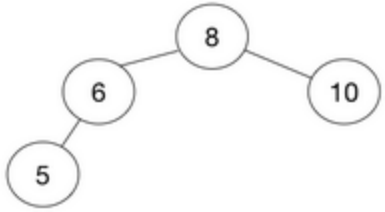
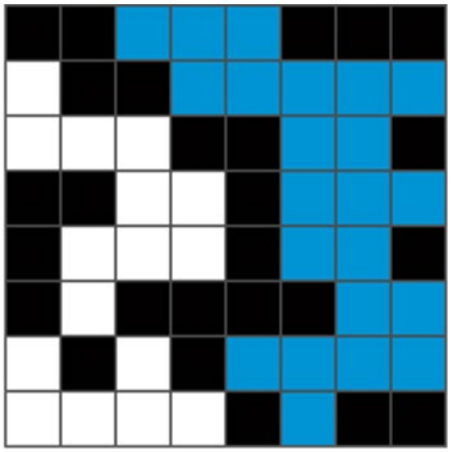
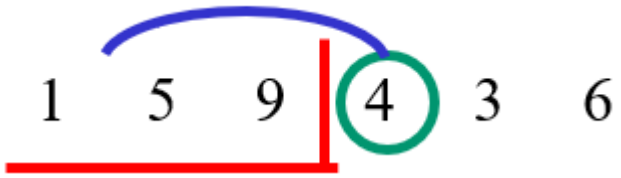
Compsci 201
Percolation, Union Find, Sorting and Priority Queues
Part 1 of 5







Susan Rodger

April 1, 2020

These are tough times

- Talk to your friends, virtually reach out to others

DUKEREACH

S is for …

- **Stack**
  - Last in, First Out, source of overflow!
- **Software**
  - Joys and sorrows, eating the world
- **Sorting**
  - From slow to quick to tim to …

Announcements

- APT-5 due Tuesday, March 31, still turn in today
  - Need extension, fill out form and take it!

- Assignment P5 Percolation – form due April 2
  - Fill out form to tell us your partner or solo

- APT-6 out and due Tuesday, April 7

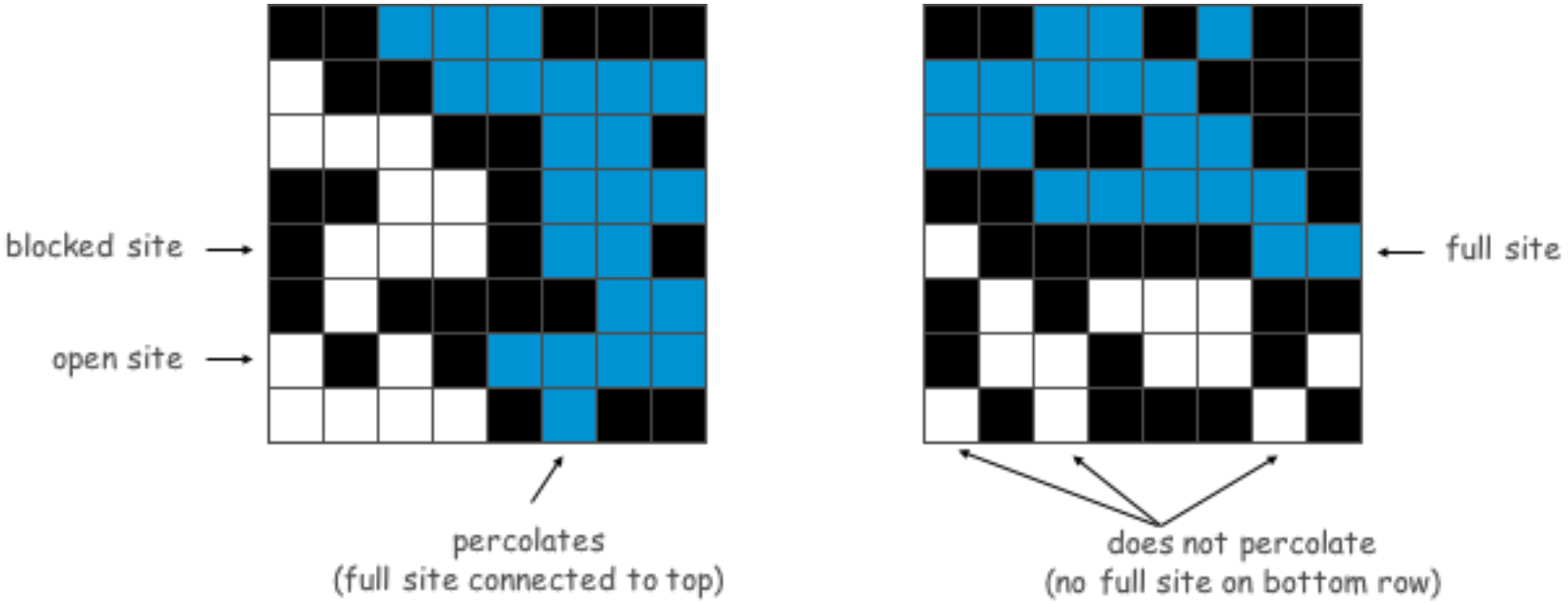
- Exam 2 is April 10
- APT Quiz 2 is April 11-15

