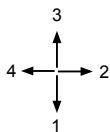


Рекурсия с връщане назад

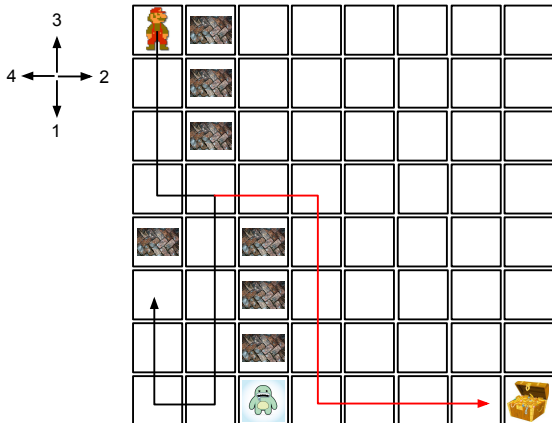
22 декември 2015 г.

Търсене на път в лабиринт

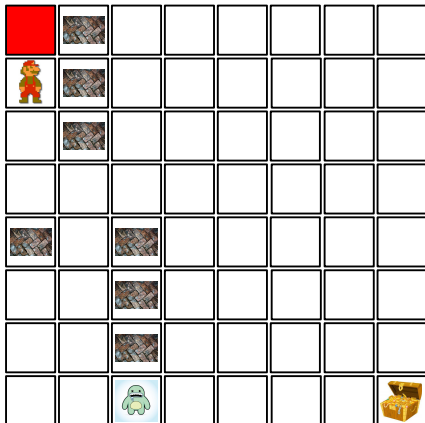
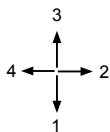
Задачата



