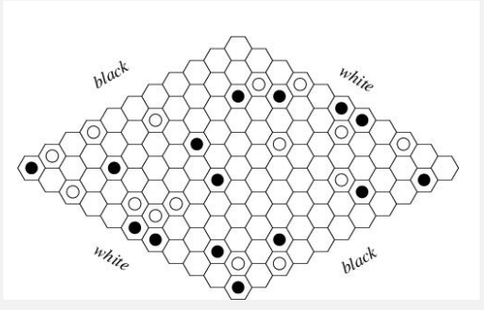
http://algs4.cs.princeton.edu

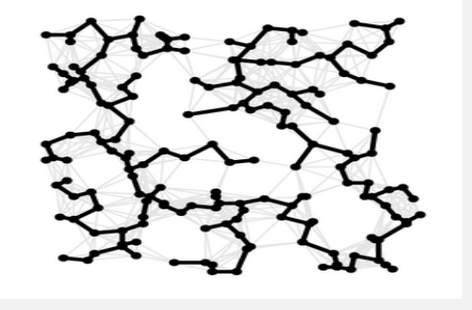
1.5 Union-Find

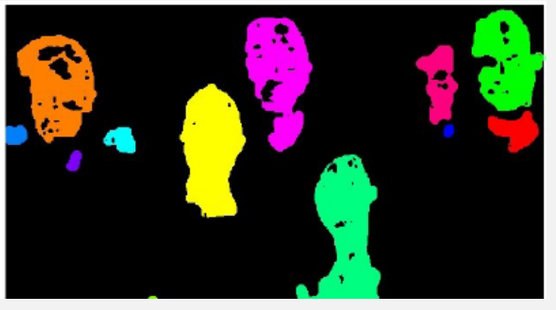
- *dynamic connectivity*
- *quick find*
- *quick union*
- *improvements*
- *applications*

Union-find applications

- Percolation.
- Games (Go, Hex).
- ✓ Dynamic connectivity.
- Least common ancestor.
- Equivalence of finite state automata.
- Hoshen-Kopelman algorithm in physics.
- Hinley-Milner polymorphic type inference.
- Kruskal's minimum spanning tree algorithm.
- Compiling equivalence statements in Fortran.
- Morphological attribute openings and closings.
- Matlab's `bwlabel()` function in image processing.



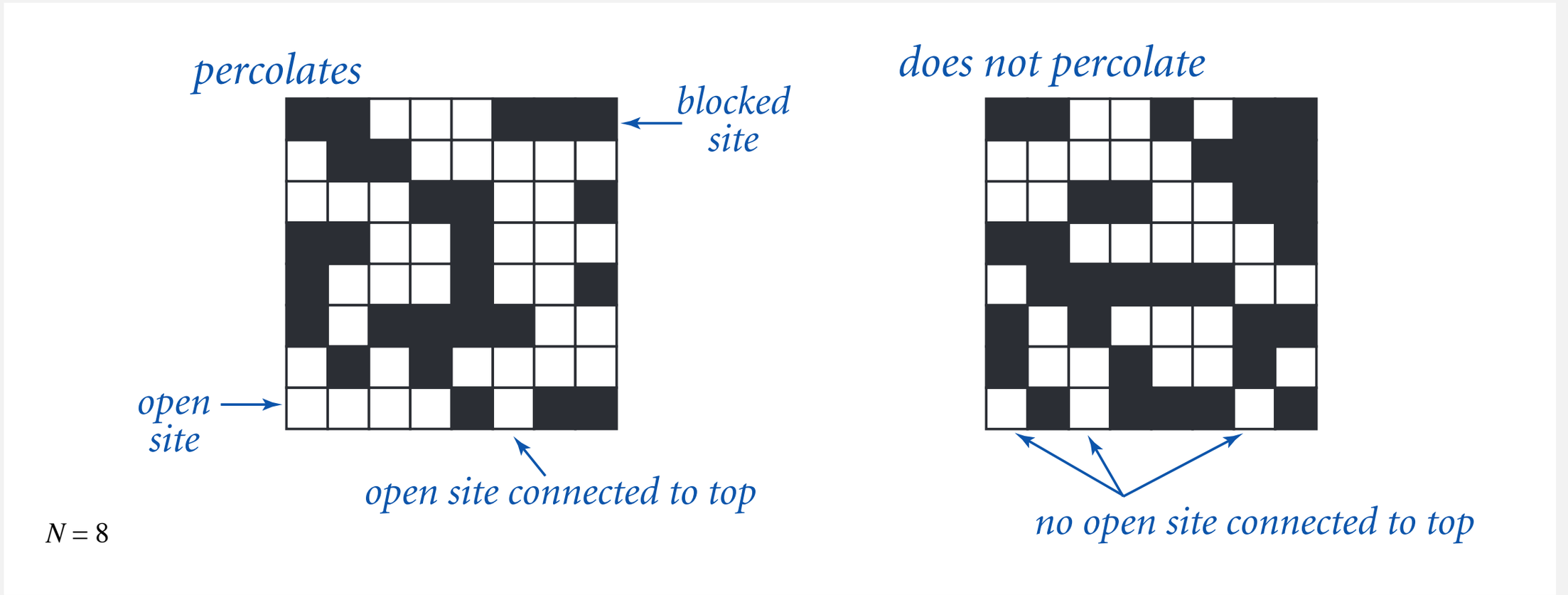




Percolation

An abstract model for many physical systems:

- N-by-N grid of sites.
- Each site is open with probability p (and blocked with probability $1 - p$).
- System percolates iff top and bottom are connected by open sites.



Percolation

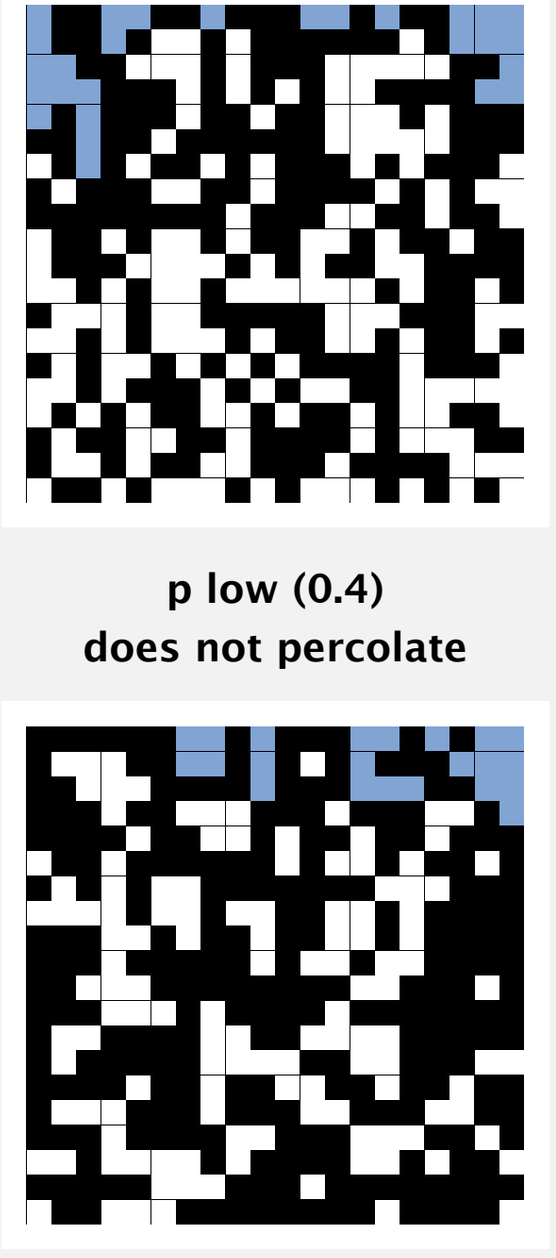
An abstract model for many physical systems:

