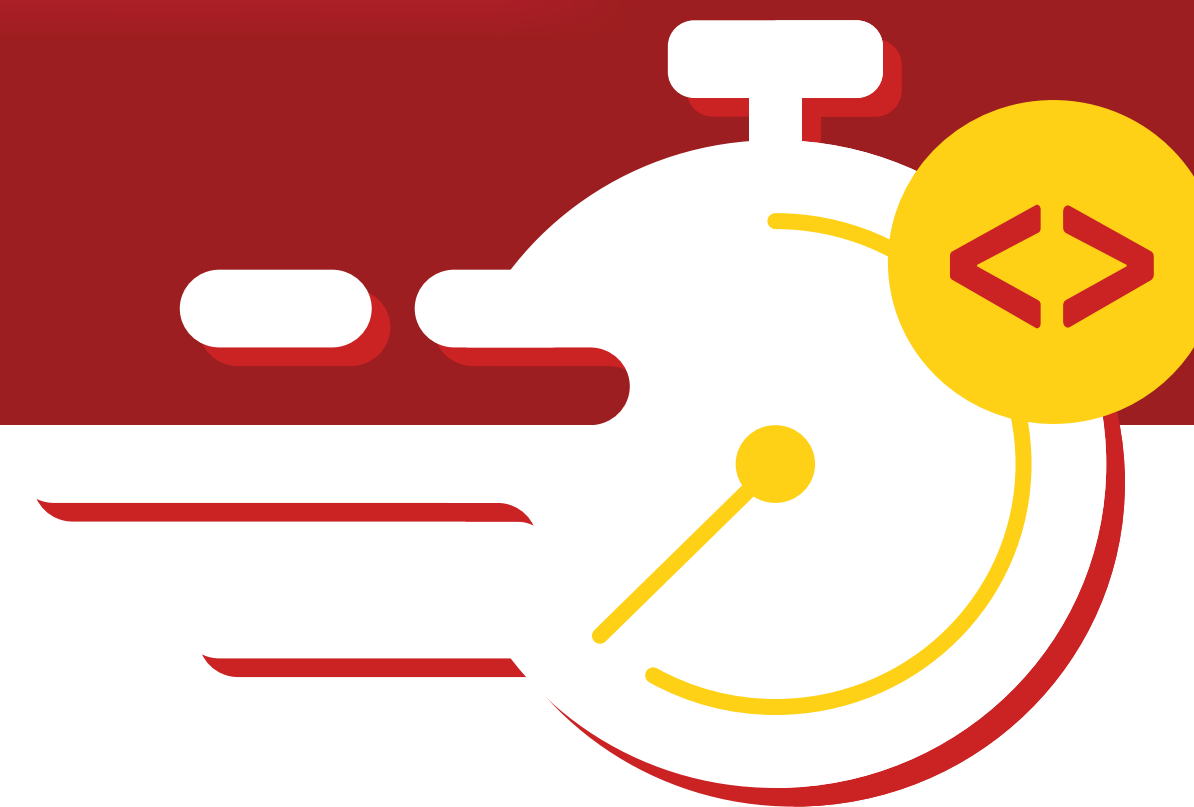
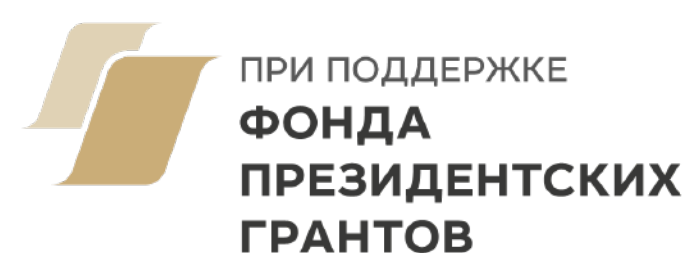
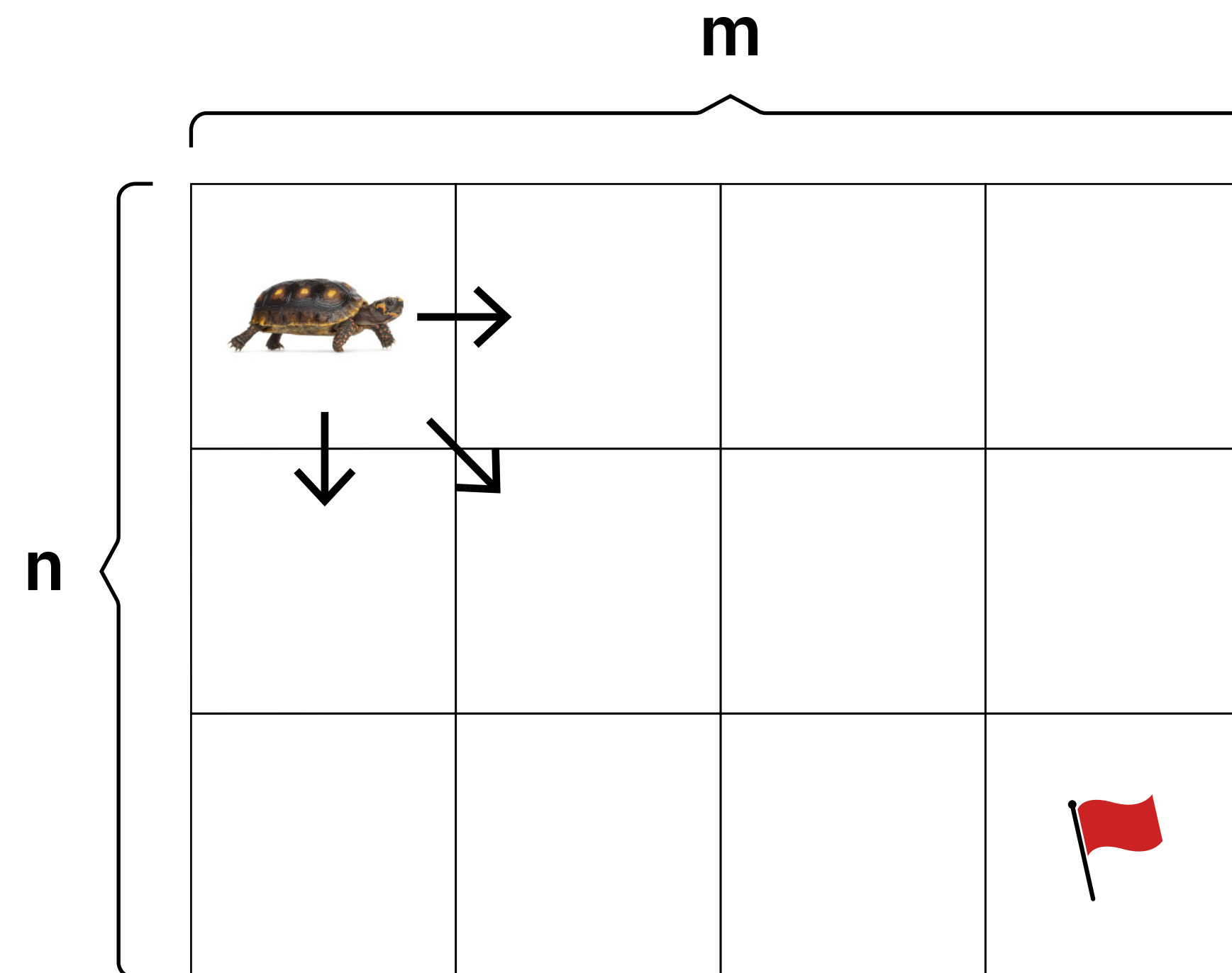


