Mathematical concepts for computer science

Graph algorithms

shortest path problem

- In graph theory, the shortest path problem is the problem of **finding a path between two vertices** (or nodes) in a graph such that the **sum of the weights of its constituent edges is minimized**.

shortest path problem

In a *shortest-paths problem*, we are given a weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow \mathbb{R}$ mapping edges to real-valued weights. The *weight* $w(p)$ of path $p = \langle v_0, v_1, \ldots, v_k \rangle$ is the sum of the weights of its constituent edges:

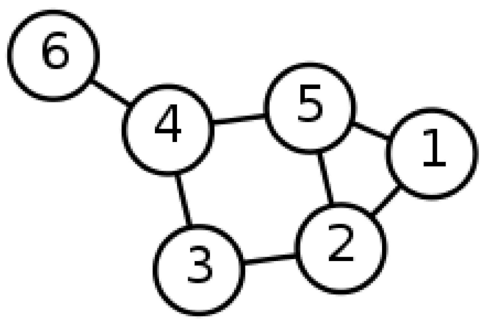
$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) .$$

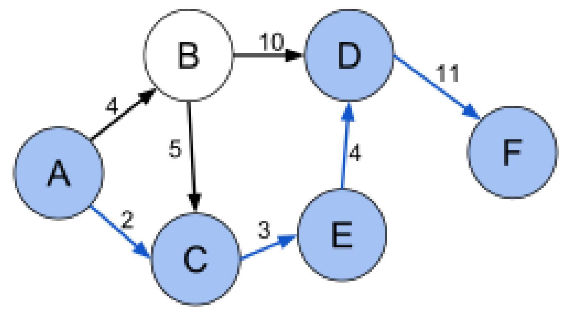
We define the *shortest-path weight* $\delta(u, v)$ from u to v by

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\leadsto} v\} & \text{if there is a path from } u \text{ to } v , \\ \infty & \text{otherwise .} \end{cases}$$

A *shortest path* from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.

shortest path problem





(6, 4, 5, 1) and (6, 4, 3, 2, 1) are both paths between vertices 6 and 1

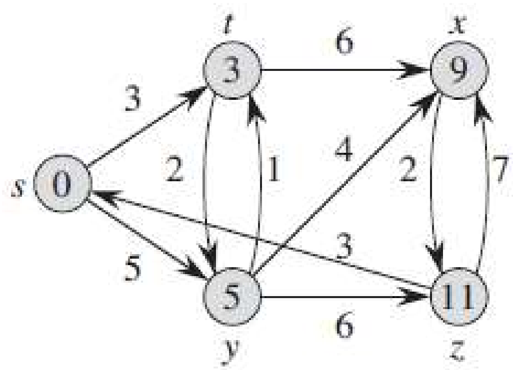
Shortest path (A, C, E, D, F) between vertices A and F in the weighted directed graph

Single-source shortest-paths problem

- Given a graph G=(V,E), we want to find a **shortest path** from a given ***source vertex*** **s** ϵ V **to each vertex v** ϵ V.
- The algorithm for the single-source problem can solve many other problems

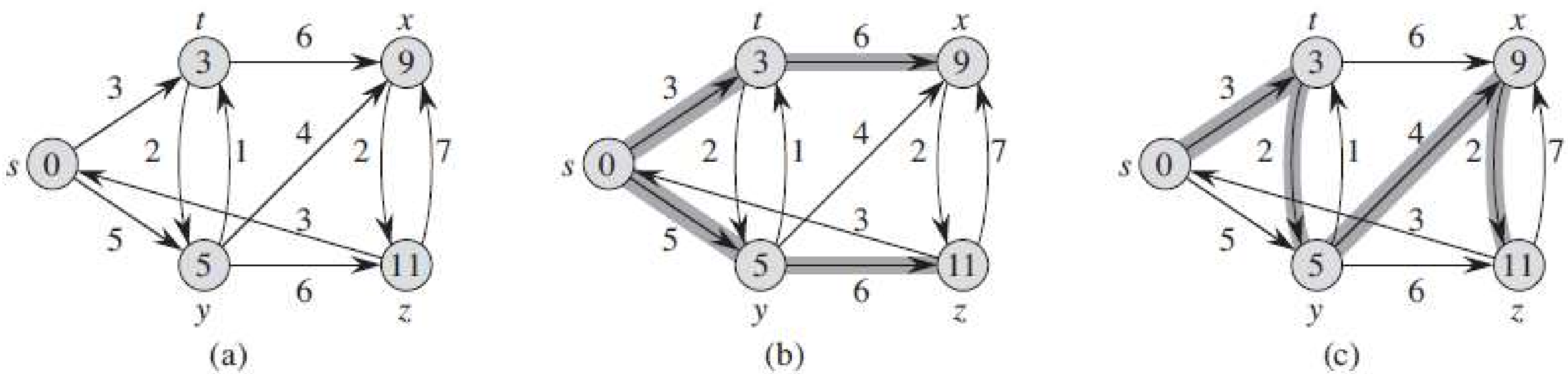
  - **Single-destination shortest-paths problem**
  - **Single-pair shortest-path problem**
  - **All-pairs shortest-paths problem**

