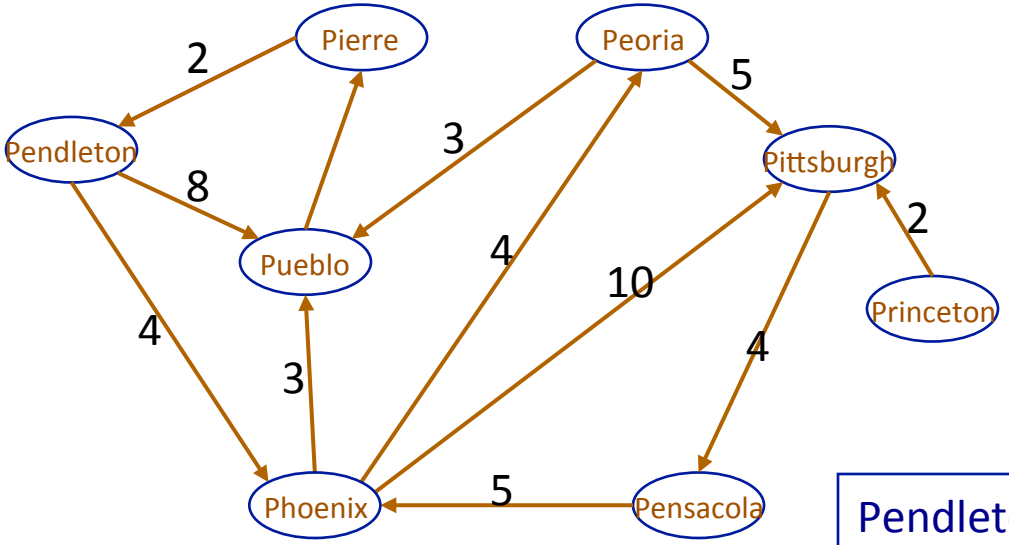
CS261 Data Structures

Dijkstra's Algorithm

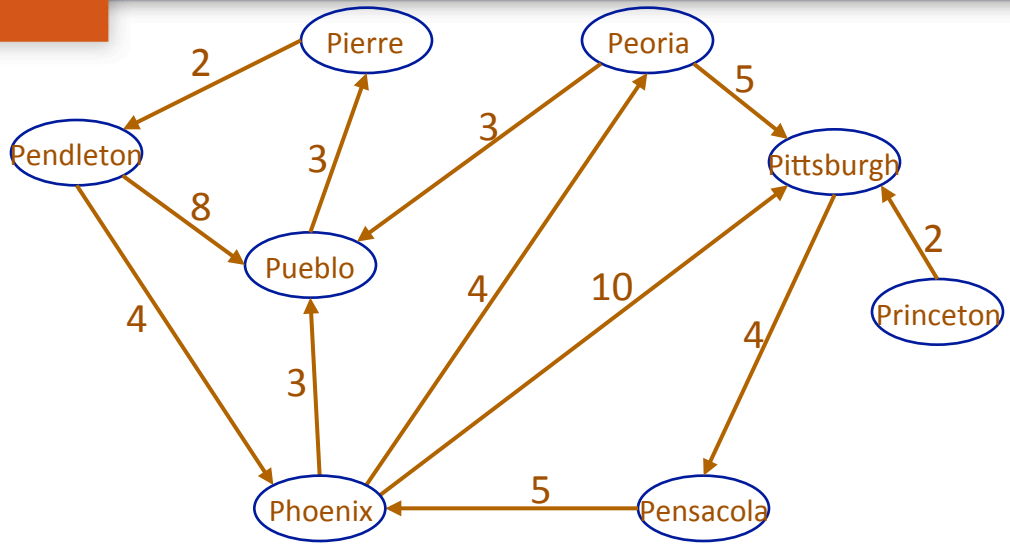
Weighted Graphs Representation: Edge List



What's reachable AND
what is the cost to get there?