# Двумерное динамическое программирование. Задача о черепашке

Урок 3.1



# Постановка задачи\_



# Пример\_

1	3	5	6
-1	4	2	9
5	-6	1	-4
-6	6	-2	2

Оптимальный штраф: -7

# Решение\_

1. Состояние
2. База
3. Формула
4. Порядок
5. Ответ

# Решение\_

1. **Состояние:**  $dp_{i,j}$  — минимальный штраф, который нужно заплатить, чтобы добраться до клетки на  $i$ -й строке и  $j$ -м столбце
2. База
3. Формула
4. Порядок
5. Ответ

# Пример состояний\_

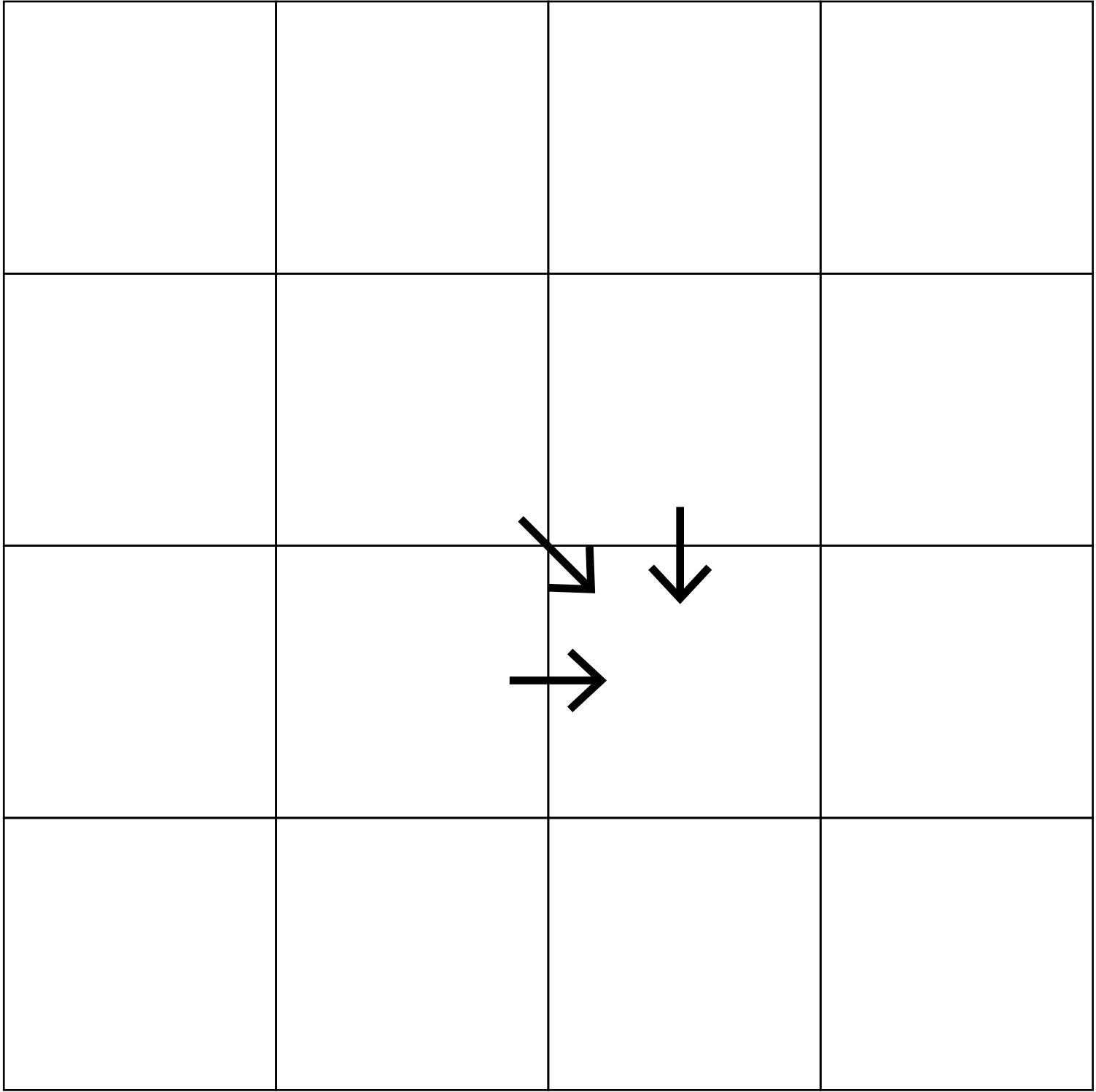
Изначальный массив

1	3	5	6
-1	4	2	9
5	-6	1	-4
-6	6	-2	2

Посчитанная динамика

1	4	9	15
0	4	6	15
5	-6	-5	-9
-1	0	-2	-7

# Формула\_



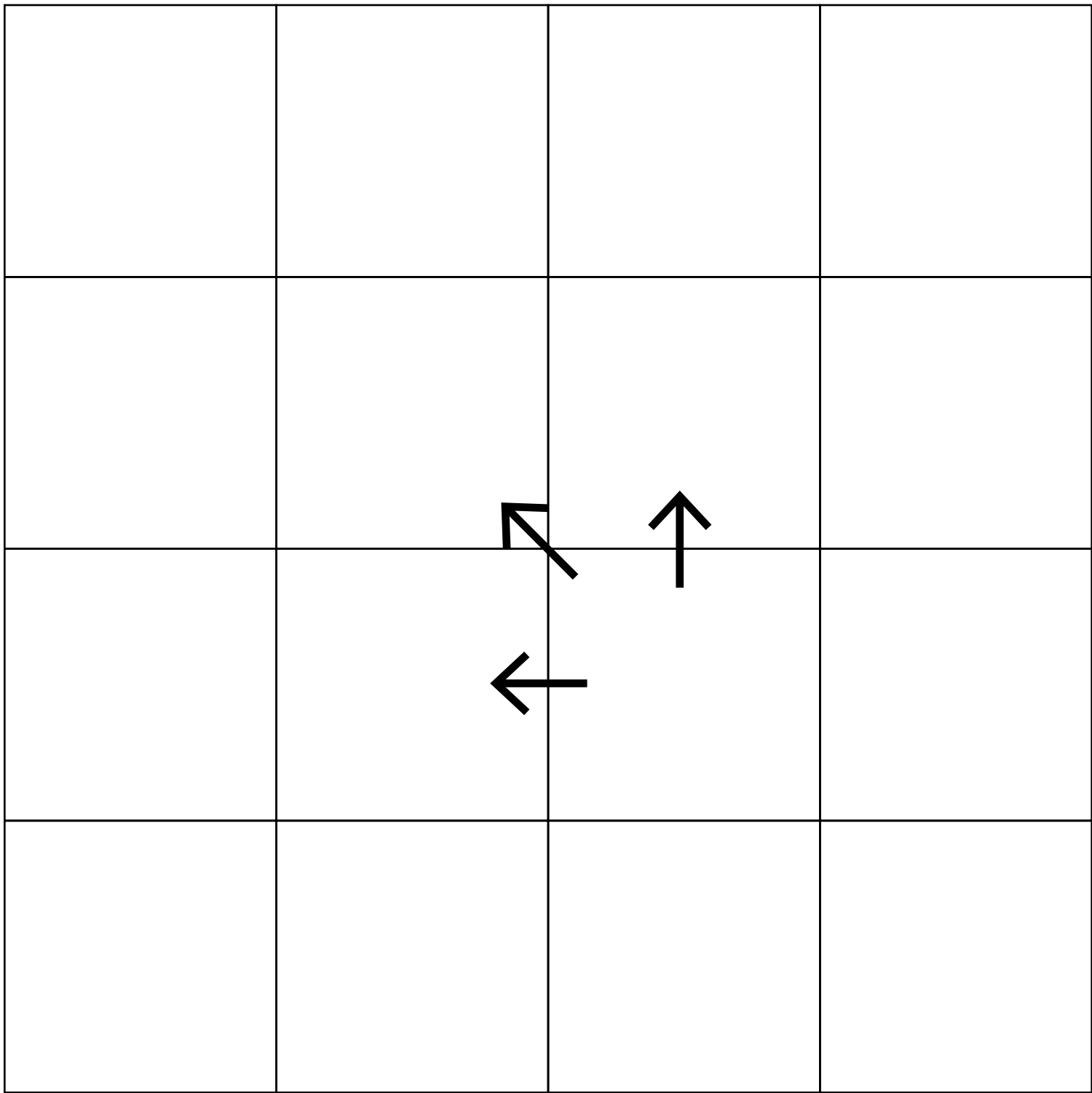
$$dp_{i,j} = \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + a_{i,j}$$