Single-source shortest-paths problem



Identify single source shortest paths starting from vertex s

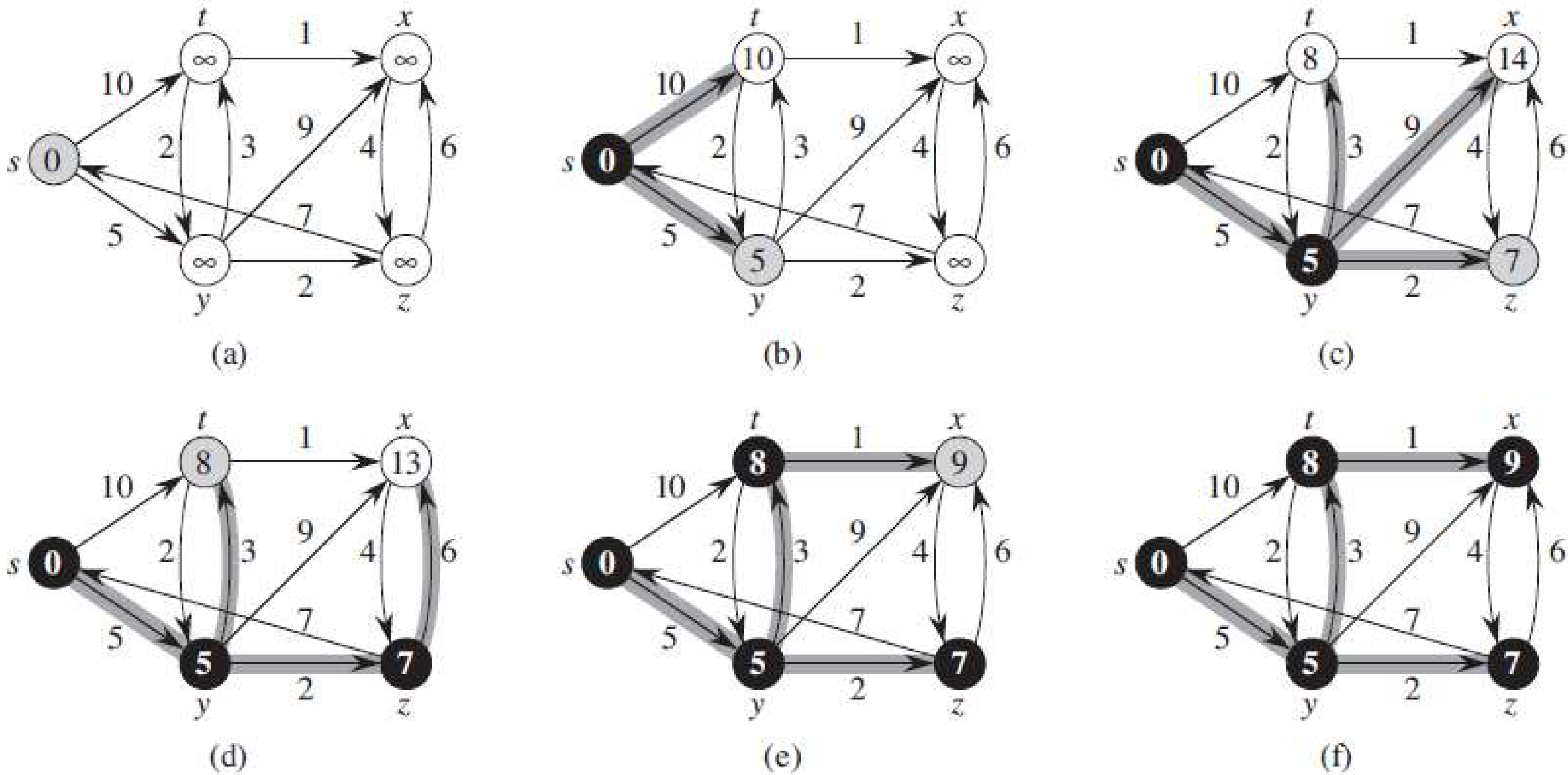
Single-source shortest-paths problem



Dijkstra’s algorithm

- Dijkstra’s algorithm solves the **single-source shortest-paths problem** on a **weighted, directed graph** G=(V,E) for the case in which **all edge weights are non-negative**.

Dijkstra’s algorithm



Dijkstra’s algorithm always chooses the “lightest” or “closest” vertex.
It uses a greedy strategy.

Dijkstra’s algorithm

Pseudocode	array	binary min-heap
INITIALIZE-SINGLE-SOURCE(G, s)		
for each vertex $v \in G.V$		
$\quad v.d = \infty$	$\Theta(V)$	$\Theta(V)$
$\quad v.\pi = \text{NIL}$		
$s.d = 0$		
RELAX(u, v, w)		
if $v.d > u.d + w(u, v)$	$O(1)$	$O(\lg V)$
$\quad v.d = u.d + w(u, v)$		
$\quad v.\pi = u$		
DIJKSTRA(G, w, s)		
INITIALIZE-SINGLE-SOURCE(G, s)		
$S = \emptyset$		
$Q = G.V$	$O(1)$	$O(\lg V)$
while $Q \neq \emptyset$	$O(V)$	$O(V)$
$\quad u = \text{EXTRACT-MIN}(Q)$	$O(V)$	$O(\lg V)$
$\quad S = S \cup \{u\}$		
