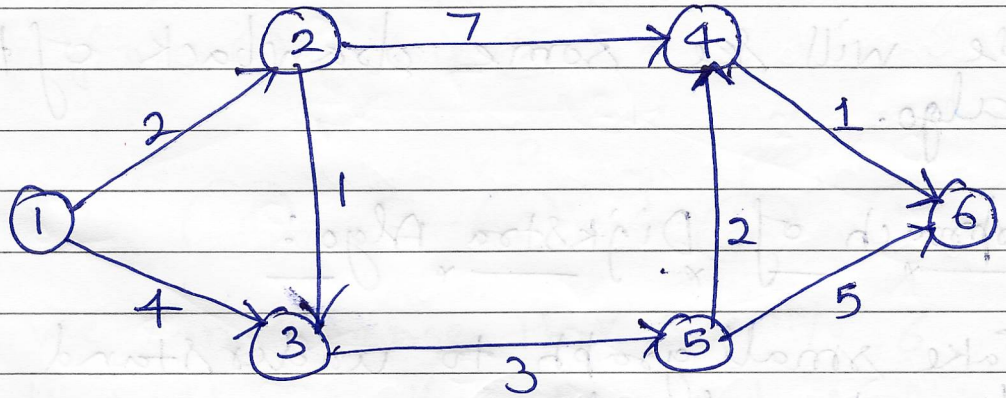
Single Source Shortest Paths

Dijkstra's Algorithm

Dijkstra's Algo is for Single Source Shortest path problem.



- If a weighted graph is given then we have to find a shortest path from some starting vertex to all other vertices.
- Let us say, If I am selecting starting vertex 1, then I have to find out shortest path to all the vertices.
- May be a direct path or via other vertices.
- And I can select any one vertex as source vertex.
- As we have to find out the shortest path, so its a minimization problem and minimization problem is an optimization problem. So optimization problems can be solved using Greedy method.

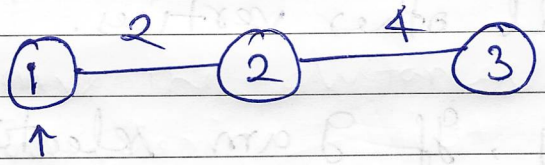
- Here Dijkstra Algo will give a procedure to find the shortest path (optimal solution)

How the algo works?

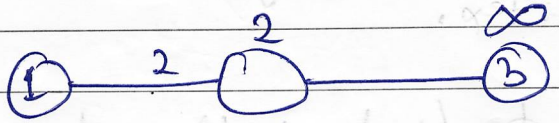
- The Dijkstra Algo can work on directed as well as non directed graph.
- We will see some drawback of this algo.

Approach of Dijkstra Algo:

- Take small graph to understand the procedure of algo.



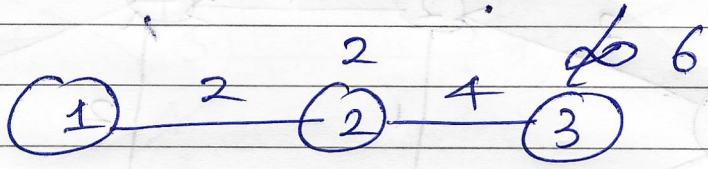
- If (1) is the starting vertex, and if we want to find the shortest path from (1) to (2) & (1) to (3).
- there is the direct path from (1) to (2) so its cost is 2 & there is no direct path between (1) & (3) so we don't know what is the path? so cost = ∞



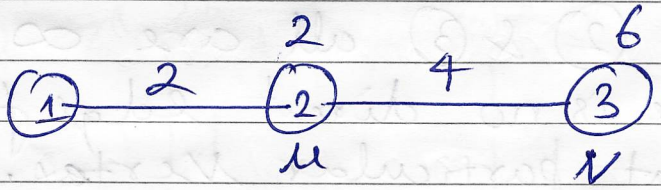
- Dijkstra algo selects first shortest path vertex.

- Which one is the shortest path vertex?
  i.e - (2)

- the algo says that once you have selected one of the shortest path, then check via that vertex there is any shortest path to other vertices.
- So Let us check, who is connected to (2)? from (2), it is (3).