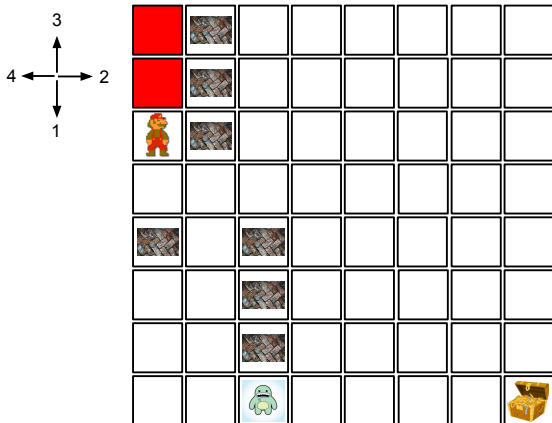
Задачата



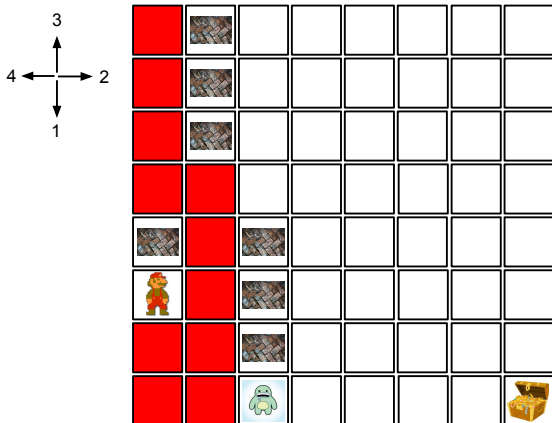
Подзадача



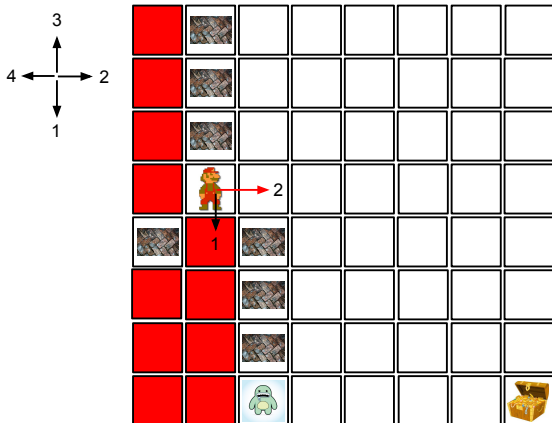
Търсене



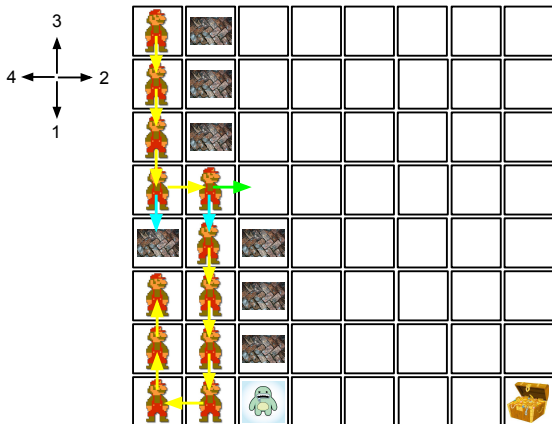
Търсене



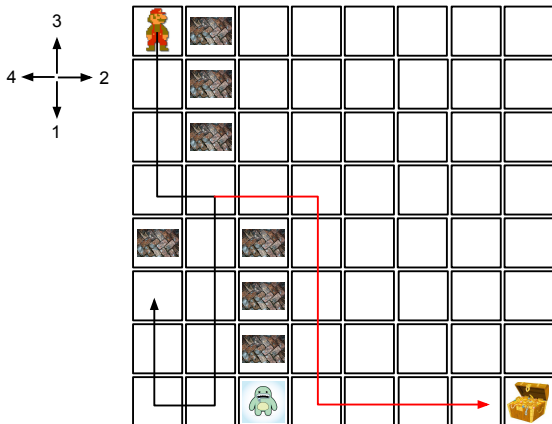
Търсене



Търсене



Търсене



Модел на лабиринта

```
const int ls = 8;  
int lab[ls][ls] = {0,1,0,0,0,0,0,0,  
                  0,1,0,0,0,0,0,0,  
                  0,1,0,0,0,0,0,0,  
                  0,0,0,0,0,0,0,0,  
                  1,0,1,0,0,0,0,0,  
                  0,0,1,0,0,0,0,0,  
                  0,0,1,0,0,0,0,0,  
                  0,0,2,0,0,0,0,0};
```

Модел на лабиринта

```
void markVisited (int lab[ls][ls], int row, int col)
{
    lab[row][col] = 9;
}

bool canStepOn (int lab[ls][ls], int row, int col)
{
    return row >= 0 &&
           col >= 0 &&
           row < ls &&
           col < ls &&
           lab[row][col] == 0;
}
```

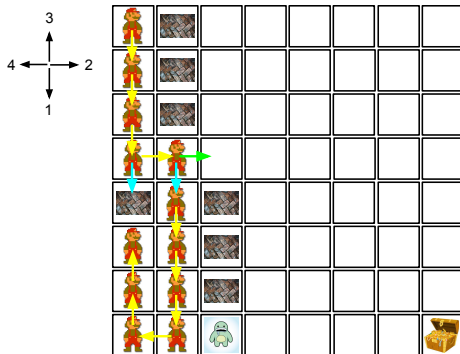
Проби и грешки

```
bool wayExists (int lab[ls][ls], int startRow, int startCol)
{
    if (startRow == ls-1 && startCol == ls-1)
    { return true; }

    markVisited (lab, startRow, startCol);

    if (canStepOn (lab, startRow+1, startCol) &&
        wayExists (lab, startRow+1, startCol))
        return true;
    if (canStepOn (lab, startRow, startCol+1) &&
        wayExists (lab, startRow, startCol+1))
        return true;
    if (canStepOn (lab, startRow-1, startCol) &&
        wayExists (lab, startRow-1, startCol))
        return true;
    if (canStepOn (lab, startRow, startCol-1) &&
        wayExists (lab, startRow, startCol-1))
        return true;
    return false;
}
```