PFFDiA

- Percolation and Simulation
  - Monte-Carlo for percolation threshold
  - DFS and BFS: limits of recursion
  - Union-Find as algorithmic alternative
- Sorting
  - Look at several slow sorts, one faster
- More Tree APTs

Percolation

- **Simulate whether an NxN grid percolates**
  - Connecting top to bottom
  - All sites *blocked*, choose at random to *open*
  - Site is *full* if in top row or connected to top



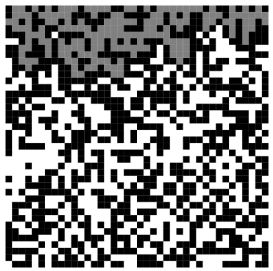
5

When Does System Percolate?

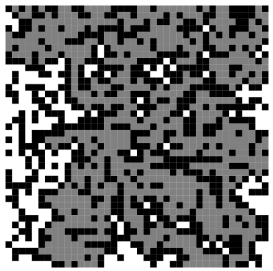
- Given an N-by-N system where each site is *open* with probability p, does system percolate?

p = 0.3
(does not percolate)

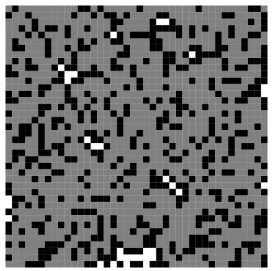
p = 0.4
(does not percolate)



p = 0.5
(does not percolate)



p = 0.6
(percolates)



p = 0.7
(percolates)

- Open question in statistical physics
- We use simulation: a computational approach

Monte Carlo Percolation

- For large N, there is a percolation threshold p*
  - Probability p < p* -- no percolation
  - Probability p > p* -- system percolates

- Simulation: take all NxN grid cells, shuffle them
  - Open one at a time until system percolates
  - How many must be opened until this happens?
  - Probability is **`count/(NxN)`** – estimate of p*

Simulating with DFS + BFS

- **Start with basic DFS, make it faster**
  - Don't test by starting at every cell in top row
  - Test after opening site **`PercolationDFSFast`**
  - Use Queue not recursion **`PercolationBFS`**

- **Base your code and ideas on BlobFill code**
  - Threshold is near 0.592, so $O(N^2)$ at least since have to open that many sites

Object-Oriented view of Percolation

- IPercolate is an interface
  - PercolateDFS, Per..DFSFast, Per..BFS, Per..UF
- Each of these can be used in a simulation
  - PercolationStats or in InteractiveVisualizer
    - Methods: **`open`**, **`isOpen`**, **`isFull`**, **`percolates`**

- PercolationUF needs a Union-Find object
  - IUnionFind: has union(x,y), connected(x,y) …
  - Different implementations, *but not a priority*

Visualize

- When automated tests aren't enough?
  - Use the visualizer to see what's happening
  - Watch a video to see what's happening

- Visualize PercolationDFSFast, BFS, and UF
  - More tests than can be done in automated way
  - Be sure you can test with concepts

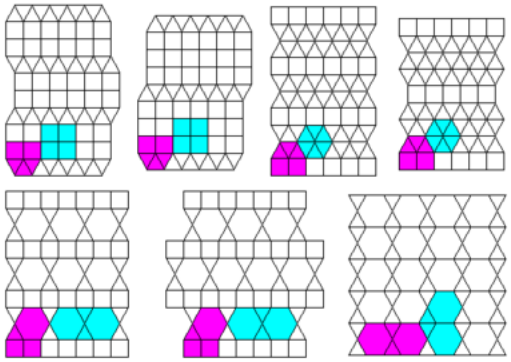
Question

- In a 10x10 grid, what is the minimum number of cells I need to mark (click on) for which it may say that it percolates?

- Is it possible to open half the cells and have the system not percolate?

Two-minute WOTO

http://bit.ly/201spring20-0401-1

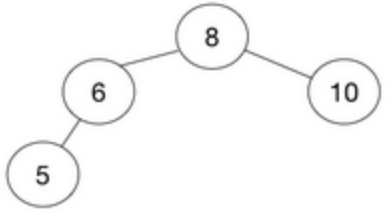


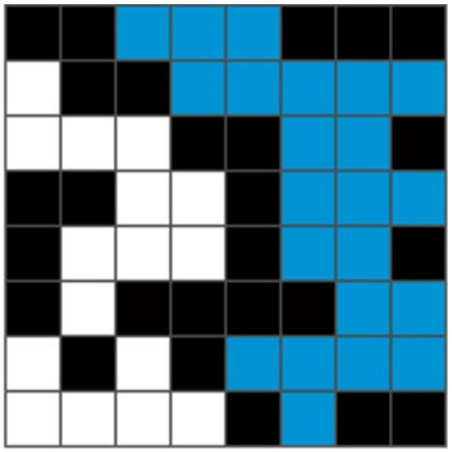
Margaret Martonosi

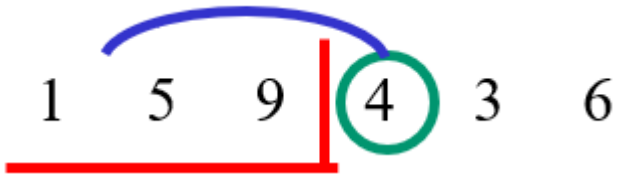
- Computer Architecture and Mobile Computing
- Designed and deployed mobile tracking with zebras
  - Low-power GPS devices
- Professor Princeton, Currently Head of National Science Foundation CISE (Computer Information Science and Engineering.

