* **Single Source Shortest Path**
  + BFS
  + Dijkstra’s
  + Bellman Ford
* **All pair shortest Path**
  + Floyd Warshall

BFS

* Simple BFS algorithm can be used for unweighted graphs.

Dijkstra’s

* Weighted graph
* SSSP
* Greedy Algorithm
* Doesn’t support -ve weight edges

Bellman Ford

* Weighted graph
* SSSP
* Supports -ve weights

Floyd Warshall

* Weighted
* APSP
* DP

**Dijkstra’s**

* Maintain a distance [] array which will store distance from source node to i node. distance[src] = 0 and others will be INT\_MAX to start with.
* Maintain a set or priority queue which will return edge with smallest weight. To initialize enter source node with weight 0.
* At any point the set will contain nodes yet to be traversed with corresponding min weight required to reach it using the edges traversed so far.
* While set isn’t empty get the node with smallest distance,
  + Traverse it’s neighbours and calculate the distance to reach them via current node.
  + If distance travelled using current node to its neighbours is smaller than the existing value of distance[neighbour] update distance value and add/replace the new route to neighbour node in set.

Bellman Ford

* Maintain a distance [] array which will store distance from source node to i node. distance[src] = 0 and others will be INT\_MAX to start with.
* Relax each edge V-1 times [most important]
  + By relaxation means, let’s say we have an edge from A->B then check if the distTillA + distFromAtoB is < dist[B], if yes then update.
  + The order in which edges are traversed can be anything, as we were traversing V-1 times, we will get right answer at max V-1iteration.
  + Why V-1 times coz, at max there will be V-1 level so to reach the node at last level we need to run max V-1 times depending on how we traverse. For eg: 1->2->3->4 and we traverse keeping edges from src at last.
* For -ve weight we have the same steps, but need to check the infinite loop scenario.eg, 1->2 w 2, 2->3 w -4,3->1 w 1. If we can relax the edges even after traversing V-1 times means there’s infinite cycle.

Floyd Warshall

* The crux of the algorithm is to pick a node, say k and use it as an intermediate node for every other pair of nodes, i and j.
* Now, picking an intermediate node can increase or decrease the overall distance, so make sure you always pick the minimum distance as shown below:
  + D[i][j] = min(D[i][j], D[i][k] + D[k][j])D[i][j]=min(D[i][j],D[i][k]+D[k][j])