Pendleton: {Pueblo:8, Phoenix:4}
Pensacola: {Phoenix:5}
Peoria: {Pueblo:3, Pittsburgh:5}
Phoenix: {Pueblo:3, Peoria:4, Pittsburgh:10}
Pierre: {Pendleton:2}
Pittsburgh: {Pensacola:4}
Princeton: {Pittsburgh:2}
Pueblo: {Pierre:3}

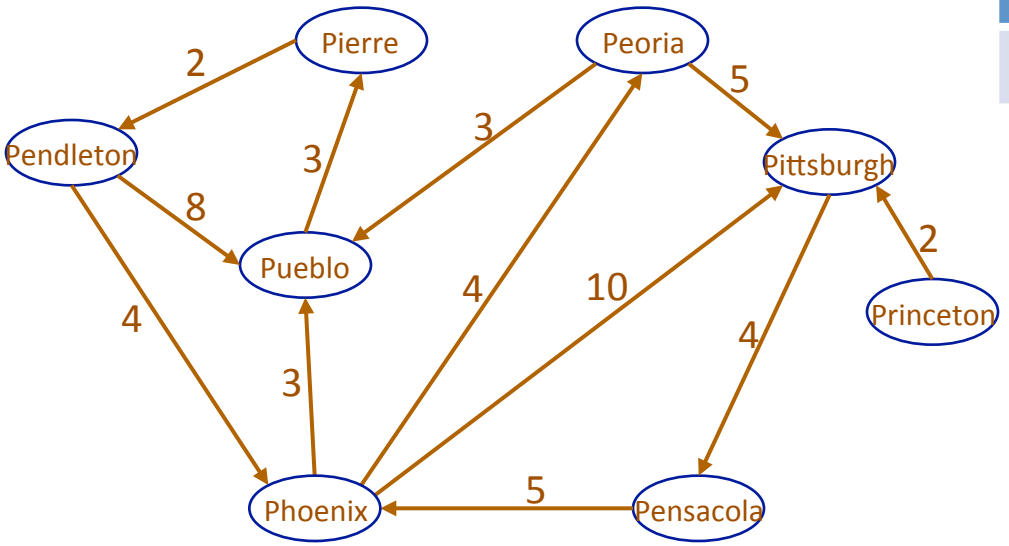
Dijkstra's Algorithm



Cost -First Search

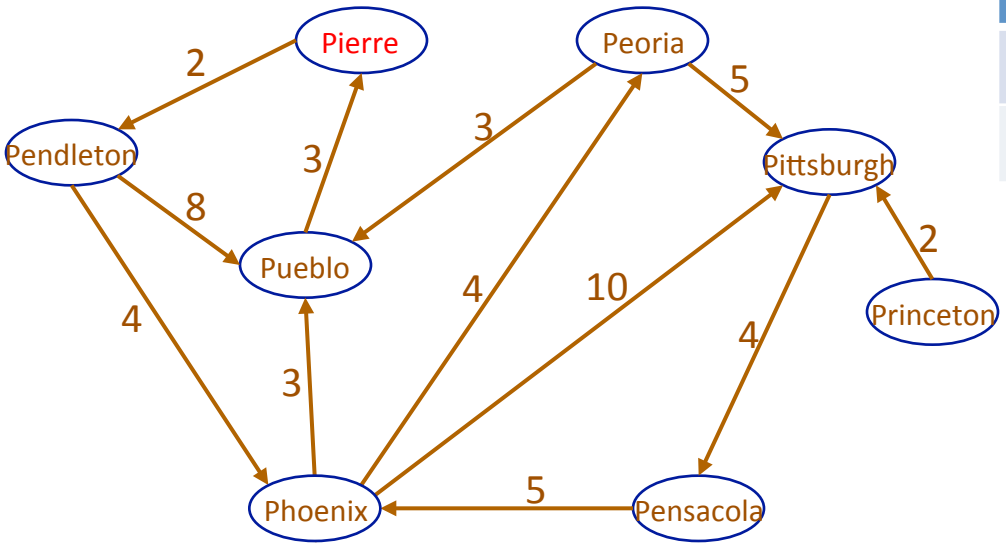
Initialize map of reachable vertices,
and add source vertex,v_i, to a priority queue with distance zero
While priority queue is not empty
    Getmin from priority queue and assign to v
    If v is not in reachable
        add v with given cost to map of reachable vertices
        For all neighbors, v_j, of v
            If v_j is not is set of reachable vertices, combine cost of reaching v with cost to travel from v to v_j, and add to priority queue

Example: What is the distance from Pierre?



