CS 4820 Algorithm Design and Analysis

www.cs.cornell.edu/courses/cs4820/2021sp/syllabus/

From 2800: sets, graphs, probabilities, methods for writing proofs, e.g. induction, contradiction.

From 3110 (2110): analyzing running time using $O(\cdot)$ notation.

Basic data structures: lists, queues, stacks, arrays. Hash tables. Running time of operations.

CS Homework Partner Finding Social (virtual)

Thurs 2/11 9–10:30pm

RSVP http://bit.ly/cisSP21

2 prelims & final exam

March 18, April 13 (late May)

Available to take on-line.

All lectures will be videorecorded and available for viewing throughout SP21. These handwritten notes will always be uploaded as well.

Most elementary algorithms are brute-force search.

Example: Count the number of Ø's in an array.

Sometimes brute force is too slow.

Example: Sort an array of n elements.
Brute force searches all permutations to find the sorted one. $O(n!)$

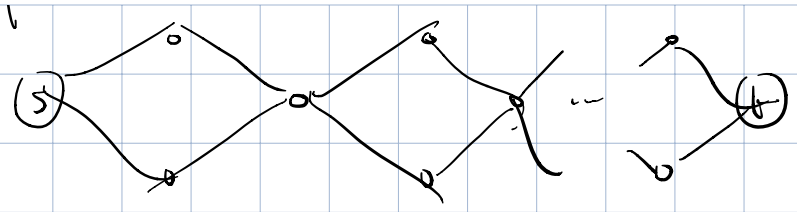
Faster sorting algorithms exists, e.g.

Bubblesort	$O(n^2)$
Mergesort	$O(n \log n)$
Quicksort	$O(n \log n)$ in expectation.
Heapsort	$O(n \log n)$

Example: Given an undirected graph with n vertices, m edges, and given two vertices s & t, is there a path from s to t?

Note: # s-t paths can be exponential in m.

Ex.

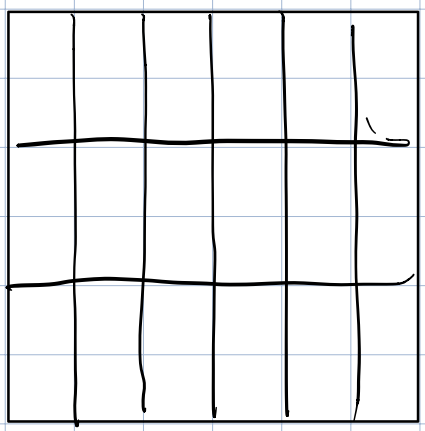


Breadth-first or depth-first search (BFS, DFS) solve this in $O(m)$.

Example. If G has edge lengths ≥ 0 find the minimum length path from s to t.

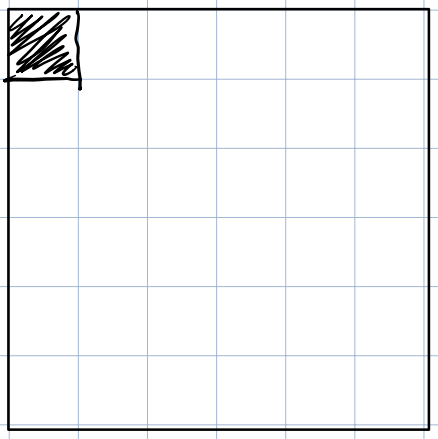
Dijkstra's Alg solves this in $O(m \log n)$ or $O(m + n \log n)$ if using Fibonacci heaps.

Case Study in Complexity of Computational Problems:
TILING PUZZLES!



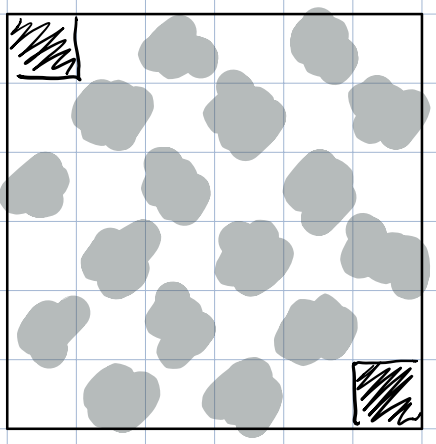
Can this 6×6 square be covered by non-overlapping dominoes?

Yes.



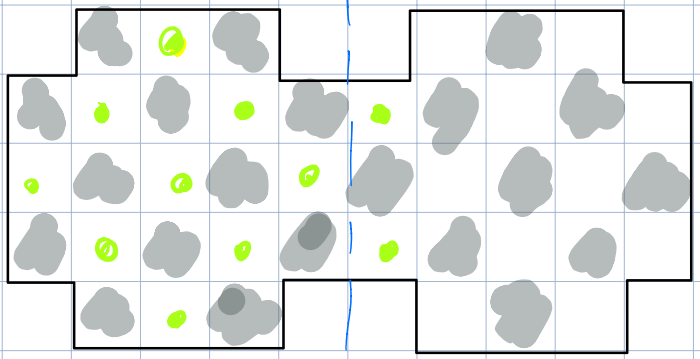
I removed one corner. Can the remaining squares be covered by dominoes?

No, dominoes can't cover odd # of squares.



Now I removed two corners. Can you cover the rest with dominoes?

18 white and 16 black squares. A domino covers one white, one black always.



Now there are 21 black and 21 white squares. Can you cover them all with 21 dominoes?

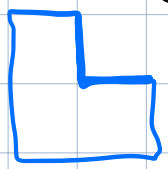
Left of the blue line there are 12 black squares. Each must be paired with a neighboring white square. But there are only 11 of those.

Aubrey obstruction: a set of k squares all of one color such that the # of oppositely colored neighboring squares is $< k$.

Lemma. If a subset of the grid has an Aubrey obstruction then it has no domino tiling.

Theorem. For any subset of the grid either it can be tiled with dominoes or there is an Aubrey obstruction and it can't be tiles.
Furthermore there's an efficient algorithm to find the tiling or obstruction.
(Why? Ch. 7, in March.)

Example: Tromino tiling.



Given a finite subset of grid cells in $\mathbb{Z}^2$, can you cover it with non-overlapping L-shaped trominoes?

This turns out to be NP-complete: any other computational search problem that can be solved by "brute force search and verify" can be efficiently reduced to this one.

Eg. if anyone discovers an efficient algorithm for tiling they can use it to mine Bitcoin efficiently.

(Why? Chapter 8, NP-Completeness.)

Final example. Given finitely many tile shapes and an infinite supply of each, can they tile the infinite grid?

This problem is undecidable: no algorithm to solve it exists, with finite running time.

Every alg either solves some instances incorrectly or sometimes runs forever.