# Решение\_

1. **Состояние:**  $dp_{i,j}$  — минимальный штраф, который нужно заплатить, чтобы добраться до клетки на  $i$ -й строке и  $j$ -м столбце
2. База
3. Формула:  $dp_{i,j} = \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + a_{i,j}$
4. Порядок
5. Ответ



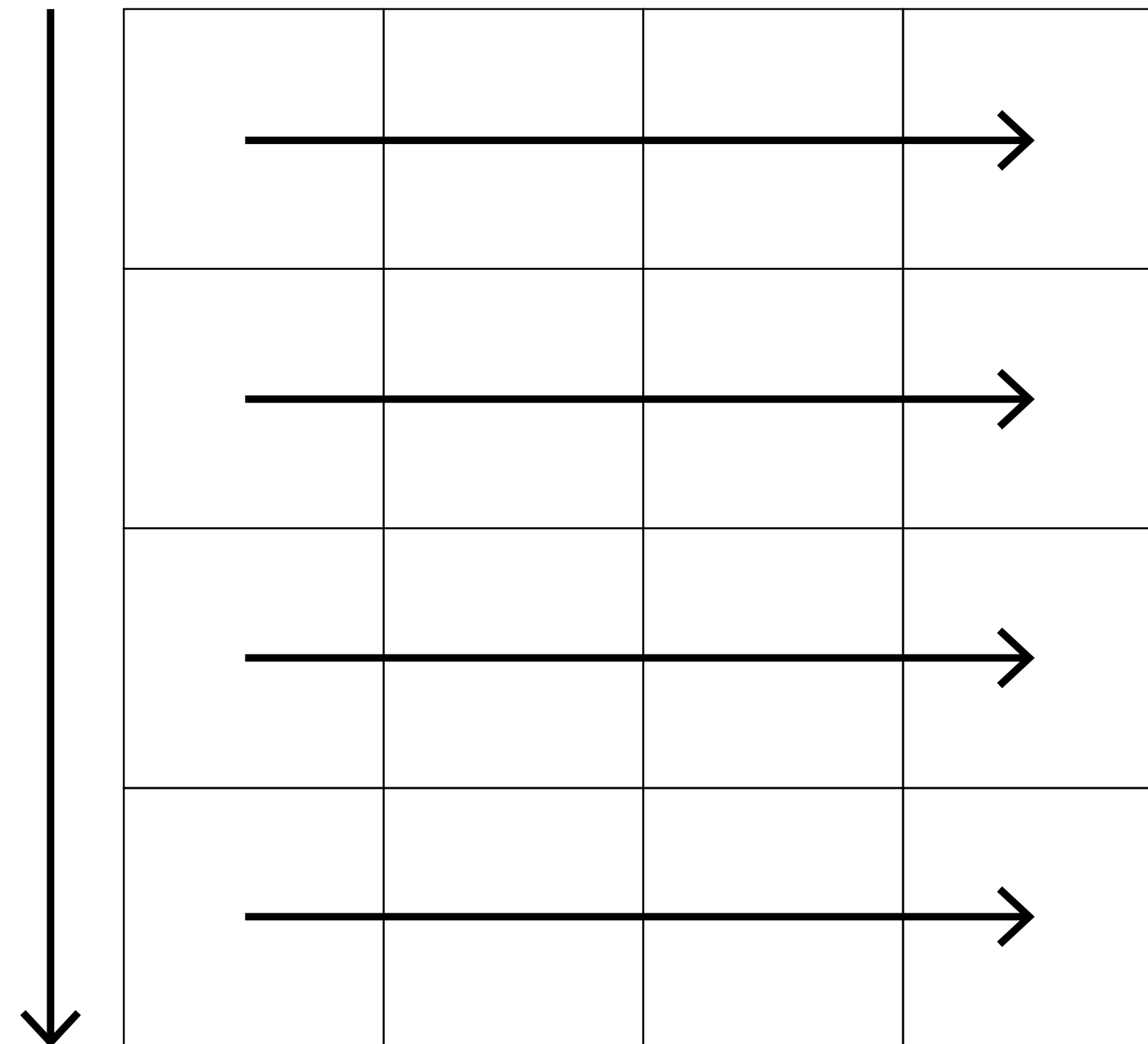
# Порядок\_



# Порядок\_

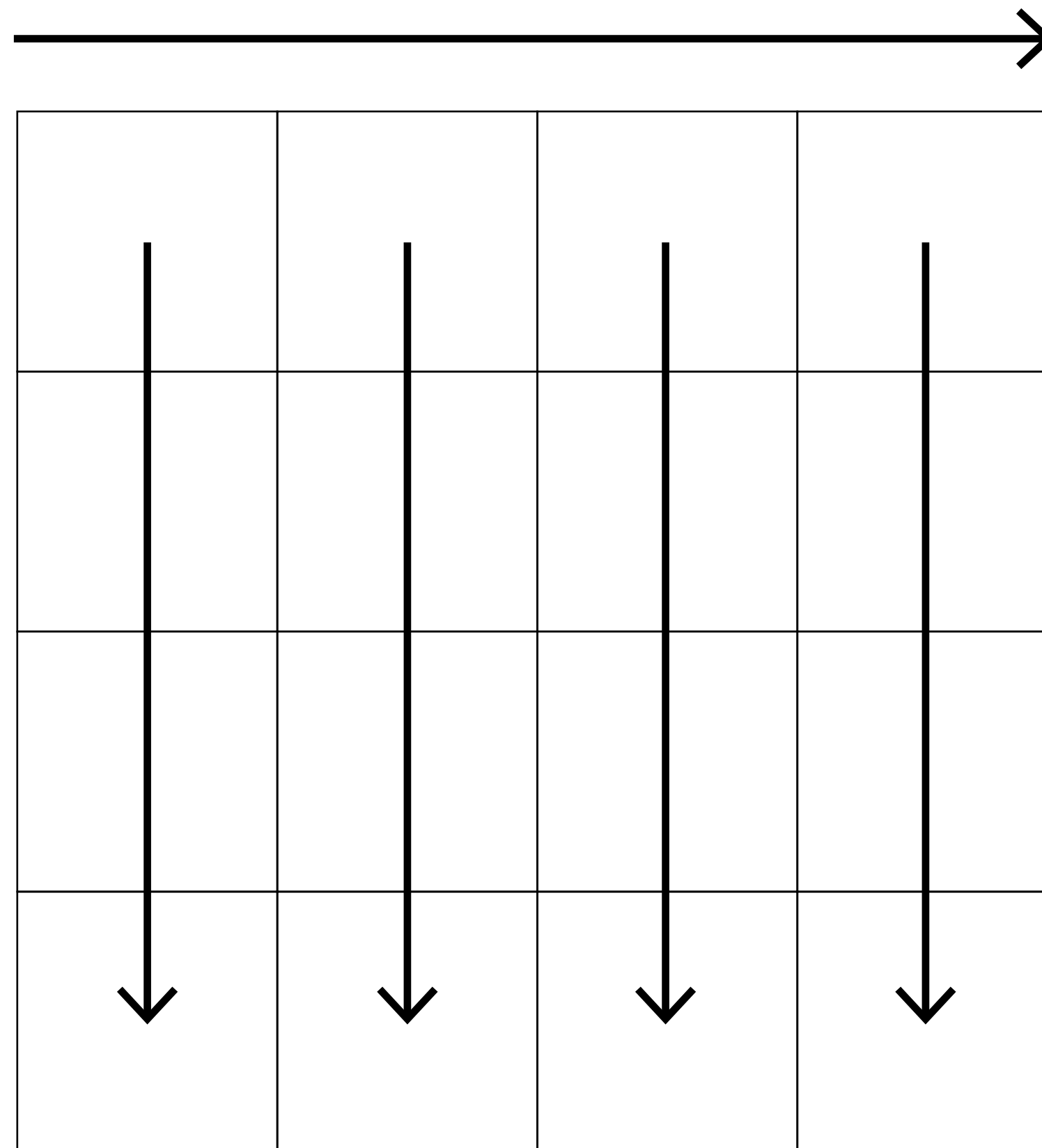



# Порядок\_

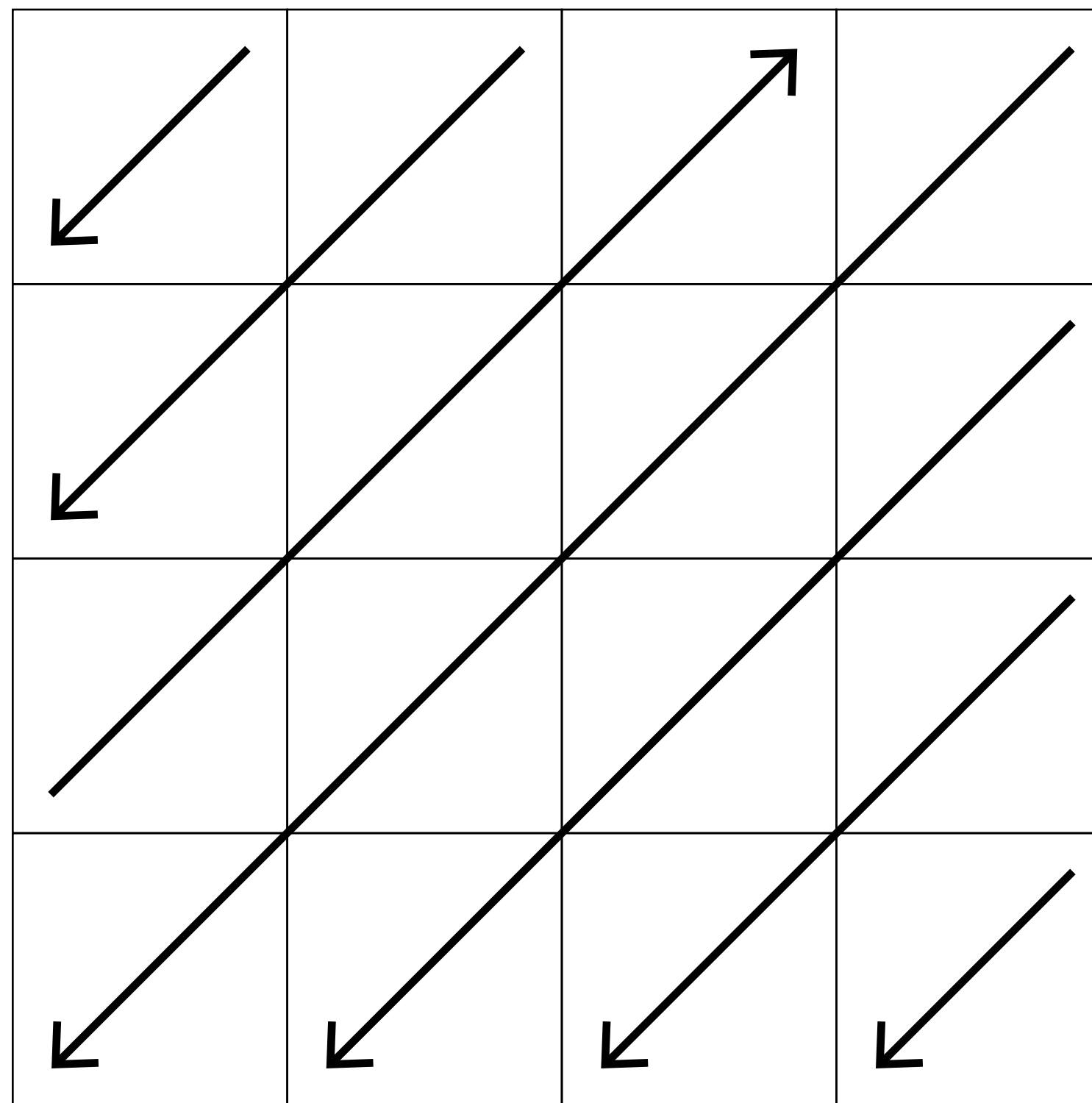


# Порядок\_


# Порядок\_

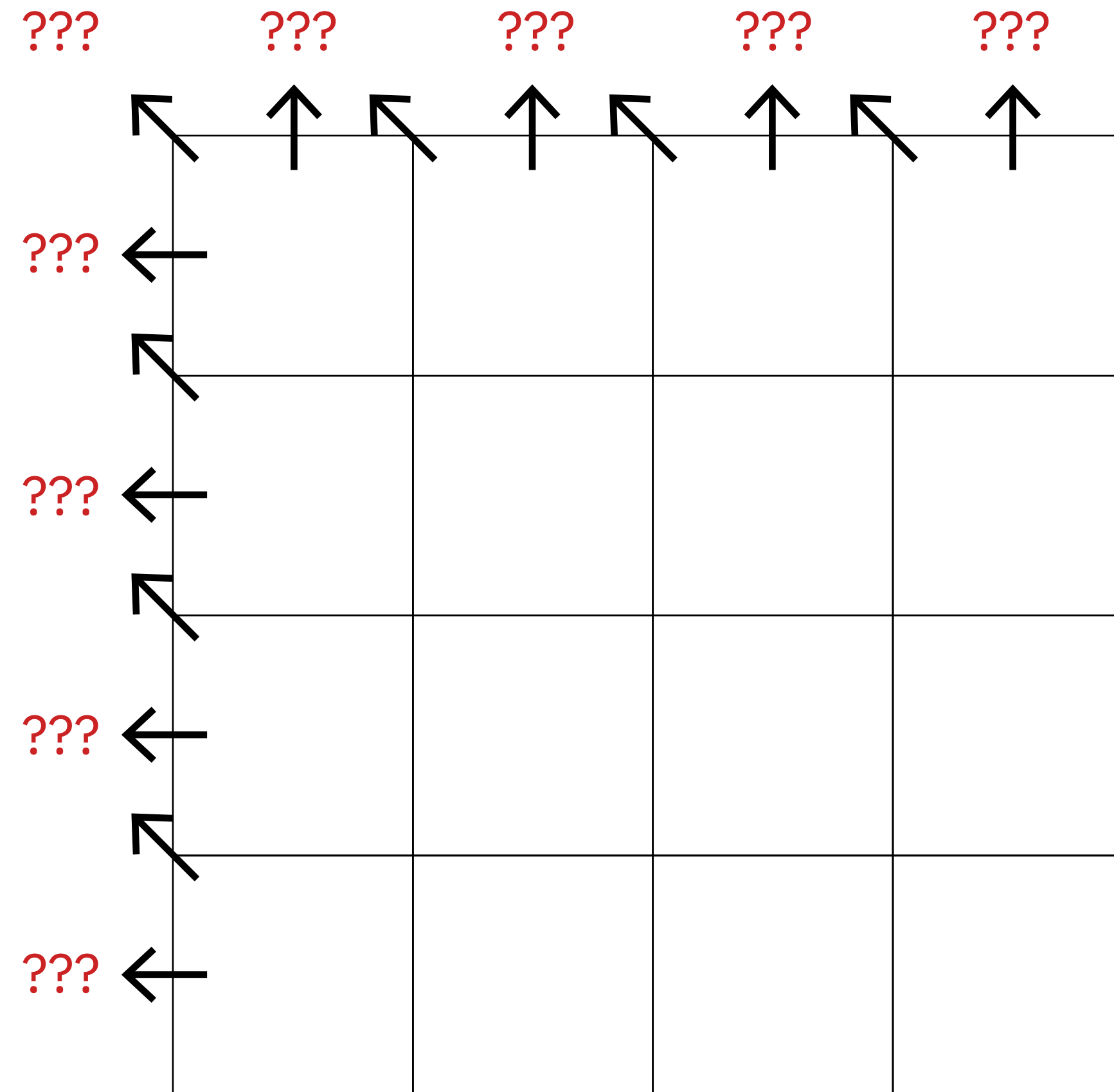