"ZebraNet was an unusual and risky project for a computer architect to embark on, but it was unique and rewarding and we learned a lot personally and technically."

Compsci 201
Percolation, Union Find, Sorting and Priority Queues
Part 2 of 5







Susan Rodger

April 1, 2020

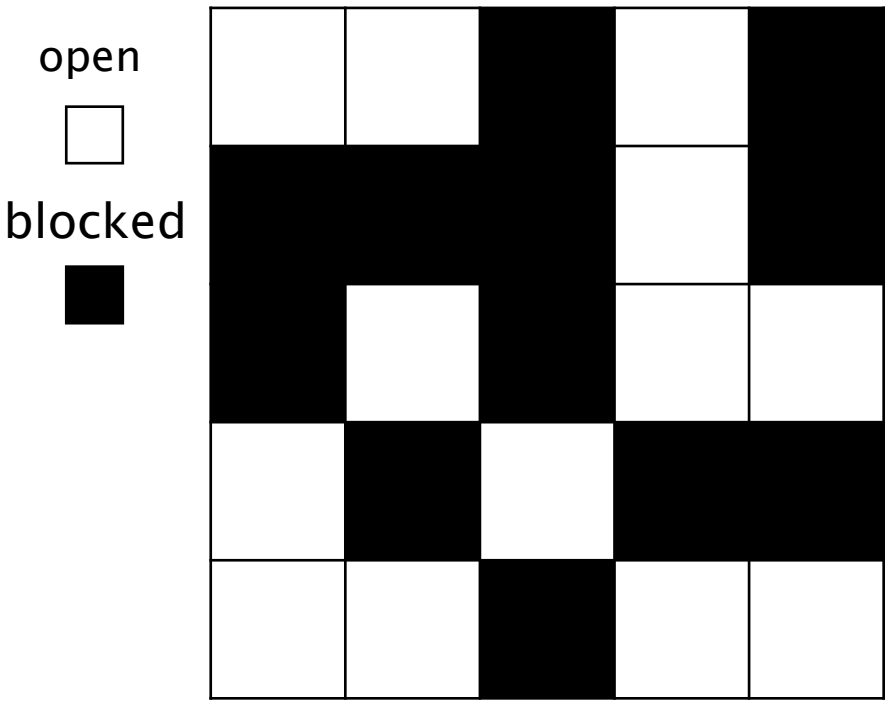
Union Find Alternative

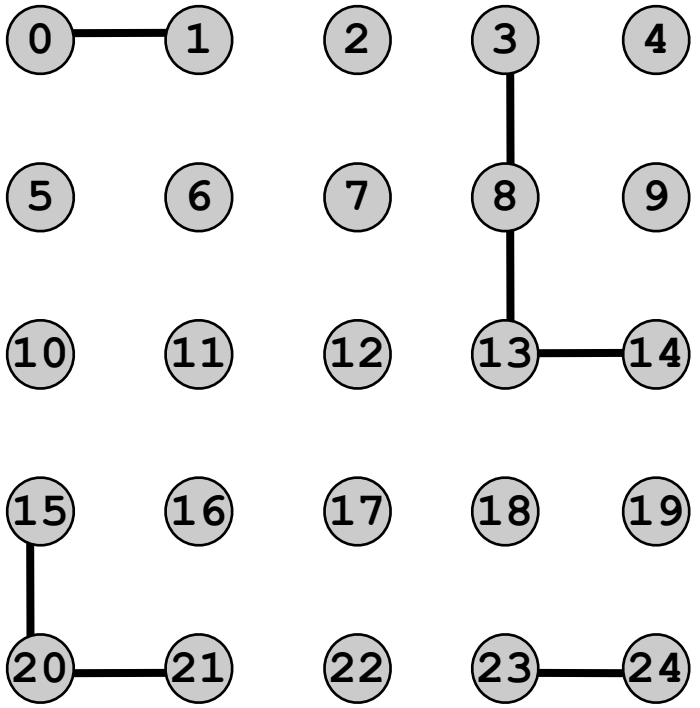
- *Union Find* aka *Disjoint Set*: Algorithm
  - Sets have empty intersection, they are disjoint
  - Creating the union of two sets should be fast
  - Finding what set an element is in should be fast

- Represent each Percolation site/grid cell as a number, initially each cell is a set by itself
  - If a site is open? Union with adjacent open sites

