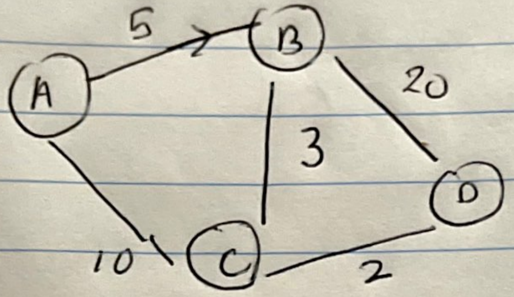
Dijkstra Algorithm -

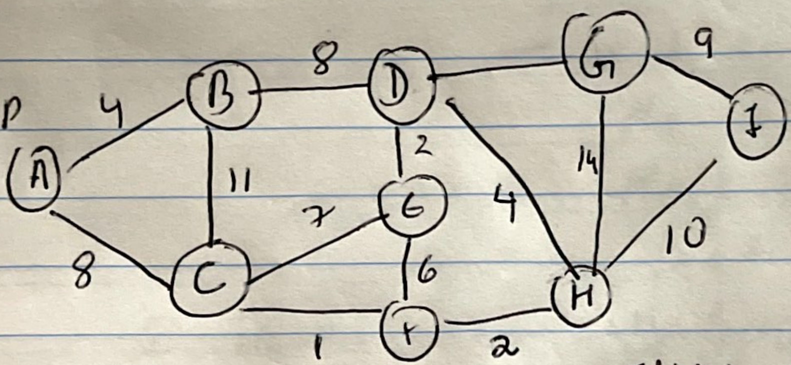
- Given a weighted graph & a source, find shortest distance from source to all other vertices



O/P → A 0
B 5
C 8
D 10

Source → A

I/P



O/P →

A	0
B	4
C	8
D	12
E	14
F	9
G	19
H	11
I	21

- ~~Initialize src to zero & other vertex to ∞~~

	A	B	C	D	E	F	G	H	I
min val	0	∞	∞	∞	∞	∞	∞	∞	∞
Pick A →	0	4	8						
Pick B →	0	4	8	12					
Pick C →	0	4	8	12	15	9			
Pick F →								11	
keep doing it									

finalized → A, B, C

Steps→

(a) begin with source vertex & we initialize to zero and other vertex as ∞

(b) At every movement, we pick a vertex & we finalize ~~is~~ its distance

- Two sets -

finalized → distance is finalized

not-finalized → distance yet to be finalized

HOW do we pick to finalized?

→ which ever vertex has the smallest distance

that's why src is picked first

(c) Travel all the adjacent of vertex & we Relax

Updating distance → Relax (U, V) →

vertex just finalized

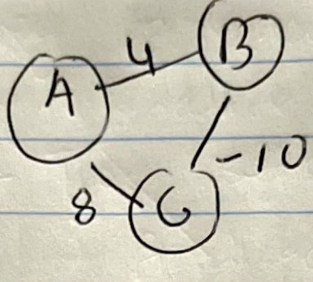
vertex yet to be finalized & adjacent of it

$$d[V] > d[U] + w(U,V)$$

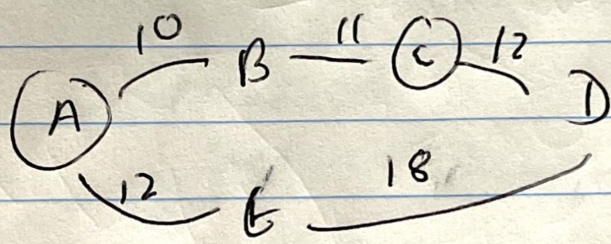
$$d(V) = d[U] + w(U+V$$

Facts about Dikstra Algorithm:

- doesnot work for negative cycle



- Does the shortest path change if add a weight to the edges of the original graph? Yes



Shortest Path - A → B → C → D

Add +10

Now, shortest path

A → E → D

Digstra Algorithms

Implementations

① Create a Priority Queue (or min Heaps)

② dist[V] = {∞, ∞, ∞ ... ∞}

③ dist[src] = 0

O(V) — ④ Insert all distance in pq

V — ⑤ while (pq is not empty)

{ u = pq.extractMin() — O(log V)

E — Relax all adjacent of v that are not in pq — O(log v)

Time Complexity → $O(V\log V + E\log V)$