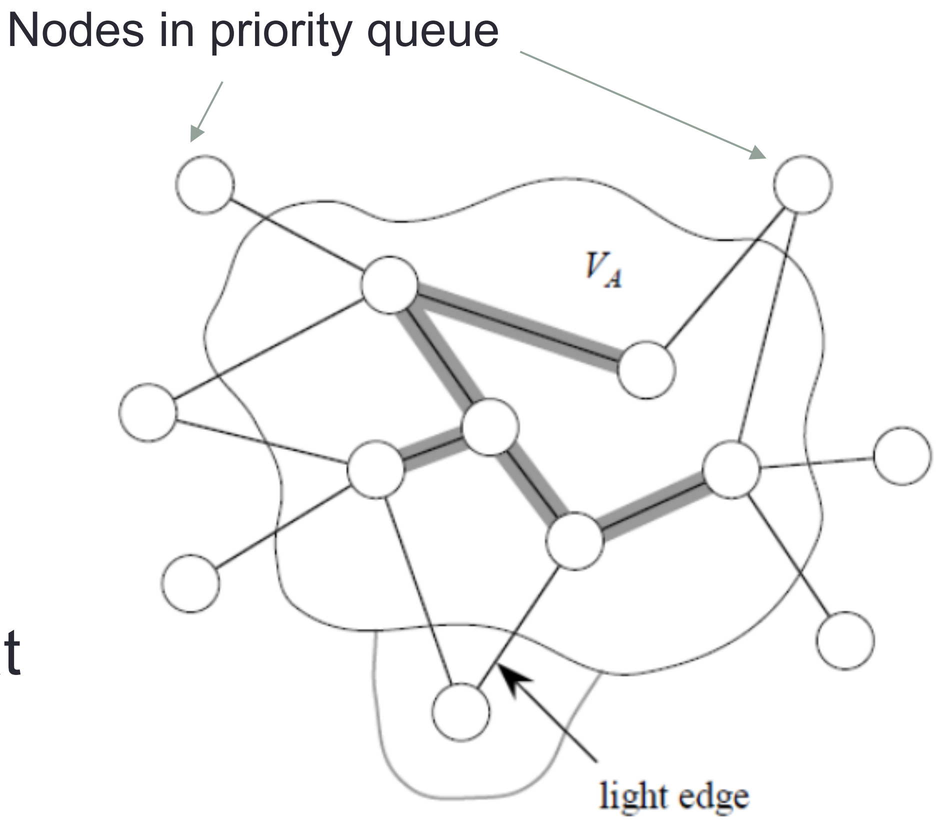
PROBLEM SOLVING 8

Prim and Dijkstra, Shortest Paths

CS340

Prim's Algorithm

- Builds one tree, so A is always a tree.
- Starts from an arbitrary “root” r.
- At each step, find a light edge crossing cut (V_A, $V-V_A$), where V_A = vertices that A is incident on. Add this edge to A.



Prim's Algorithm

- How to find the lightest edge quickly?
  Use a priority queue:
  - Each object is a vertex in $V - V_A$.
  - Key is minimum weight of any edge (u,v) where $u \in V_A$.
  - Then the vertex returned by EXTRACT-MIN is v such that there exists $u \in VA$ and (u,v) is light edge crossing $(V_A, V\text{-}V_A/$.
- Key of v is ∞ if v is not adjacent to any vertices in V_A.

Prim's Algorithm

```
PRIM(G, w, r)
  Q = ∅
  for each u ∈ G.V
      u.key = ∞
      u.π = NIL
      INSERT(Q, u)
  DECREASE-KEY(Q, r, 0)      // r.key = 0
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      for each v ∈ G.Adj[u]
          if v ∈ Q and w(u, v) < v.key
              v.π = u
              DECREASE-KEY(Q, v, w(u, v))
```

Notice that u is not changed.
U is added to the MST.
V is changed.

