Today’s announcements:

Final exam 12/15, 7-10p, Locations TBA. Format…
Code challenge #2, 12/10, 9p, Siebel 0224.
Exam Review - 12/13, 2p, in Siebel ????

How would you characterize the difference between these graphs?
Prim’s algorithms (1957) is based on the Partition Property:

Consider a partition of the vertices of G into subsets U and V.

Let e be an edge of minimum weight across the partition.

Then e is part of some minimum spanning tree.

Proof:
See cs374
MST - minimum total weight spanning tree

Theorem suggests an algorithm...
Example of Prim’s algorithm -

Initialize structure:
1. For all $v$, $d[v] = \text{“infinity”}$, $p[v] = \text{null}$
2. Initialize source: $d[s] = 0$
3. Initialize priority (min) queue
4. Initialize set of labeled vertices to $\emptyset$.
Example of Prim’s algorithm -

Initialize structure:
1. For all $v$, $d[v] = \text{“infinity”}$, $p[v] = \text{null}$
2. Initialize source: $d[s] = 0$
3. Initialize priority (min) queue
4. Initialize set of labeled vertices to $\emptyset$.

Repeat these steps $n$ times:
• Find & remove minimum $d[]$ unlabelled vertex: $v$
• Label vertex $v$
• For all unlabelled neighbors $w$ of $v$,
  If cost($v,w$) < $d[w]$
  $d[w] = \text{cost}(v,w)$
  $p[w] = v$
Prim’s Algorithm (undirected graph with unconstrained edge weights):

**Initialize structure:**
1. For all v, d[v] = “infinity”, p[v] = null
2. Initialize source: d[s] = 0
3. Initialize priority (min) queue
4. Initialize set of labeled vertices to ∅.

Repeat these steps n times:
- Remove minimum d[] unlabeled vertex: v
- Label vertex v (set a flag)
- For all unlabeled neighbors w of v,
  - If cost(v,w) < d[w]
    - d[w] = cost(v,w)
    - p[w] = v

<table>
<thead>
<tr>
<th>adj mtx</th>
<th>adj list</th>
</tr>
</thead>
<tbody>
<tr>
<td>heap</td>
<td>O(n)</td>
</tr>
</tbody>
</table>

| Unsorted array | O(n) | O(n) |

**Which is best?**

*Depends on density of the graph:*

- Sparse
- Dense
Single source shortest path

Given a start vertex (source) s, find the path of least total cost from s to every vertex in the graph.
Single source shortest path:

Input: directed graph $G$ with non-negative edge weights, and a start vertex $s$.

Output: A subgraph $G'$ consisting of the shortest (minimum total cost) paths from $s$ to every other vertex in the graph.
Single source shortest path (directed graph with non-negative edge weights): Dijkstra’s Algorithm (1959)

Given a source vertex \( s \), we wish to find the shortest path from \( s \) to every other vertex in the graph.

Initialize structure:

Repeat these steps:
1. Label a new (unlabelled) vertex \( v \), whose shortest distance has been found
2. Update \( v \)'s neighbors with an improved distance
Single source shortest path *(directed graph w non-negative edge weights)*:

**Dijkstra’s Algorithm (1959)**

**Why non-negative edge weights??**

![Graph](image)

**Initialize structure:**

**Repeat these steps:**

1. Label a new (unlabelled) vertex v, whose shortest distance has been found

2. Update v’s neighbors with an improved distance
**Single source shortest path** (directed graph w non-negative edge weights):

Initialize structure:
1. For all v, d[v] = “infinity”, p[v] = null
2. Initialize source: d[s] = 0
3. Initialize priority (min) queue

Repeat these steps n times:
- Find minimum d[] unlabelled vertex: v
- Label vertex v
- For all unlabelled neighbors w of v,
  - If (____________ < d[w])
    - d[w] = ______________
    - p[w] = v

Running time?