- N-by-N grid of sites.
- Each site is open with probability p (and blocked with probability $1 - p$).
- System percolates iff top and bottom are connected by open sites.

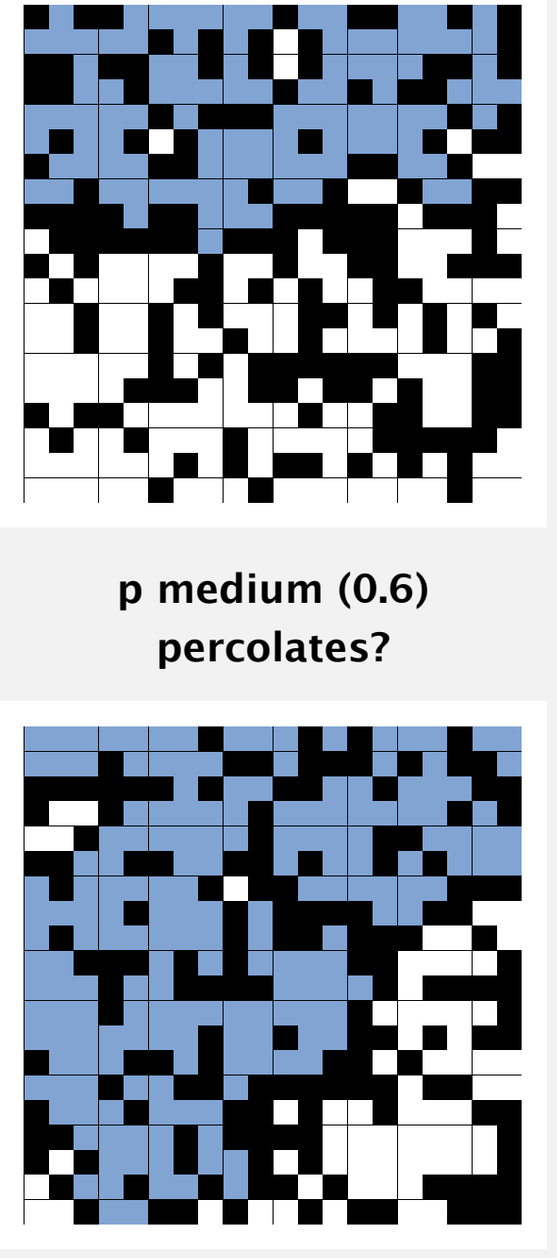
model	system	vacant site	occupied site	percolates
electricity	material	conductor	insulated	conducts
fluid flow	material	empty	blocked	porous
social interaction	population	person	empty	communicates

Likelihood of percolation

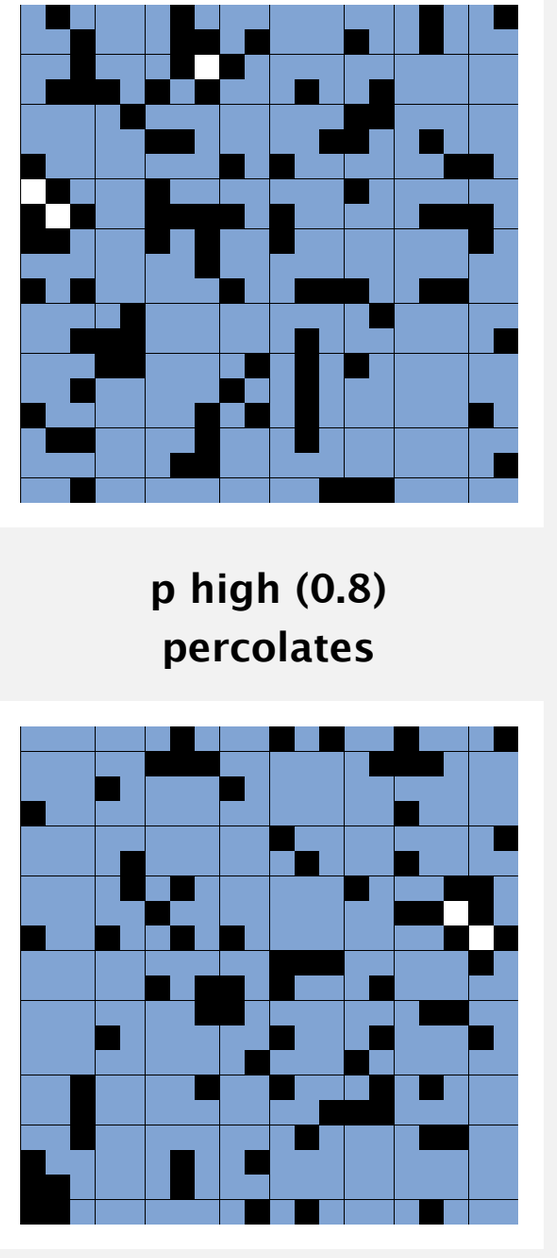
Depends on grid size N and site vacancy probability p.



p low (0.4)
does not percolate



p medium (0.6)
percolates?



