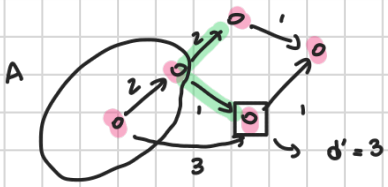
Algorithms

DATE : 22/03/2022



DIJKSTRA'S ALGO (G(V,E) , s) :

$S \leftarrow \{s\}$ // S is the set of nodes visited so far

$d(s) = 0$ // $d(v)$ will contain the length of a shortest path from s to v

$P_s = [s]$ // P_v is a shortest path from s to v

WHILE $S \neq V$: ($\leq n$ iterations)

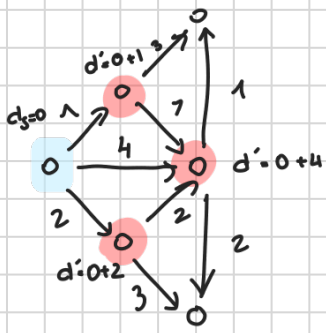
- $T = \{ w \mid w \in V-S$ AND THERE EXISTS $v \in S$ S.T. $(v,w) \in E \}$ (O(n))
- $\forall w \in T$, Let $d'(w) = \min_{\substack{(u,w)\in E \\ u \in S}} (d(u) + \ell(u,w))$ (O(n·|S|))
  ↘ shortest path from source to w
- LET $v \in T$ be a node of minimum $d'(v)$ (O(n))
- LET $u \in S$ be such that $d'(v) = d(w) + \ell(w,v)$ (O(1))

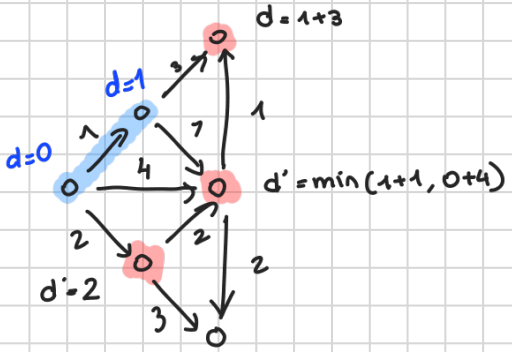
NOTE: There will always be many shortest paths to choose from. The algo just provides one of the many possibee solutions.

- $S \leftarrow S \cup \{v\}$
- $d(v) \leftarrow d'(v)$ } (O(1))
- $P_v \leftarrow P_u + [v]$

$O(n^2)$

RETURN d, P_v, P_{v_2}, ..., P_{v_n}





(it doesn't necessarily need to find path from the "last" node)

Questo algoritmo ci dà in output tutti i percorsi più brevi avendo come punto di partenza un origine s.

DIJKSTRA'S ALGO ($G(V,E)$, s):

$S \leftarrow \{s\}$ // S is the set of nodes visited so far. At the beginning this set contains only the starting/source node.

$d(s) = 0$ // $d(v)$ will contain only the length of a shortest path from s to v.

$P_s = [s]$ // P_v is a shortest path from s to v. This bunch of lists contain the "names" of the paths.

a bunch of lists

// The "INPUT" set of vertices/nodes.

WHILE $S \neq V$: ($\leq n$ iterations) // At each iteration, the algorithm
* "removes" one node from V and adds it to S.
* Actually it doesn't really remove it, but let's say that it removes it.

// The set of nodes that we haven't visited yet and that are reachable by one step from s (a node already picked).

- $T = \{ w \mid w \in V-S$ AND THERE EXISTS $v \in S$ S.T. $(v,w) \in E \}$

// A generic node that belongs to the set of nodes that we haven't visited yet.

- $\forall w \in T$, Let $d'(w) = \min_{\substack{(u,w) \in E \\ u \in S}} (d(u) + \ell(u,w))$ → shortest path from source to w

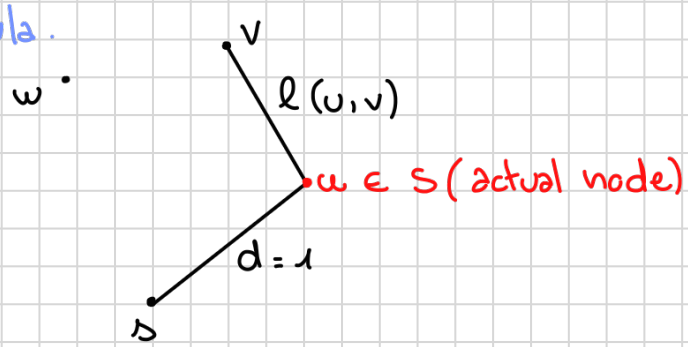
// It's the minimum distance that we have to do with only one step to get to a node w, from the actual one.

- LET $v \in T$ be a node of minimum $d'(v)$

// It's all the specific node that the algorithm chooses between the nodes of T.
The algorithm chooses v because it's the nearest node to the actual one, that can be reached by one step and with a minimum distance.

- LET $u \in S$ be such that $d'(v) = d(u) + \ell(u, v)$

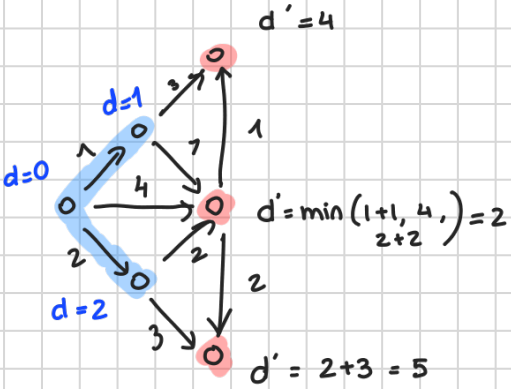
// u is the actual node, so it belongs to S.
Moreover we have to define the actual distance from the starting point s, and this distance can be established by that formula.

