Partition Property
Consider an arbitrary partition of the vertices on G into two 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 in CS 374!

Partition Property Algorithm

<table>
<thead>
<tr>
<th>Pseudocode for Prim’s MST Algorithm</th>
</tr>
</thead>
</table>
| PrimMST(G, s):
| 1 Input: G, Graph;
| 2 s, vertex in G, starting vertex of algorithm
| 3 Output: T, a minimum spanning tree (MST) of G
| 4 foreach (Vertex v : G):
| 5 d[v] = +inf
| 6 p[v] = NULL
| 7 d[s] = 0
| 8 PriorityQueue Q // min distance, defined by d[v]
| 9 Q.buildHeap(G.vertices())
| 10 Graph T // "labeled set"
| 11 repeat n times:
| 12 Vertex m = Q.removeMin()
| 13 T.add(m)
| 14 foreach (Vertex v : neighbors of m not in T):
| 15 if cost(v, m) < d[v]:
| 16 d[v] = cost(v, m)
| 17 p[v] = m
| 18 return T

Heap

Adj. Matrix | Adj. List
---|---

Unsorted Array
Running Time of MST Algorithms

Kruskal's Algorithm:

Prim's Algorithm:

**Q:** What must be true about the connectivity of a graph when running an MST algorithm?

...what does this imply about the relationship between $n$ and $m$?

<table>
<thead>
<tr>
<th>Kruskal’s MST</th>
<th>Prim’s MST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Q:** Suppose we built a new heap that optimized the decrease-key operation, where decreasing the value of a key in a heap updates the heap in amortized constant time, or $O(1)^*$. How does that change Prim's Algorithm runtime?

---

Final big-O Running Times of classical MST algorithms:

<table>
<thead>
<tr>
<th>Kruskal’s MST</th>
<th>Prim’s MST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

CS 225 – Things To Be Doing:

1. Get your projects approved and start work on them.
2. mp_mazes due today.
3. No new mp this week.
4. Daily POTDs are ongoing.