COS 226 Lecture 19: Shortest paths

Classic algorithms for solving "network" problems

MINIMAL SPANNING TREE
- Kruskal's algorithm
- Prim's algorithm

SHORTEST PATHS
- Dijkstra's algorithm
- Floyd's algorithm

Related to generalized graph search
using a priority queue

Ex: telephone wiring
Ex: supplies to troops

Generalized search (Priority-first)

Use PQ, rather than stack or queue

TREE vertices
    visited
FRINGE vertices
    adjacent to visited
UNSEEN vertices
    others

Put all fringe vertices on PQ

Search algorithm
- take next vertex off PQ
- change status to TREE
- for each adjacent vertex
    - UNSEEN: move to FRINGE
    - FRINGE: update prioirity

Encompasses several basic graph algorithms
- depth-first search
    - (use decreasing counter for priority)
- breadth-first search
    - (use increasing counter for priority)
- minimal spanning tree
- shortest path tree

Shortest paths

SINGLE-SOURCE shortest paths:

    shortest path from x to each other node

Algorithm: PFS starting at x

val[k]:

    shortest distance from x to k

priority on FRINGE:

    shortest known distance to node

smallest fringe value:

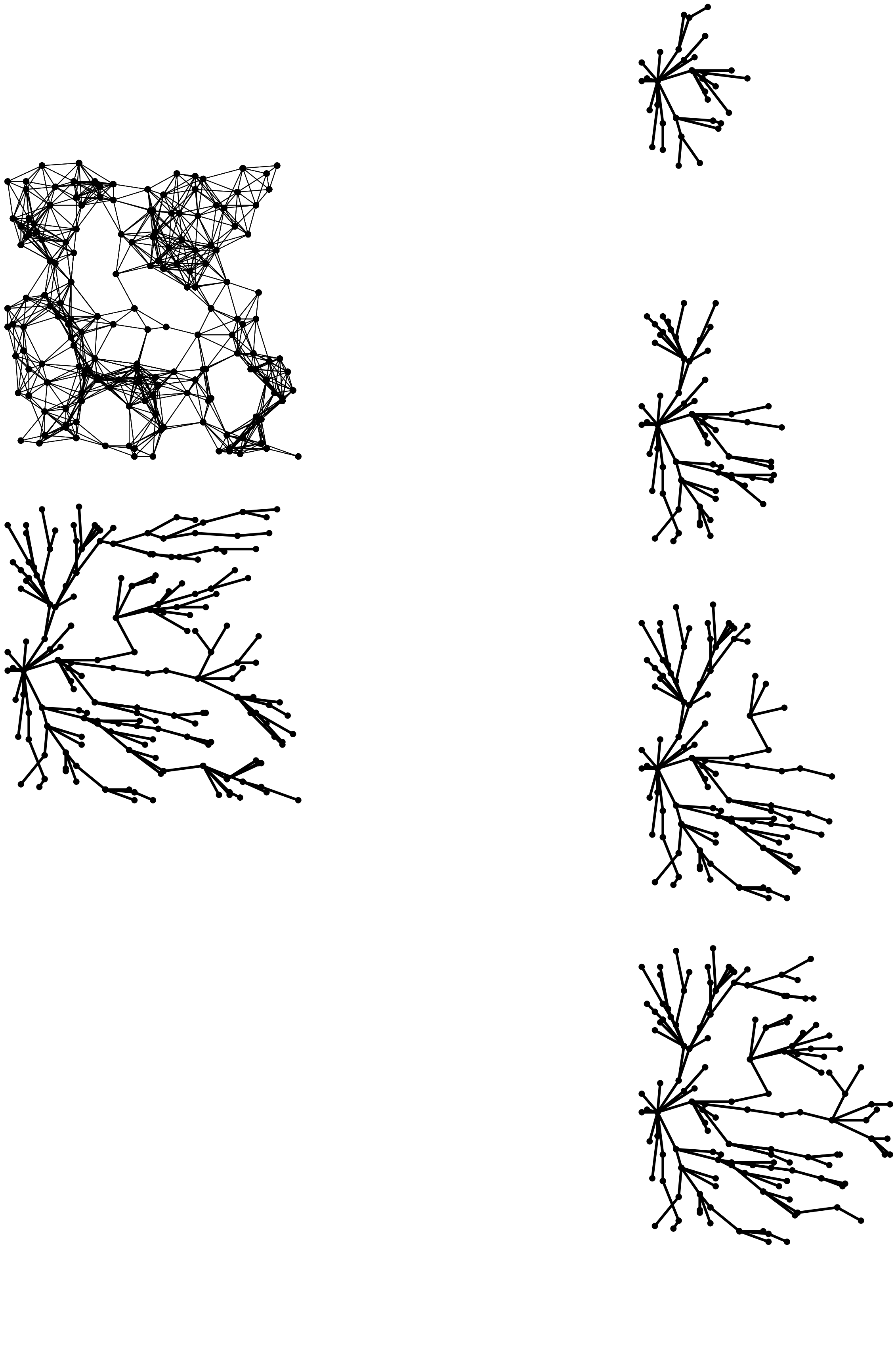
    shortest distance to *any* node not on tree
    (won't find a shorter path)

```
#define priority val[k]+t->w
```