- we can change the ∞ to 6. this means that there is the shortest path from vertex (1) to (3) of distance 6. It's not a direct path, its comming via (2) but there is the path. that's How this algo always select a vertex with shortest path then it will update the shortest path to other vertices if possible. And this updation is called Relaxation.



- How this relaxation works? let us consider (2) as u & (3) as v.

Relaxation:

if (d[u] + c(u,v) < d[v])

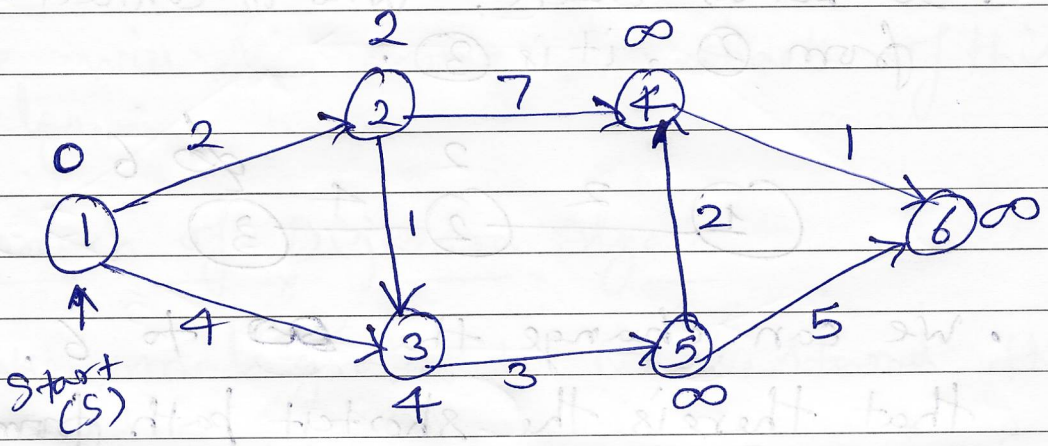
initially 2 + 4 < ∞

but we modified, therefore d[v] = d[u] + c(u,v)

d[v]

- We do the relaxation for vertices, whenever we select shortest path, we will try to relax other vertices.

- Let us follow this procedure of relaxation, and solve the below graph problem to find shortest path.



- Start with source vertex (1)

- Now give the distances for all the vertices, by considering just single edge. So label the distances above those vertex only.

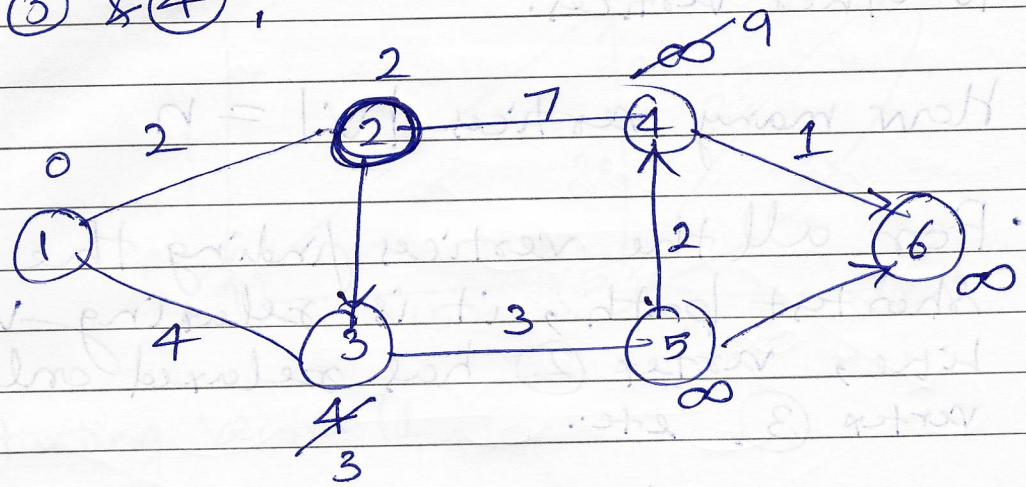
- except (2) & (3) all are ∞, because there is no direct edge from source to that particular vertex.

⇒ - First step: Slect the shortest path out of 2, 4, ∞, ∞, ∞, which one is smallest?

vertex.

- Select (2), then perform the relaxation.