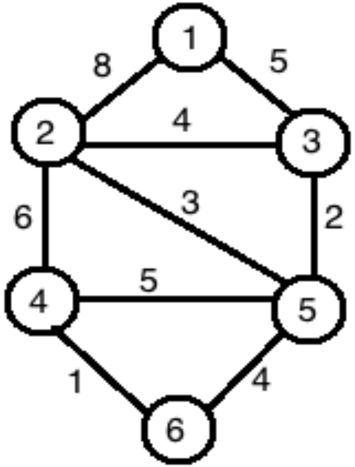
We don’t know where v is in the priority queue, so use the location table.

Prim Efficiency

- Suppose Q is a binary heap.
- Initialize Q and first for loop: $O(V \lg V)$
- Decrease key of r: $O(\lg V)$
- while loop: $|V|$ EXTRACT-MIN calls $= O(V \lg V) \geq$
  $|E|$ DECREASE-KEY calls $= O(E \lg V)$
- Total: $O(E \lg V)$

Interview Questions

parent	nil					
step	1	2	3	4	5	6
init						
1						
2						
3						
4						
5						
6						



Interview Questions

- Can Prim and/or Kruskal be used to find a maximum spanning tree?

SHORTEST PATHS

Optimal substructure of a shortest path

- A shortest path between two vertices contains other shortest paths within it.

Initialization

- All the shortest-paths algorithms start with INIT-SINGLE-SOURCE.