p high (0.8)
percolates

Percolation phase transition

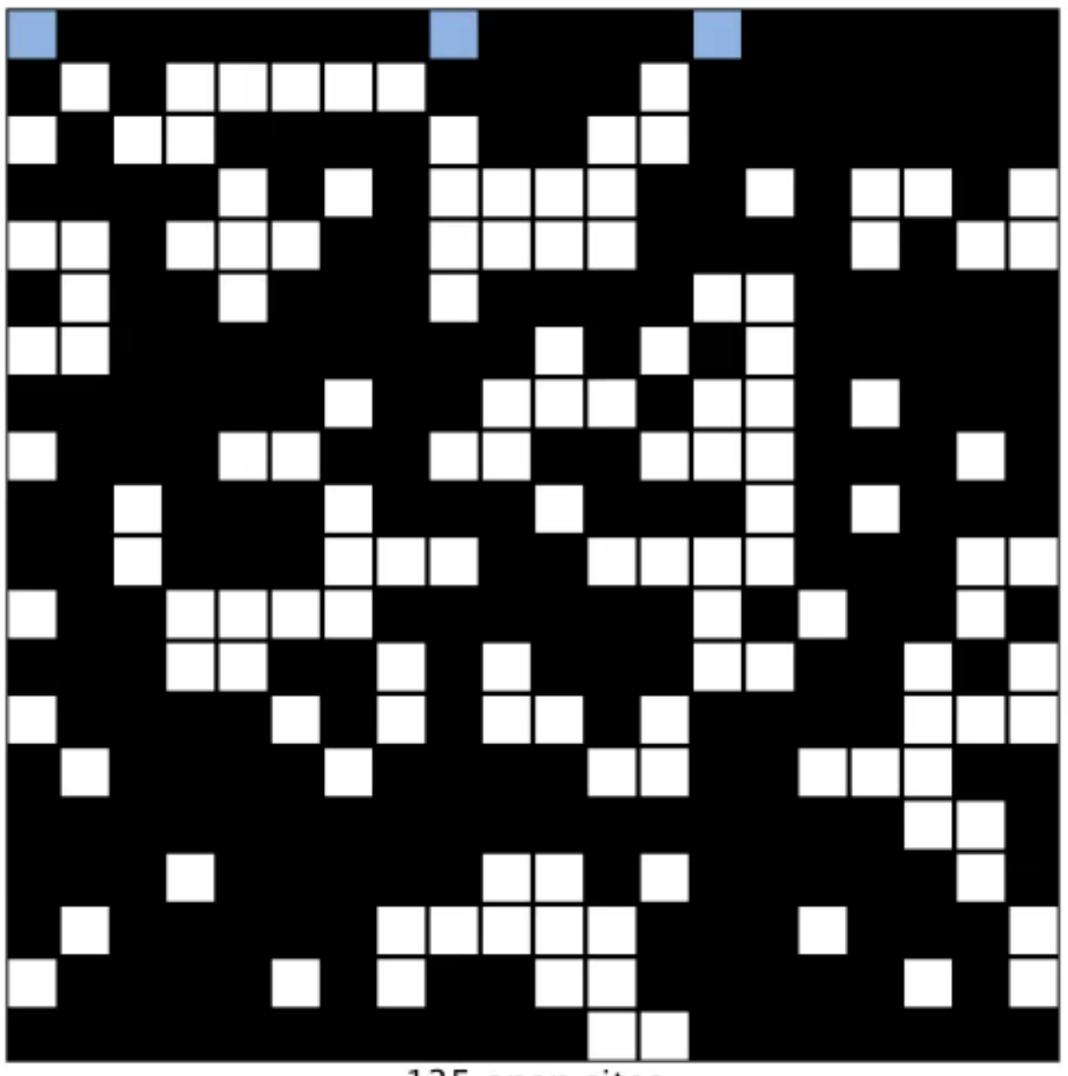
When N is large, theory guarantees a s

- $p > p^*$: almost certainly percolates.
- $p < p^*$: almost certainly does not p

Q. What is the value of p^* ?

Monte Carlo simulation

- Initialize all sites in an N-by-N grid to be blocked.
- Declare random sites open until top connected to bottom.
- Vacancy percentage estimates p^*.