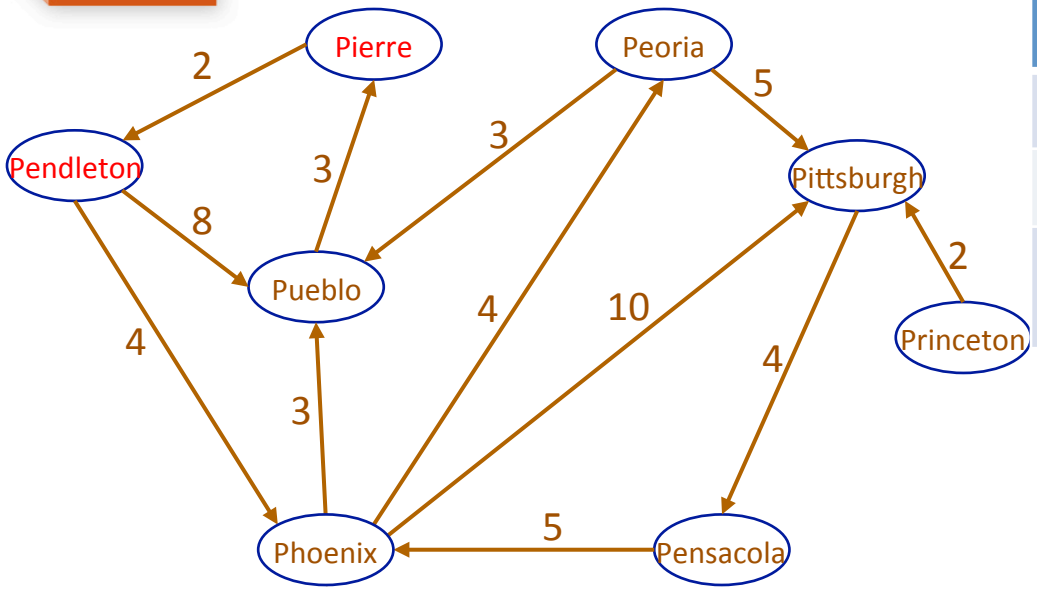
	Pqueue	Reachable
0	pierre(0)	{}

Example: What is the distance from Pierre?



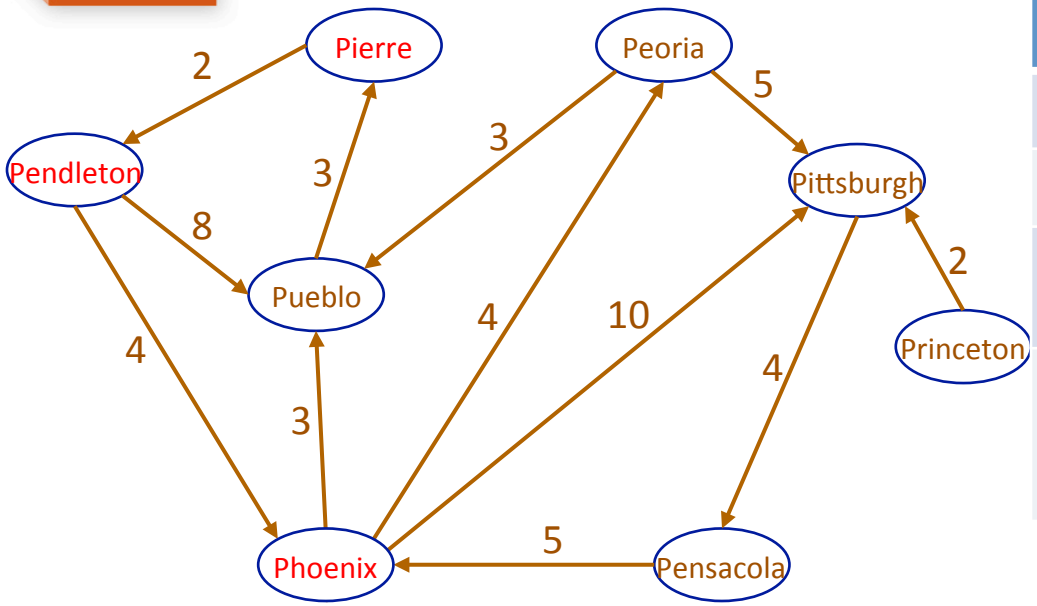
	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)

Example: What is the distance from Pierre?



