

CS 4820

Design & Analysis of Algorithms

Instructors

Prof. Michael Kim (me)

Prof. Bobby Kleinberg

Plan for today 22 Jan 2024

① Motivation for 4820

② Brief Administrivia

③ Case Study: Tiling Problems

# ① Motivation

Algorithms is the study of how to solve computational problems

## Computational Problem Examples

- \* Given a road map, find the shortest path from A to B
- \* Given a list of integers, return them in sorted order.
- \* Given a social network, find a large clique  
(mutually-connected accounts)

# ① Motivation

Algorithms is the study of how to solve computational problems

## Computational Problem Examples

- \* Given a road map, find the shortest path from A to B
- \* Given a list of integers, return them in sorted order.
- \* Given a social network, find a large clique  
(mutually-connected accounts)

Can we solve these problems? Can we solve them efficiently?

# ① Motivation

Algorithms is the study of how to solve computational problems

## Computational Problem Examples

- \* Given a road map, find the shortest path from A to B
- \* Given a list of integers, return them in sorted order.
- \* Given a social network, find a large clique  
(mutually-connected accounts)

Can we solve these problems? Can we solve them efficiently?

An algorithm is a well-defined sequence of steps  
(i.e., a recipe) to solve a problem.

Why 4820?

\* It's a requirement.

→ Why? Algorithms show up in every area of CS.

Why 4820?

\* It's a requirement.

→ Why? Algorithms show up in every area of CS.

\* The "algorithmic lens" on other fields

→ Algorithms give new perspectives in  
Econ, Bio, Physics, Sociology

# Why 4820?

\* It's a requirement.

→ Why? Algorithms show up in every area of CS.

\* The "algorithmic lens" on other fields

→ Algorithms give new perspectives in  
Econ, Bio, Physics, Sociology

\* Algorithmic Thinking is problem solving

→ Designing algorithms requires creativity

→ Analyzing algorithms requires clarity of thought

↓  
Precise definitions &  
mathematical proofs.

# Guiding Philosophy

For many computational problems there is a simple, but inefficient algorithm.



# Guiding Philosophy

For many computational problems there is a simple, but inefficient algorithm.

eg. Shortest Path from A to B:

- Iterate through every path in the road map.
- If the path connects A to B,
  - ↳ record the path and its distance
- Return the A→B path of minimum distance

# Guiding Philosophy

For many computational problems there is a simple, but inefficient algorithm.

eg. Shortest Path from A to B:

- Iterate through every path in the road map.
- If the path connects A to B,
  - ↳ record the path and its distance
- Return the A→B path of minimum distance

Concern. There are exponentially many paths in a map.

# Guiding Philosophy

For many computational problems there is a simple, but inefficient algorithm.

eg. Shortest Path from A to B:

- Iterate through every path in the road map.
- If the path connects A to B,  
↳ record the path and its distance
- Return the A→B path of minimum distance

Concern. There are exponentially many paths in a map.

Key Question: Can we do better?

## ② Intermission: Administrivia

All information @ course website

[cs.cornell.edu/courses/cs4820/2024sp](https://cs.cornell.edu/courses/cs4820/2024sp)

↳ Get on Ed Discussions  
& Gradescope

## Course Structure

\* "Required" Lecture MWF

\* Homework (Roughly) Weekly

\* Exams

— Prelim #1, 20 Feb

— Prelim #2, 11 Apr

— Final Exam, Finals Week TBD.

} 0% → XC

} 25%

} 75%

## Prerequisites.

### \* CS 2800 .

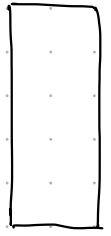
- Discrete math (sets, graphs, basic probability)
- Formal proofs (induction, contradiction, basic logic)

### \* CS 3110

(or A-in CS 2110)

- Big-O Notation (for runtime analysis)
- Basic Data Structures (lists, queues, stacks, arrays, hash tables)

### ③ Case Study in Computational Problems: Tiling Problems



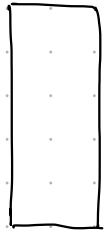
Shape  
 $S$



Tile  
 $t$

Definition. A tiling of a shape  $S$  with tile  $t$  is an arrangement of non-overlapping copies of  $t$  that completely covers  $S$ .

