Minimum Spanning Trees:
- Given a connected, weighted graph \( G = (V, E) \)
  \( (u, v) \in E \rightarrow w(u, v) = \text{weight} \)
- Spanning Tree:
  - A subset \( T \subseteq E \) of edges that connect all the vertices and is acyclic
  - Total weight: \( w(T) = \sum_{(u, v) \in T} w(u, v) \)

Minimum Spanning Tree (MST):
- A spanning tree of min. weight

Facts:
- If \( G \) has \( n \) vertices, any spanning tree has \( n-1 \) edges

How are data structures used?
- Transaction/Query:
  - Insert new student
    name = "Mary" ID = 1234...
  - Closest coffee to my location
- Algorithms:
  - Dijkstra - Fibonacci Heap
  - Kruskal - Union/Find

Algorithms for MSTs:
- Based on greedy construction
  - Add the lightest edge that causes no cycle

Kruskal's, Prim's, Borůvka's

Lemma: Given any cut \( (S, P \setminus S) \)
  always safe to add lightest edge \( (p_i, p_j) \) \( p_i \in S, p_j \in P \setminus S \)

Applications:
- Clustering (Machine Learning)
- Approximation (TSP)
- Networking

Data Structures & Algorithm Design:
Euclidean Min Spanning Tree (I)

Euclidean Graph:
Given a set \( P = \{p_1, \ldots, p_n\} \)
  of pts in \( \mathbb{R}^d \), this is a complete graph (all \( \binom{n}{2} \) edges)
  where:
  \( w(p_i, p_j) = \text{dist}(p_i, p_j) = \frac{1}{\sqrt{(x_i-x_j)^2 + (y_i-y_j)^2}} \)

Euclidean MST (EMST):
- The MST of \( P \)

Euclidean MST

Dependents list \( \text{dep}(p_j) \) is list of all points \( p_i \) that depend on \( p_j \).
addEdge(Pair(Point) edge)

 Helpers:

 - Initialize (Point start)
 - addEdge (Pair(Point) edge)
 - add new edge to EMST
 - add new NN pair (pt, nn)

 Euclidean MSTs (III)

Q: Why check nn3 == null?
- On adding last pt to EMST, the kd-tree is empty.

addNN(Point pt, Point nn)
- QuakHeap<Double, Pair>
- dist = distanceSq(pt, nn)
- pair = new Pair(pt, nn)
- insert pair in heap w. priority dist
- add pt to dep[nn]

initialize (Point start)
- clear: edgelist in EMST
- heap + kdTree
- for each (dep in dependents)
  - clear dep
  - for each(pt in P)
    - if (pt != start) insert pt in kdTree

That's it!

Is this efficient?
- Assuming NN queries in O(log n) time

Total time = O(Cn^2log n + mlog n)
m = # of NN updates

⇒ Much depends on m, m depends on pt. distrib.
1. Prog Assign 2 due tonight
2. Prog Assign 3 out
   - can drop lowest score
3. HW 4?  - extra credit if you
do all three
4. Grades  - material covered on final