	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)

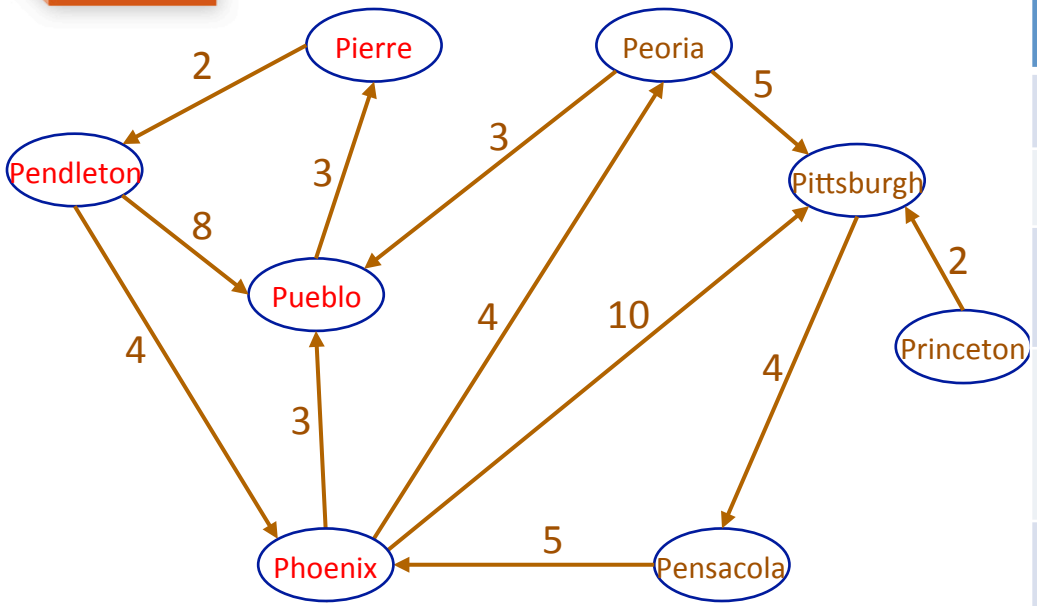
Example: What is the distance from Pierre?



	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)

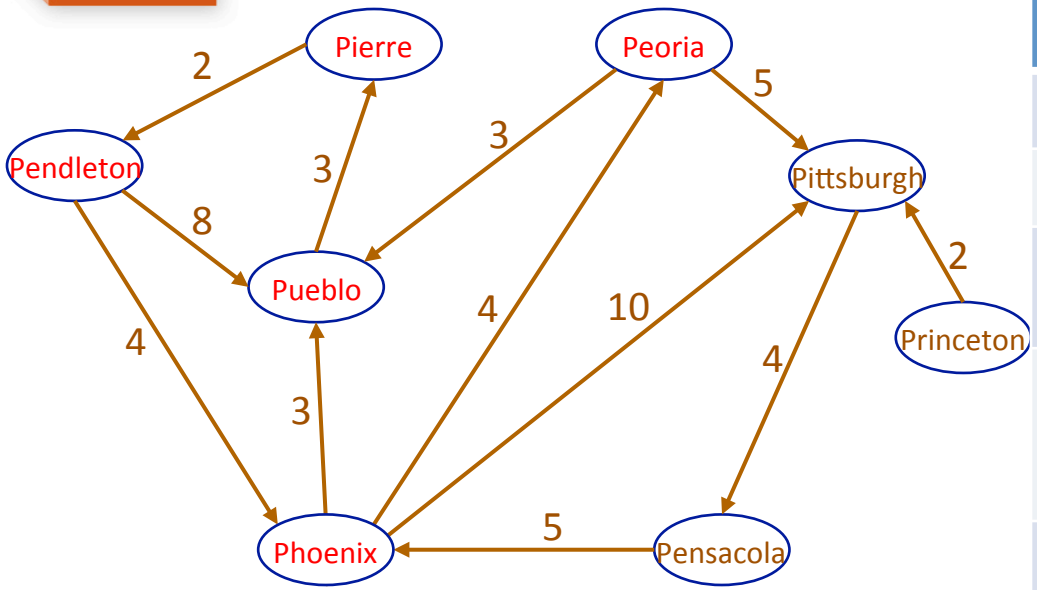
NOTE: Reachable is only showing the latest node added to the collection!

