As always, sit with a partner and work through these together.

**Activity 1: Prim-Jarnik**

while PQ not empty:
N = removeMin and circle it
connect N to MST (except start node)
for each neighbor M of N:
update distance to M if smaller
Activity 2: Runtime of Prim-Jarnik’s
Find the runtime of Prim-Jarnik’s Algorithm based on the following pseudocode by filling in the runtime for each appropriate line of code.

function prim(G):
  for all v in V:----------------------------------------- 1. O(_______)
    v.cost = \infty
    v.prev = null
  source = a random v in V
  source.cost = 0
  MST = []
PQ = PriorityQueue(V)---------------------------------- 2. O(_______)
while PQ is not empty:----------------------------------- 3. O(_______)
  v = PQ.removeMin()---------------------------------- 4. O(_______)
  if v.prev != null:
    MST.append((v, v.prev))
    for all incident edges (v,u) of v:------------------- 5. O(_______)
      if u.cost > (v,u).weight:
        u.cost = (v,u).weight
        u.prev = v
        PQ.replaceKey(u, u.cost)-------------------------- 6. O(_______)
return MST

Runtime of Prim-Jarnik’s: ____________________