### ③ Case Study in Computational Problems: Tiling Problems



Shape  
S



Tile  
t

Q: Can S be tiled w/ t?

Definition. A tiling of a shape S with tile t is an arrangement of non-overlapping copies of t that completely covers S.

### ③ Case Study in Computational Problems: Tiling Problems



Shape  
S



Tile  
t

Q: Can S be tiled w/ t?

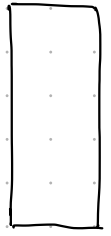
A: Yes!



Definition. A tiling of a shape S with tile t is an arrangement of non-overlapping copies of t that completely covers S.



### ③ Case Study in Computational Problems: Tiling Problems



Shape  
S



Tile  
t

Q: Can S be tiled w/ t?

A: Yes!



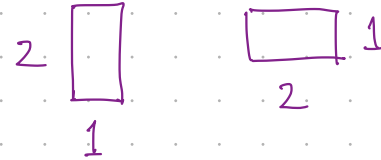
Definition. A tiling of a shape S with tile t is an arrangement of non-overlapping copies of t that completely covers S.

The t-Tiling Problem.

Given Shape S, can S be tiled with t?

# Example #1. Domino-Tiling

\* Dominos are  $2 \times 1$  tiles

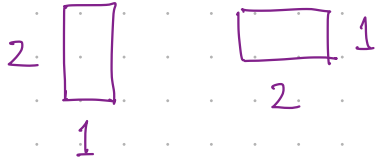


The Domino-Tiling Problem.

Given Shape  $S$ , can  $S$  be tiled with Dominos?

# Example #1. Domino-Tiling

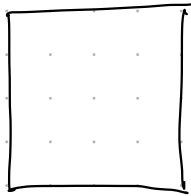
\* Dominos are  $2 \times 1$  tiles



The diagram shows two purple-outlined rectangles representing dominoes. The first is a vertical rectangle with a height of 2 and a width of 1. The second is a horizontal rectangle with a height of 1 and a width of 2.

## The Domino-Tiling Problem.

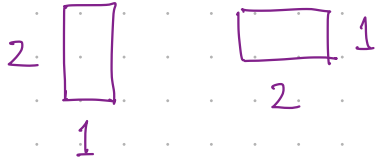
Given Shape  $S$ , can  $S$  be tiled with Dominos?



$S$ :  $4 \times 4$  square

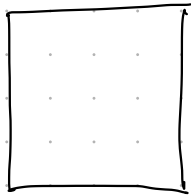
# Example #1. Domino-Tiling

\* Dominos are  $2 \times 1$  tiles



## The Domino-Tiling Problem.

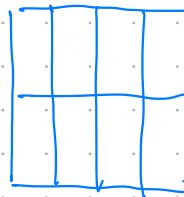
Given Shape  $S$ , can  $S$  be tiled with Dominos?

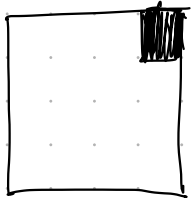


$S: 4 \times 4$  square

Claim.  $S$  can be Domino-tiled.

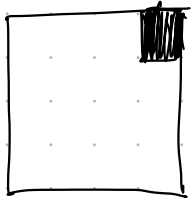
Pf.





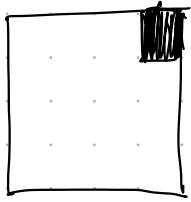
$S'$ :  $4 \times 4$  square w/ corner removed

Can  $S'$  be Domino-tiled?



$S'$ :  $4 \times 4$  square w/ corner removed

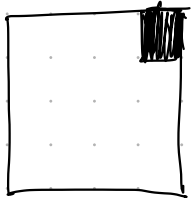
Claim.  $S'$  CANNOT be Domino-Tiled.



