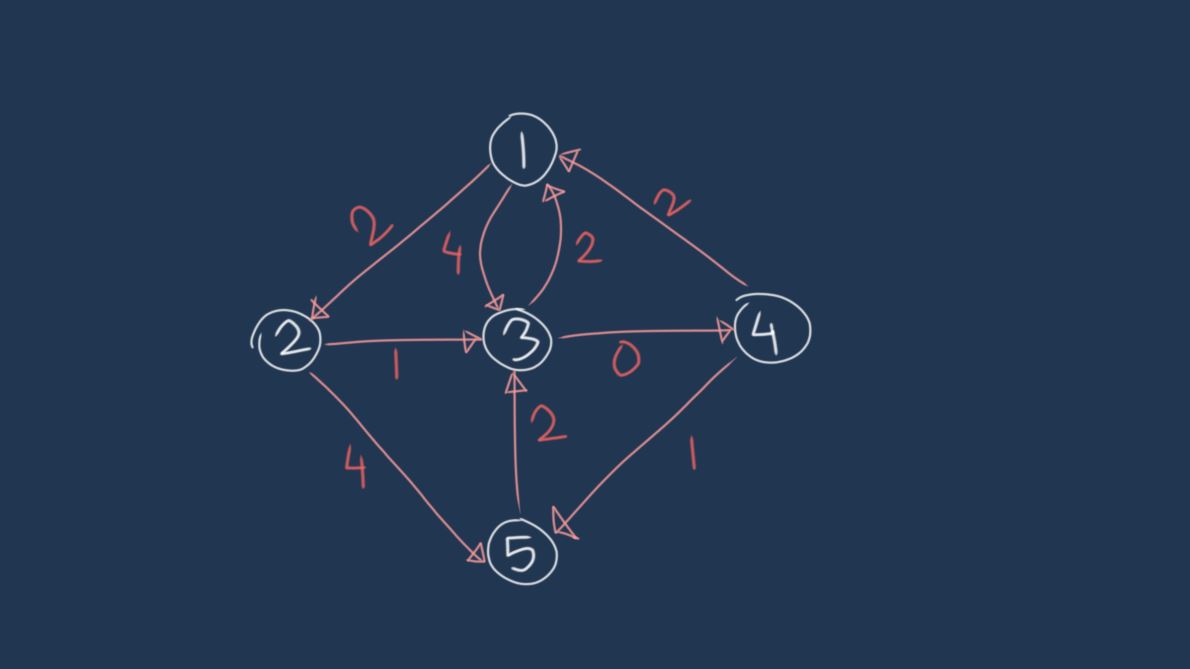
**Dijkastra’s Algorithm**

Best known algorithm for shortest path problems is Dijkastra’s Algorithm.

* It’s used for **single source shortest path problems**.
* It works on both directed or non-directed graphs.
* It **can’t handle negative edges** because it’s unable to detect negative cycles.
* It relies on **heap (Or priority queue)** **just like Prim’s algorithm.**
* It’s a **greedy** method. (As we have to find a shortest path here, so it’s a minimization problem. And minimization problem is an optimization problem which can be solved using Greedy method)

Let’s see how it’s done. Given a weighted graph below which has no negative weights:



**Pseudo Code**

Dijkastra(s)

for (x: x ∈ V)

distance[x] = INF

parent[x] = Null

distance[s] = 0

//declare minheap

//put all (distance, node) pairs in minheap

while(minheap not empty)

pair(distance,u) = extract\_min(minheap)

for(v: v ∈ adj[u])

if(d[u]+w < d[v])

d[v] = d[u] + w

**Time Complexity**

**It is divided into two parts to discuss:**

**A, join point, exhaustive side: each point is only added once, each side is only exhaustive, just the same as a Graph Traversal time.**

**B. Look for the next point: look for the minimum value from the array of size V, which is O(V); find a total of V times, which is O(V2).**

**The addition of A and B is the overall time complexity. If the data structure of the graph is adjacency matrix, it is O(V2); if the data structure of the graph is adjacency lists, it is O(V2).**