Example: What is the distance from Pierre?



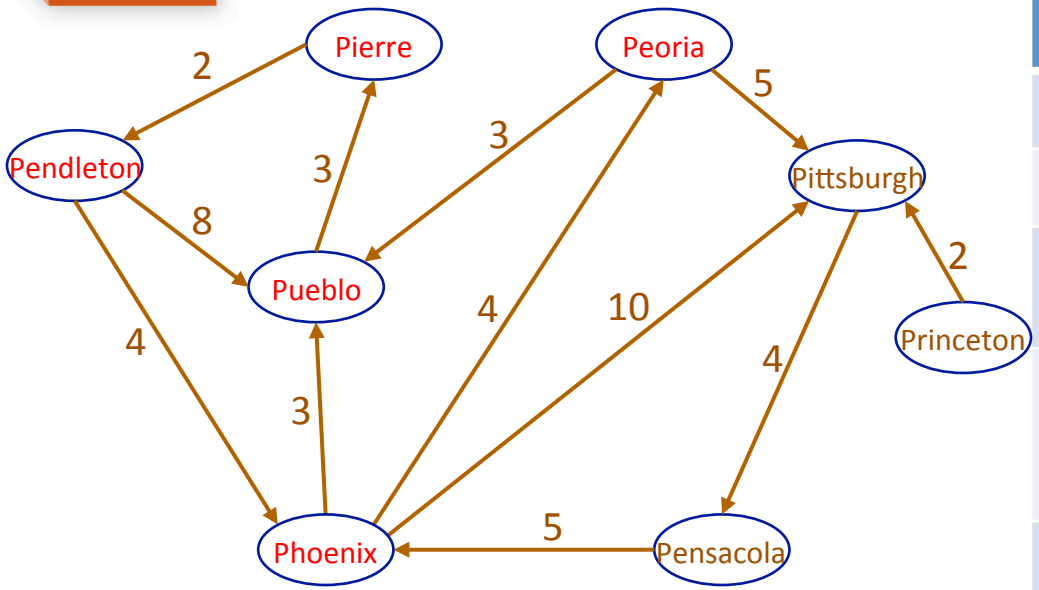
	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)

Example: What is the distance from Pierre?



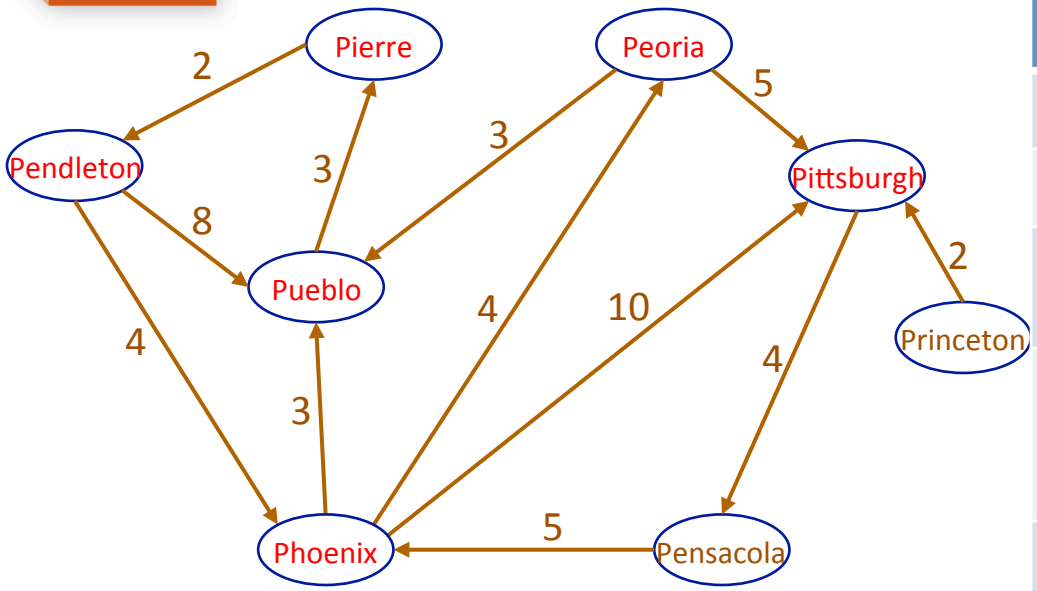
	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)
5	pueblo(10) pittsburgh(15), pittsburgh(16),	peoria(10)