$\quad$ **for** each vertex $v \in G.Adj[u]$	$\lvert E \rvert$	$\lvert E \rvert$
$\quad\quad$ RELAX(u, v, w)		
	$O(V^2 + E) = O(V^2)$	$O((V + E) \lg V)$

Theorem (Correctness of Dijkstra's algorithm)

Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $u.d = \delta(s, u)$ for all vertices $u \in V$.

Proof We use the following loop invariant:

> At the start of each iteration of the **while** loop of lines 4–8, $v.d = \delta(s, v)$ for each vertex $v \in S$.

It suffices to show for each vertex $u \in V$, we have $u.d = \delta(s, u)$ at the time when u is added to set S. Once we show that $u.d = \delta(s, u)$, we rely on the upper-bound property to show that the equality holds at all times thereafter.

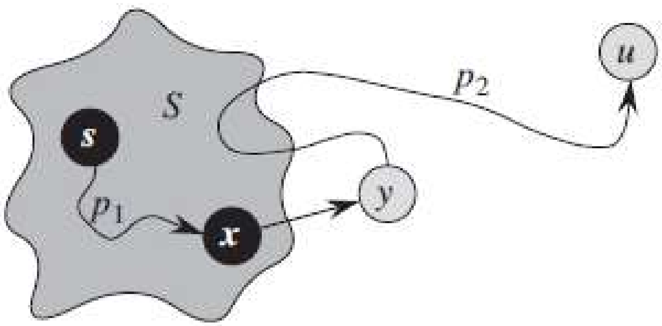
Theorem (Correctness of Dijkstra's algorithm)

Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $u.d = \delta(s, u)$ for all vertices $u \in V$.

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Initialization: Initially, $S = \emptyset$, and so the invariant is trivially true.

