Shortest Paths in a graph

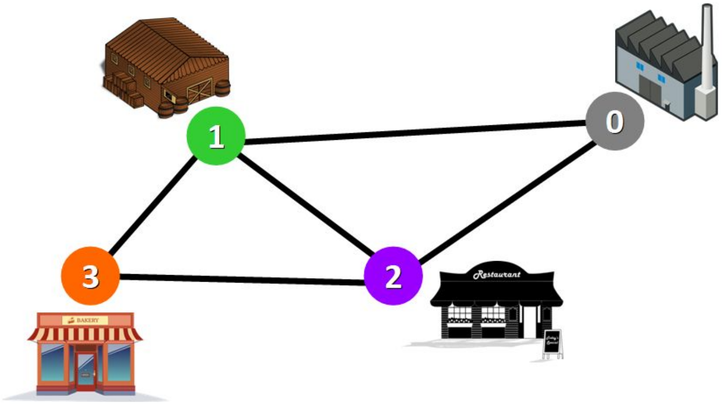
Refer to Chap 4 from Tardos

Sakeena Shahid | 11-02-2022

1

Problem Statement

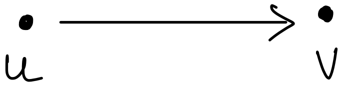
- **Aim:** to find the shortest paths between nodes of a graph.
- Graphs - model networks
- Formally:
    - Given nodes u and v, what is the shortest u-v path?

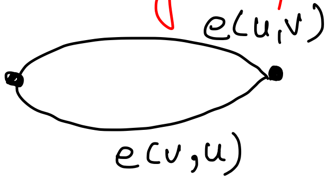


Problem statement

- Given a graph G(**V**,**E**) with a start node s. [Assume there is a path from s to every node]
- **V** = set of vertices/nodes
- **E** = set of edges. Each edge ‘e’ has a length $l_e \geq 0$. This denotes cost, distance, time.
- Path (P) is a sequence of such edges connecting distinct vertices.
- Length of path **l(P)**: sum of all edge lengths in the path.
- **Goal:** Find a path with the minimum **l(P)**.
- Directed versus undirected graphs?

↳ Replace directed edge w/ two edges





Dijkstra's Algorithm

4

- Used to solve single-source shortest path problem
- Steps:
  - S set of vertices u for which the shortest path d(u) from s is already determined -> **explored part**.
  - Initially S = {s}, d(s) =0, d(s') = ∞ where s!= s'
  - Then for each node v ∈ V - S, we determine the shortest path from s along the explored path.
  - The new shortest distance $\mathbf{d'(v) = \min_{e=(u,v):u \in S} d(u) + l_e}$
  - We choose v that minimises this quantity, then d(v) = d'(v)
  - Repeat until S = V

Dijkstra's Algorithm

Dijkstra's Algorithm (G, ℓ)

Let S be the set of explored nodes

    For each $u \in S$, we store a distance $d(u)$

Initially $S = \{s\}$ and $d(s) = 0$

While $S \neq V$

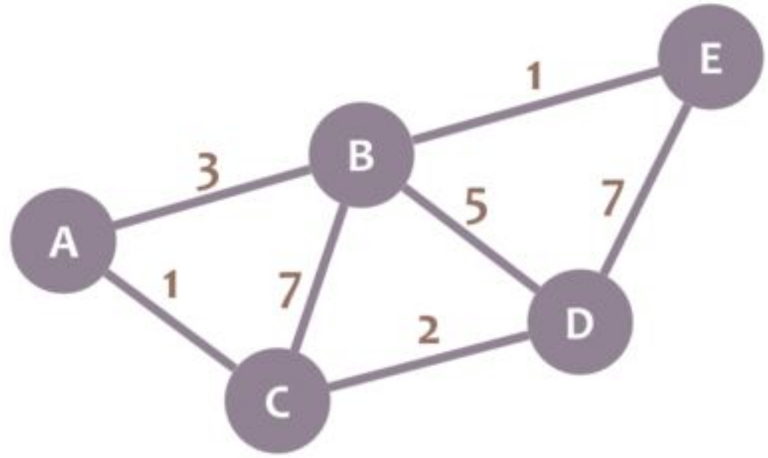
    Select a node $v \notin S$ with at least one edge from S for which

