انا فاهم ال subroutines كويس !A

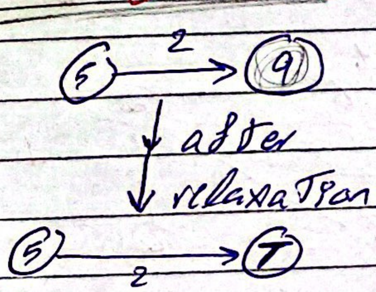
Common subroutines: دي حاجة ثابتة بنعملها في أغلب ال algorithms

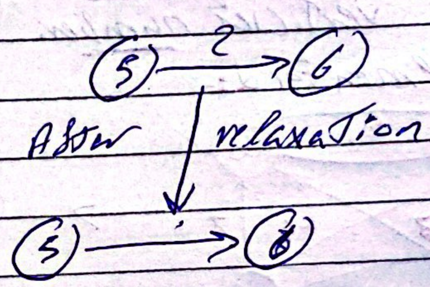
Initializing single source (G, s)

1. For each vertex $v \in G.V$ — $G \rightarrow$ graph.
2- $v.d = \infty$ // distance from $s = \infty$ — $s \rightarrow$ node to be source
3- $v.\pi = NIL$ // no parents.
4- $s.d = 0$ // distance of source = 0.

بعرف ال path عن طريق ال π

Relaxing an edge:





$(5+2) \overset{?}{<} 6$
no
do nothing

$5+2 \overset{?}{<} 9$
yes then put 7
& make 5 the parent of 7

فكرة ال relaxation اني أشوف ال edge ده مفيد ولا
لا ، لو مفيد بجمعه على ال source و احطه في ال destination
واخلي ال source هو ال π (parent) بتاعة ال destination ، لو
مش مفيد مبعملش حاجة

☆ مفيد ← قيمة ال source + ال edge < من قيمة ال destination

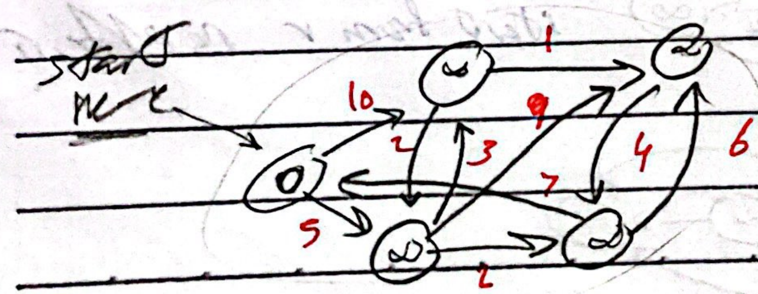
$$\boxed{\begin{array}{l} \text{Relax}(u, v, w) \\ 1.\ \text{if } v.d > \underline{u.d} + \underline{w(u,v)} \\ 2\text{-}\ \ v.d = u.d + w(u,v) \\ 3\text{-}\ \ v.\pi = u \end{array}}$$

$u \rightarrow$ source
$v \rightarrow$ destination
$w \rightarrow$ edge weight

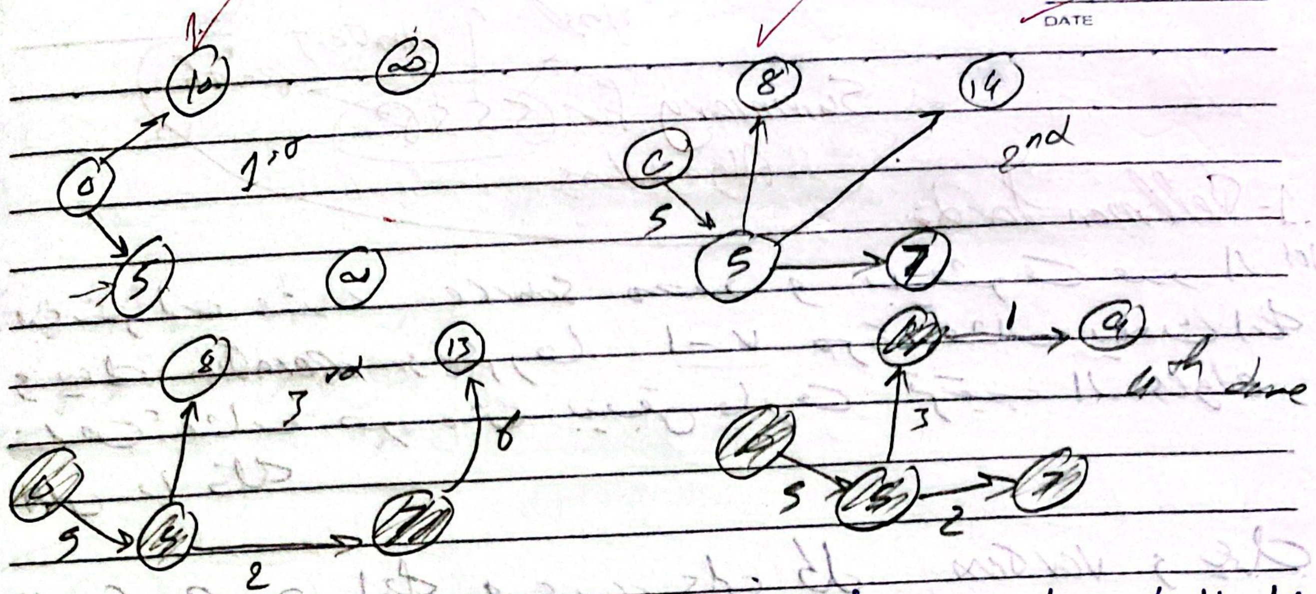
3. Dijkstra's algorithm:

- works only with positive edges.
- Greedily takes the nodes with smallest distance from source & relax its edges.
- Similar to BFS but in the non-decreasing order of node distance from root.
- BFS can be seen as special case of Dijkstra when all edges have the same weight.

Example: on the following graph apply Dijkstra.



Scanned with CamScanner



كل الفكرة اني هبدأ من ال 0 واعمل relax ll edges
اللي متوصلة بال $\boxed{0}$ ولو في حاجة اصغر بختارها بعدها

بشوف فين اصغر Node واعمل relax ll edges بتاعتها
وفي حالتنا دي هتبقى ال $\boxed{5}$ بعدها هلاقي ان اصغر Node
عندي هي ال $\boxed{7}$ اعملها relax واشوف مين اصغر.
هلاقي إنها ال $\boxed{8}$ بعد ال $\boxed{8}$ هعمله relax وهبقى خلصت

Dijkstra (G, w, s)

1- Initialize single source (G, s)
2- S = ∅ // empty set → contain finished nodes
3- Q = G.V // place all vertices in queue. → Build heap → to get minimum of them
4- while Q.size
5- u = Extract_min (Q)
6- S = S ∪ {u} // بحط el u جوه el finished nodes
7- for each vertex v ∈ G.Adj[u]
8- Relax (u, v, w) → decrease key

Complexity differs according to the used data structure

	Array (once / Total)	Min heap	Fibonacci heap
Insert (1/T)	O(1) / O(V)	- / O(V)	- / O(V)
Extract min	O(V) / O(V^2)	O(lg V) / O(V lg V)	O(lg V) / O(V lg V)
Decrease key	O(1) / O(E)	O(lg V) / O(E lg V)	O(1) / O(E)
Total	O(V^2) → unsorted array	O(E lg V)	O(V lg V + E)