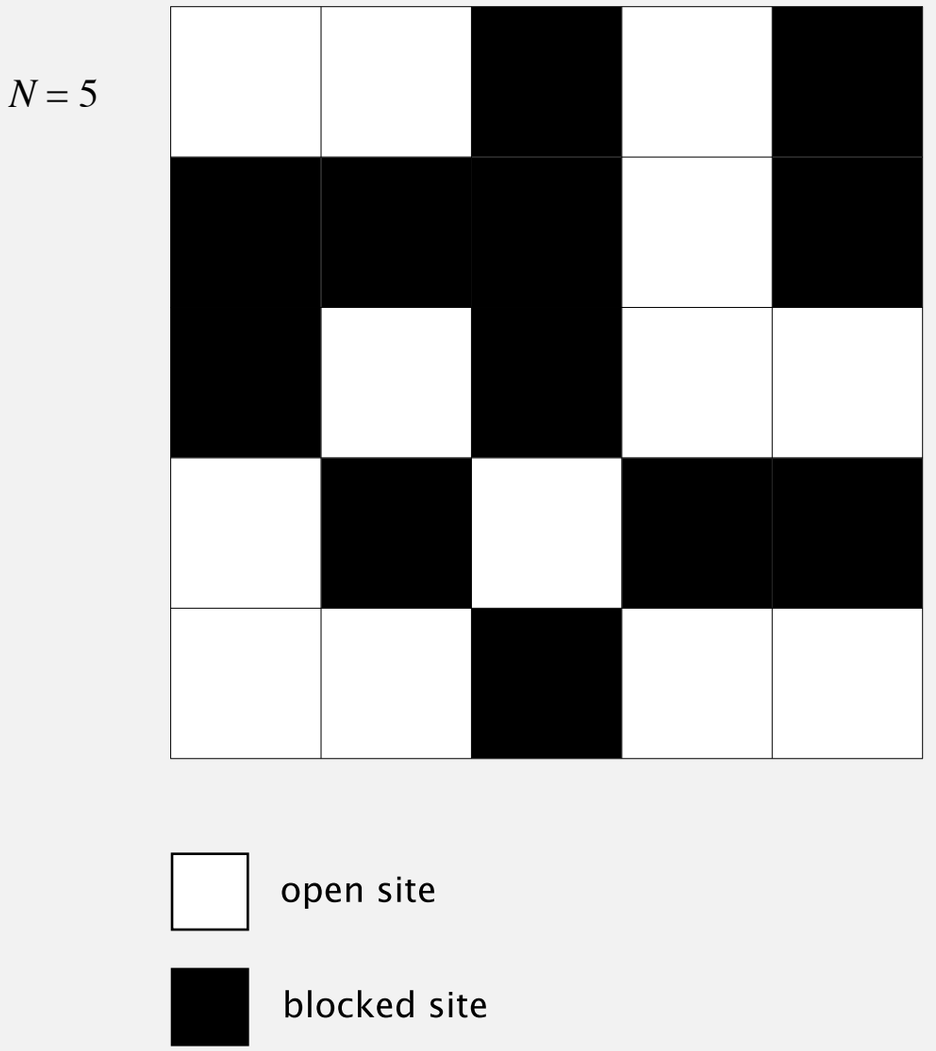


$N = 20$

Dynamic connectivity solution to estimate percolation threshold

Q. How to check whether an N-by-N system percolates?

A. Model as a dynamic connectivity problem and use union-find.

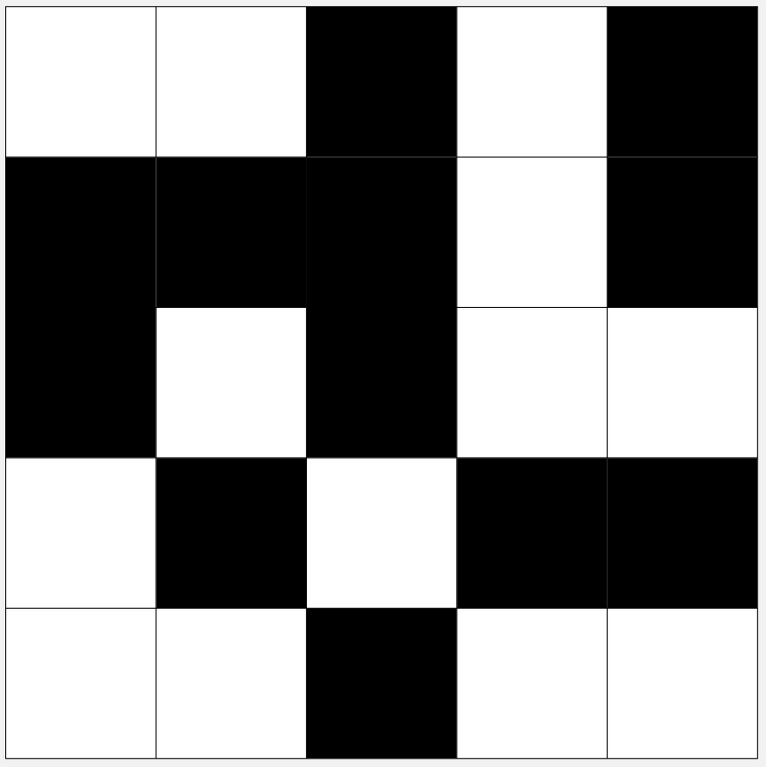


Dynamic connectivity solution to estimate percolation threshold

Q. How to check whether an N-by-N system percolates?

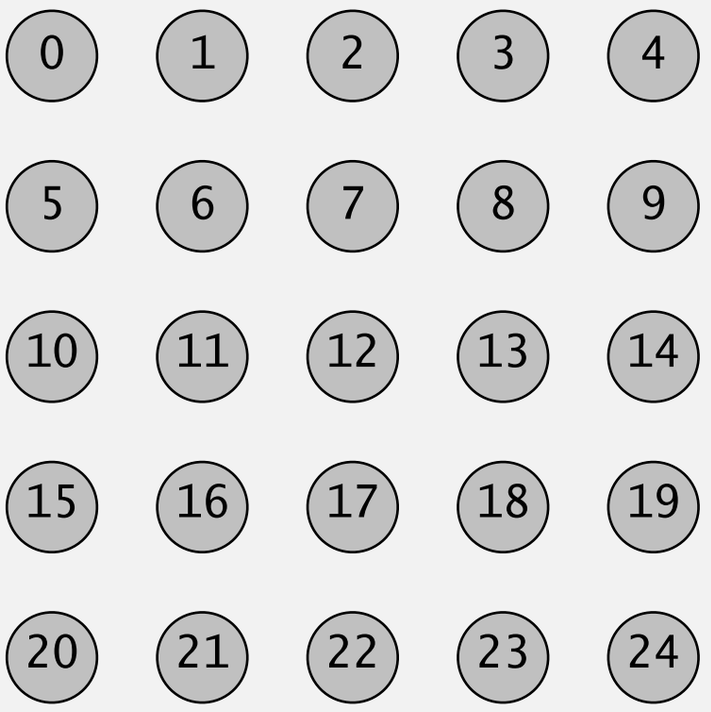
- Create an object for each site and name them 0 to $N^2 - 1$.