$S'$ :  $4 \times 4$  square w/ corner removed

Claim.  $S'$  CANNOT be Domino-Tiled.

Pf. By Parity Argument.



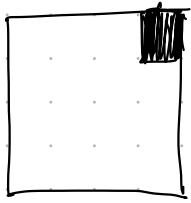
$S'$ :  $4 \times 4$  square w/ corner removed

Claim.  $S'$  CANNOT be Domino-Tiled.

Pf. By Parity Argument.

- Every Domino covers exactly 2 squares.



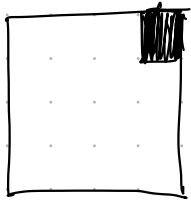


$S'$ :  $4 \times 4$  square w/ corner removed

Claim.  $S'$  CANNOT be Domino-Tiled.

Pf. By Parity Argument.


- Every Domino covers exactly 2 squares.
- By non-overlapping property, every shape that can be Domino-Tiled has an even number of squares.

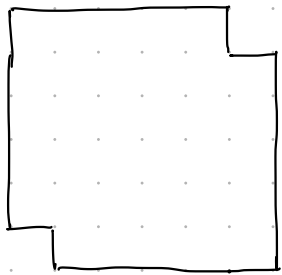


$S'$ :  $4 \times 4$  square w/ corner removed

Claim.  $S'$  CANNOT be Domino-Tiled.

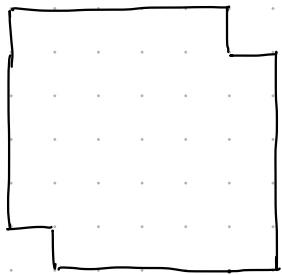
Pf. By Parity Argument.

- Every Domino covers exactly 2 squares.
- By non-overlapping property, every shape that can be Domino-Tiled has an even number of squares.
- But  $S'$  has an odd (15) number of squares. 



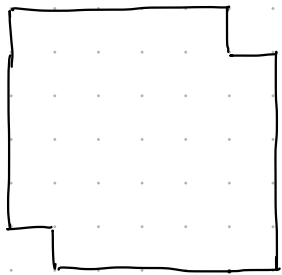
$S''$  6x6 square w/ opposing corners removed.

Can  $S''$  be Domino-Tiled?



$S''$  6x6 square w/ opposing corners removed.

Claim:  $S''$  CANNOT be Domino-Tiled.

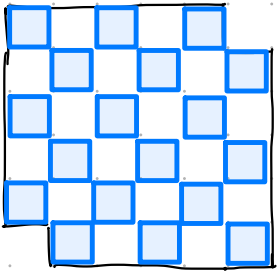


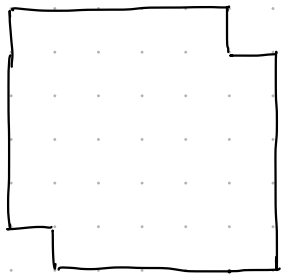
$S''$  6x6 square w/ opposing corners removed.

Claim:  $S''$  CANNOT be Domino-Tiled.

Pf. By coloring argument.

— Consider coloring the squares of  $S$  w/ white & blue  
s.t. adjacent squares are of different colors.



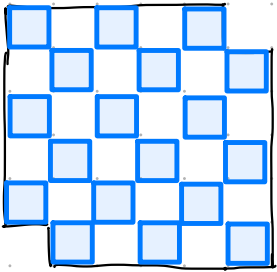


$S''$  6x6 square w/ opposing corners removed.

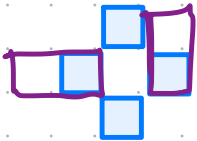
Claim:  $S''$  CANNOT be Domino-Tiled.

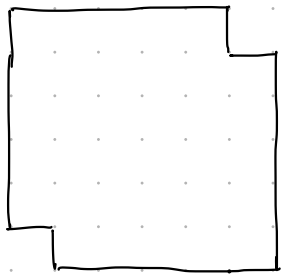
Pf. By coloring argument.