Проби и грешки



```

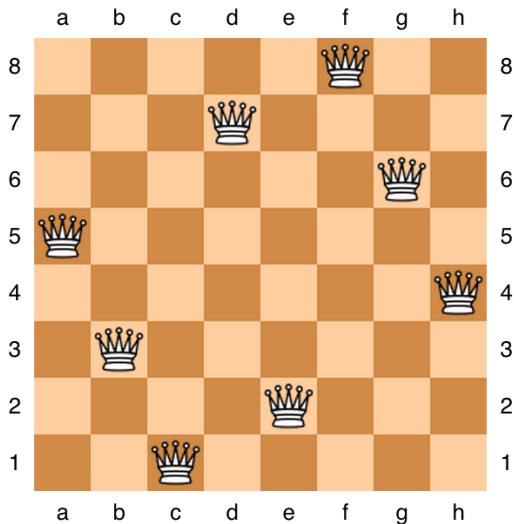
if (canStepOn (lab, startRow+1, startCol) &&
    wayExists (lab, startRow+1, startCol))
    return true;
if (canStepOn (lab, startRow, startCol+1) &&
    wayExists (lab, startRow, startCol+1))
    return true;

```

...

Пъзел с 8 царици

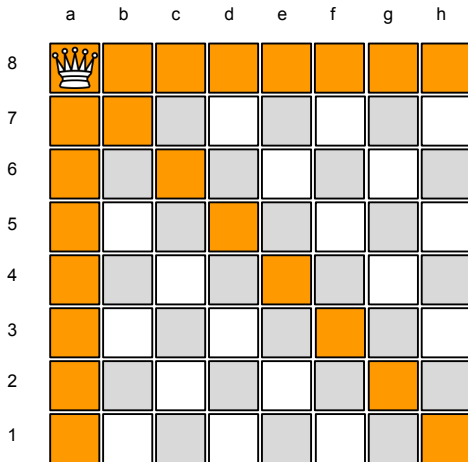
Задачата



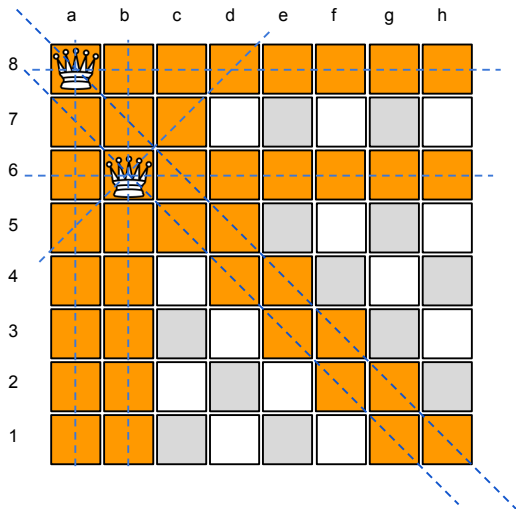
Търсене

	a	b	c	d	e	f	g	h
8		■		■		■		■
7	■		■		■		■	
6		■		■		■		■
5	■		■		■		■	
4		■		■		■		■
3	■		■		■		■	
2		■		■		■		■
1	■		■		■		■	