Percolation: Union-Find

- Adjacent open sites/cells are in the same set
  - Initially each site/cell its own set: 0, 1, … N^2-1





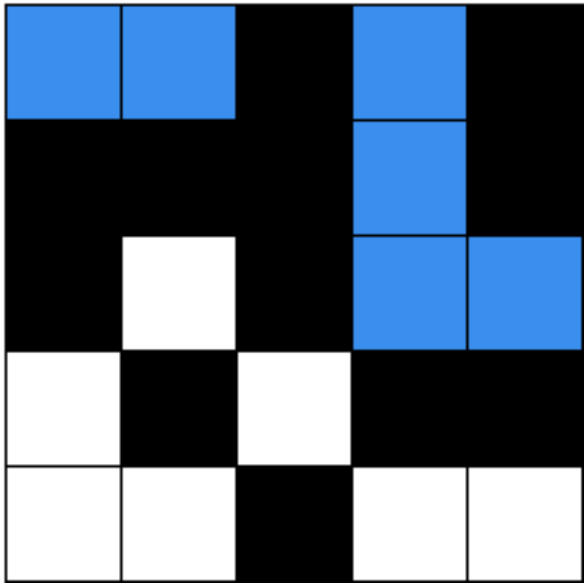
Union and Connected

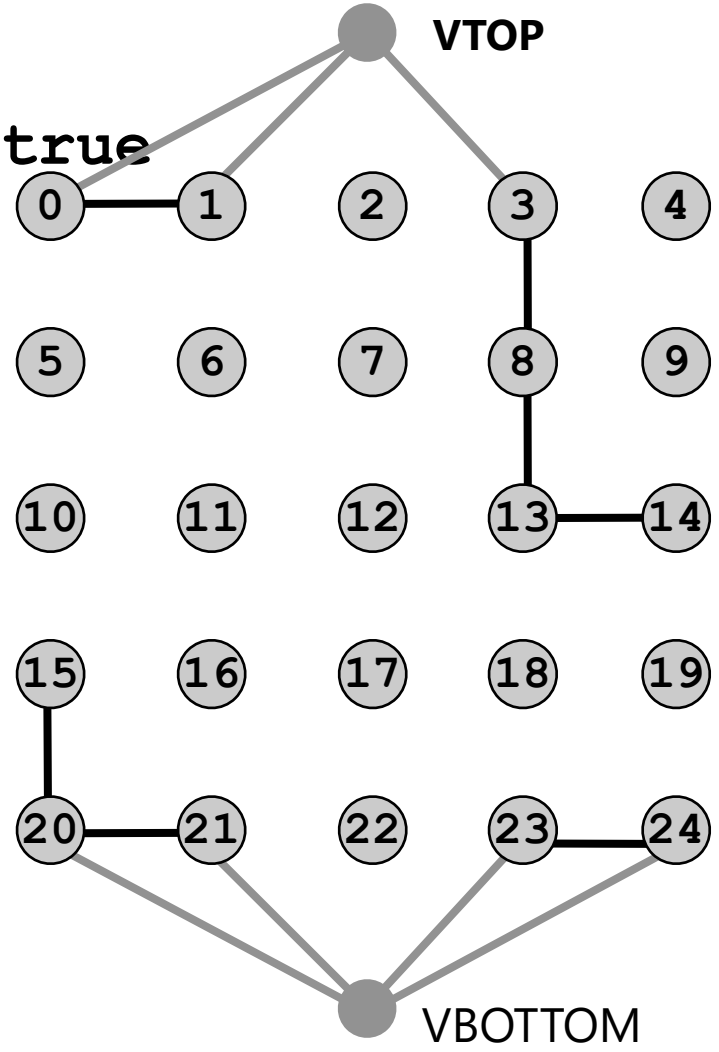
- System percolates concept and code …
  - Conceptually: open path top to bottom
  - Code: adjacent open cells in same set
  - Code: **`connected(VTOP, VBOTTOM)`**

- Each time a cell is open … NO RECURSION
  - Check adjacent cells: if open? Union sets
  - VTOP is open and VBOTTOM is open

Union and Connected

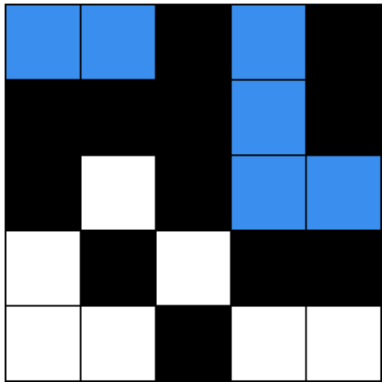
- Open? **`myGrid[r][c] == true`**
  - Calling union: open 18?

