1D1020 2018-09-11 #4

Union-Find

To develop an algorithm (solve a problem)
Model, Design API, Does it scale?, Analyze, Find better solutions.

Ex. N objects with M edges connecting them

Union: connect two objects
Find: to which set does an object belong?

Applications:
- Pixels
- Computers
- Friends
- Transistors
- Mathematical sets
- Variables in Fortran program

Number the objects 0-N-1 and use the numbers as indeces in an array.

Assume
- Reflexive
- Symmetric
- Transitive

Equivalence relation, equivalence classes
Connected component

Find: In which component do we find p
Connected: Does p and q belong to the same component?
Union: If p&q is not connected, merge the components into one.

```
UF(int N)
void union(int p, int q)
int find(int p)
boolean connected(int p, int q)
```

Quick-find
- Integer array id[] of size N

Many array accesses (Quadratic increase)
id[p] contains the ids of the component that contains p.

Quick-union
- Represent a component as a tree
- id[i] = index of parent of i

Trees may become un-balanced (to deep)
Find/connected can be expensive

Weighted quick-union: Avoid deep trees
- array with tree sizes
- Depth is at most lg N

Quick-union with path compression
The depth of a node x is at most $\lg N$
Don't balance trees "just because".

alg.	worst-case (M operations on N objects
QF	MN
QU	MN
WQU	$N + M \log N$
QUPC	$N + M \log N$
WQUPC	$N + M \lg N$

Applications
- Percolation
- Dynamic connectivity
- Games
- Least common ancestor
- ⋮

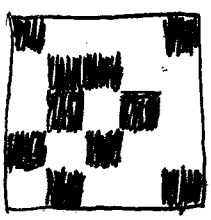
Percolation

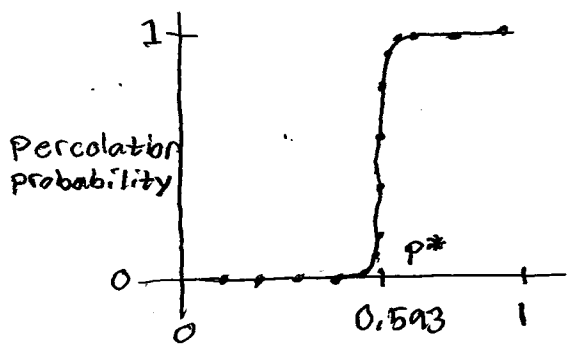
The movement and filtering of a fluid through porous material
An abstract model of many physical systems:

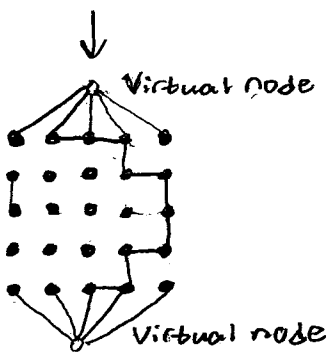
- $N \times N$ grid of sites
- Each site is open with a probability p
- The system percolates iff the top and bottom is connected through a set of connected sites.

Electricity, fluid flow, social interaction

Ex.







Check if virtual nodes are connected