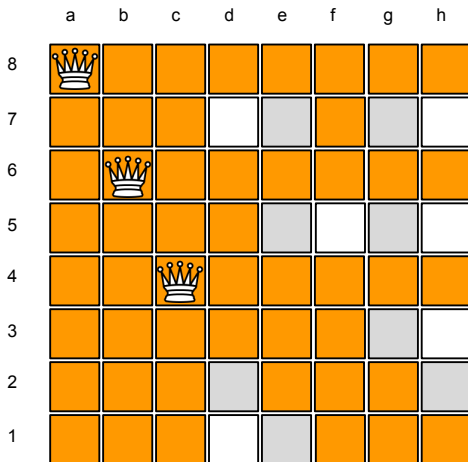
Търсене



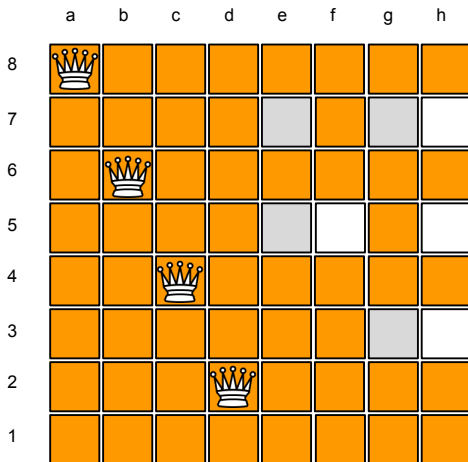
Търсене



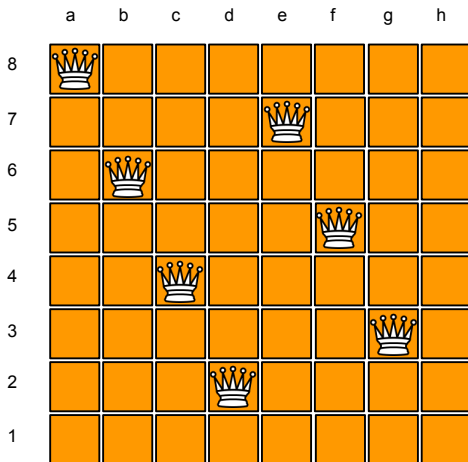
Търсене



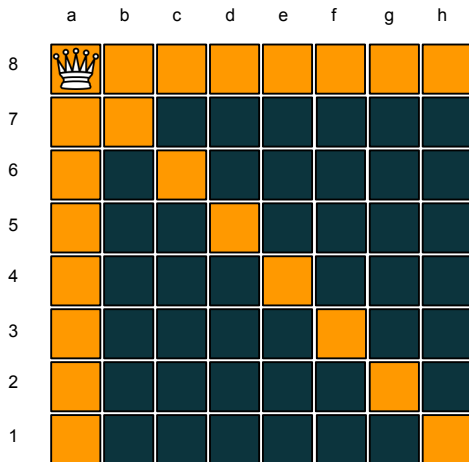
Търсене



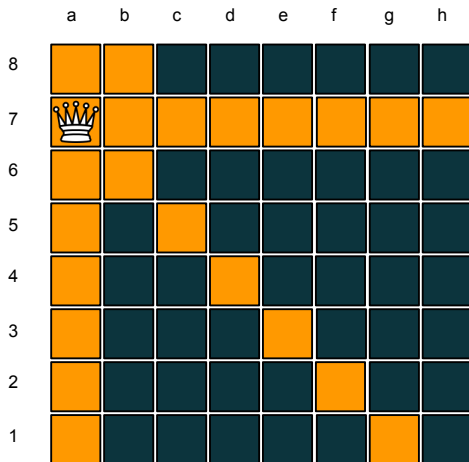
Търсене



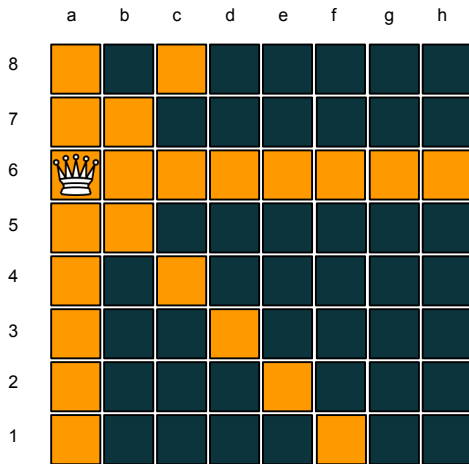
Подзадача



Търсене



Търсене



Модел на играта

```
const int bs = 5;  
bool board[bs][bs] = {false};
```

Модел на играта

```
void placeQueen (bool board[bs][bs], int row, int col)
{
    board[row][col] = true;
}
bool canPlaceQueen (bool board[bs][bs], int row, int col)
{
    for (int count = 0; count < bs; count++)
    {
        if (board[row][count] || board[count][col])
            return false;
    }
    //.....
}
```

