

# Domino Magic Threefold Problem Set

Andreas Bärtschi, Daniel Graf, Przemek Uznański

ETH Zürich

December 7, 2016

**Fetch two sheets of paper at the entrance!**

# Threefold Problem Set

**Goal:** simulate the thinking steps of a six hour exam in one hour.

## First Hour: (17:15-18:00)

- 3 problems on paper
- think about them
- sketch your solution on paper
- no coding required
- provide us with some feedback

## Break: (18:00-18:15)

- Q &A session

## Second Hour: (18:15-18:45)

- solution discussion
- feedback collection

**Fetch two sheets of paper at the entrance!**

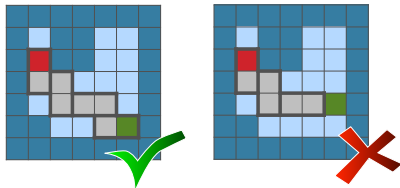
# Domino Snake – The problem

Given:

- $h \times w$  grid with obstacles
- $p$  queries of point pairs  $((q, r), (s, t))$

Wanted:

- y or n per query:  
Does a domino snake between the two points exist?



Rephrased in graph theory language:

What corresponds to a *domino snake*?

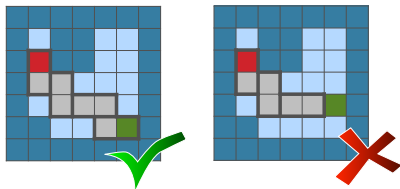
- a  $(q, r)$ - $(s, t)$ -path on the 4-neighborhood grid graph with holes
- this path needs to have even length (i.e. an even number of vertices)

Is even length necessary and sufficient?

- *necessary*: odd length paths can not be tiled into dominos of area 2.
- *sufficient*: a path  $P = (p_1, p_2, \dots, p_l)$  of even length  $l = 2k$  can always be tiled into dominoes of the form  $(p_1, p_2), (p_3, p_4), \dots, (p_{l-1}, p_l)$ .

# Domino Snake – Handling a single query

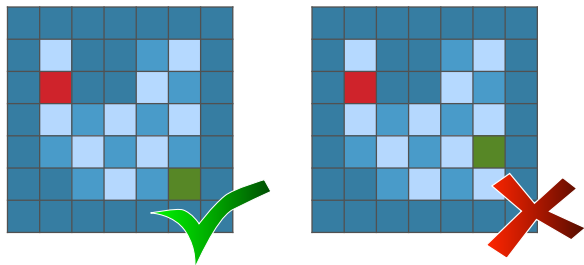
BFS can answer if any path from  $(q, r)$  to  $(s, t)$  exists and if possible gives us the shortest such path.



What if this path is of odd length?

Can we add a detour to find a path of the right parity?

No! Chessboard coloring argument.  
We require:  $(q + r) \not\equiv_2 (s + t)$



## Domino Snake – Handling multiple queries

$p = 1$ : check that BFS from  $(q, r)$  visits  $(s, t)$  and  $(q + r) \not\equiv_2 (s + t)$ .

$p > 1$ :

- Running BFS over and over is expensive  $\rightarrow \mathcal{O}(hwp)$
- We do not really need the shortest path.
- We just ask: Are  $(q, r)$  and  $(s, t)$  in the same connected component?
- Precompute the connected components in  $\mathcal{O}(hw)$ .
- Each of the  $p$  queries can then be answered in  $\mathcal{O}(1)$  by checking
  - $\text{component}((q, r)) \stackrel{?}{=} \text{component}((s, t))$
  - $(q + r) \not\equiv_2 (s + t)$ .

Overall runtime:  $\mathcal{O}(hw + p)$

# New Tiles – The problem

## Problem

Given a  $h \times w$  matrix of 0's and 1's.

Find the maximum number of non-overlapping  $2 \times 2$  matrices of the form:

$$\begin{matrix} 1 & 1 \\ 1 & 1 \end{matrix}$$

Is there a greedy solution, ideas?

Greedily take every  $2 \times 2$  free space.

Counterexample:

0110

1111

1111

Greedy gives answer 1. Maximum is 2.

(We ignore the zeros at the boundary from now on.)