$N = 5$



open site

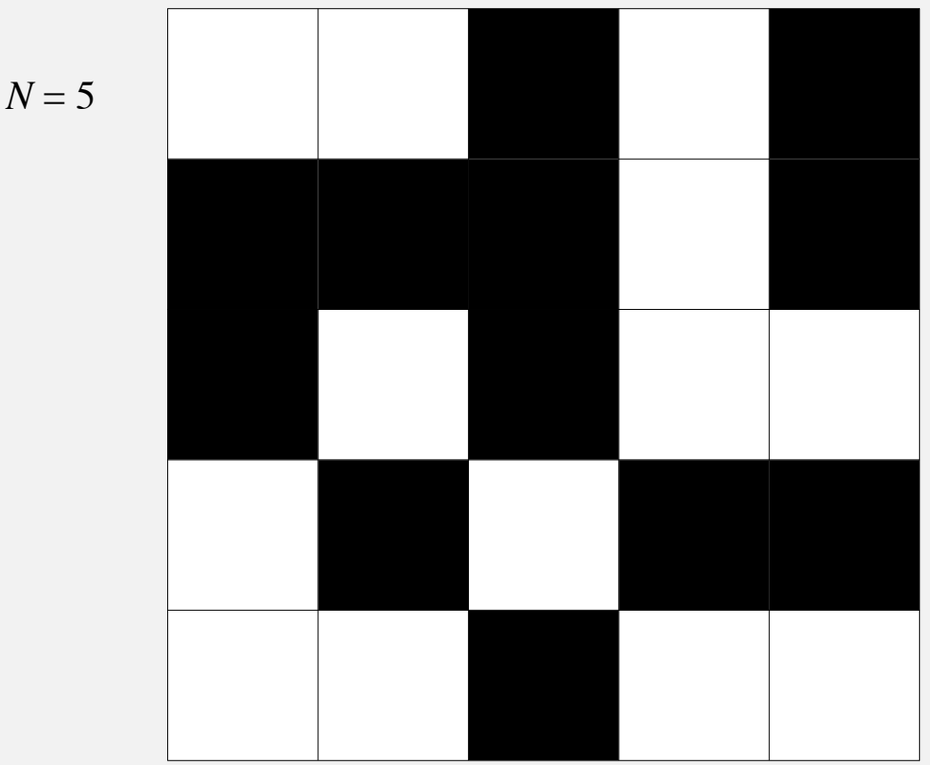
blocked site

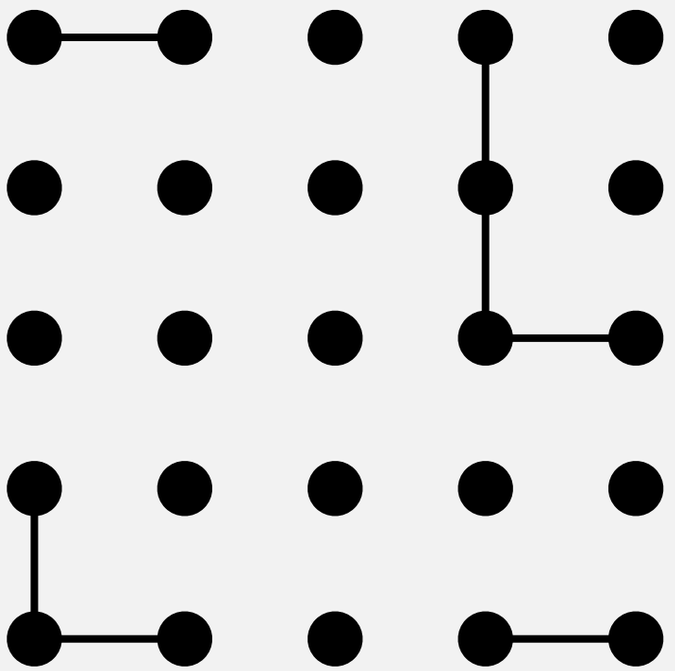


Dynamic connectivity solution to estimate percolation threshold

Q. How to check whether an N-by-N system percolates?

- Create an object for each site and name them 0 to $N^2 - 1$.
- Sites are in same component iff connected by open sites.





open site

blocked site

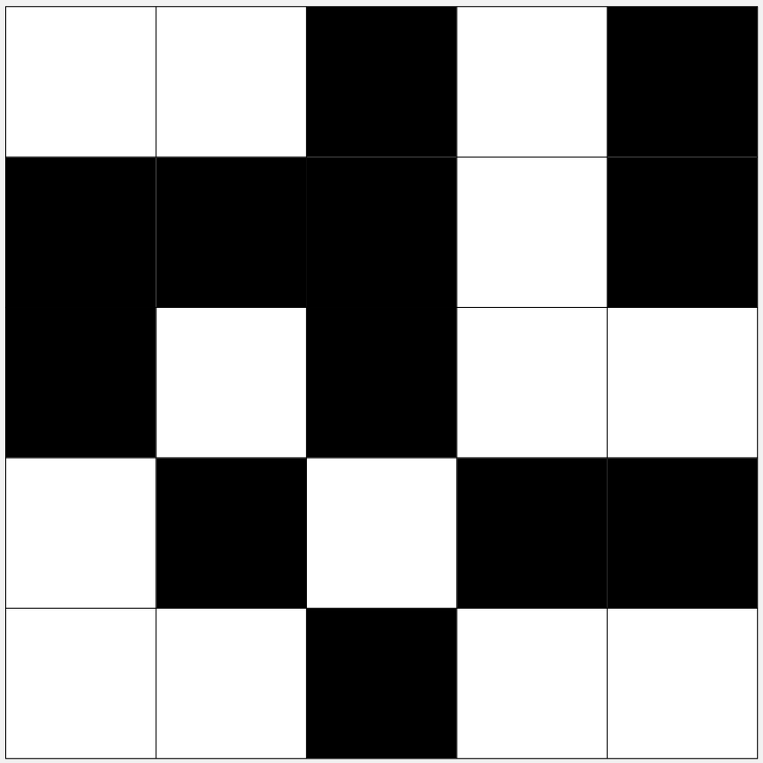
Dynamic connectivity solution to estimate percolation threshold

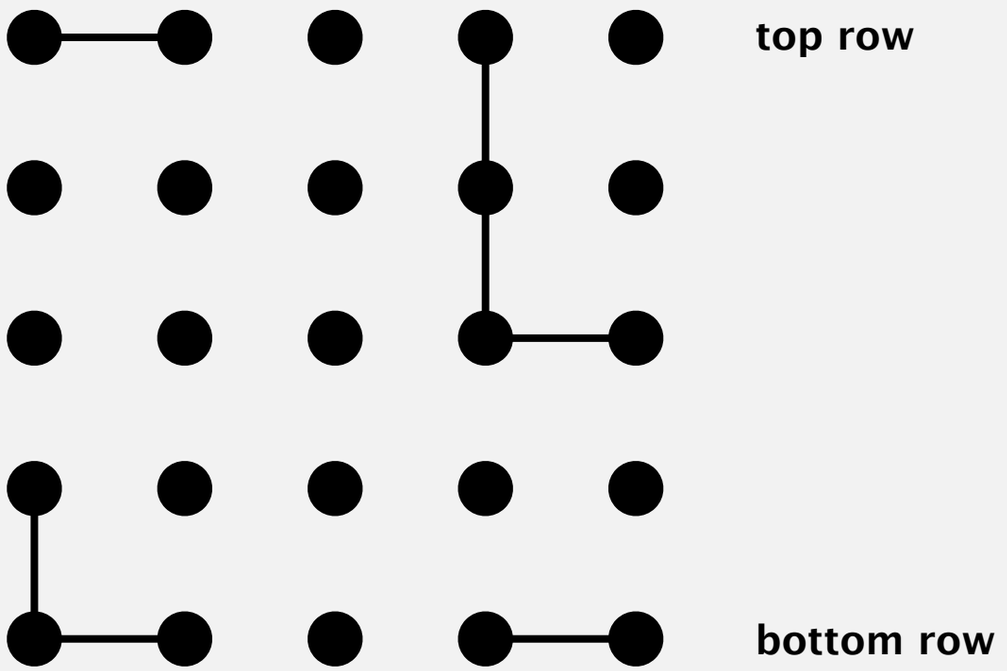
Q. How to check whether an N-by-N system percolates?

- Create an object for each site and name them 0 to $N^2 - 1$.
- Sites are in same component iff connected by open sites.
- Percolates iff any site on bottom row is connected to any site on top row.

brute-force algorithm: N 2 calls to connected()