Theorem (Correctness of Dijkstra's algorithm)



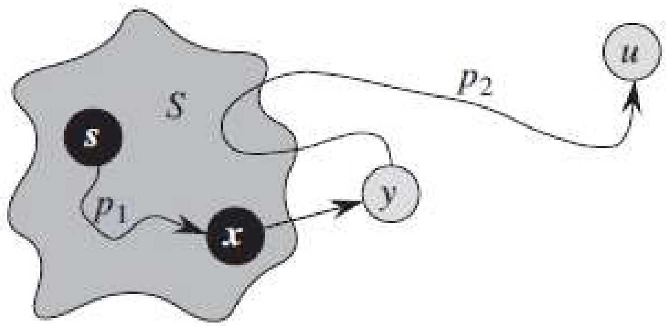

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

We wish to show that in each iteration, **u.d= δ(s, u)** for the vertex added to set S.

For the purpose of contradiction

let u be the first vertex for which **u.d ≠ δ(s, u)** when it is added to set S.

Theorem (Correctness of Dijkstra's algorithm)



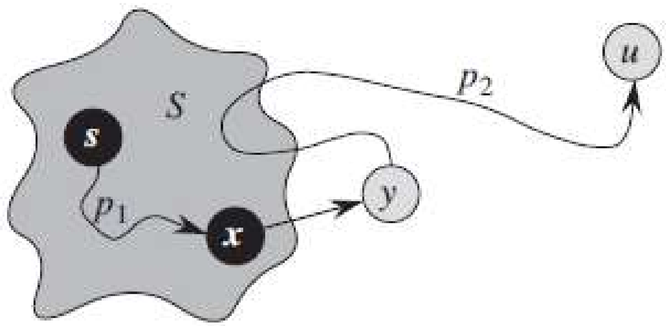

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

u ≠ s because s is the first vertex added to set S and **s.d=δ(s, s)=0** at that time.

Because **u ≠ s** , we also have that **S ≠ ϕ** ; just before u is added to S.

There must be some path from s to u, for otherwise **u.d=δ(s, u)=α** by the no-path property, which would violate our assumption that **u.d ≠ δ(s, u).**

Theorem (Correctness of Dijkstra's algorithm)

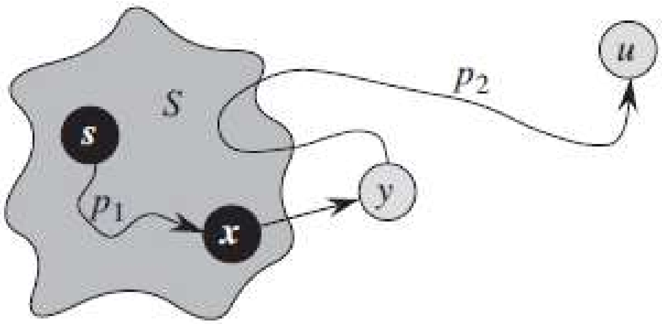



```
DIJKSTRA(G, w, s)
  INITIALIZE-SINGLE-SOURCE(G, s)
  S = ∅
  Q = G.V
  while Q ≠ ∅
      u = EXTRACT-MIN(Q)
      S = S ∪ {u}
      for each vertex v ∈ G.Adj[u]
          RELAX(u, v, w)
```

S $\quad V - S$

We claim that $y.d = \delta(s, y)$ when u is added to S.

Theorem (Correctness of Dijkstra's algorithm)



We can now obtain a contradiction to prove that $u.d = \delta(s, u)$. Because y appears before u on a shortest path from s to u and all edge weights are non-negative (notably those on path p_2), we have $\delta(s, y) \leq \delta(s, u)$, and thus

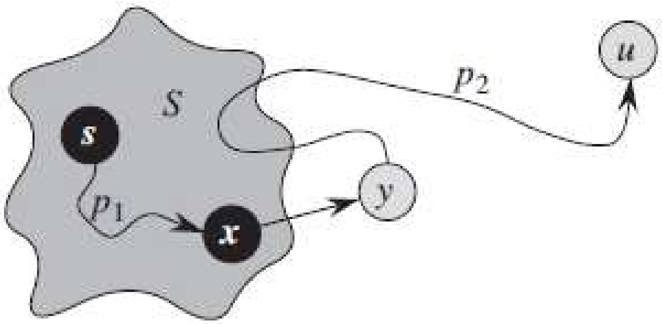
$$
\begin{aligned}
y.d &= \delta(s, y) \\
&\leq \delta(s, u) \qquad\qquad (24.2) \\
&\leq u.d \qquad \text{(by the upper-bound property)} .
\end{aligned}
$$