— Consider coloring the squares of  $S$  w/ white & Blue s.t. adjacent squares are of different colors.



— Every Domino covers exactly 1 white & 1 Blue square



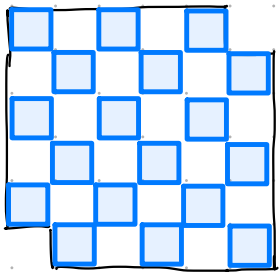


$S''$  6x6 square w/ opposing corners removed.

Claim:  $S''$  CANNOT be Domino-Tiled.

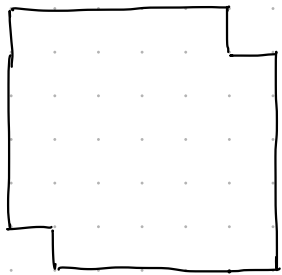
Pf. By coloring argument.

— Consider coloring the squares of  $S$  w/ white & blue s.t. adjacent squares are of different colors.



— Every Domino covers exactly 1 white & 1 Blue square

— Thus, every shape that can be Domino-Tiled has an equal number of white & blue squares.

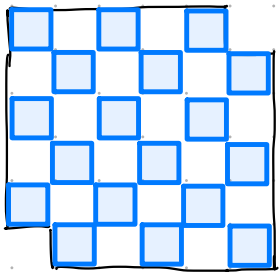


$S''$  6x6 square w/ opposing corners removed.

Claim:  $S''$  CANNOT be Domino-Tiled.

Pf. By coloring argument.

— Consider coloring the squares of  $S$  w/ white & blue s.t. adjacent squares are of different colors.



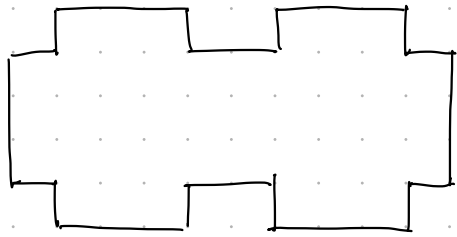
— Every Domino covers exactly 1 white & 1 Blue square

— Thus, every shape that can be Domino-Tiled has an equal number of white & Blue squares.

— But  $S''$  has 16 white squares &  
18 Blue squares

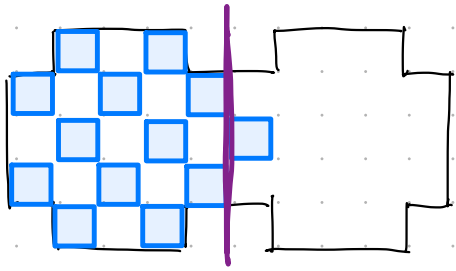




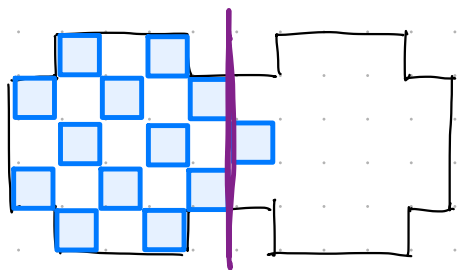


$S_{++}$

Can  $S_{++}$  be Domino-Tiled?

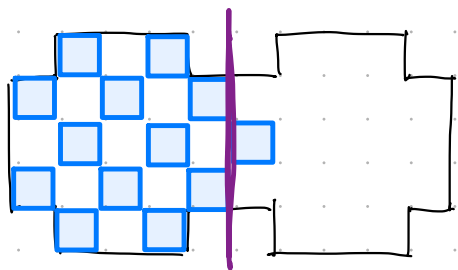


Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .



Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

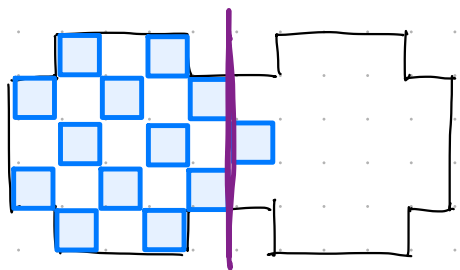
Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.



Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.

Pf. By contradiction. Suppose  $S$  has a Domino-Tiling.

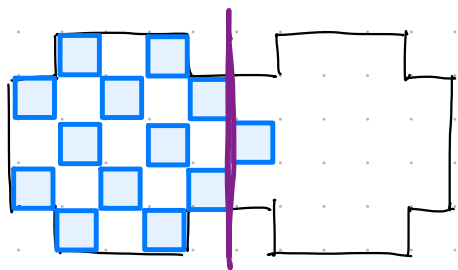


Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.

Pf. By contradiction. Suppose  $S$  has a Domino-Tiling.

— To cover every Blue square in  $B$ ,  $|B|$  Dominos must be used

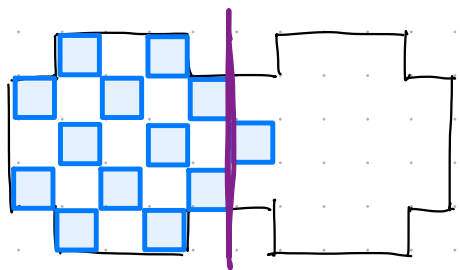


Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.

Pf. By contradiction. Suppose  $S$  has a Domino-Tiling.

- To cover every Blue square in  $B$ ,  $|B|$  Dominos must be used
- Each of these dominos also covers an adjacent White square.




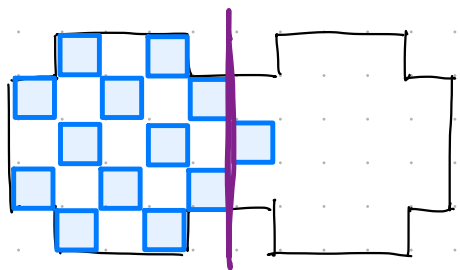
Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.

Pf. By contradiction. Suppose  $S$  has a Domino-Tiling.

- To cover every Blue square in  $B$ ,  $|B|$  Dominos must be used
- Each of these dominos also covers an adjacent White square.
- But if  $|W| < |B|$ , then by the Pigeonhole Principle some Dominos must share a White square, violating the non-overlapping property.

Contradiction! 




Definition. A pigeonhole obstruction is a set of Blue squares  $B$ , with adjacent White squares  $W$  s.t.  $|W| < |B|$ .

Lemma. If a shape  $S$  has a pigeonhole obstruction, then it cannot be Domino-Tiled.

Pf. By contradiction. Suppose  $S$  has a Domino-Tiling.

- To cover every Blue square in  $B$ ,  $|B|$  Dominos must be used
- Each of these dominos also covers an adjacent White square.
- But if  $|W| < |B|$ , then by the Pigeonhole Principle some Dominos must share a White square, violating the non-overlapping property.

Contradiction! 

Corollary.  $S_{++}$  cannot be Domino-Tiled.



## The Domino-Tiling Problem.

Given Shape  $S$ , can  $S$  be tiled with Dominos?

Obstruction Lemma. If  $S$  contains a pigeonhole obstruction, then it cannot be Domino-Tiled.

## The Domino-Tiling Problem.

Given Shape  $S$ , can  $S$  be tiled with Dominos?

Obstruction Lemma. If  $S$  contains a pigeonhole obstruction, then it cannot be Domino-Tiled.

Reverse Obstruction Lemma. (Deep, Non-trivial fact!)

If  $S$  cannot be Domino-Tiled, then it must contain a pigeonhole obstruction.

## The Domino-Tiling Problem.

Given Shape  $S$ , can  $S$  be tiled with Dominos?

Theorem. There is an efficient (i.e., polynomial-time) algorithm that solves the Domino-Tiling Problem.

Namely, Given shape  $S$ , the algorithm either

(a) Returns a valid Domino-Tiling, OR

(b) Returns a pigeonhole obstruction.