UNIT 4: Greedy Technique

Greedy Technique:

Dijkstra’s algorithm

Variants of the shortest path problem:

- Single- source shortest - paths problem
- Single- destination shortest - paths problem
- Single - pair shortest - path problem
- All - pairs shortest - paths problem

Algorithms to solve shortest path problem:

- **Dijkstra's algorithm** solves the single-source shortest paths problem with non-negative edge weight.
- **Bellman–Ford algorithm** solves the single-source shortest paths if edge weights may be negative. *But do not solve if there is a negative cycle.*
- **Floyd–Warshall algorithm** solves all pairs shortest paths.
- **Johnson's algorithm** solves all pairs shortest paths, and may be faster than Floyd–Warshall on sparse graphs.
- **A* search algorithm** solves for single-pair shortest path using heuristics to try to speed up the search.

Shortest paths applications:

- Driving directions on web mapping
- If a nondeterministic abstract machine is represented as a graph where vertices describe states and edges describe possible transitions, shortest path algorithms can be used to find an optimal sequence of choices to reach a certain goal state, or to establish lower bounds on the time needed to reach a given state

  if vertices represent the states of a puzzle like a Rubik's Cube and each directed edge corresponds to a single move or turn, shortest path algorithms can be used to find a solution that uses the minimum possible number of moves.

- networking or telecommunications
- operations research: plant and facility layout planning , robotics, transportation
- VLSI design

Dijkstra’s algorithm to find Single Source Shortest Paths

Single Source Shortest Paths Problem

- For a given vertex called the source in a weighted connected graph, find shortest paths to all its other vertices.
- The single-source shortest-paths problem asks for a family of paths, each leading from the source to a different vertex in the graph, though some paths may, of course, have edges in common.

NOTE:

BFS algorithm finds shortest path; works on unweighted graphs; each edge is considered to have unit weight.

Dijkstra's algorithm is usually the working principle behind

- link-state routing protocols,
- Open Shortest Path First (OSPF) and
- Intermediate System to Intermediate System(IS-IS)

```
ALGORITHM  Dijkstra(G, s)
    //Dijkstra's algorithm for single-source shortest paths
    //Input: A weighted connected graph G = ⟨V, E⟩ with nonnegative weights
    //          and its vertex s
    //Output: The length d_v of a shortest path from s to v
    //           and its penultimate vertex p_v for every vertex v in V
    Initialize(Q)   //initialize vertex priority queue to empty
    for every vertex v in V do
        d_v ← ∞;   p_v ← null
        Insert(Q, v, d_v)   //initialize vertex priority in the priority queue
    d_s ← 0;   Decrease(Q, s, d_s)   //update priority of s with d_s
    V_T ← ∅
    for i ← 0 to |V| − 1 do
        u* ← DeleteMin(Q)   //delete the minimum priority element
        V_T ← V_T ∪ {u*}
        for every vertex u in V − V_T that is adjacent to u* do
            if d_u* + w(u*, u) < d_u
                d_u ← d_u* + w(u*, u);   p_u ← u*
                Decrease(Q, u, d_u)
```

Time efficiency of Dijkstra's algorithm

- Depends on the data structures used for implementing the priority queue and for representing an input graph itself.

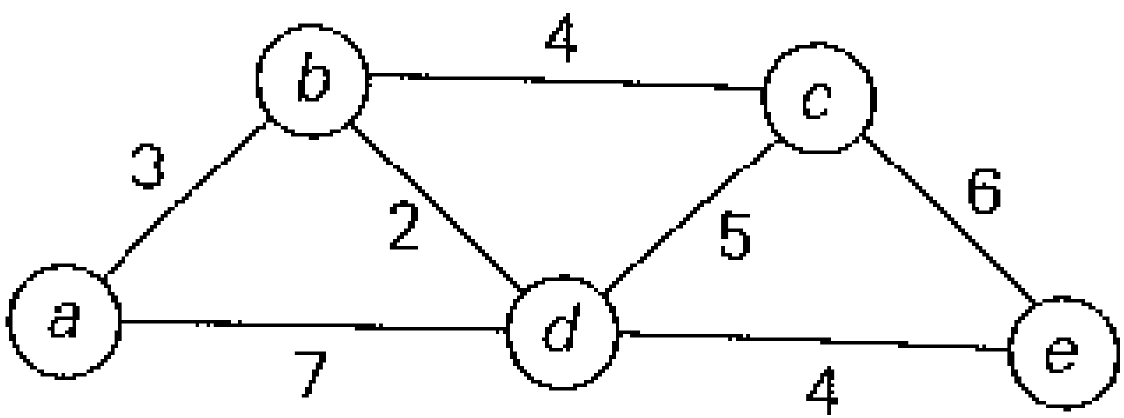
- Weight matrix + priority queue implemented as an unordered array: $\mathbf{\Theta(|v|^2)}$
- Adjacency list + priority queue implemented as a min-heap: $\mathbf{O(|E| \log|V|)}$

