

CSE 332: Minimum Spanning Trees

Richard Anderson
Spring 2016

Announcements

- No class on Monday

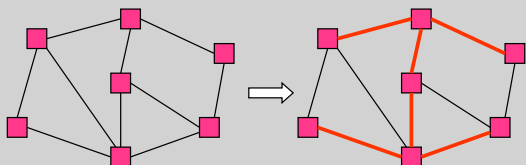
2

Union Find Review

- Data: set of pairwise **disjoint sets**.
- Operations
 - **Union** – merge two sets to create their union
 - **Find** – determine which set an item appears in
- Amortized complexity
 - M Union and Find operations, on a set of size N
 - Runtime $O(M \log^* N)$

3

Spanning Tree in a Graph



Spanning tree
- Connects all the vertices
- No cycles

4

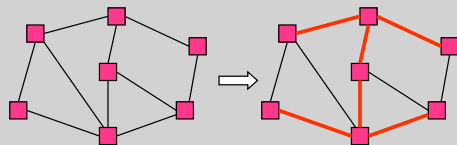
Undirected Graph

- $G = (V, E)$
 - V is a set of vertices (or nodes)
 - E is a set of unordered pairs of vertices
- $V = \{1, 2, 3, 4, 5, 6, 7\}$
 $E = \{(1, 2), (1, 6), (1, 5), (2, 7), (2, 3), (3, 4), (4, 7), (4, 5), (5, 6)\}$
- 2 and 3 are adjacent
2 is incident to edge (2, 3)

5

Spanning Tree Problem

- Input: An undirected graph $G = (V, E)$. G is connected.
- Output: $T \subset E$ such that
 - (V, T) is a connected graph
 - (V, T) has no cycles



6

Spanning Tree Algorithm

```

ST(Vertex i) {
  mark i;
  for each j adjacent to i {
    if (j is unmarked) {
      Add (i,j) to T;
      ST(j);
    }
  }
}

```

```

Main() {
  T = empty set;
  ST(1);
}

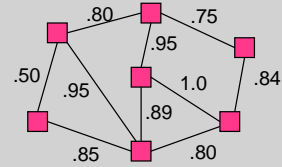
```

7

Best Spanning Tree

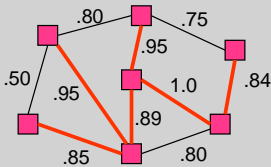
Finding a reliable routing subnetwork:

- edge cost = probability that it won't fail
- Find the spanning tree that is least likely to fail



8

Example of a Spanning Tree



Probability of success = $.85 \times .95 \times .89 \times .95 \times 1.0 \times .84$
 $= .5735$

9

Minimum Spanning Trees

Given an undirected graph $G=(V,E)$, find a graph $G'=(V,E')$ such that:

- E' is a subset of E
- $|E'| = |V| - 1$
- G' is connected
- $\sum_{(u,v) \in E'} c_{uv}$ is minimal

G' is a **minimum spanning tree**.

10

Minimum Spanning Tree Problem

- Input: Undirected Graph $G = (V,E)$ and $C(e)$ is the cost of edge e .
- Output: A spanning tree T with minimum total cost. That is: T that minimizes

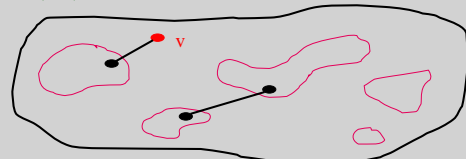
$$C(T) = \sum_{e \in T} C(e)$$

11

Kruskal's MST Algorithm

Idea: Grow a **forest** out of edges that do not create a cycle. Pick an edge with the smallest weight.

$G=(V,E)$



12

Kruskal's Algorithm for MST

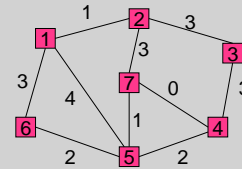
An *edge-based* greedy algorithm
Builds MST by greedily adding edges

1. Initialize with
 - empty MST
 - all vertices marked unconnected
 - all edges unmarked
2. While there are still unmarked edges
 - a. Pick the lowest cost edge (u, v) and mark it
 - b. If u and v are not already connected, add (u, v) to the MST and mark u and v as connected to each other

Sound familiar?

13

Example of for Kruskal



(7,4) (2,1) (7,5) (5,6) (5,4) (1,6) (2,7) (2,3) (3,4) (1,5)
0 1 1 2 2 3 3 3 3 4

14

Data Structures for Kruskal

- Sorted edge list

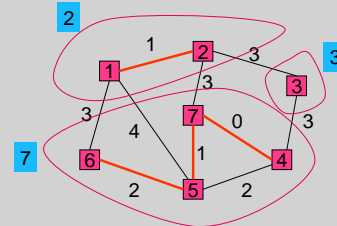
(7,4) (2,1) (7,5) (5,6) (5,4) (1,6) (2,7) (2,3) (3,4) (1,5)
0 1 1 2 2 3 3 3 3 4

