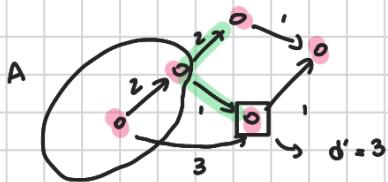


# NOTE BOOK

Rokshana Ahmed

# Algorithms

DATE : 22/03/2022



$\rightarrow \langle \overline{X} \overline{X} \overline{X} \rangle t$

D1JKSTRA'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)  $(O(n^2))$

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

-  $\forall w \in T$ , Let  $d'(w) = \min_{\substack{(u, w) \in E \\ u \in S}} (d(u) + l(u, w))$   $(O(n \cdot |E|))$

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

- LET  $u \in S$  be such that  $d'(v) = d(u) + l(u, v)$   $(O(1))$

$O(n^2)$

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

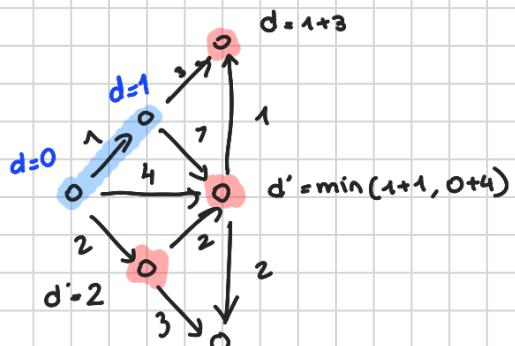
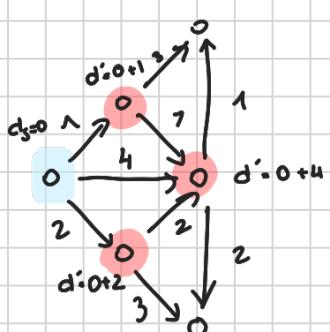
-  $S \leftarrow S \cup \{v\}$

-  $d(v) \leftarrow d'(v)$

-  $P_v \leftarrow P_u + [v]$

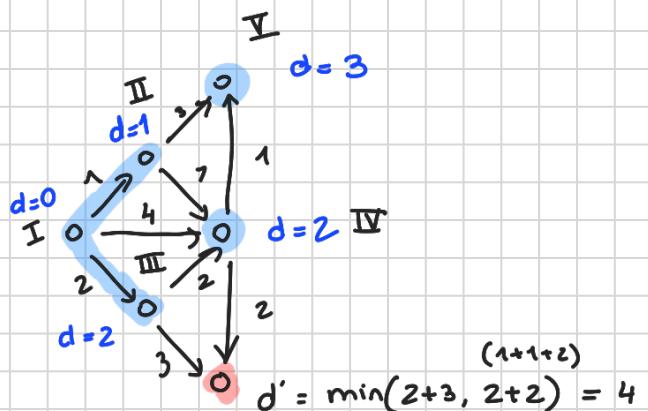
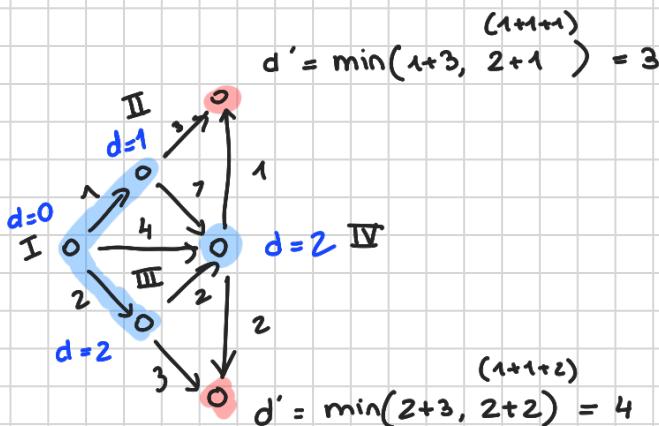
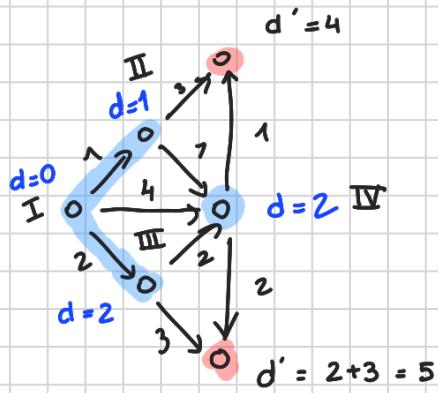
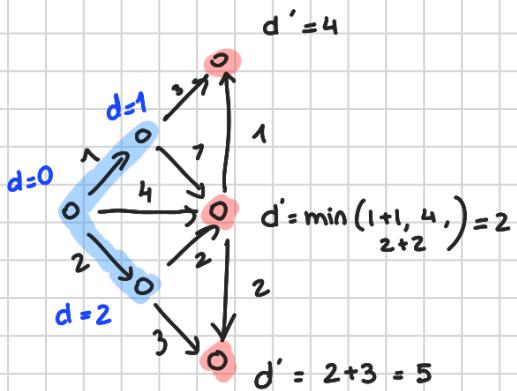
$\} (O(1))$

RETURN  $d$ ,  $P_v$ ,  $P_{v_2}, \dots, P_{v_n}$



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

- ①  $S$  is the set that contains the nodes visited so far.
- ②  $P[S] \rightarrow$  contains all the shortest paths
- ③ while  $S \neq V \rightarrow$  loop will go on until all nodes have been visited so far.
- ④  $T$  is the SET that contains all the nodes that haven't been visited so far.
- ⑤  $d'(w)$  is the minimum distance required to reach a  $w$  node by using only one step
- ⑥  $w$  is a generic node (that hasn't been visited yet) while  $v$  is the



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

P: We prove the claim by induction on  $|S|$ .  
If  $|S|=1$ , then  $S=\{s\}$ , so  $d(s)=0$  and  $P_s=1s1$ .  
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 its 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  $|S'|=k+1$ .

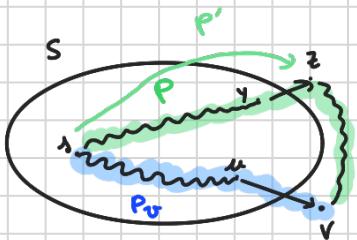
Let  $\mu \in S$  be the node that the algorithm chose to get to  $v$  then,  $P_v = P_\mu + [v]$

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

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

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:** DIJKSTRA's algorithm finds all the shortest paths from  $s$ .  
↓  
**COROLLARY**

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  $O(n \log n)$  time