        $d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$ is as small as possible

    Add v to S and define $d(v) = d'(v)$

EndWhile

Example - Dijkstra's Algorithm



Dijkstra's Algorithm (G, ℓ)

Let S be the set of explored nodes

  For each $u \in S$, we store a distance $d(u)$

Initially $S = \{s\}$ and $d(s) = 0$

While $S \neq V$

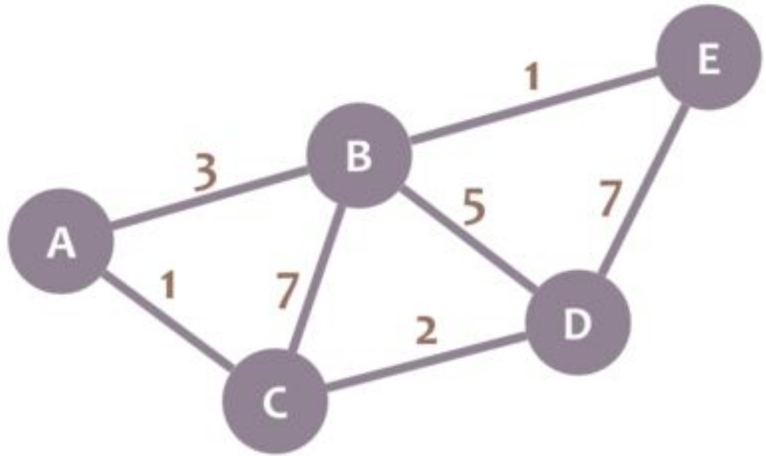
  Select a node $v \notin S$ with at least one edge from S for which

    $d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$ is as small as possible

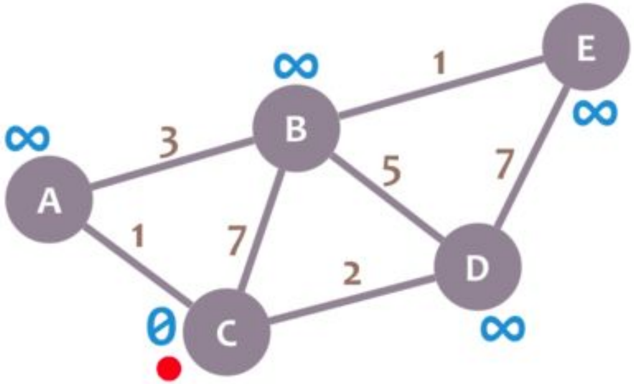
  Add v to S and define $d(v) = d'(v)$

EndWhile

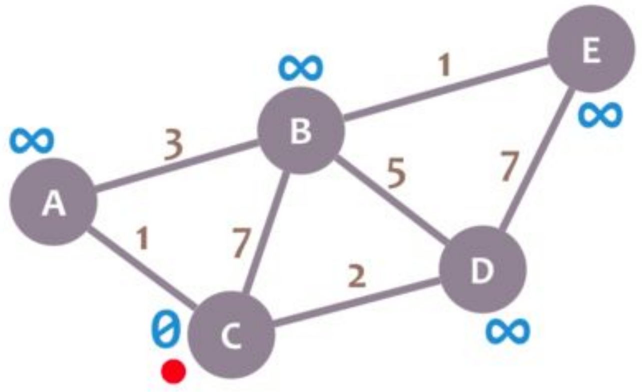
Example - Dijkstra's Algorithm



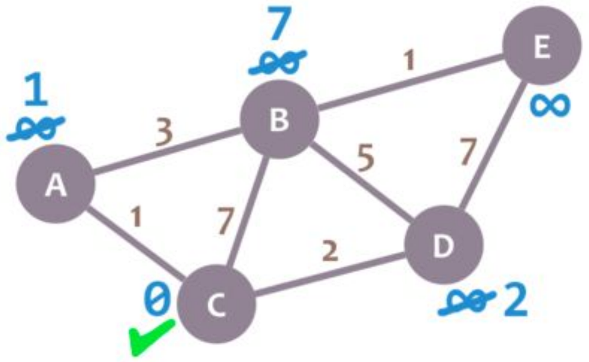
- Start node: C
- d(C) = 0
- d(A), d(B), d(D), d(E) = ∞