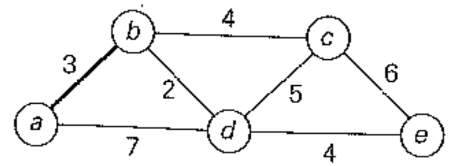
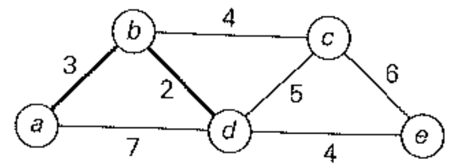
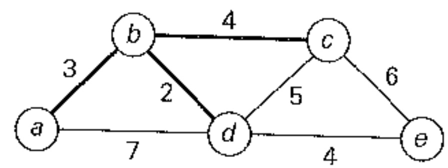
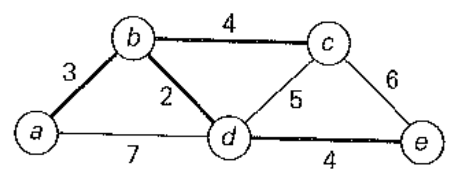
Dijkstra’s algorithm visualization

https://www.cs.usfca.edu/~galles/visualization/Dijkstra.html

https://visualgo.net/en/sssp

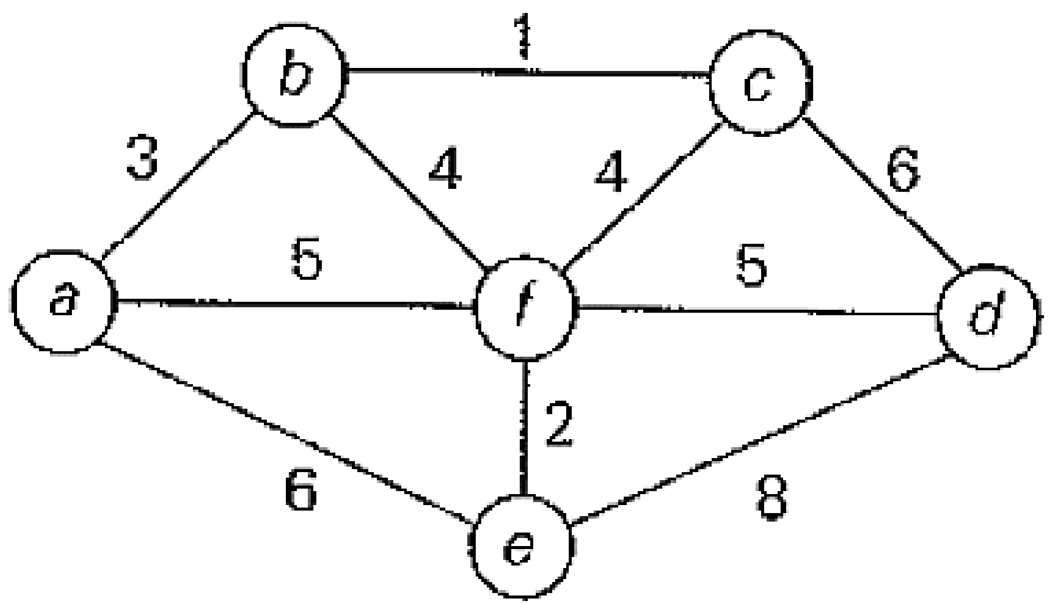
Apply Dijkstra's algorithm to find single source shortest paths from source vertex 'a'



