**Shortest Path Problems**

* Given directed/undirected weighted graph and two vertices u and v. A shortest path from u to v is a path of minimum weight over all paths connecting u and v.
* A shortest path tree (SPT) is a tree subgraph rooted at a source node such that the path from source node to each node in the tree is a shortest path connecting the source node and the destination node in the super-graph.
* Example: the red edges in the right diagram represent an SPT rooted at A



**Problems**

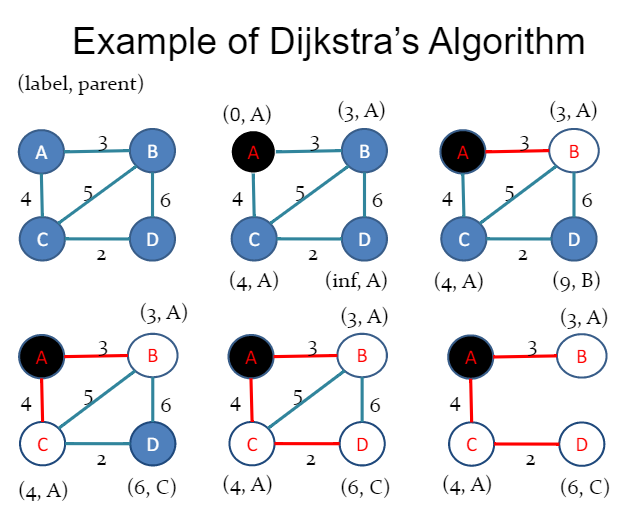
* Simple shortest path problem: given graph G and two vertices u and v, find a shortest path from u to v
* Single-source shortest path problem: given graph G and a vertex u, for any other vertex v, find the shortest path from u to v
* All-pairs shortest path problem: given G, for any pair of vertices u and v, find the short problem from u to v

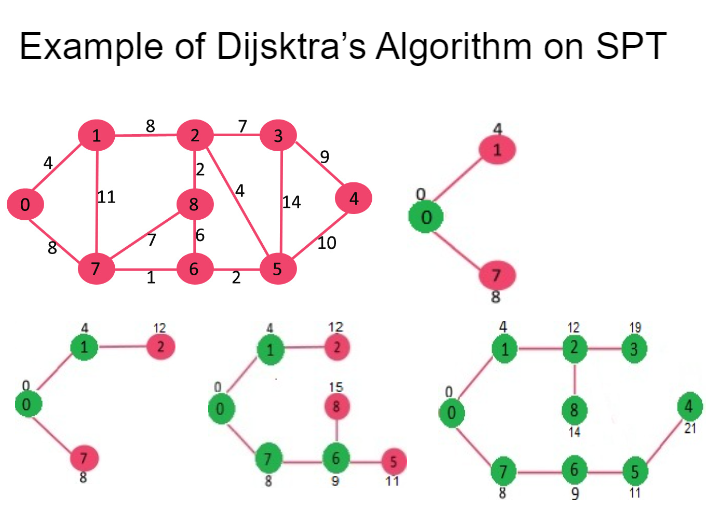
**Dijkstra’s Algorithm**

* Dijkstra’s algorithm, proposed by Dutch scientist Edsger Dijkstra in 1959, is used to find the shortest path tree / path.
* Idea: starting from the source node, growing the shortest path tree by adding the shortest reachable node and edge till no more node and edges can be added.
* Similar to Prim’s algorithm, Dijkstra’s algorithm grows the current SPT by the greedy strategy of adding shortest reachable node and edge.
* Dijkstra’s algorithm is also used for finding costs of shortest paths from a source node to a destination node.

Input: Weighted graph G, source node s

1. Create a graph T to represent current SPT T. Create array label [|V(G)|], initializing to infinite. Set label [s] = 0, add s to T, let u be s, the node newly added to T. Create array parent [|V(G)|], represent the parent node id, initializing to s.
2. If V(G)-V(T) is empty, go to step 5
3. For every neighbor v of u in V(G)-V(T), if label[u]+weight(u,v) < label[v]
   1. then label [v] = label [u] + weight(u,v)
   2. parent [v] = u
4. find the node x in V(G)-V(T) of minimum label. If label [c] != infinite, add x to T, let u be x. Go to step 2.
5. Stop, output T.





**Dijkstra’s Algorithm implementation using min-heap**

Main operation: find a node not in V(G)-V(T) that has the smallest label assigned to it and add it to N

How to find a node with the smallest label efficiently? Solution to the main operation is to use min-heap

1. For each node I, use a key variable to represent the label (shortest distance from the source) to the node through the current tree, and a parent variable to represent the other end of edge in the current tree
2. Use min-heap to store the node key
3. Use extract min to find the node of the smallest label node
4. Use decrease key to update the label

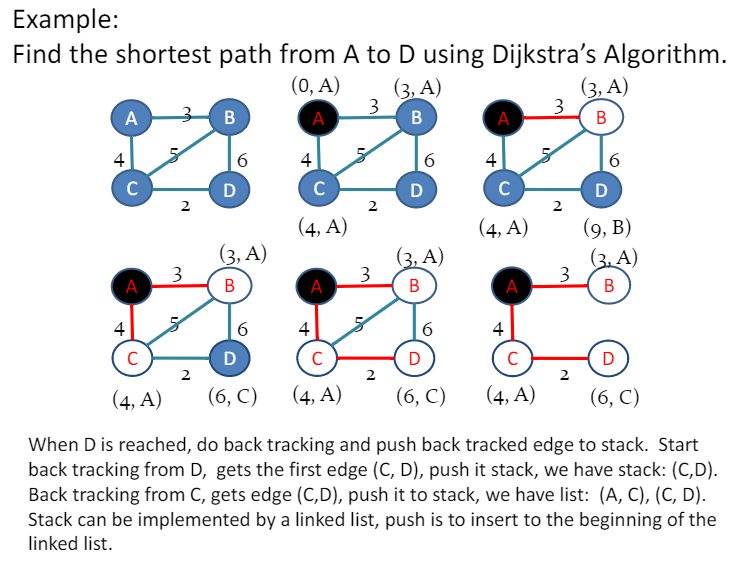
**Dijkstra’s algorithm for single source and destination**

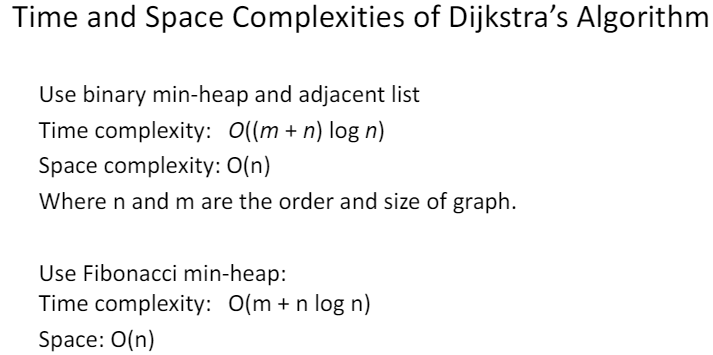
Similar to Dijkstra’s algorithm for SPT, when growing the SPT, if the newly added node is the destination node, then stop growing SPT

Back tracking

Use edge list to represent a shortest path.

1. Starting from the destination node u, (u is not the source node)
2. Add edge (parent[u], u) to the beginning of the edge list
3. If parent[u] is not source node, set u = parent[u], go to step 2
4. Else stop, return the edge list





**FYP idea**

* No edge objects
* Node objects have a property called location, edge weight is calculated by simple subtraction
* This would prevent recursive definitions
* Still need a way to store graph that isn’t the most aids matrix of all time