# Альтернативный порядок\_

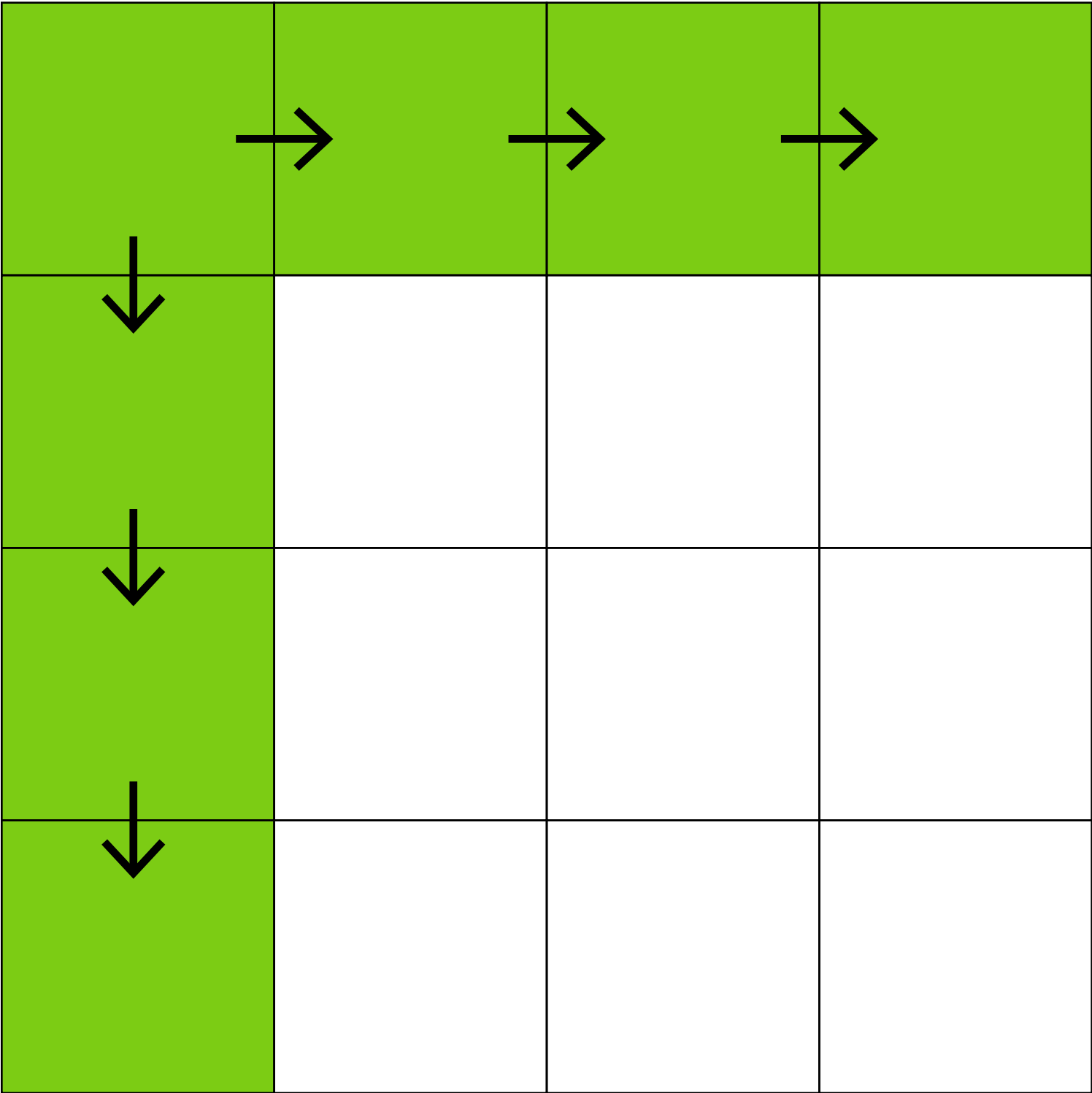


# Решение\_

1. **Состояние:**  $dp_{i,j}$  — минимальный штраф, который нужно заплатить, чтобы добраться до клетки на  $i$ -й строке и  $j$ -м столбце
2. **База**
3. **Формула:**  $dp_{i,j} = \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + a_{i,j}$
4. **Порядок:** любой, например по возрастанию  $i$ , а затем по возрастанию  $j$
5. **Ответ:** лежит в  $dp_{n,m}$