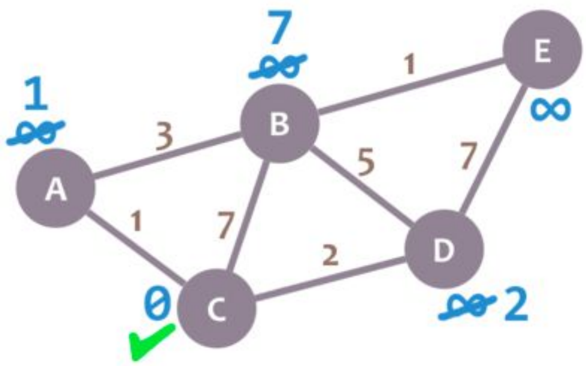
Example - Dijkstra’s Algorithm



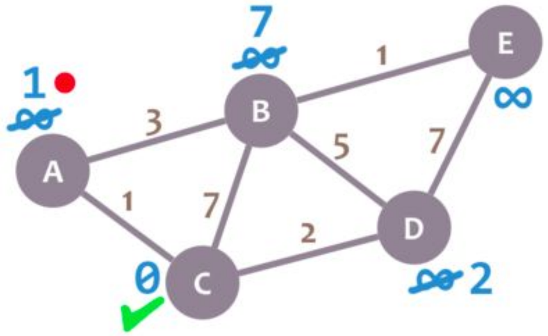
- For the neighbour of C, find the shortest paths
- Neighbours: A, B , D
- No particular order



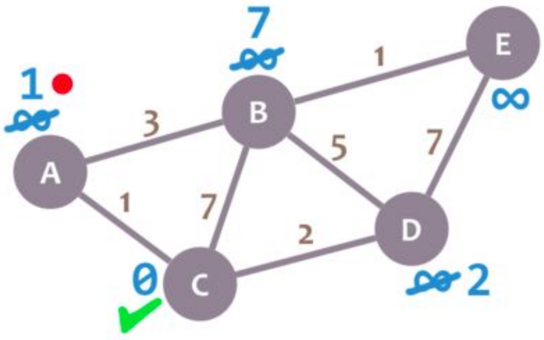
Example - Dijkstra’s Algorithm



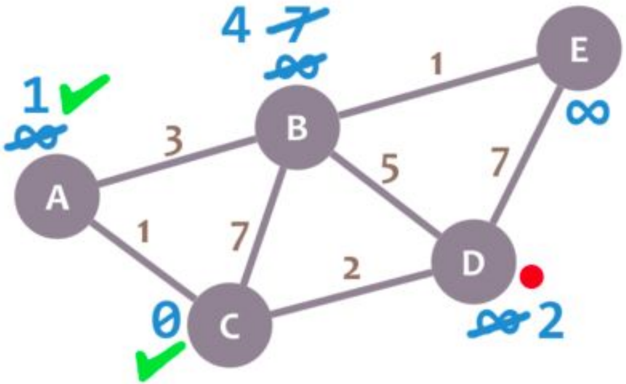
- C is now explored
- Select new node. Which one? A because of minimum distance.
- Repeat previously applied steps on A.