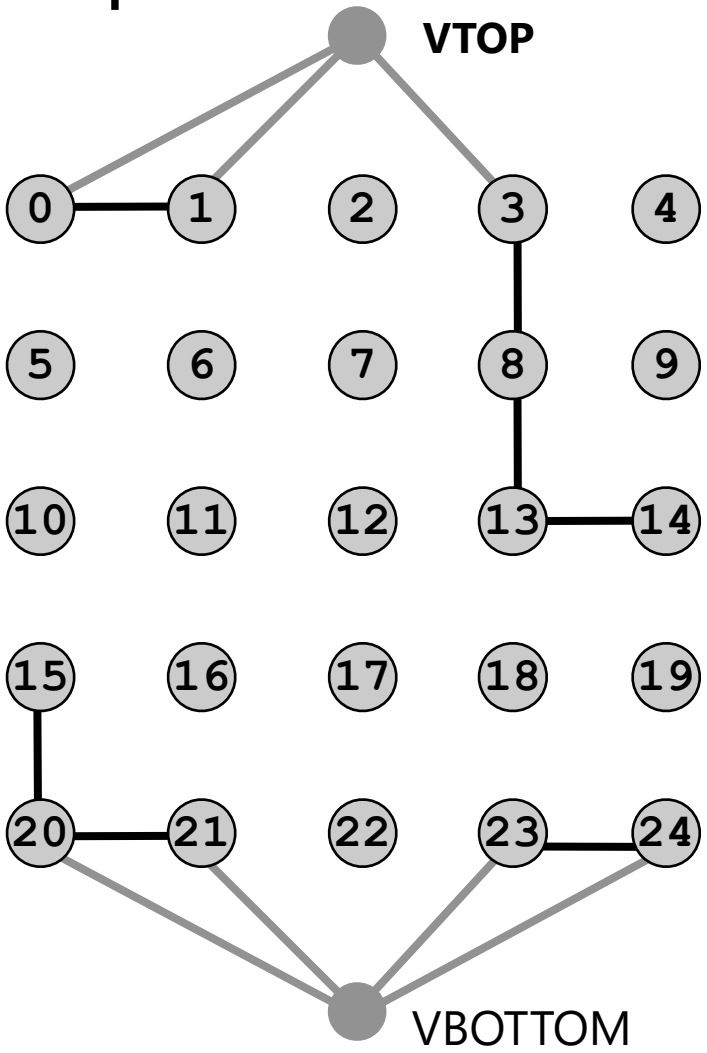




When 18 is open …

- Adjacent: 13, 17, 19, 23
  - 13 and 23 are open
  - Union(13,18)
  - Union(18,23)
- System percolates!
  - Connected(VTOP,VBOTTOM)





Union-Find Algorithms

- Initialize with N disjoint sets: O(N) for all
  - In Percolation we have $O(N^2)$ disjoint sets

- Implementations: union and find *both* efficient
  - Easy: QuickFind or QuickUnion: **`O(N)`**
  - Medium: WeightedQuickUnion: **`O(log N)`**
  - Harder: + Path-compression: **`O(1)`**
    - Technically not O(1), but in our universe it is

Two-minute WOTO

http://bit.ly/201spring20-0401-2

Craig Gentry, Duke '95

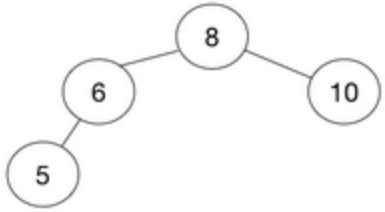
- Harvard Law, Stanford Compsci PhD
- ACM 2010 Hopper Award
- MacArthur Fellow 2014

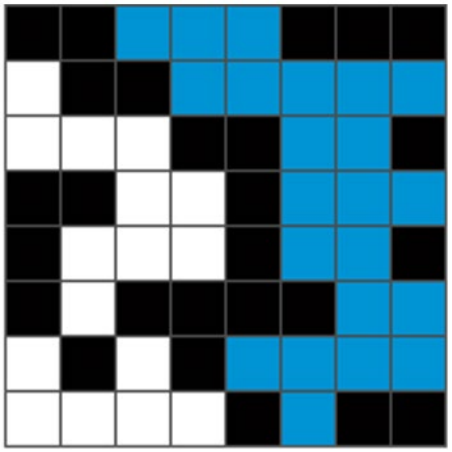
"Fully homomorphic encryption is a bit like enabling a layperson to perform flawless neurosurgery while blindfolded, and without later remembering the episode. We believe this breakthrough will enable businesses to make more informed decisions, based on more studied analysis, without compromising privacy."

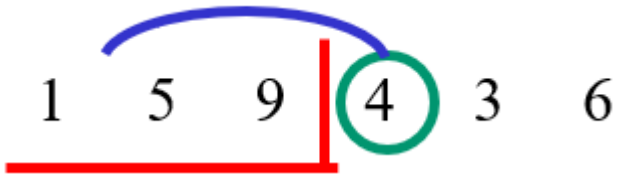
Research Scientist at IBM

Now at Algorand Foundation

Compsci 201
Percolation, Union Find, Sorting and Priority Queues
Part 3 of 5







Susan Rodger

April 1, 2020

PriorityQueues top to bottom

- All operations are O(log N) where N size of PQ
  - This for add and remove; can peek in O(1)
  - Details after midterm

- Always remove the smallest element, minPQ
  - Can change by providing a **`Comparator`**