## New Tiles – Dynamic programming

**State of the subproblem:**  $[i, bitmask]$ ,  $1 \leq i \leq h$ ,  $0 \leq bitmask \leq 2^w - 1$

$DP[i][bitmask]$  = maximum number of 2x2 matrices possible, if we consider:

- the first  $i - 1$  rows completely,
- in the  $i$ -th row only the cells of column  $j$  with  $j$ -th bit of the  $bitmask$  set.

### Initialization:

- In a single row, no tiles fit:  $DP[1][bitmask] = 0$ , for any  $bitmask$
- If we do not use the new row at all:  $DP[i][\underbrace{00 \dots 0}_w] = \max_{bitmask} (DP[i - 1][bitmask])$
- We might always leave a square of the new row empty:  
$$DP[i][bitmask] = \max_{j \in \{j \mid j\text{-th bit is set in } bitmask\}} DP[i][bitmask \text{ with } j\text{-th bit unset}]$$
- If we compute the entries in lexicographical order, we avoid all dependencies.

# New Tiles – Dynamic programming

## Recurrent formula for placing new tiles:

We have to take some  $2 \times 2$  matrices in the  $i - 1, i$  strip

1	1	1	1	0	0	1	0	0	0	0	1	1	0	0	lookup in row $i-1$
0	0	0	0	1	1	0	1	1	1	1	0	0	1	1	state in row $i$

0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	row $i-1$ of the input
0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	row $i$ of the input

Try only those *bitmasks* that have even number of consecutive 1s.

All others are covered by the initialization (by a case with some of the bits removed).

Check whether this maximal amount of  $2 \times 2$  matrices is compatible with the input matrix. If yes, check whether this would lead to a larger tiling.



# New Tiles – Dynamic programming

## Recurrent formula for placing new tiles:

We have to take some  $2 \times 2$  matrices in the  $i - 1, i$  strip

1	1	1	1	0	0	1	0	0	0	0	1	1	0	0	lookup in row $i-1$
0	0	0	0	1	1	0	1	1	1	1	0	0	1	1	state in row $i$
0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	row $i-1$ of the input
0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	row $i$ of the input

For the example:

$$DP[i][000011011110011] = \max(DP[i][000011011110011], \\ DP[i-1][111100100001100] + 4).$$

# New Tiles – Dynamic programming

## Recurrent formula for placing new tiles:

We have to take some  $2 \times 2$  matrices in the  $i - 1, i$  strip

1	1	1	1	0	0	1	0	0	0	0	1	1	0	0	lookup in row $i-1$
0	0	0	0	1	1	0	1	1	1	1	0	0	1	1	state in row $i$

0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	row $i-1$ of the input
0	1	1	1	1	1	0	1	1	1	1	0	1	1	1	row $i$ of the input

In general: if the bitmask fits in row  $i$  and  $i - 1$  of the input, perform this step:

$$DP[i][bitmask] = \max(DP[i][bitmask], \\ DP[i - 1][\text{negated } bitmask] + \text{bitcount}(bitmask)/2)$$

## New Tiles – Dynamic programming

### **Runtime:**

Size of the table:  $2^w \cdot h$ .

Update step per entry: two loops over  $w$  (one in the initialization, one for the check).

Runtime:  $\mathcal{O}(2^w \cdot h \cdot w)$ , which is fast enough.

### **Slower solution for first subtask:**

For smaller  $w$ , we could also check for a given bitmask all of its subsets, resulting in running time  $\mathcal{O}(3^w \cdot w \cdot h)$ .

### **Implement it yourself:**

Try it out! This task is available on the judge.

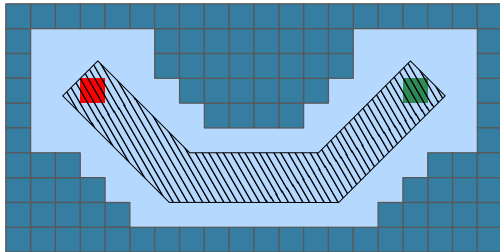
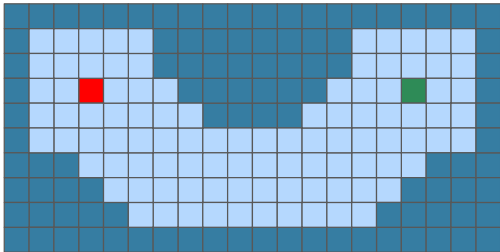
# Snakes strike back – The problem

Given:

- $h \times w$  grid with obstacles (snake cages) and entrance/exit pair.
- Minimum path width  $p$ , safety distance  $p/2$ . (Cases:  $p = 1$ ,  $p = 2$ ,  $p \leq 20$ )

Wanted:

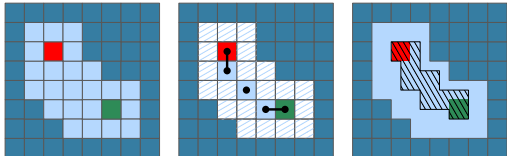
- yes or no: Is there a path between the entrance and the exit with minimum width  $p$  which keeps clear from obstacles by  $p/2$ ?



