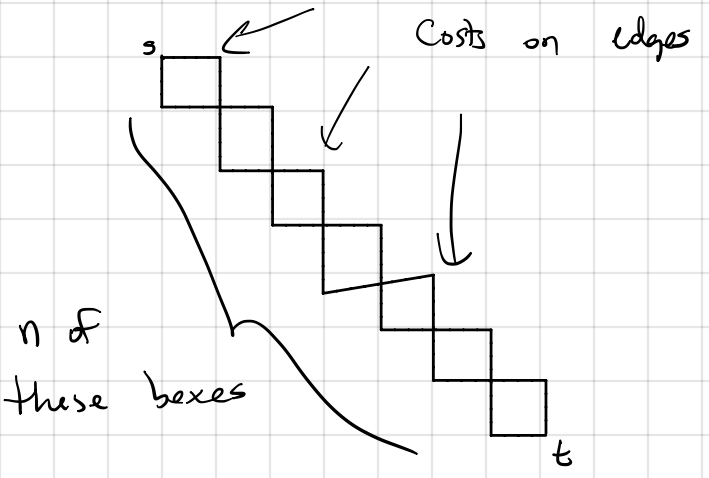
CS 4820: Intro. to Analysis of Algorithms

23 Jan 2019



Find cheapest s-t path

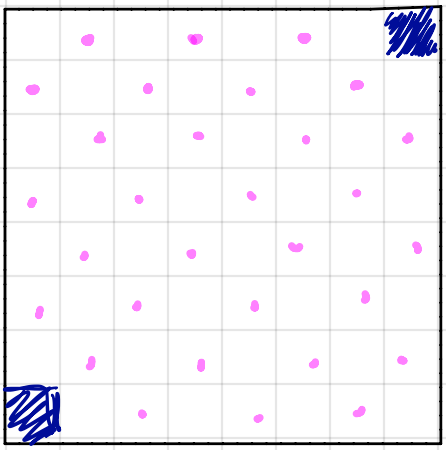
2^n paths from s to t.

Yet it only takes $O(n)$ time to find the shortest one.

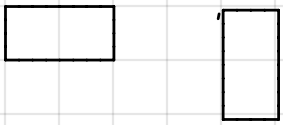
In general graphs with n vertices, m edges,

Dijkstra's algorithm takes $O(m \log n)$ time to find cheapest paths, actually $O(m + n \log n)$ if you the right data structures.

Basic question of algorithms: Why are some computational problems easy to solve rapidly, while others are not?



A "domino" is a rectangular tile made up of 2 squares.

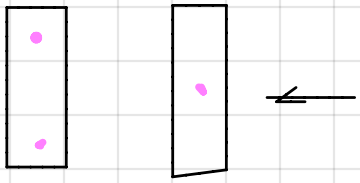


Can you cover this 8×8 square with non-overlapping dominoes?

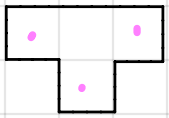
Can you cover all squares except top right corner with non-overlapping dominoes?

No! Non-overlapping dominoes cover an even number of squares. "Junxi obstruction"

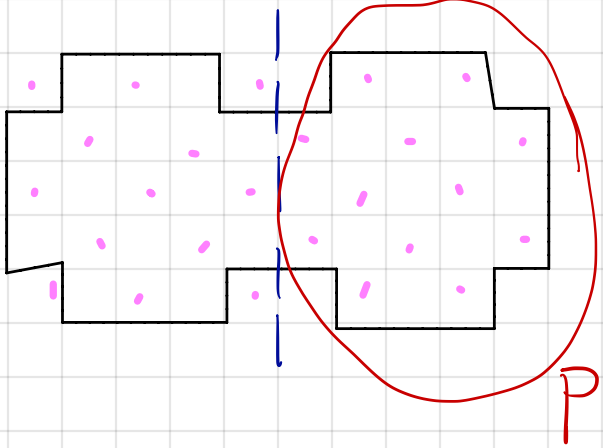
Can you cover all but the top right + bottom left corners?
32 white but only 30 pink squares. Each domino covers an equal number of white and pink squares.
So, impossible! "Daniel obstruction"



← No Daniel obstruction, but no tiling.



← Daniel obstruction.

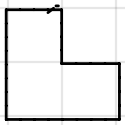


42 squares: 21 pink, 21 white.

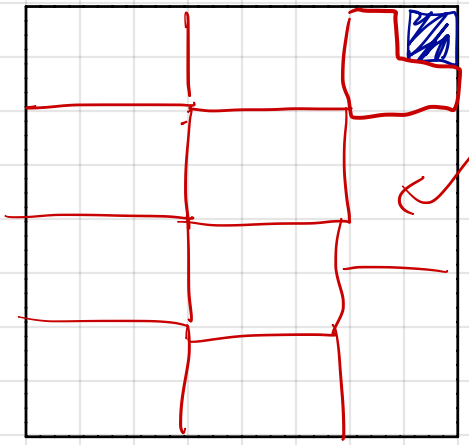
Peter obstruction: a set of pink squares, P, that has strictly fewer than |P| neighboring white squares.

$|P| = 12$, $|\text{Neighbors of } P| = 11$

FACT: For every 2-D domino tiling problem, either ∃ a tiling or ∃ a "Peter obstruction" and there's an efficient algorithm to find one or the other.



An L-shaped tromino.



Tromino tiling problem ("Given a 2-D set of grid squares, can it be covered by non-overlapping trominoes?") is NP-complete.

If an ~~efficient~~ algorithm exists, it can be used as a subroutine to solve MANY other problems believed to be computationally hard.