$N = 5$





open site

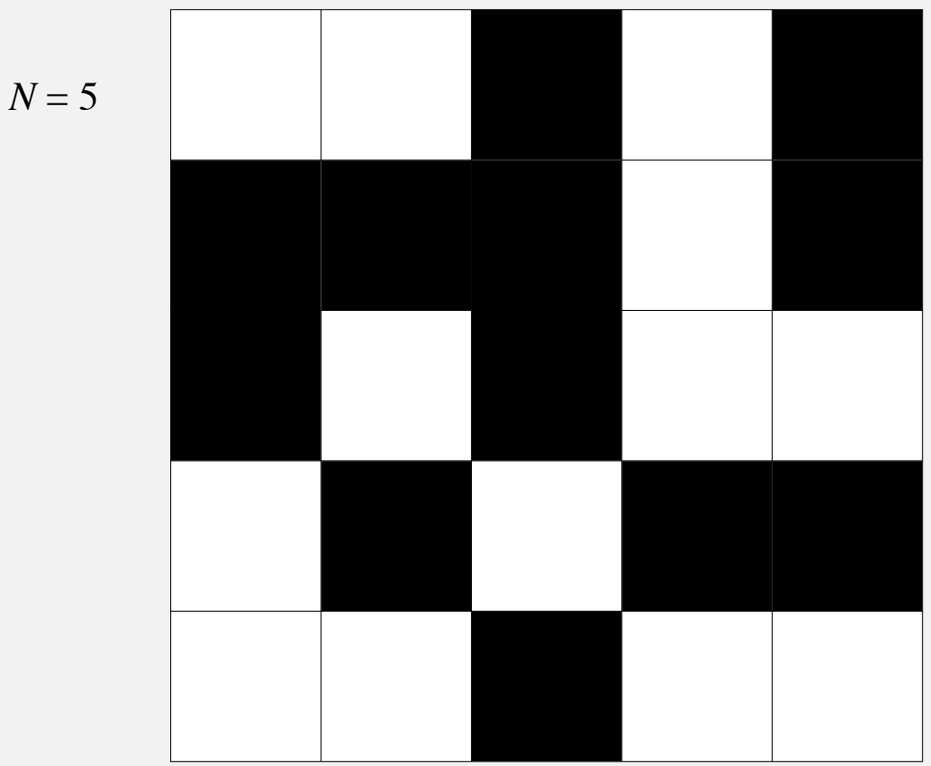
blocked site

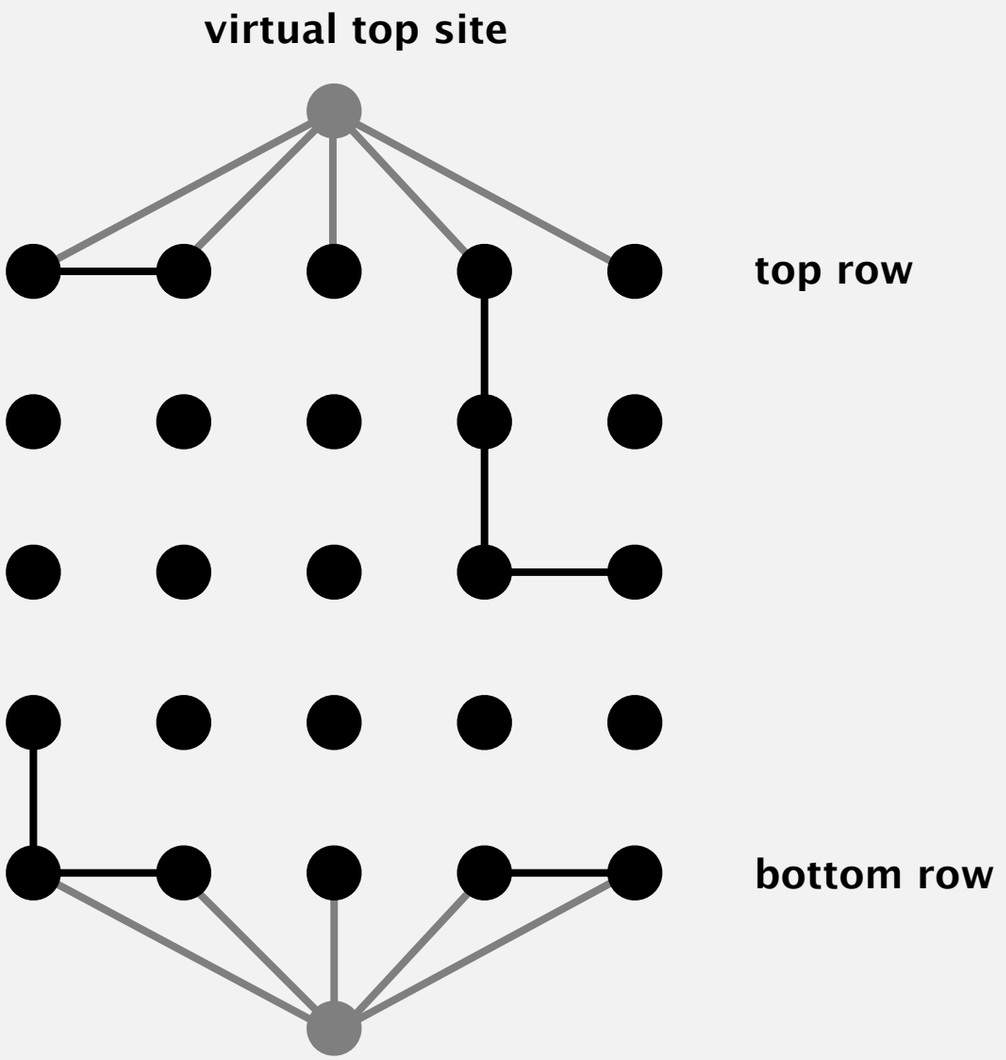
Dynamic connectivity solution to estimate percolation threshold

Clever trick. Introduce 2 virtual sites (and connections to top and bottom).

- Percolates iff virtual top site is connected to virtual bottom site.

more efficient algorithm: only 1 call to connected()