# Snakes strike back – Case $p = 1$

Rough Idea:

Use graph from Domino Snake, but delete squares which are adjacent to snake cages.



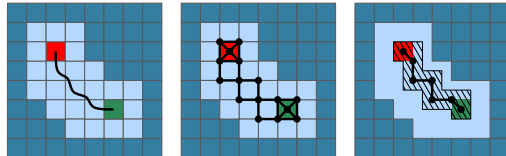
**Problem:** Fails, graph is disconnected!

**Need** better grasp on path width  $p$  and safety distance  $p/2$ .

**Equivalent:** **Curve** (has 0 width!),  
Safety distance:  $p$ .

Solution Approach:

If there is a curve: Shift and bend, such that it runs along the boundary of cells which are not adjacent to snake cages.



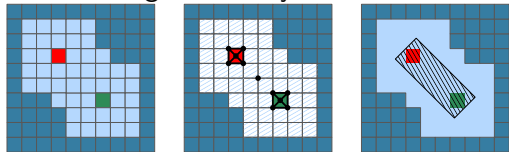
**Solution ( $p = 1$ ):**

BFS on grid graph given by vertices and by boundary segments with distance  $\geq 1$  to snake cage boundaries.

Can we adapt for larger  $p$ ?

## Snakes strike back – Case $p = 2$

Rough Idea (first case solution adapted):  
BFS on grid graph given by vertices and  
by boundary segments with distance  $\geq 2$   
to snake cage boundaries.

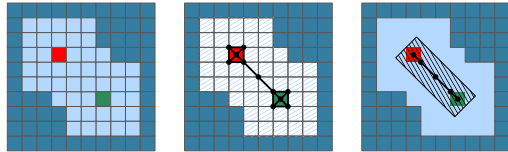


**Problem:** When can we pass between  
two obstacles / their vertices?

- $\Delta x$  or  $\Delta y$  at least  $2p$ , or
- $\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq 2p$ .

Solution Approach:

Because  $p = 2$ , second case only appears  
for  $\Delta x = \Delta y = 3$ . Hence add diagonals  
between vertices  $u, v$  of the same cell.



**Solution ( $p = 2$ ):**

BFS on grid graph given by

- vertices and boundary segments  
with safety distance  $\geq 2$ ,
- diagonals between vertices if they  
belong to the same cell.

## Snakes strike back – Arbitrary $p$

**Question:** Can we extend the  $p = 2$  solution to arbitrary  $p$ ?

**Recall:** Main problem for  $p = 2$  were obstacles which lie diagonally apart, i.e. pairs  $(\Delta x, \Delta y)$  such that  $\sqrt{(\Delta x)^2 + (\Delta y)^2} \geq 2p$ , but  $\Delta x, \Delta y < 2p$ .

**Answer:** For  $p > 2$  there will be many different kinds of pairs  $(\Delta x, \Delta y)$ , not only one! Things start to get messy!

Look at the problem statement again:

**Originally:** Is there a path of width  $p$  which keeps clear from obstacles by  $p/2$ ?

**Redefined:** Is there a path of width 0 which keeps clear from obstacles by  $p$ ?

**Redefinition II:** Is there a path of width  $2p$  which keeps clear from obstacles by 0?

# Snakes strike back – Arbitrary $p$

**Draw path with a brush of Diameter  $2p$ .**

This sounds familiar:

H1N1 – How to move a disk  $D$  without colliding with a given point set  $P$ ?

Move disk along Voronoi Diagram edges / in the Delaunay triangulation.

Problem: Here our obstacles are cells, not points.

Solution: Replace every snake cage square by its 4 vertices.

**Careful:**

- We need vertices of each square!  
(Only considering the convex & concave vertices of the boundary of the full obstacle is not enough.)