- Shortest-path, e.g., Google Maps. Best-first search in games
  - Best element removed from queue, not first

PriorityQueues top to bottom

- **How can we sort elements using Priority Queue?**
  - Add all elements to pq, then remove them
  - Every operation is O(log N), so this sort?
  - O(N log N) – basis for ***heap sort***

```
104     void sort(List<T> list) {
105         PriorityQueue<T> pq = new PriorityQueue<>(list);
106         list.clear();
107         while (pq.size() > 0) {
108             list.add(pq.remove());
109         }
110     }
```

Problem: Finding Top M or N

- Given N items. Find the Top M largest items. Return them with largest first, then second largest…

- Example: [5, 9, 2, 32, 8, 41, 27, 11, 7, 24]
- Find top 4:

- If strings, find “largest” means “those last in alphabetical order”

First way: Finding top M of N

- Sort all and get first (or last) M
  - O(N log N) to sort, then O(M), typically N >> M
- Code below doesn't alter list parameter
  - Why is **`comp.reversed()`** used?

```
21      public static List<String>
22          sortTopM(List<String> list, int mSize,
23                   Comparator<String> comp){
24          List<String> copy = new ArrayList<>(list);
25          Collections.sort(copy,comp.reversed());
26          return copy.subList(0, mSize);
27      }
```

Faster way: Finding top M of N

- Can do this in O(N log M) using priority queue
  - Not intuitive? largest M using min PQ?

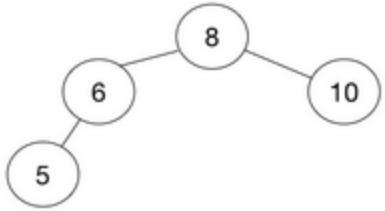
```
29    public static List<String>
30        pqTopM(List<String> list, int mSize,
31               Comparator<String> comp) {
32        PriorityQueue<String> pq = new PriorityQueue<>(comp);
33        for(String s : list) {
34            pq.add(s);
35            if (pq.size() > mSize) {
36                pq.remove();
37            }
38        }
39        LinkedList<String> ret = new LinkedList<>();
40        while (pq.size() > 0) {
41            ret.addFirst(pq.remove());
42        }
43        return ret;
44    }
```

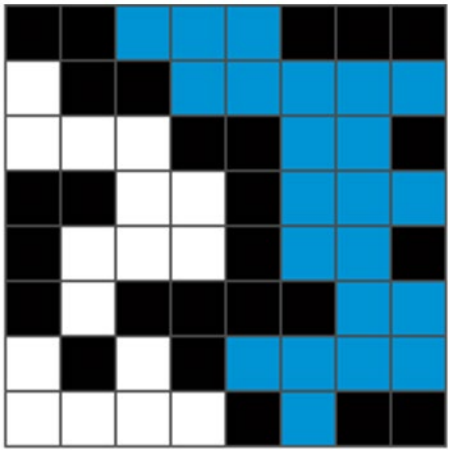
Details for M of N

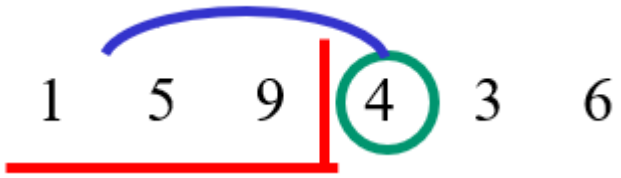
- **Keep only M elements in the priority queue**
  - Every time one removed? It's the smallest
  - When done? Top M remain, removed smallest!
  - First element removed? Smallest, so …
    - Why is LinkedList used? O(1) add to front

```
39          LinkedList<String> ret = new LinkedList<>();
40          while (pq.size() > 0) {
41              ret.addFirst(pq.remove());
42          }
43          return ret;
```

Compsci 201
Percolation, Union Find, Sorting and Priority Queues
Part 4 of 5







Susan Rodger

April 1, 2020

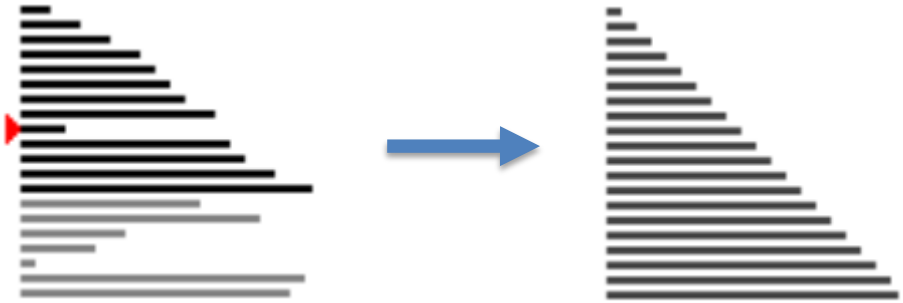
Sorting: 201 in a 

- Algorithms: traditionally foundation of compsci
  - Study, analyze, develop, use
- APIs: tested, proven, configurable
  - Algorithms encapsulated and usable

- You should know how to write your own sort
- You should know how to call library sort

Sorting: From Theory to Practice

- **Why do we study more than one algorithm?**
  - Paradigms of trade-offs and algorithmic design
  - Know your history



Review - Selection Sort

- Sort a list of numbers.
- Idea:
  - Repeat til sorted
    - Find the smallest element in part of list not sorted
    - Put it where it belongs in sorted order.
      - Swap it with the element where it should be
- Sort example

Sorted, won't move final position	???