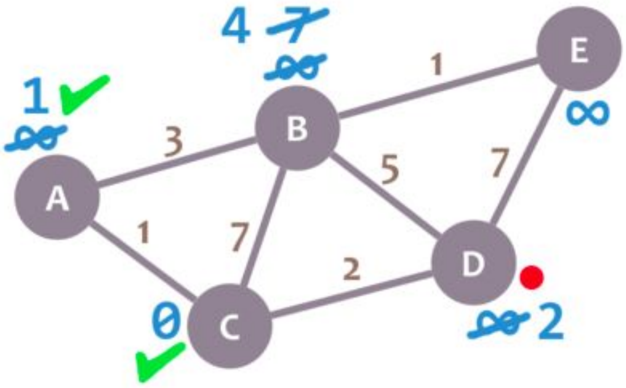
Example - Dijkstra's Algorithm



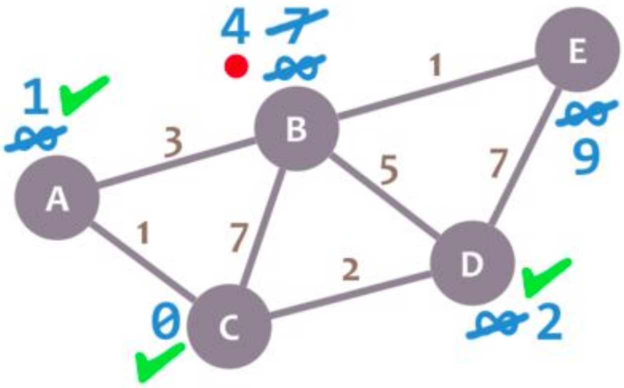
- Update distances of A's neighbours.
- Neighbours: B
- Add A to explored.
- Next node with minimum distance? D



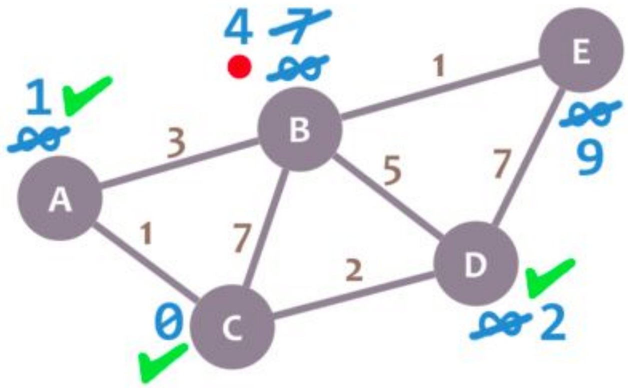
Example - Dijkstra's Algorithm



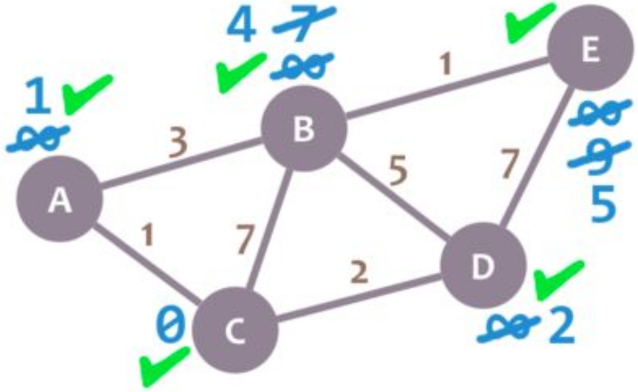
- Update distances of D's neighbours.
- Neighbours: E,B
- Add D to explored.
- Next node with minimum distance? B