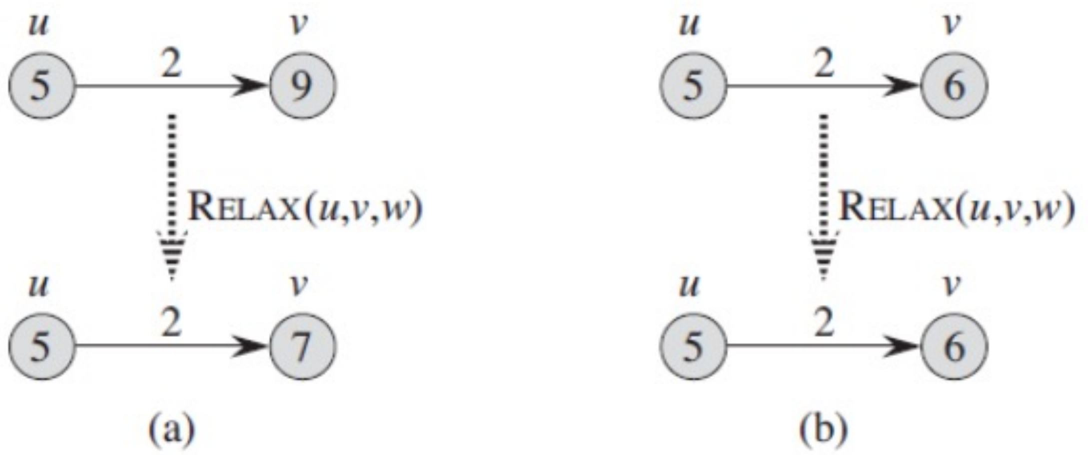
INITIALIZE-SINGLE-SOURCE(G, s)

```
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

This was called "key" in Prim's algorithm

Relaxing an edge

- Can we improve the shortest-path estimate for v by going through u and taking (u,v)?



Relaxing an edge

RELAX(u, v, w)

1 **if** $v.d > u.d + w(u, v)$

2 $\quad v.d = u.d + w(u, v)$

3 $\quad v.\pi = u$

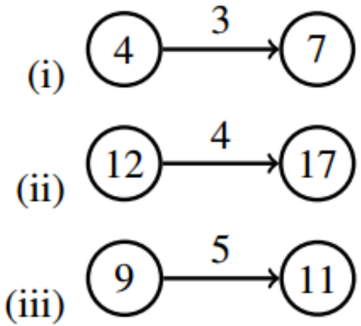
This is a decreaseKey()
Or for us, changeKey()

Properties of shortest paths and relaxation

- Path-relaxation property
  - If $p = <v_0, v_1,\ldots,v_k>$ i is a shortest path from $s = v_0$ to v_k, and we relax the edges of p in the order $(v_0, v_1), (v_1,v_2),\ldots(v_{k-1},v_k)$. then $v_k.d = \delta(s, v_k)$. This property holds **regardless** of any other relaxation steps that occur, even if they are intermixed with relaxations of the edges of p.

Interview Questions

What is the result of relaxing the following edges?



Dijkstra's Algorithm

- No **negative-weight** edges.
- Essentially a weighted version of breadth-first search.
- Instead of a FIFO queue, uses a priority queue.
- Keys are shortest-path weights (v.d).
- Have two sets of vertices:
  - S = vertices whose final shortest-path weights are determined
  - Q = priority queue = V-S.

Dijkstra's Algorithm

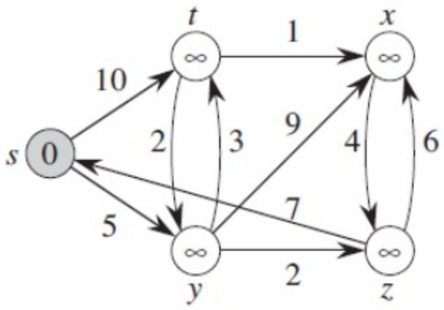
```
DIJKSTRA(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  S = ∅
  Q = G.V
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      S = S ∪ {u}
      for each vertex v ∈ G.Adj[u]
          RELAX(u, v, w)
```

Note that u,v are the IDs of the nodes!

Dijkstra's Algorithm

- Looks a lot like Prim’s algorithm, but computing v.d, and using shortest-path weights as keys.
- Dijkstra’s algorithm can be viewed as greedy, since it always chooses the “lightest” (“closest”) vertex in V-S to add to S.

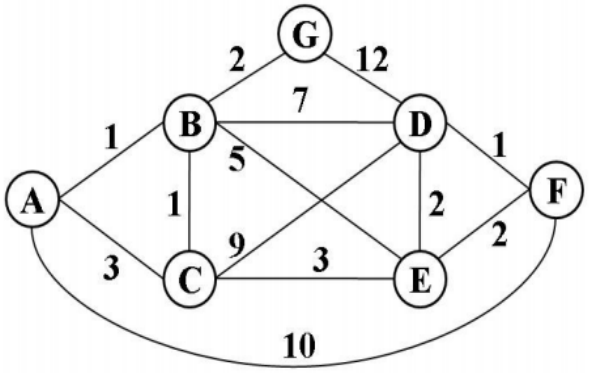
Dijkstra's Algorithm



parent	NIL				
step	s	t	x	y	z
init					
1					
2					
3					
4					
5					

Dijkstra’s Algorithm

Parent							
vertex	A	B	C	D	E	F	G
Init							
Step 1							
Step 2							
Step 3							
Step 4							
Step 5							
Step 6							
Step 7							



Time Complexity of Dijkstra

- Time complexity depends on how it is implemented
- Matrix:
  - Each EXTRACT-MIN takes O(V) time to look through the array
  - There are V EXTRACT-MIN instructions for $O(V^2)$
- Priority Queue
  - The algorithm is only 1 line different from Prim
  - O(E lg V)

Interview Questions

- Suppose we change Dijkstra’s algorithm such that the last vertex is not removed from the priority queue, and the while loop to executes |V|-1 times instead of |V| times. Is this proposed algorithm correct?

Interview Question

- To implement Dijkstra’s shortest paths algorithm on unweighted graphs so that it runs in linear time, what data structure can be used?
- 1. Stack
- 2. Queue
- 3. Priority Queue
- 4. All of the above
- 5. None of the above

Interview Questions

- In a weighted graph, assume that the shortest path from a source s to a destination t is correctly calculated using Dijkstra's algorithm. Is the following statement true? If we increase weight of every edge by 1, the shortest path always remains the same.

Interview Questions

- Given a graph, suppose we have calculated shortest path from a source to all other vertices. If we modify the graph such that the weights of all edges are doubled, does the shortest path remain the same?
- Given a weighted graph where weights of all edges are unique (no two edges have same weights), is there always a unique shortest path from a source to destination in such a graph?
- Each edge in a connected, unweighted graph G is colored either red or blue. Present an algorithm to compute a path between s and t that traverses the fewest number of red edges. Analyze its running time.