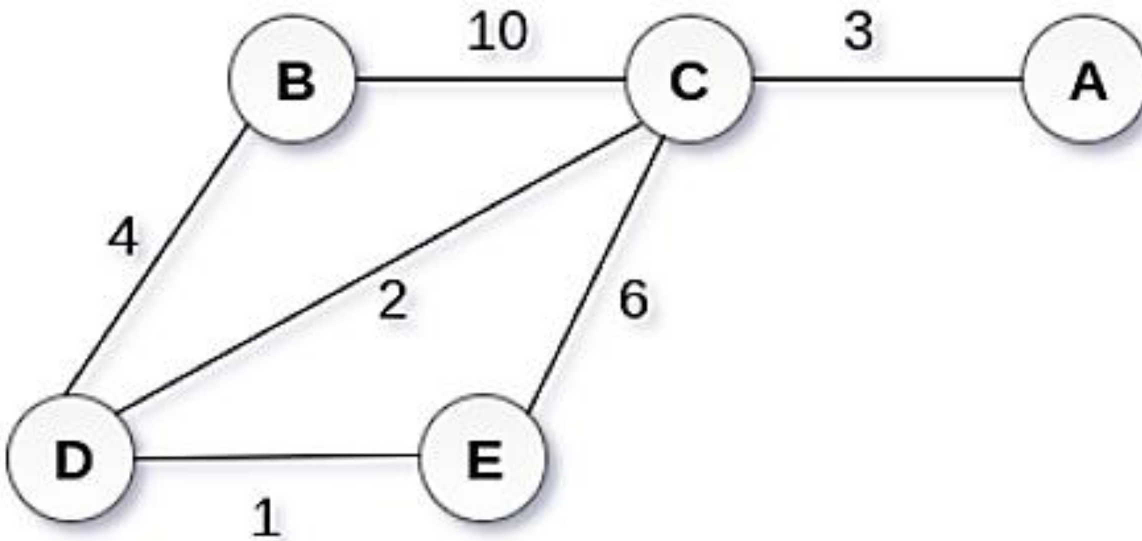
<table>
<tr><th>Tree vertices</th><th>Remaining vertices</th><th>Illustration</th></tr>
<tr><td>a(−, 0)</td><td>b(a, 3) c(−, ∞) d(a, 7) e(−, ∞)</td><td>
</td></tr>
<tr><td>b(a, 3)</td><td>c(b, 3 + 4) d(b, 3 + 2) e(−, ∞)</td><td>
</td></tr>
<tr><td>d(b, 5)</td><td>c(b, 7) e(d, 5 + 4)</td><td>
</td></tr>
<tr><td>c(b, 7)</td><td>e(d, 9)</td><td>
</td></tr>
<tr><td>e(d, 9)</td><td></td><td></td></tr>
</table>

from a to b :	$a - b$	of length 3
from a to d :	$a - b - d$	of length 5
from a to c :	$a - b - c$	of length 7
from a to e :	$a - b - d - e$	of length 9

Apply Dijkstra's algorithm to find single source shortest paths from source vertex 'a'