Example - Dijkstra’s Algorithm

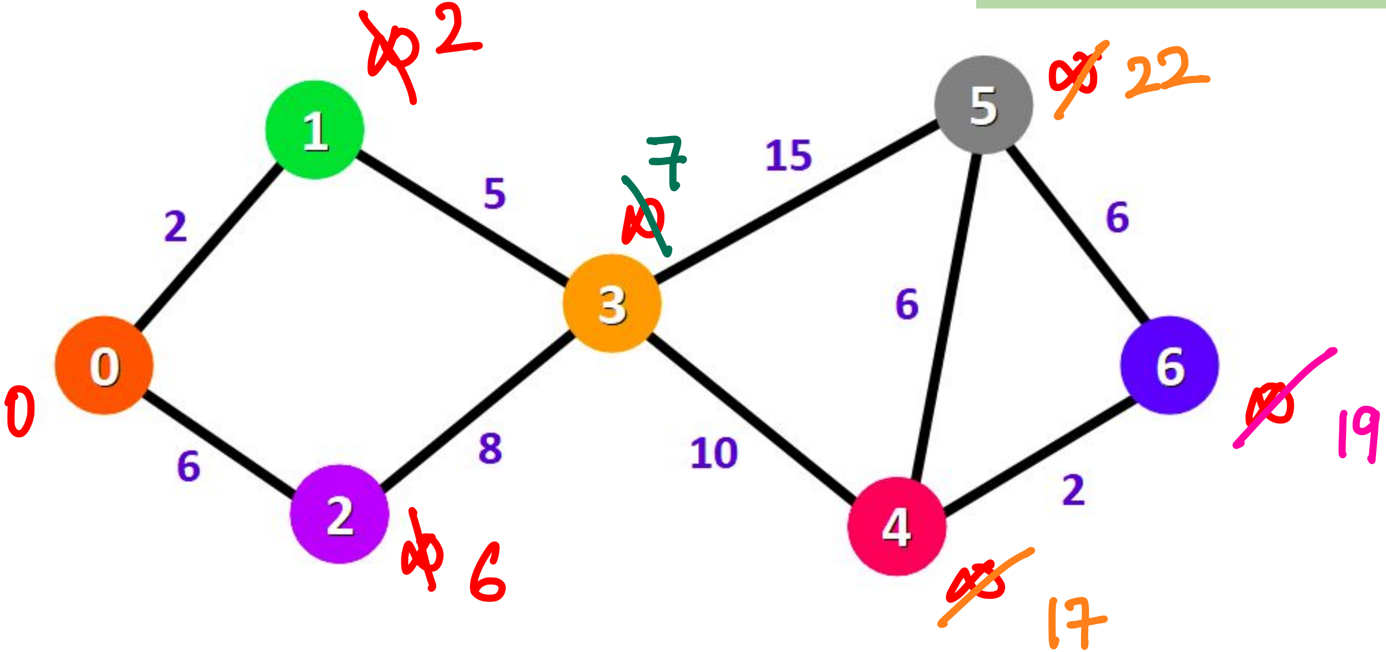


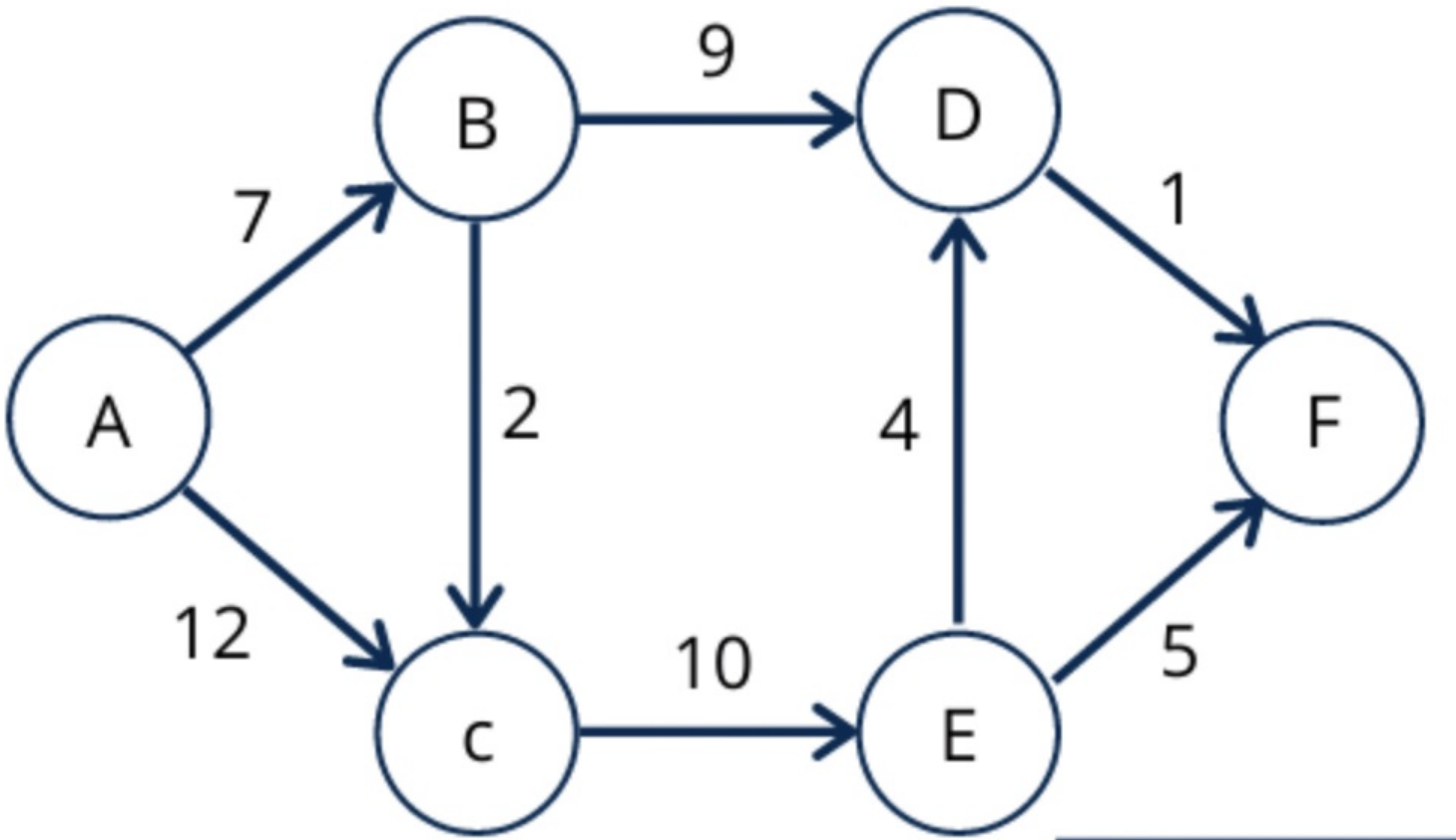
- Update distances of B’s neighbours.
- Neighbours: E
- Add B to explored.
- Next node with minimum distance? none