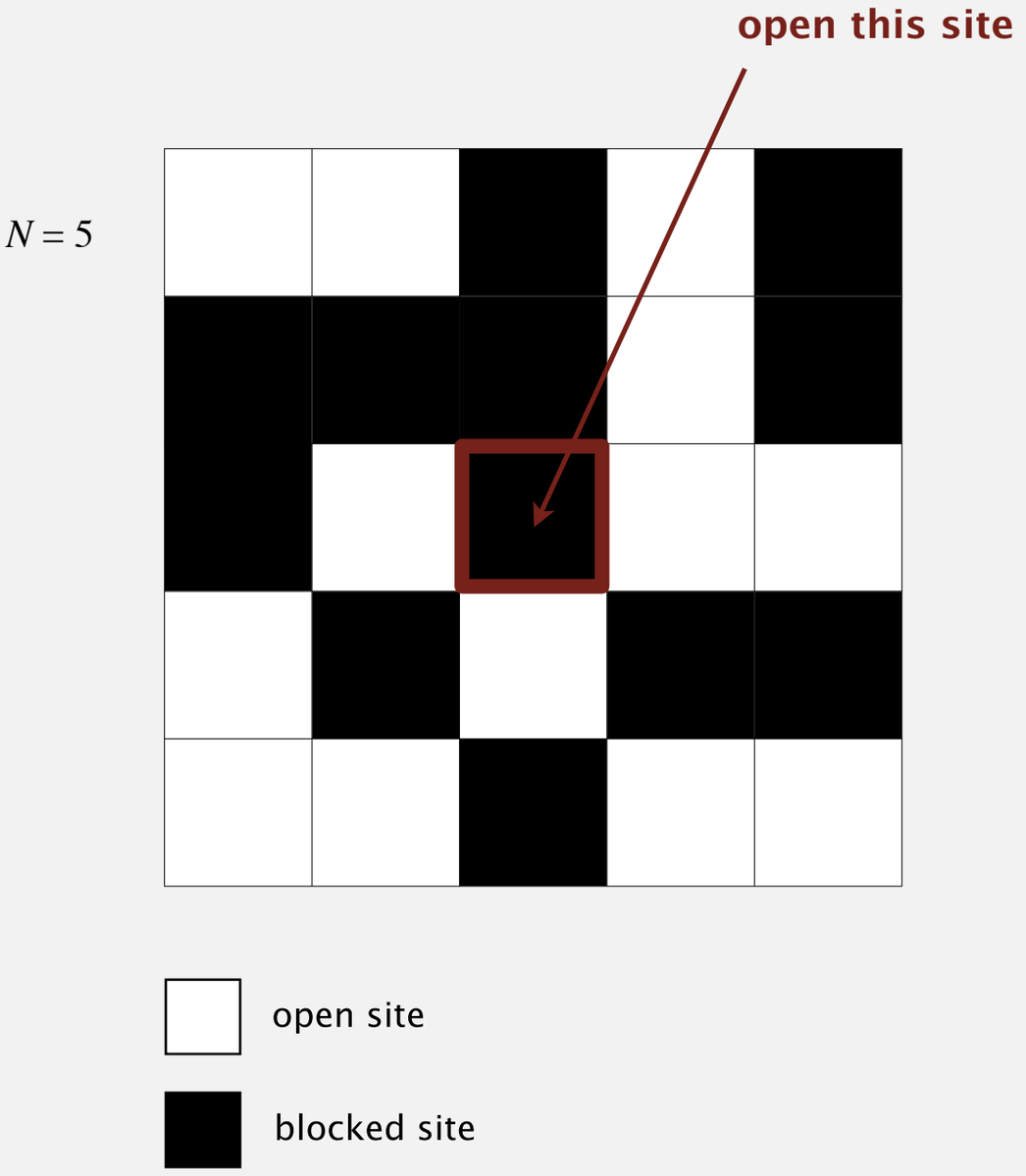


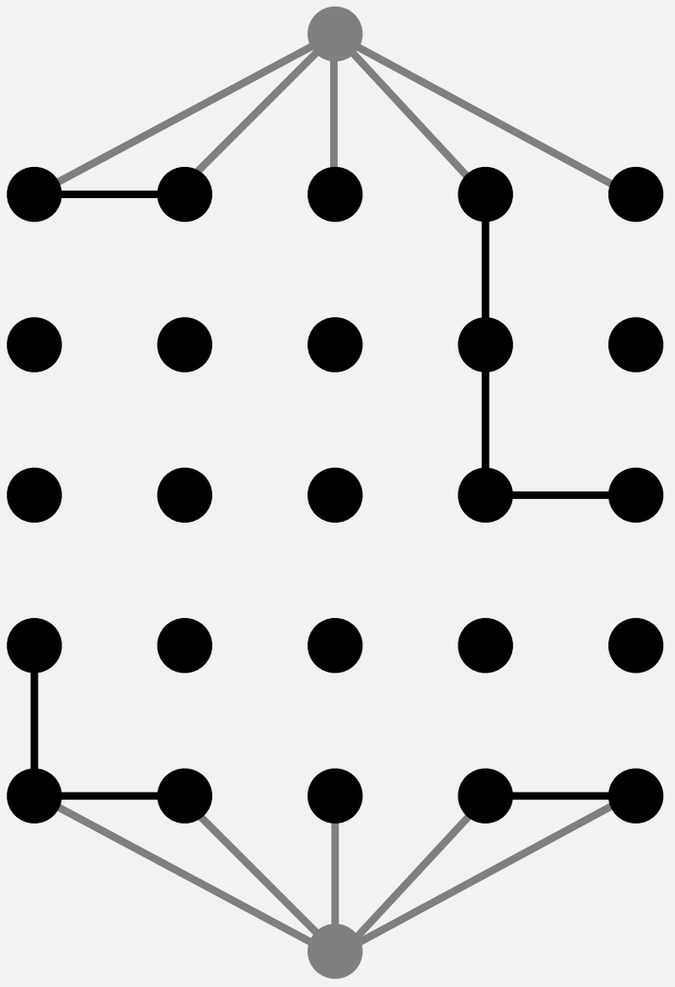
open site

blocked site

Dynamic connectivity solution to estimate percolation threshold

Q. How to model opening a new site?



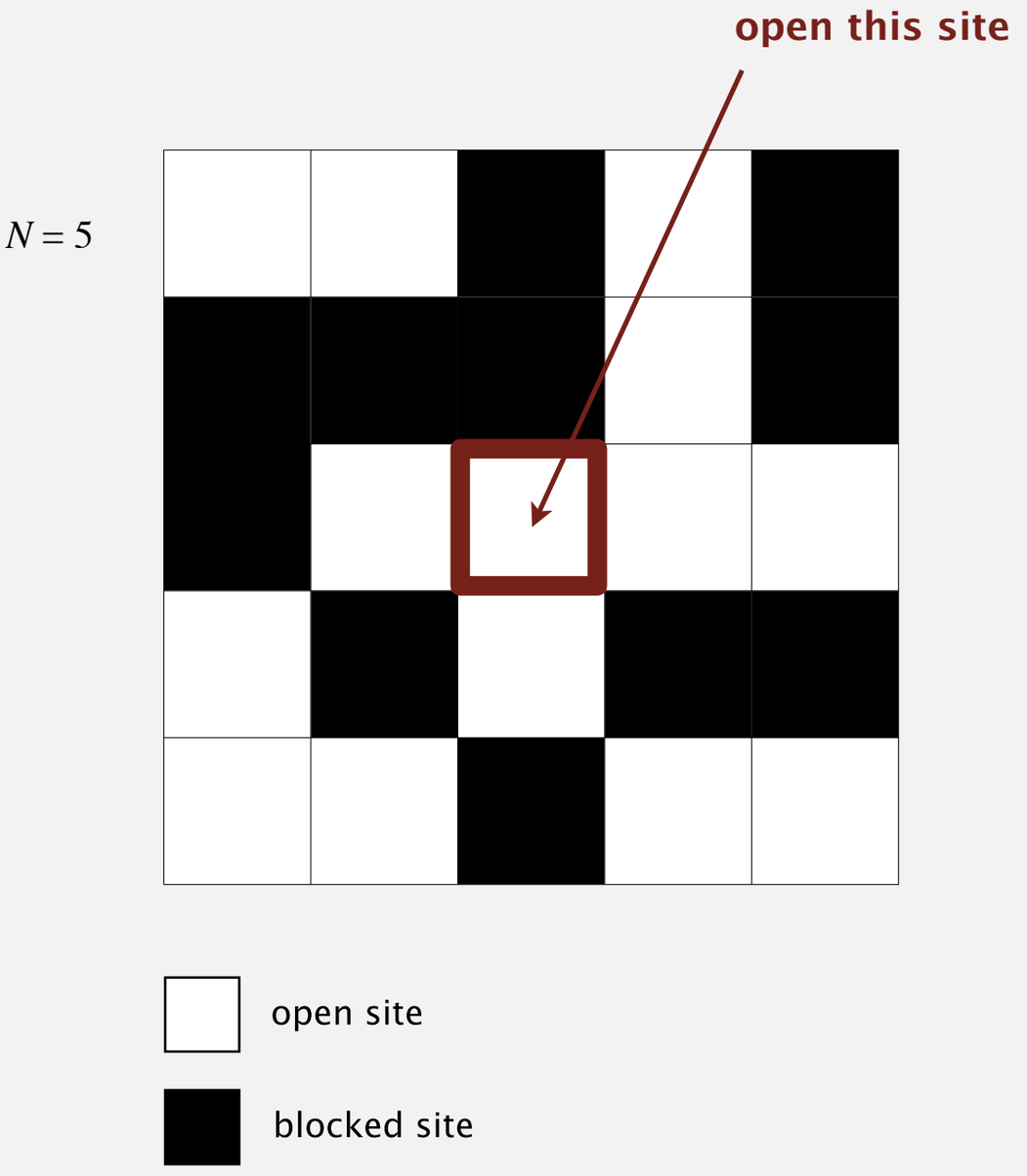


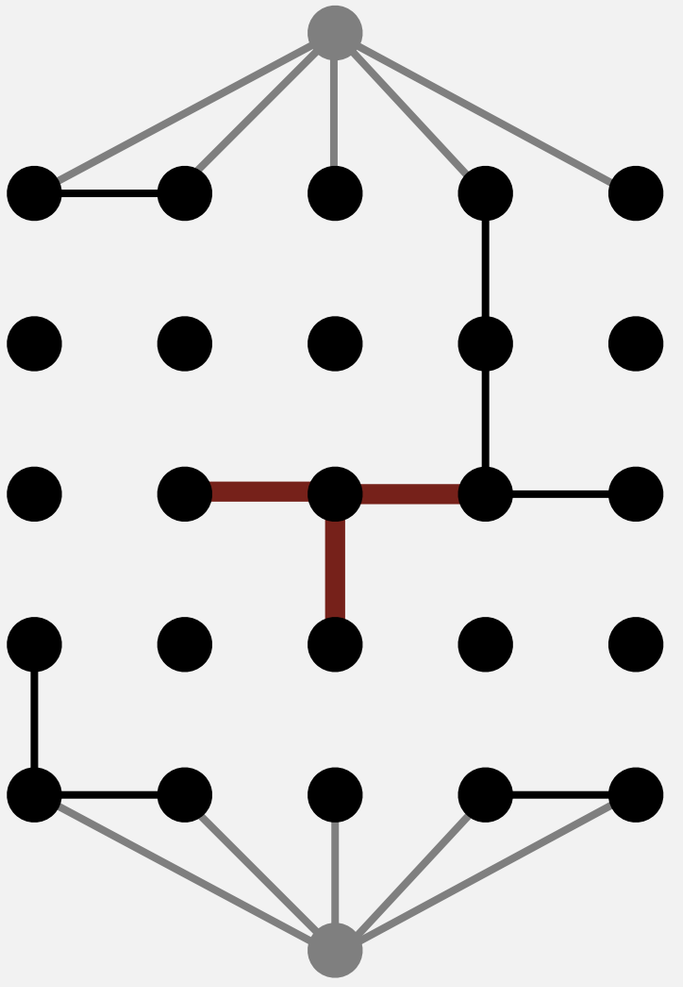
Dynamic connectivity solution to estimate percolation threshold

Q. How to model opening a new site?

A. Mark new site as open; connect it to all of its adjacent open sites.

up to 4 calls to `union()`