But because both vertices u and y were in $V - S$ when u was chosen in line 5, we have $u.d \leq y.d$. Thus, the two inequalities in (24.2) are in fact equalities, giving

$$y.d = \delta(s, y) = \delta(s, u) = u.d .$$

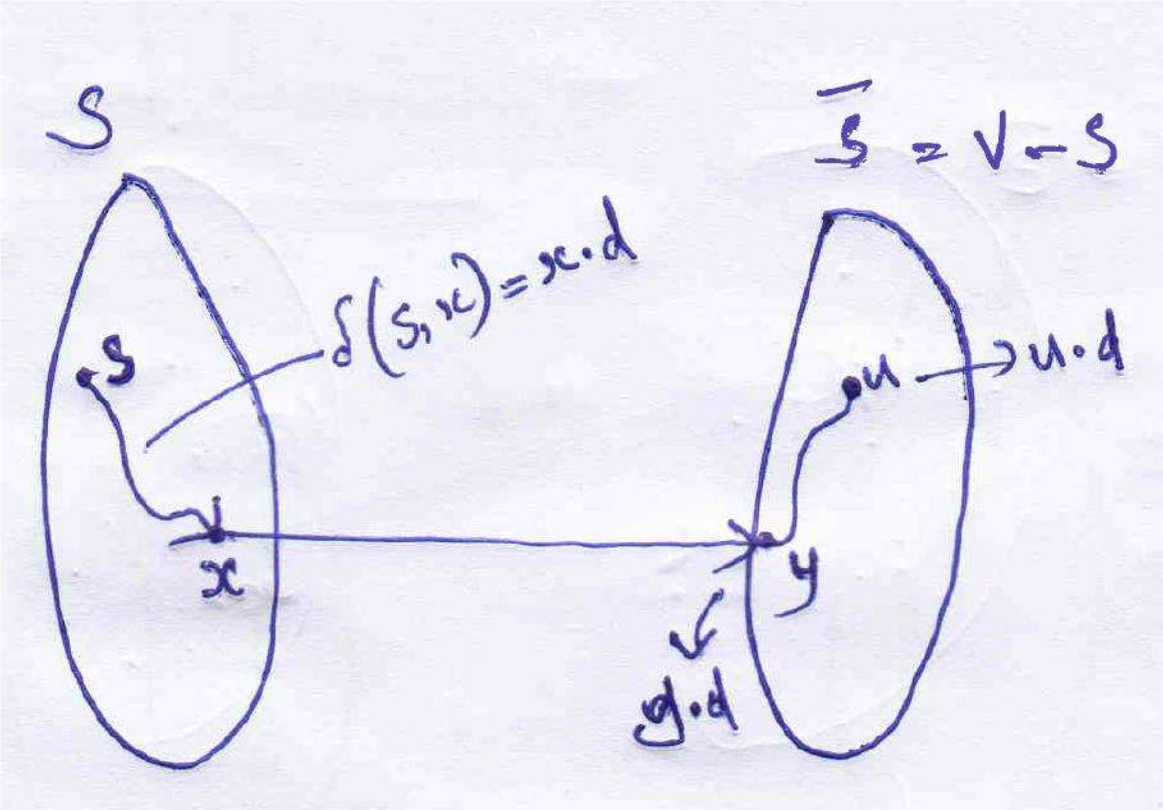
Consequently, $u.d = \delta(s, u)$, which contradicts our choice of u. We conclude that $u.d = \delta(s, u)$ when u is added to S, and that this equality is maintained at all times thereafter.