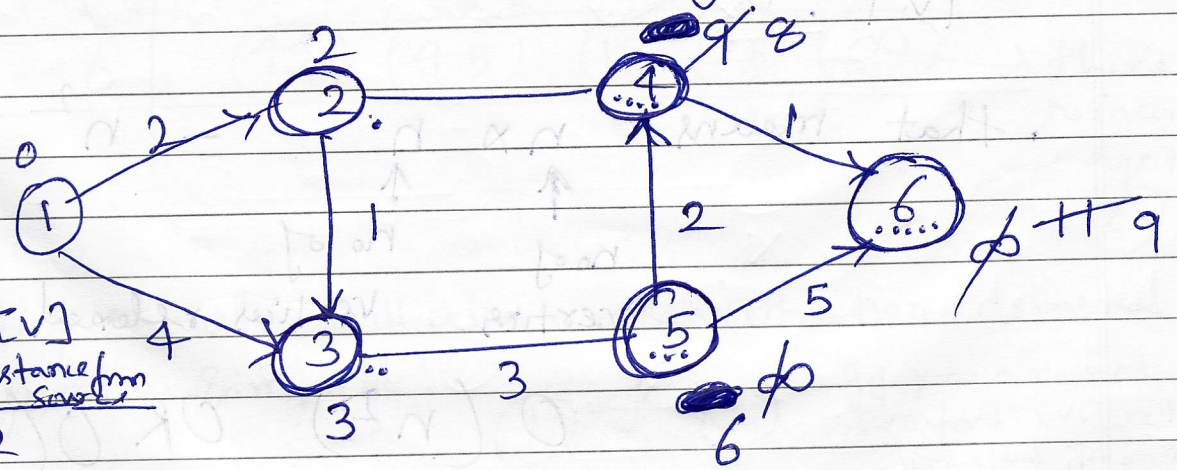
- Check who are connected, vertex (3) & vertex (4), modify the distance at vertex (3) & (4).



⇒ Second step → Repeat the step:

Select the smallest one among 3, 9, ∞, ∞ ? Which one is shortest ?

- Check the connected vertices. check if any vertex get relaxed.



[v] Vertex	d[v] distance from source
2	2
3	3
4	8
5	6
6	9

Analyze the time complexity of Algo:

What the algo do?

- It finds the shortest path from source to other vertices.

- How many vertices $|V| = n$

- For all the vertices finding the shortest path, it is relaxing vertices. Like, vertex (2) has relaxed only vertex (3) etc..

- So total How many vertices will be relaxing? we don't know it depends on the graph.

- At most How many vertices may be relaxed, all n vertices. Suppose (2) is connecting to all then it may be relaxing all vertices. So again it is $|V|$.

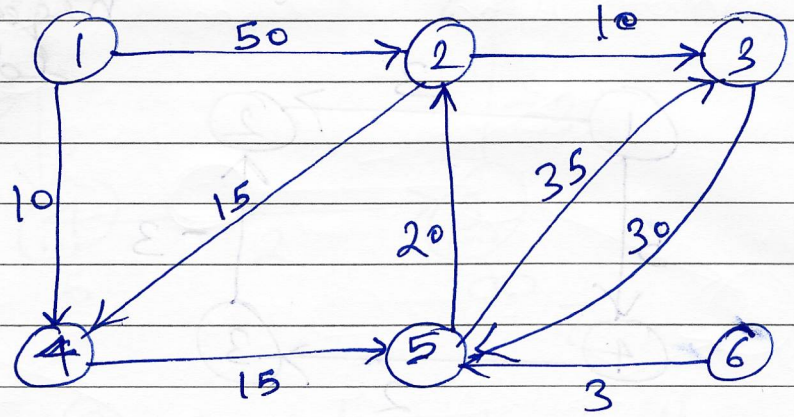
- that means $n \times n = n^2$

  (n: no of vertices; n: no of vertices relaxed)

$O(n^2)$ OR $O(|V|^2)$

Another Ex:

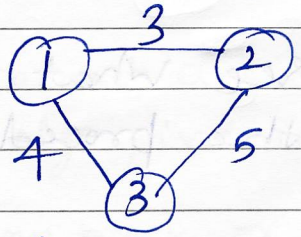
no edge between ① – ⑤ & ① – ⑥.