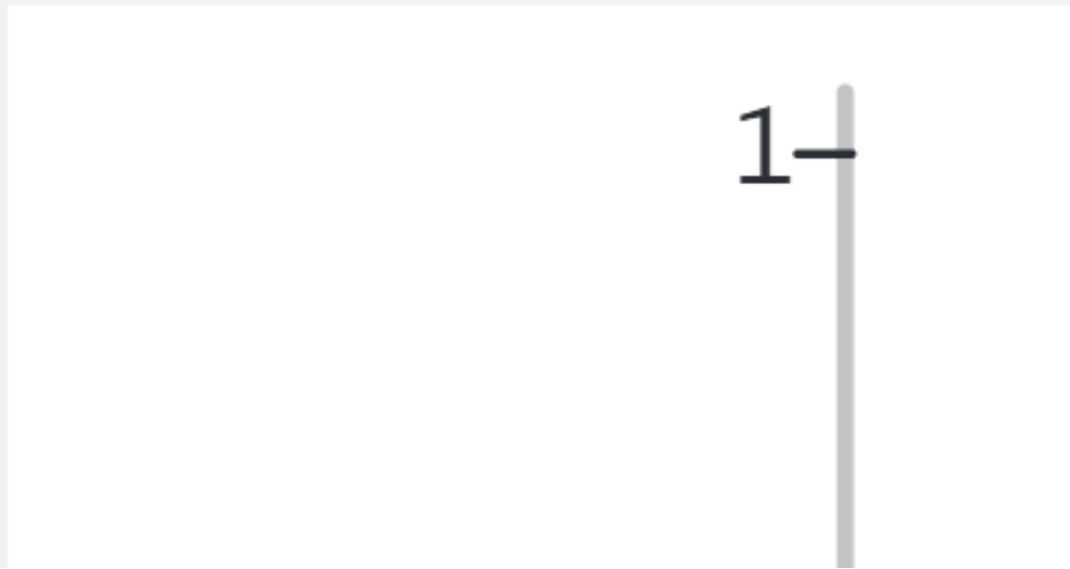


Percolation threshold

Q. What is percolation threshold p^* ?

A. About 0.592746 for large square latt

constant known only via simula



Subtext of today's lecture (and this course)

Steps to developing a usable algorithm.

- Model the problem.
- Find an algorithm to solve it.
- Fast enough? Fits in memory?
- If not, figure out why.
- Find a way to address the problem.
- Iterate until satisfied.

The scientific method.

Mathematical analysis.