(Обхождане на обратния диагонал)

			(r-1,c+1)
		(r,c)	
	(r+1,c-1)		
(r+2,c-2)			

$$(row + i, col - i), i = -1, \dots, 2$$

(Обхождане на обратния диагонал)

			(r-1,c+1)
		(r,c)	
	(r+1,c-1)		
(r+2,c-2)			

$$row + i > 0$$

$$row + i < 4$$

$$col - i > 0$$

$$col - i < 4$$

$$i > -row$$

$$i < 4 - row$$

$$i < col$$

$$i > col - 4$$

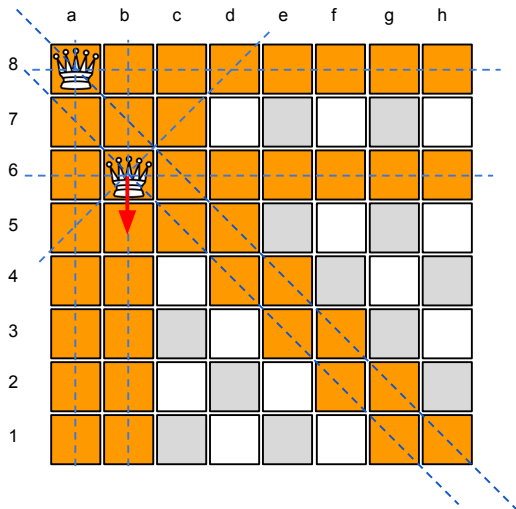
$$i = \max(-row, col - 4), \dots, \min(col, 4 - row)$$

Модел на играта

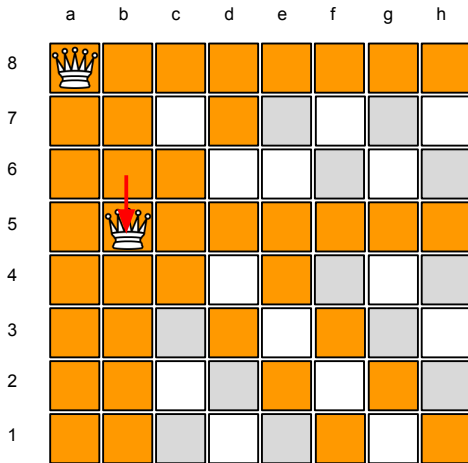
$$i = \max(-row, col - bs), \dots, \min(col, bs - row)$$

```
//.....
for (int count = -min(row, col);
     count < min (bs-row, bs-col);
     count++)
{
    if (board[row+count][col+count])
        return false;
}
for (int count = max(-row, col-bs);
     count < min (col, bs-row);
     count++)
{
    if (board[row+count][col-count])
        return false;
}
return true;
}
```

Връщане назад



Връщане назад



Връщане назад

```
void replaceQueen (bool board[bs][bs], int row, int col)
{
    board[row][col] = false;
}
```

Проби и грешки

```
void placeQueens (bool board[bs][bs], int number)
{
    if (number == 0)
    {
        printBoard(board);
        return;
    }
    for (int row = 0; row < bs; row++)
    {
        for (int col = 0; col < bs; col++)
        {
            if (canPlaceQueen (board,row,col))
            {
                placeQueen (board,row,col);
                placeQueens (board,number-1);
                replaceQueen (board,row,col);
            }
        }
    }
}
```

Благодаря за вниманието!