Selection Sort – red area sorted

9 5 4 1 3 6 - find smallest, swap

Insertion Sort

- Sort a list of numbers.
- Idea:
  - Sort by repeated inserting another element
    - Leftmost element is sorted part of list
    - Insert another element in that sublist keeping it sorted
    - Insert another element in that sublist keeping it sorted
    - Etc.
- Sort example

Sorted relative to each other	???

Insertion Sort – red area sorted

9 5 1 4 3 6 - insert 5

Review Bubble Sort

- Sort a list of numbers.
- Idea:
  - Repeat til sorted
    - Compare all adjacent pairs, one at a time. If out of order then swap them
- Sort example

???	***Sorted, won't move final position***

Bubble Sort – red area sorted

9 5 4 1 3 6 - compare, swap

Simple, $O(n^2)$ sorts

- *Selection sort* --- n^2 comparisons, n swaps
  - Find min, swap to front, increment front, repeat
- *Insertion sort* --- n^2 comparisons, no swap, shift
  - *stable*, fast on sorted data, slide into place
- *Bubble sort* --- n^2 everything, slow*
  - Catchy name, but slow and ugly*

*this isn't everyone's opinion, but it should be

- *Shell sort*: quasi-insertion, fast in practice
  - Not quadratic with some tweaks

Sorting: Visualizations

- Good site for sorting visualizations:
  - http://www.sorting-algorithms.com/
- There are lots of sorting demonstrations out there:
  - Google dance sorting algorithms

Sorting algorithms demonstrated with Hungarian folk dance

WOTO on sorting

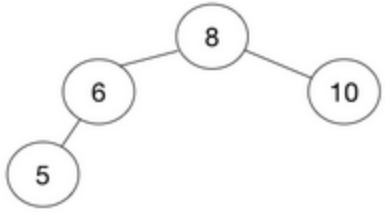
http://bit.ly/201spring20-0401-3

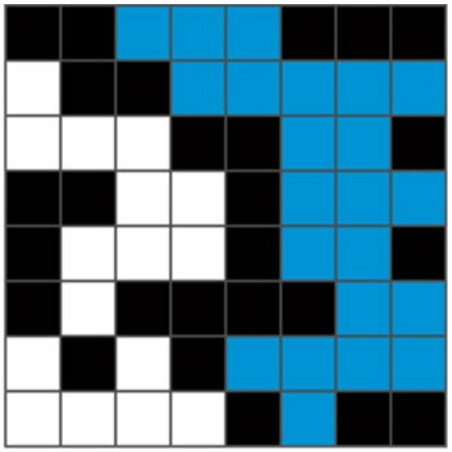
Alan Turing

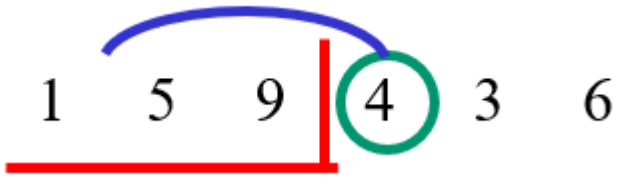
- 2:46 marathon
- 15:20 three mile
- Enigma machine and WWII
- *Entscheidungsproblem*
- *He was gay at a time when society was not very accepting*

Those who can imagine anything, can create the impossible.

Compsci 201
Percolation, Union Find, Sorting and Priority Queues
Part 5 of 5







Susan Rodger

April 1, 2020

Tree APTs: Recursion Idiom for Trees

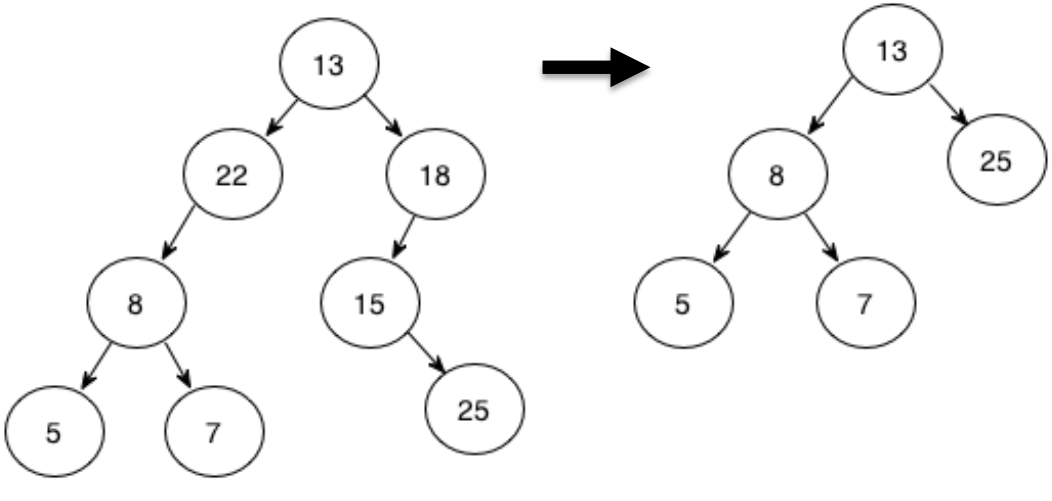
- Use **`root = caller(root, val);`**
  - Modifies Tree at root and returns the result
  - Recursive calls works similarly
    - **`root.left = caller(root.left, val);`**
    - Process the subtree, use the result