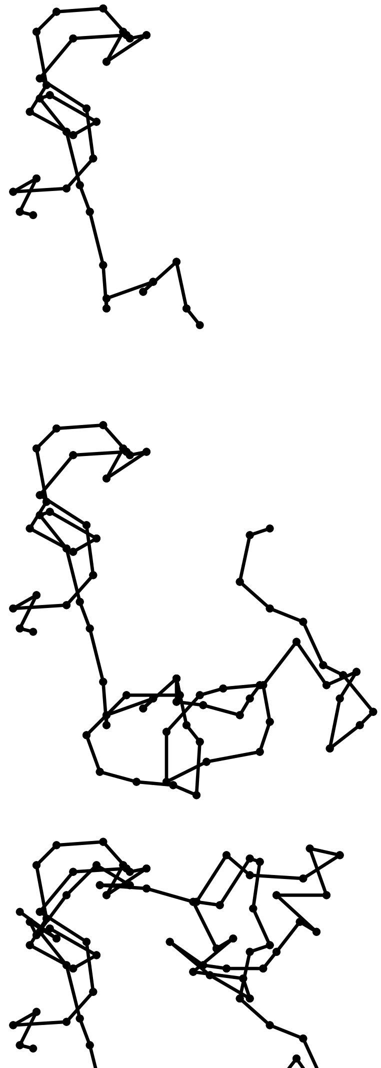
DFS and BFS also immediate from PFS

    BFS: priority is "time since start"
    DFS: priority is "time to finish"

PFS SPT example



PFS BFS and DFS examples



Shortest paths in Euclidean graphs

Problem:

    find shortest path from x to y

Algorithm:

    start shortest-path PFS at x
    stop when reaching y

val[k]:

    shortest distance from x to k in TREE
    PLUS distance from k to y as the crow flies

priority on FRINGE:

    shortest possible way from x to y

```
#define priority
  val[k]+t->w + distance(t_>w, y)
```

All shortest paths

Problem:

Given weighted graph, compute
a table giving the shortest weighted
path from each node to each other node

Ex: map of New England
nodes: cities
edges: roads connecting cities

.	P	W	L	N
. **Providence**	**0**	**53**	**54**	**48**
. **Westerley**	**53**	**0**	**18**	**101**
. **newLondon**	**54**	**18**	**0**	**12**
. **Norwich**	**48**	**101**	**12**	**0**

Norwich-Westerly: 101 miles??

12 miles Norwich-New London
18 miles New London-Westerly
----
30 miles

Floyd-Warshall algorithm

Another ancient algorithm (1962)

Want shorter path from x to j?
    take x to y, then y to j, if shorter

For each x-y pair, do all j

```
for (y = 1; y <= V; y++)
  for (x = 1; x <= V; x++)
    for (j = 1; j <= V; j++)
      if (a[x][y] + a[y][j] < a[x][j])
```

Correctness proof:
    induction on y

Running time: O(V^3)

Negative edge weights?
- much more difficult to solve
- could get negative cycles