- Disjoint Union / Find

- Union(a,b) - merge the disjoint sets named by a and b
- Find(a) returns the name of the set containing a

15

Example of DU/F

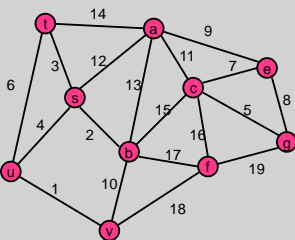


~~(7,4) (2,1) (7,5) (5,6) (5,4) (1,6) (2,7) (2,3) (3,4) (1,5)~~
~~0 1 1 2 2 3 3 3 3 4~~

16

Kruskal's Algorithm

- Add the cheapest edge that joins disjoint components



Kruskal's Algorithm with DU / F

```
Sort the edges by increasing cost;
Initialize A to be empty;
for each edge (i,j) chosen in increasing order do
  u := Find(i);
  v := Find(j);
  if not(u = v) then
    add (i,j) to A;
    Union(u,v);
```

This algorithm will work, but it goes through all the edges.

Is this always necessary?

18

Kruskal code

```

void Graph::kruskal(){
  int edgesAccepted = 0;
  DisjSet s(NUM_VERTICES);
  while (edgesAccepted < NUM_VERTICES - 1){
    e = smallest weight edge not deleted yet;
    // edge e = (u, v)
    uset = s.find(u);
    vset = s.find(v);
    if (uset != vset){
      edgesAccepted++;
      s.unionSets(uset, vset);
    }
  }
}
  
```

Total Cost:

19

Kruskal's Algorithm: Correctness

It clearly generates a spanning tree. Call it T_K .

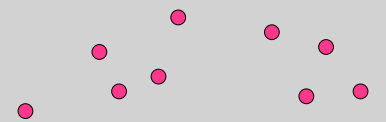
Suppose T_K is *not* minimum:

- Pick another spanning tree T_{min} with *lower cost* than T_K
- Pick the smallest edge $e_1=(u,v)$ in T_K that is not in T_{min}
- T_{min} already has a path p in T_{min} from u to v
- ⇒ Adding e_1 to T_{min} will create a cycle in T_{min}
- Pick an edge e_2 in p that Kruskal's algorithm considered *after* adding e_1 (must exist: u and v unconnected when e_1 considered)
- ⇒ $cost(e_2) \geq cost(e_1)$
- ⇒ can replace e_2 with e_1 in T_{min} without increasing cost!
- Keep doing this until T_{min} is identical to T_K
- ⇒ T_K must also be minimal – contradiction!

20

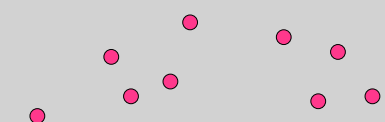
MST Application: Clustering

- Given a collection of points in an r -dimensional space, and an integer K , divide the points into K sets that are closest together

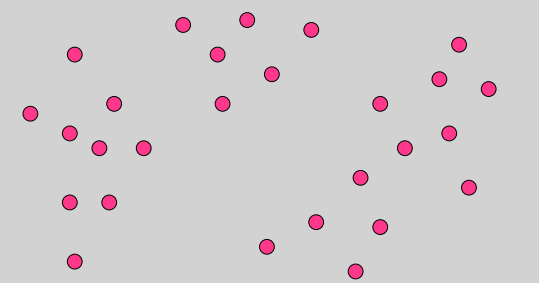


Distance clustering

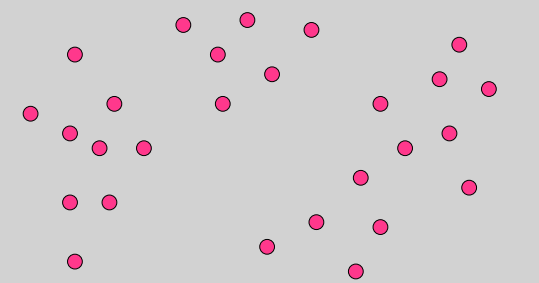
- Divide the data set into K subsets to maximize the distance between any pair of sets
- $dist(S_1, S_2) = \min \{dist(x, y) \mid x \text{ in } S_1, y \text{ in } S_2\}$



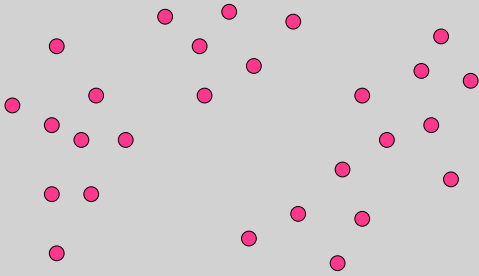
Divide into 2 clusters



Divide into 3 clusters



Divide into 4 clusters



Distance Clustering Algorithm

Let $C = \{\{v_1\}, \{v_2\}, \dots, \{v_n\}\}$; $T = \{\}$

while $|C| > K$

Let $e = (u, v)$ with u in C_i and v in C_j be the minimum cost edge joining distinct sets in C

Replace C_i and C_j by $C_i \cup C_j$

K-clustering