- Where do we see this? Insertion into search tree
  - We also see it in tree tighten APT

Tightening a Tree (hint1)

Remove all the nodes that have only one child. Return modified tree.

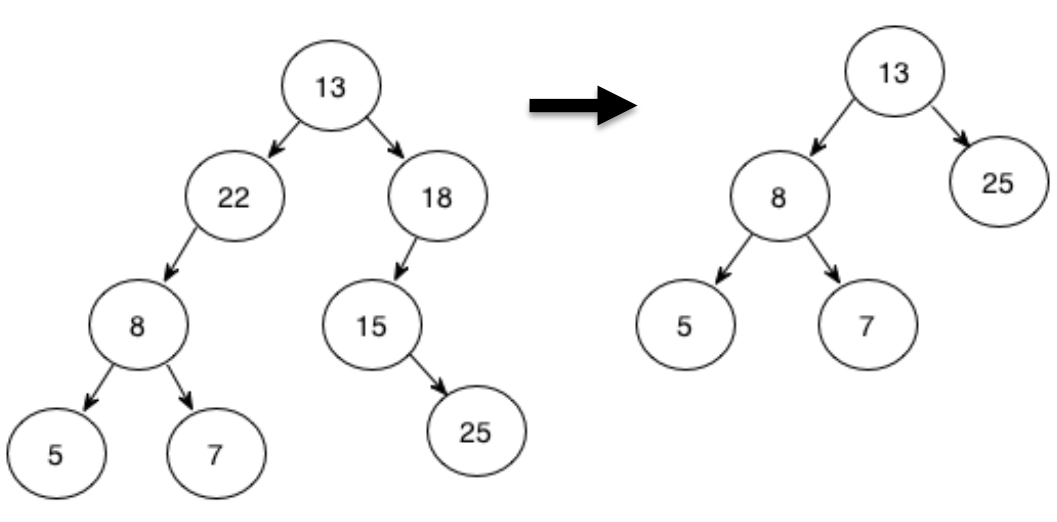
- https://www2.cs.duke.edu/csed/newapt/treetighten.html
- Call: **`tree = tighten(tree)`**
- A node has two children (13 and 8)
  - They are still in modified tree
  - Reset left, reset right, return itself
    - `tree.left = tighten(tree.left)`



Tightening a Tree (hint2)

Remove all the nodes that have only one child. Return modified tree.

- A node has no children (5, 7, and 25)
  - They are still in modified tree
- A node has one child (22, 18 and 15)
  - Don't want the node, instead return the child.

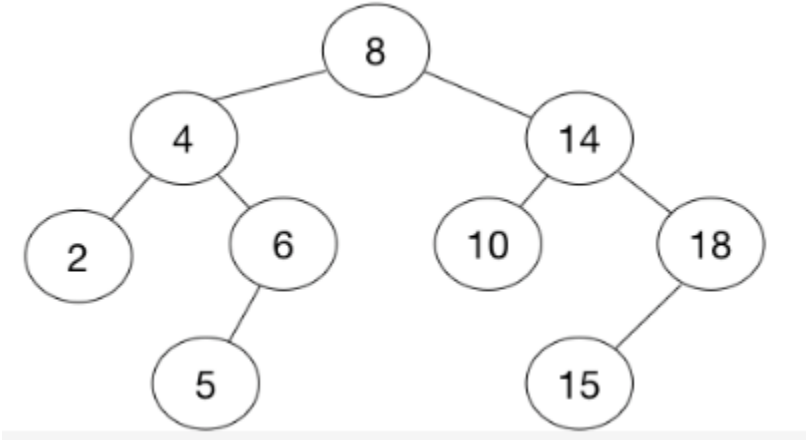



```
if (t.left == null && t.right != null) return tighten(t.right);
```

APT TrimTree

Remove all nodes from tree NOT in range [x,y]

- https://www2.cs.duke.edu/csed/newapt/trimtree.html
- It's a search tree, so we can use range search
  - Left subtree <= root. Right subtree > root
    - With no duplicates, < and not <=
- trim(root, 5, 12) for [5,12]



APT TrimTree (hints)

- If root in range?
  - Process subtrees, return root
    - `t.left = trim(t.left, …)`
    - `t.right = trim(t.right, …)`

- If root not in range? Return result of one call!
  - `return trim(t.left, ..)` or …