# Решение\_


1. **Состояние:**  $dp_{i,j}$  — минимальный штраф, который нужно заплатить, чтобы добраться до клетки на  $i$ -й строке и  $j$ -м столбце
2. **База:** префиксные суммы для первого столбца и первой строки
3. **Формула:**  $dp_{i,j} = \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + a_{i,j}$
4. **Порядок:** любой, например по возрастанию  $i$ , а затем по возрастанию  $j$
5. **Ответ:** лежит в  $dp_{n,m}$

# Реализация\_

```
1  int n, m;
2  cin >> n >> m;
3
4  vector <vector <int> > a(n + 1, vector <int> (m + 1));
5  vector <vector <int> > dp(n + 1, vector <int> (m + 1));
6
7  // Чтение массива a
8  // Тут должно быть два вложенных цикла
9
10 dp[1][1] = a[1][1];
11
12 for (int j = 2; j <= m; ++j) {
13     dp[1][j] = dp[1][j - 1] + a[1][j];
14 }
15
16 for (int i = 2; i <= n; ++i) {
17     dp[i][1] = dp[i - 1][1] + a[i][1];
18 }
19
20 for (int i = 2; i <= n; ++i) {
21     for (int j = 2; j <= m; ++j) {
22         dp[i][j] = min(min(dp[i - 1][j], dp[i][j - 1]),
23                         dp[i - 1][j - 1] + a[i][j]);
24     }
25 }
26
27 cout << dp[n][m] << endl;
```

# Итоги\_

1. **Состояние:**  $dp_{i,j}$  — минимальный штраф, который нужно заплатить, чтобы добраться до клетки на  $i$ -й строке и  $j$ -м столбце
2. **База:** префиксные суммы для первого столбца и первой строки
3. **Формула:**  $dp_{i,j} = \min(dp_{i-1,j}, dp_{i,j-1}, dp_{i-1,j-1}) + a_{i,j}$
4. **Порядок:** любой, например по возрастанию  $i$ , а затем по возрастанию  $j$
5. **Ответ:** лежит в  $dp_{n,m}$



# Следующее занятие — другая база для черепашки

---