Example: What is the distance from Pierre?



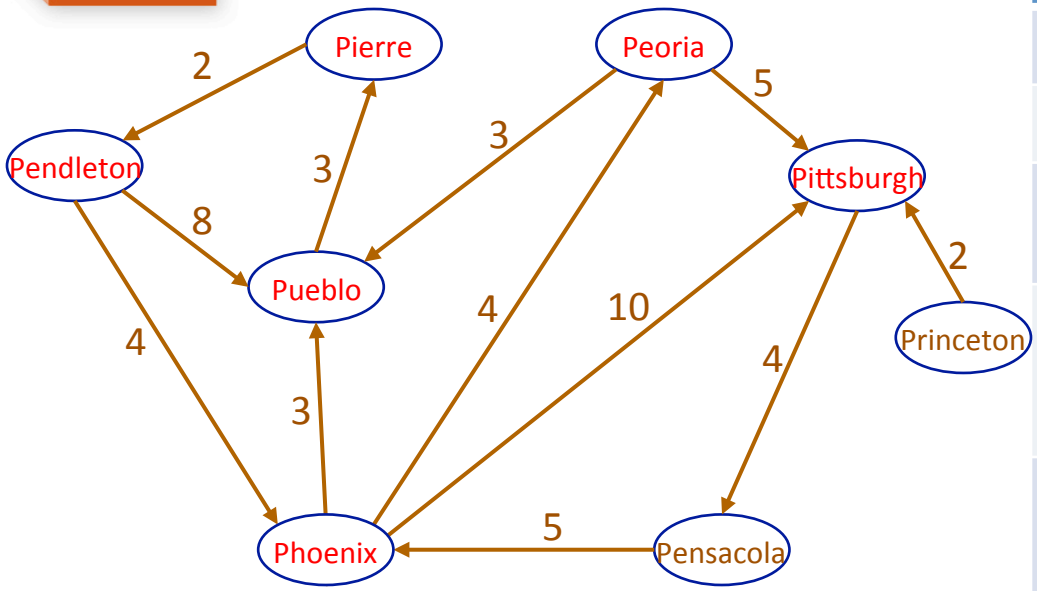
	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)
5	pueblo(10) pittsburgh(15), pittsburgh(16),	peoria(10)
6	pittsburgh(15), pittsburgh(16)	--

Example: What is the distance from Pierre?