Starting vertex ①

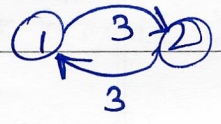
Selected Vertex	2	3	4	5	6
Smallest → 4 Vertex.	50	45	(10)	∞	∞
5	50	45	(10)	(25)	∞
2	(45)	45	(10)	(25)	∞
3	(45)	(45)	(10)	(25)	(∞)
6	(45)	(45)	(10)	(25)	(∞) → there is no incoming edge to (6)

Dijkstra algo will also work with non-directed graph. Simple Ex:



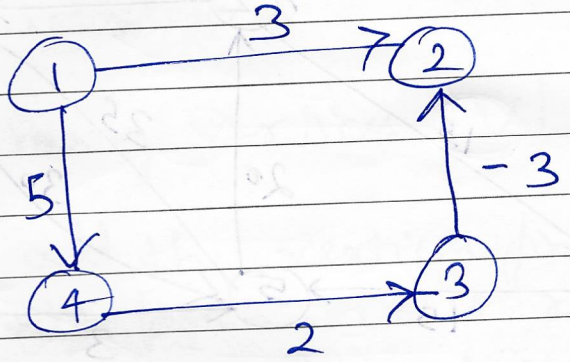
How algo will work here?

If you are not comfort with non-directed graph then, convert it into directed by adding parelled edges.



Drawback of Dijkstra Algo:

- take a simple graph: having a negative weighted edge.



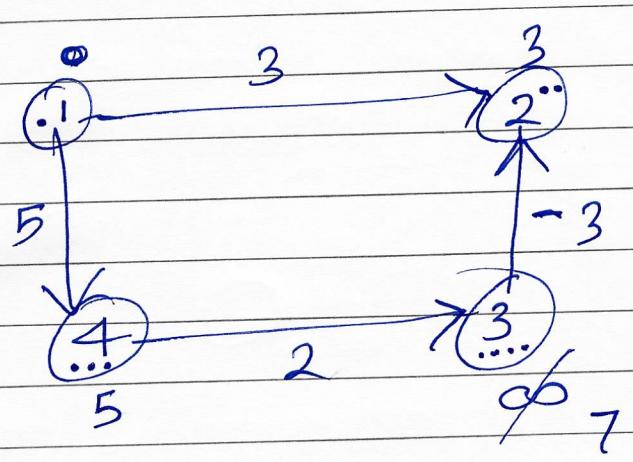
- How it will be negative? See this is not a distance, as distance is not measured in negative.

- Any ~~Example~~ Business problem, we can represent using graphs. most of the problems of real world are mapped into the graph. If you are representing something in form of graph then it may have some negative or positive value to the edge.

- Dijkstra has not considered, if there are negative edge then what should we do?

- Let us see what it happens if we follow the procedure of algo.

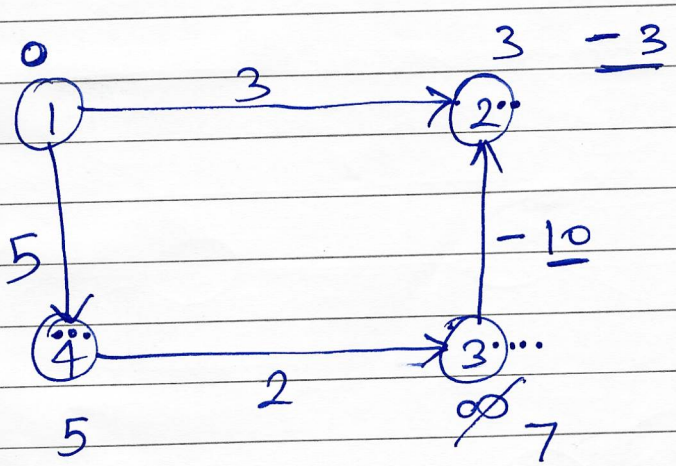
- If start vertex is (1)



- Once (3) is selected, shall we relax (2)? no need, because it has been already relaxed in previous steps.

- So Dijkstra algo works, even though there is negative weight edge.

now let us do small change:



Conclusion: actual correct answer is −3, but here dijkstra algo gives incorrect answer +3.

So Dijkstra algo may work or may not work in case of negative edges.