CS200: Graphs

Rosen Ch. 9.1-9.4, 9.6, 10.4-10.5
Walls and Mirrors Ch. 14

Trees as Graphs

- Tree: an undirected connected graph that has no cycles.

Rooted Trees

- A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.

When is a graph a Tree?

- Can explicitly check that the graph is connected and has no cycles. (How?)
- Want an alternative characterization

When is a graph a Tree?

- A connected undirected graph with n vertices must have at least n-1 edges (proof: by induction on the number of vertices)
- A connected undirected graph that has n vertices and exactly n-1 edges cannot contain a cycle (proof: by contradiction with previous statement)
- A connected undirected graph that has n vertices and more than n-1 edges must contain a cycle.
When is a graph a Tree?

- Conclusion: A connected graph with \( n \) vertices and \( n-1 \) edges is a tree.
- In order to check if a graph is a tree need to check that it is connected and count the number of edges and vertices.

Spanning Trees

- Spanning tree: A subgraph of a connected undirected graph \( G \) that contains all of \( G \)'s vertices and enough of its edges to form a tree.
- How to get a spanning tree:
  - Remove edges until you get a tree.
  - Add edges until you have a spanning tree

Spanning Trees - DFS algorithm

\[
\text{dfsTree}(\text{in } v: \text{vertex})
\]
Mark \( v \) as visited
for (each unvisited vertex \( u \) adjacent to \( v \))
  Mark the edge from \( u \) to \( v \)
  \text{dfsTree}(u)

Spanning Tree – Depth First Search Example

Minimum Spanning Tree

- Minimum spanning tree
  - spanning tree minimizing the sum of edge weights
- Example use
  - Connecting each house in the neighborhood to cable – graph where each house is a vertex. Need the graph to be connected, and minimize the cost of laying the cables.

Prim’s Algorithm

- Idea: incrementally build spanning tree by adding the least-cost edge to the tree
MST – Example

Prim’s Algorithm (Start at d)

![Graph diagram]

Prim’s Algorithm

```java
prims(in: G=(V,E):Graph)

//V_T – current vertices in spanning tree
//E_T – edges belonging to the spanning tree
V_T = {w} // w is an arbitrarily chosen vertex
E_T = φ // spanning tree contains no vertices initially

for i = 1 to |V| - 1 do

find a minimum-weight edge e=(u,v) among edges that connect a vertex in V_T with a vertex in V – V_T

add v to V_T

add e to E_T

return E_T
```

Implementing Prim’s Algorithm

- Each node not in the tree is associated with an attaching cost – the weight of the smallest edge that connects it to the forming tree (infinity if no such edge exists).
- At each iteration we retrieve the node with the smallest attaching cost and update the attaching cost of its neighbors.
- Can use a priority queue! (need to add a method for updating priorities).

Data Clustering

- Given a collection of objects – divide them into groups where objects in a group are “similar” (e.g. groups of users with similar taste).
- Graph abstraction:
  - Nodes – objects to be clustered
  - Edges – represent the similarity between objects
- Discuss the issue with the clustering scheme from the homework

Agglomerative Hierarchical Clustering

- Iteratively merge the two most similar clusters

![Hierarchical Tree diagram]
Single Linkage Clustering

- Single linkage: Similarity between clusters is the similarity between the two most similar objects
- Algorithm:
  - Sort the edges of the graph by weight
  - Add an edge between two most similar objects
  - Iteratively add edges, merging the two most similar connected components
- This is Kruskal’s algorithm for finding a minimum spanning tree!

Kruskal’s Algorithm

- Sort the edges in the graph by increasing weight
- Iterate through the sorted list of edges, inserting an edge as long it does not create a cycle

How do we know when to stop?

The Union-Find Data Structure

- Maintains a collection of disjoint sets
- Supports the following operations:
  - makeSet(u) – create a set containing element u
  - find(u) – returns the set that contains u
  - union(A, B) – merges A and B

Kruskal’s Algorithm with Union-Find

kruskals(in G=(V,E):Graph)

sort edges E by increasing weight
f = union-find structure with all vertices of G in it
E_T = \emptyset //spanning tree initially contains no vertices
for e = (u,v) in E //iterate through the edges
  if E_T has n-1 edges : return E_T
  if f.find(u) != f.find(v)
    add e to E_T
    f.union(f.find(u), f.find(v))
Union-Find: Tree-based Implementation
- Each set is a tree with parent pointers (the root is its own parent)
- The root of a tree represents the set. For example: the sets “1”, “2”, and “5”.

Union-Find Operations
- union - attach the root of one tree to the root of the other. Time: O(1)
- find - follow parent pointers from the starting node until reaching a node whose parent pointer refers back to itself

Union-Find Improvement 1
- Union by size:
  - When performing a union, make the root of smaller tree point to the root of the larger
  - O(log n) time per find operation:
    - Each time we follow a pointer this represents a merge that has at least doubled the size of the subtree.
    - Thus, we will follow at most O(log n) pointers for any find.

Union-Find Improvement 2
- Path compression:
  - After performing a find, compress all the pointers on the path just traversed so that they all point to the root
  - Yields O(n log^* n) time for performing n union-find operations (proof rather involved)

Analysis of Kruskal’s Algorithm
- Sorting the edges: O(|E| log |E|)
- We go through |E| edges at the most, and for each edge we do:
  - two find operations O(log |V|)
  - possibly a union operation O(1)
- Total: O(|E| log |E|) (same as O(|E| log |V|))

Shortest Path Algorithms
(Dijkstra’s Algorithm)
- Graph G(V,E) with non-negative weights (“distances”)
- Compute shortest distances from vertex s to every other vertex in the graph
Dijkstra's Algorithm

(Initialize)

Dijkstra's Algorithm

(Step 1)

Dijkstra's Algorithm

(Step 2)

Dijkstra's Algorithm

(Step 3)

Dijkstra's Algorithm

(Step 4)
Dijkstra’s Algorithm
(Done)

Dijkstra’s Algorithm
Using a priority queue

Dijkstra’s Algorithm

- How to obtain the shortest paths themselves?
  - At each vertex maintain a pointer that tells you from which vertex you arrived.

Shortest Path Algorithms
(Dijkstra’s Algorithm)

- Algorithm
  - Maintain array d (minimum distance estimates)
  - Init: d[s]=0, d[v]=∞
  - Priority queue of vertices not yet visited
  - select minimum distance vertex, visit v, update neighbors

MST as an optimization problem

We can formulate the MST problem as follows:
- Given a spanning tree of a weighted graph G=(V,E) with weights w(e) assigned to edges in E define a cost function for a spanning tree T as:
  \[ S(T) = \sum_{e \in E} w(e) \]
- Objective: find a tree T* that minimizes S(T)
Greedy Algorithms

- MST and shortest path are examples of optimization problems. In principle need to consider all possible solutions.
- Not feasible in most cases.
- We used a greedy approach instead: at each step we modified the solution in a way which was best given the limited information we had.
- In some cases the greedy approach leads to an optimal solution (Prim's, Kruskal's, Dijkstra's)