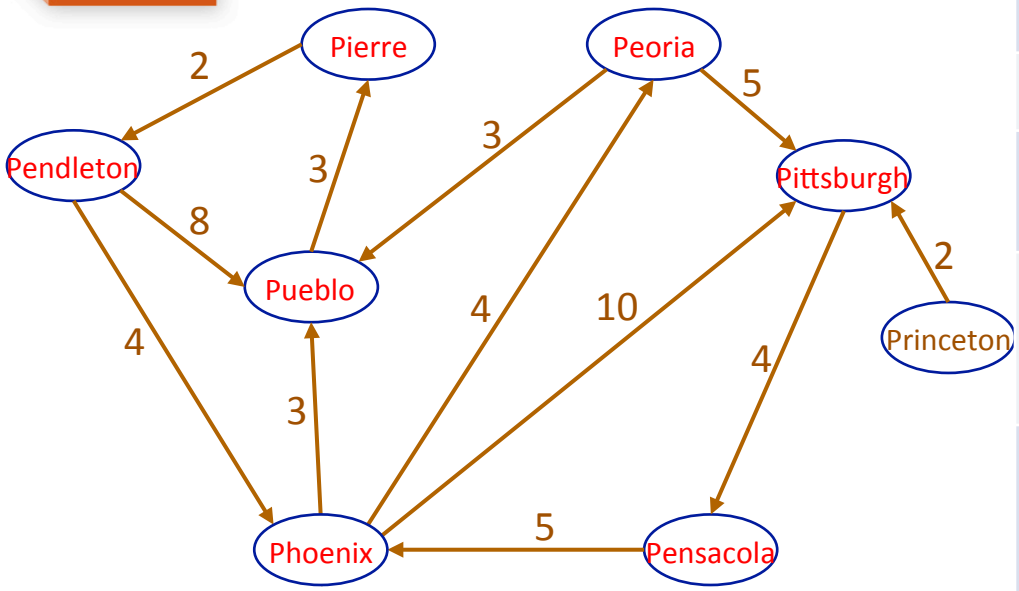
	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)
5	pueblo(10) pittsburgh(15), pittsburgh(16),	peoria(10)
6	pittsburgh(15), pittsburgh(16)	--
7	pittsburgh(16), pensacola(19)	pittsburgh(15)

Example: What is the distance from Pierre?



	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)
5	pueblo(10) pittsburgh(15), pittsburgh(16),	peoria(10)
6	pittsburgh(15), pittsburgh(16)	--
7	pittsburgh(16), pensacola(19)	pittsburgh(15)
8	pensacola(19)	--

Example: What is the distance from Pierre?



	Pqueue	Reachable
0	pierre(0)	{}
1	pendleton(2)	pierre(0)
2	phoenix(6), pueble(10)	pendleton(2)
3	pueblo(9), peoria(10), pittsburgh(16), pueblo(10)	phoenix(6)
4	peoria(10), pittsburgh(16), pueblo(10)	pueblo(9)
5	pueblo(10) pittsburgh(15), pittsburgh(16),	peoria(10)
6	pittsburgh(15), pittsburgh(16)	--
7	pittsburgh(16), pensacola(19)	pittsburgh(15)
8	pensacola(19)	--
9	{}	pensacola(19)

Dijkstra's

- Cost-first search
- Always explores the next node with the CUMULATIVE least cost
- Our implementation: O(V+E Log E)
  - Key observation: Inner loop runs at most E times
  - Time to add/rem from pqueue is bounded by logE since all neighbors, or edges, can potentially be on the pqueue
  - V comes from the initialization of reachable

- O(V+ E Log E)
  - Key observation: Inner loop runs at most E times
  - Time to add/rem from pqueue is bounded by logE since all neighbors, or edges, can potentially be on the pqueue

Initialize map of reachable vertices,
and add source vertex,v_i, to a priority queue with distance zero
While priority queue is not empty
    Getmin from priority queue and assign to v
    If v is not in reachable
        add v with given cost to map of reachable vertices
        For all neighbors, v_j, of v
            If v_j is not is set of reachable vertices, combine cost of reaching v with cost to travel from v to v_j, and add to priority queue

Summary

- Same code, three different ADTs result in three kinds of searches!!!
  - DFS (Stack)
  - BFS (Queue)
  - Dijkstras Cost-First Search (Pqueue)

Initialize set of *reachable* vertices and add v_i to a [stack, queue, pqueue]
While [stack, queue, pqueue] is not empty
    Get and remove [top, first, min] vertex *v* from [stack, queue, pqueue]
    if vertex v is not in reachable,
        add it to reachable
        For all neighbors, v_j, of *v* , not already in reachable
                add to [stack, queue, pqueue]
                (in case of pqueue, add with cumulative cost)

Implementation of Dijkstra's

- Pqueue: dynamic array heap
- Reachable:
  - Array indexed by node num
  - map: name, distance
  - hashMap
- Graph Representation: edge list with weights[map of maps]
  - Key: Node name
  - Value: Map of Neighboring nodes
    - Key: node name of one of the neighbors
    - Value: weight to that neighbor

Your Turn

- Complete Worksheet #42: Dijkstra's Algorithm