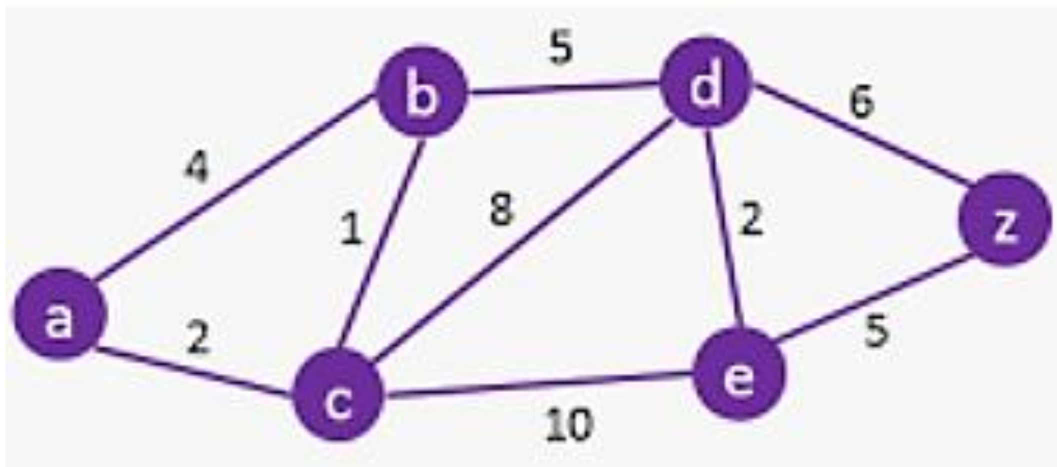


Let’s check our understanding



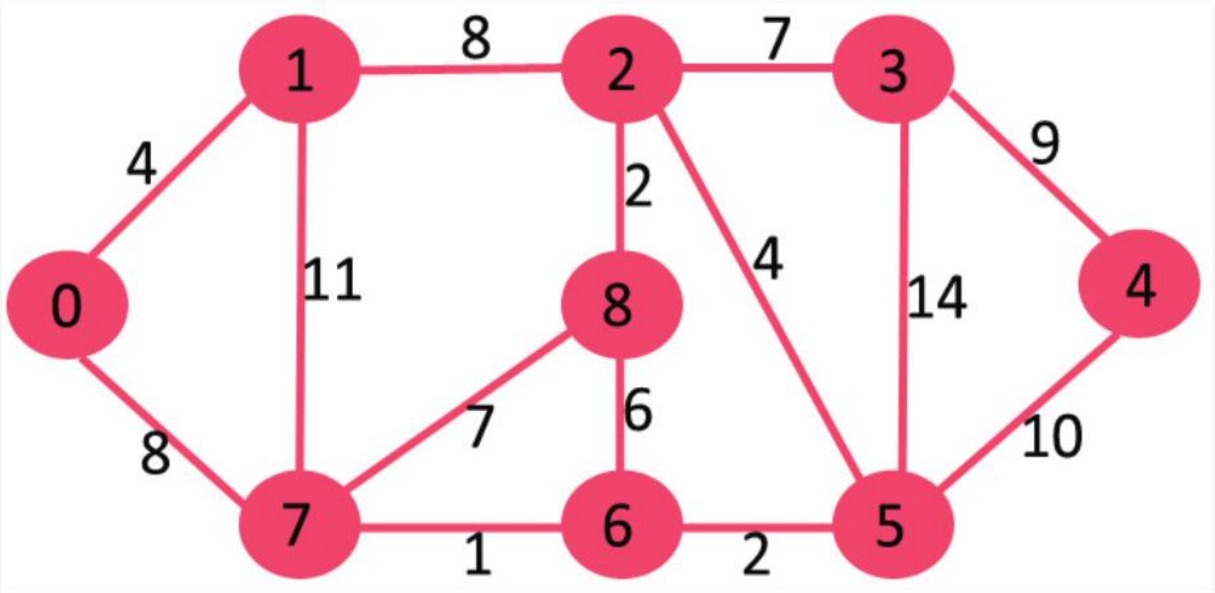
Apply Dijkstra’s algorithm to find single source shortest paths from source vertex ‘a’

Let's check our understanding



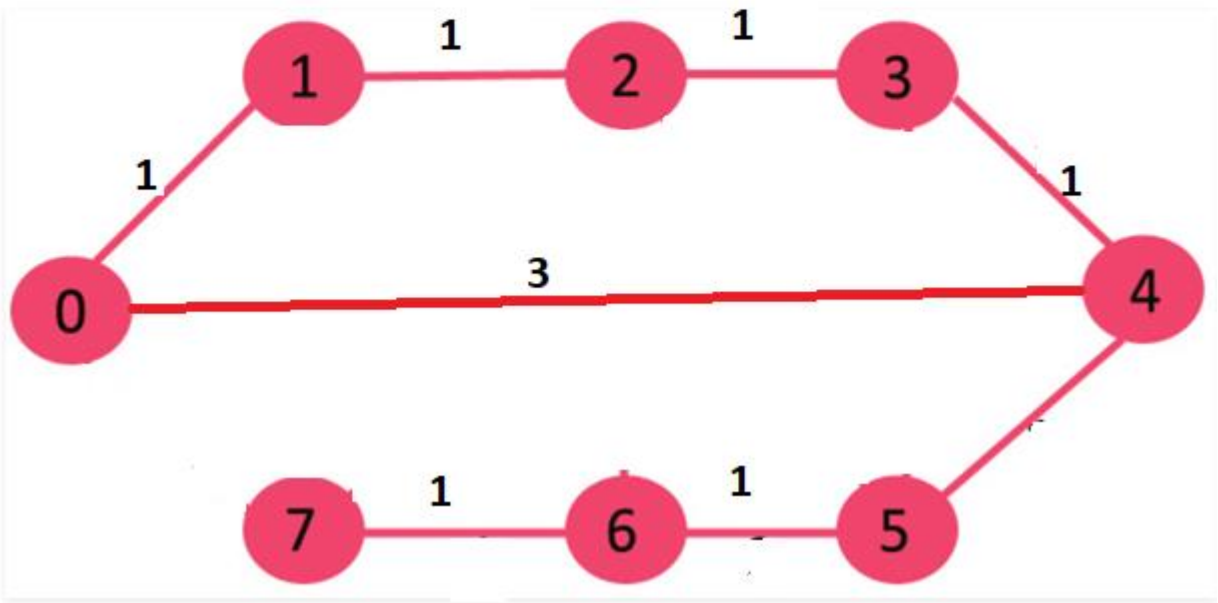
Apply Dijkstra's algorithm to find single source shortest paths from source vertex 'a'

Let’s check our understanding



Apply Dijkstra’s algorithm to find single source shortest paths from source vertex ‘a’

MST vs Shortest Paths



Find MST, Single source shortest paths : consider 0 as the source vertex

Extra Miles…

Bellman Ford Algorithm