Let's Do It

Start node: 0 (the one in red)




A
B
C
D
E
F
7
12
9
2
10
4
1
5

Analysis of Dijkstra's Algorithm

- **To prove:** It is always true that when Dijkstra's Algorithm adds a node v to set S, we get the true shortest-path distance to v from s.
- Use stay ahead style
- Consider the set S at any point in the algorithm's execution. For each $u \in S$, the path P_u is a shortest s-u path. When S contains all nodes, we can say Dijkstra's algorithm is correct.

Analysis of Dijkstra's Algorithm

Consider the set S at any point in the algorithm's execution. For each $u \in S$, the path P_u is a shortest s-u path. When S contains all nodes, we can say Dijkstra's algorithm is correct.

- Use Induction
- Base case: The case $|S| = 1$ is easy, since then we have $S = \{s\}$ and $d(s) = 0$.
- Inductive step: claim holds when $|S| = k$ for some value of $k \geq 1$;
- To prove: true after growing S to size $k + 1$ by adding the node v. Let (u, v) be the final edge on our s-v path P_v

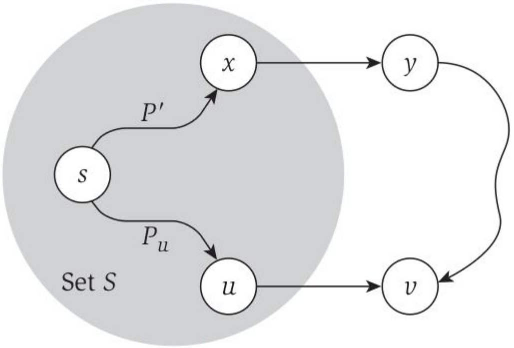
Analysis of Dijkstra's Algorithm

16

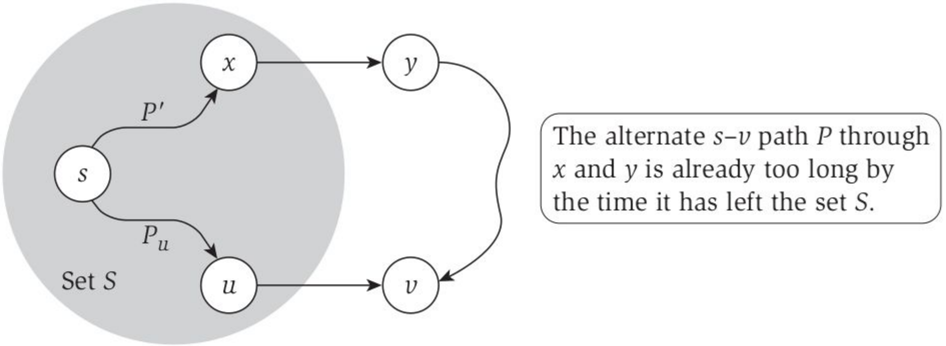
- To prove: true after growing S to size k + 1 by adding the node v. Let (u, v) be the final edge on our s-v path P_v