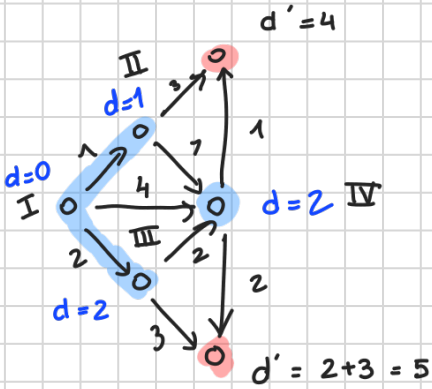


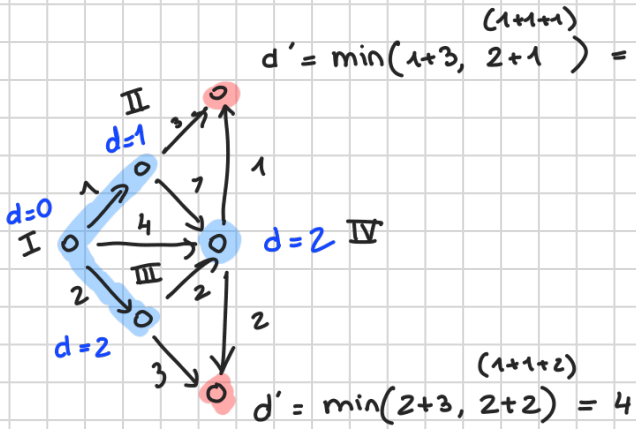
NOTE: There will always be many shortest paths to choose from. The algo just provides one of the many possibee solutions.

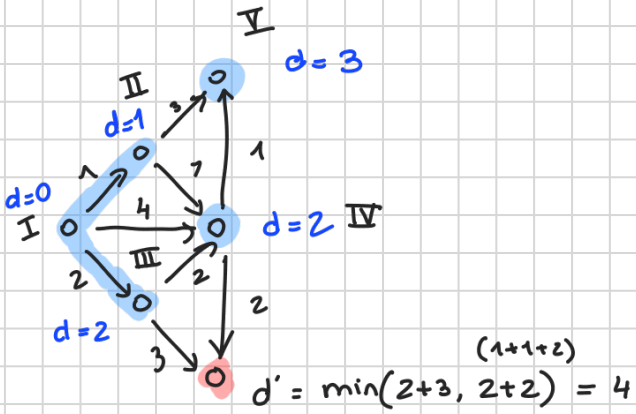
- $S \leftarrow S \cup \{v\}$ // Adding the "actual" node to S.
- $d(v) \leftarrow d'(v)$ // Adding the "actual" distance.
- $P_v \leftarrow P_u + [v]$ // Adding the "actual" path.

RETURN d, P_v, P_{v_2}, ..., P_{v_n} // Returning the distance and the "labeled" path.









L: At any time during the execution of the algorithm, if $u \in S$, then P_u is a shortest path from s to u.

P: We prove the claim by induction on $|S|$.
If $|S|=1$, then $S=\{s\}$, so $d(s)=0$ and $P_s = |s|$.
The base case has been proved.

Let us suppose that the claim holds for $|S|=K$.
(we prove it for $K+1$).

In the iteration in which, at it's beginning, S has cardinality K, the algorithm will add some new node v to S, obtaining $S' = S \cup \{v\}$.
Since $v \notin S$, it holds $|S'| = K+1$.

Let $u \in S$ be the node that the algorithm chose to get to v then, $P_v = P_u + [v]$

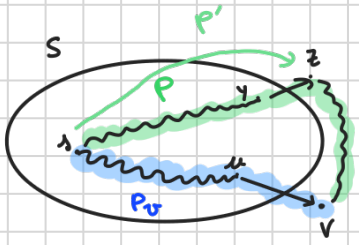
By the inductive hypothesis, since $u \in S$, P_u is a shortest path to u.

We would like to prove that P_v is a shortest path to v. Clearly, P_v is a path to v.

// We want to prove that it's the shortest.

Consider any other path P from s to v.
Since $v \notin S$, P must leave S at some point.

Let z be the first node of P that is not in S and Let y be the node preceding z in P.



We want to prove that P cannot be shorter than P_v. We will prove this by showing that the prefix P' of P that goes from s to z is not shorter than P_v.

In iteration $k+1$, the algorithm considered adding z to S via the edge (y, z). The algorithm has also rejected this choice, in favor of v.

Thus, there exists no path from s to z that is shorter than P_v.

Since edge-lengths are non-negative, the path P (which includes P') cannot be shorter than P_v.
Thus, P_v is a shortest path from s to v.
and the induction policy has been proved ■

COR: ↓ COROLLARY DIJKSTRA'S algorithm finds all the shortest paths from s.

THUS, shortest paths can be found in polynomial time (we showed $O(n^3)$ time).

Next time, we will introduce a data structure (called "priority queue", or "heap") that will allow us to implement DIJKSTRA'S algo in $O(m \log n)$ time