P_u is the shortest s-u path for each u ∈ S. Now consider any other s-v path P; we wish to show that it is at least as long as P_v .

In order to reach v, this path P must leave the set S somewhere; let y be the first node on P that is not in S, and let x ∈ S be the node just before y.



The alternate s–v path P through x and y is already too long by the time it has left the set S.



Let P' be the subpath of P from s to x. Since $x \in S$, we know by the induction hypothesis that P_x is a shortest s-x path (of length $d(x)$), and so $l(P') \geq l(P_x) = d(x)$. Thus the subpath of P out to node y has length $l(P') + l(x, y) \geq d(x) + l(x, y) \geq d'(y)$, and the full path P is at least as long as this subpath.

Finally, since Dijkstra’s Algorithm selected v in this iteration, we know that $d'(y) \geq d'(v) = l(P_v)$. Combining these inequalities shows that $l(P) \geq l(P') + l(x, y) \geq l(P_v)$.

18

Implementation and running time

Dijkstra's Algorithm (G, ℓ)

Let S be the set of explored nodes

For each $u \in S$, we store a distance $d(u)$

Initially $S = \{s\}$ and $d(s) = 0$

While $S \neq V$ → runs $n-1$ times

Select a node $v \notin S$ with at least one edge from S for which

$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$ is as small as possible

Add v to S and define $d(v) = d'(v)$

EndWhile

if considering all v at every iteration ⇒ $O(mn)$

Implementation and running time

```
Dijkstra's Algorithm (G, ℓ)
Let S be the set of explored nodes
    For each u ∈ S, we store a distance d(u)
Initially S = {s} and d(s) = 0
While S ≠ V
    Select a node v ∉ S with at least one edge from S for which
        d'(v) = min_{e=(u,v):u∈S} d(u) + ℓ_e is as small as possible
    Add v to S and define d(v) = d'(v)
EndWhile
```

- Store d'(v) explicitly for all v ∈ V - S
- Also, store V-S nodes in a priority queue with d'(v) as the key.

Recall priority queue

- Read about priority queues in heap sort.
- Operations: insert, delete, maximum/minimum, extract-max/extract-min, increase-key/decrease-key

Dijkstra using Priority Queues

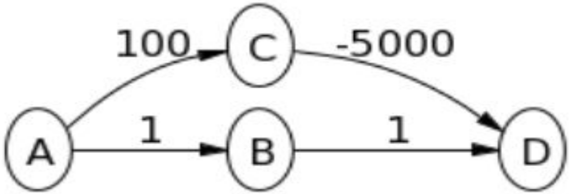
- Put nodes V in a priority queue with d'(v) as key for $v \in V$
- To add a new node to set S, we need to extract-min
- How to update the keys [d'(v)]?
- Consider: v is added to S and $w \notin S$, that is w is still in the priority queue.
- How do you update d'(w)?

How do you update d'(w)?

- If (v,w) is not an edge => w is not a neighbour of v, do nothing.
- However, if e'= (v,w) ∈ E, then, new value for d'(w) is min(d'(w), d(v)+$l_{e'}$)
- If d'(w) > d(v) + $l_{e''}$, then, we need the change-key operation.
- Because each edge is considered once, the algorithm runs in O(m) + time for extract-min and change-key => **O(mlogn).**

To think and answer

1. Will the shortest path between the start node u and destination node v change if we subtract 1 from all edges?
2. Will the shortest path between the start node u and destination node v change if we multiply 5 with all edges?
3. Can you apply Dijkstra's to the following?



4. Will dijkstra work if all edges have the same weight?