Theorem (Correctness of Dijkstra's algorithm)



Termination: At termination, $Q = \emptyset$ which, along with our earlier invariant that $Q = V - S$, implies that $S = V$. Thus, $u.d = \delta(s, u)$ for all vertices $u \in V$. ■

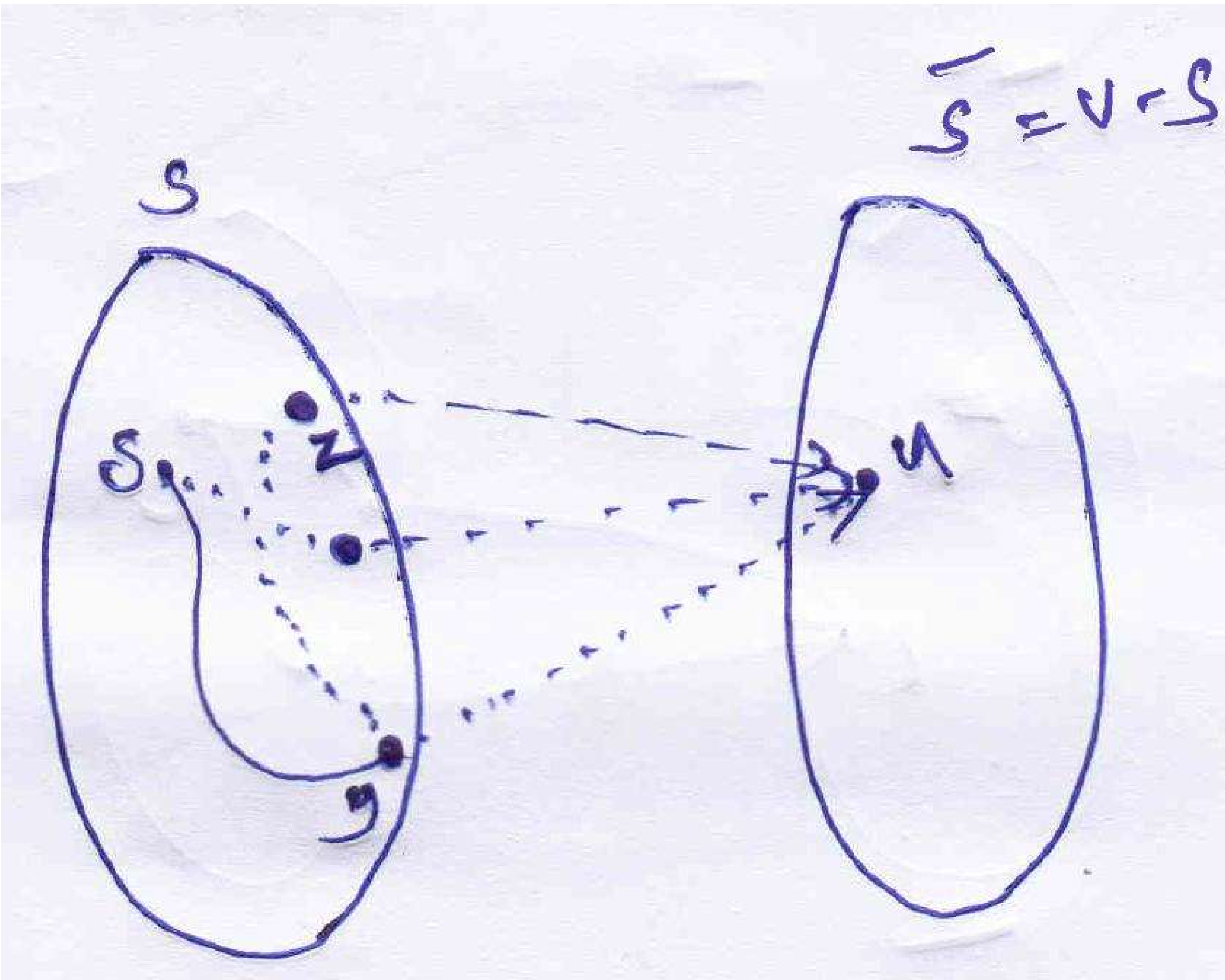
Theorem (Correctness of Dijkstra's algorithm)



Assume that this is the optimal path from s to u

What are the other paths possible ?

Theorem (Correctness of Dijkstra's algorithm)



RELAX(u, v, w)

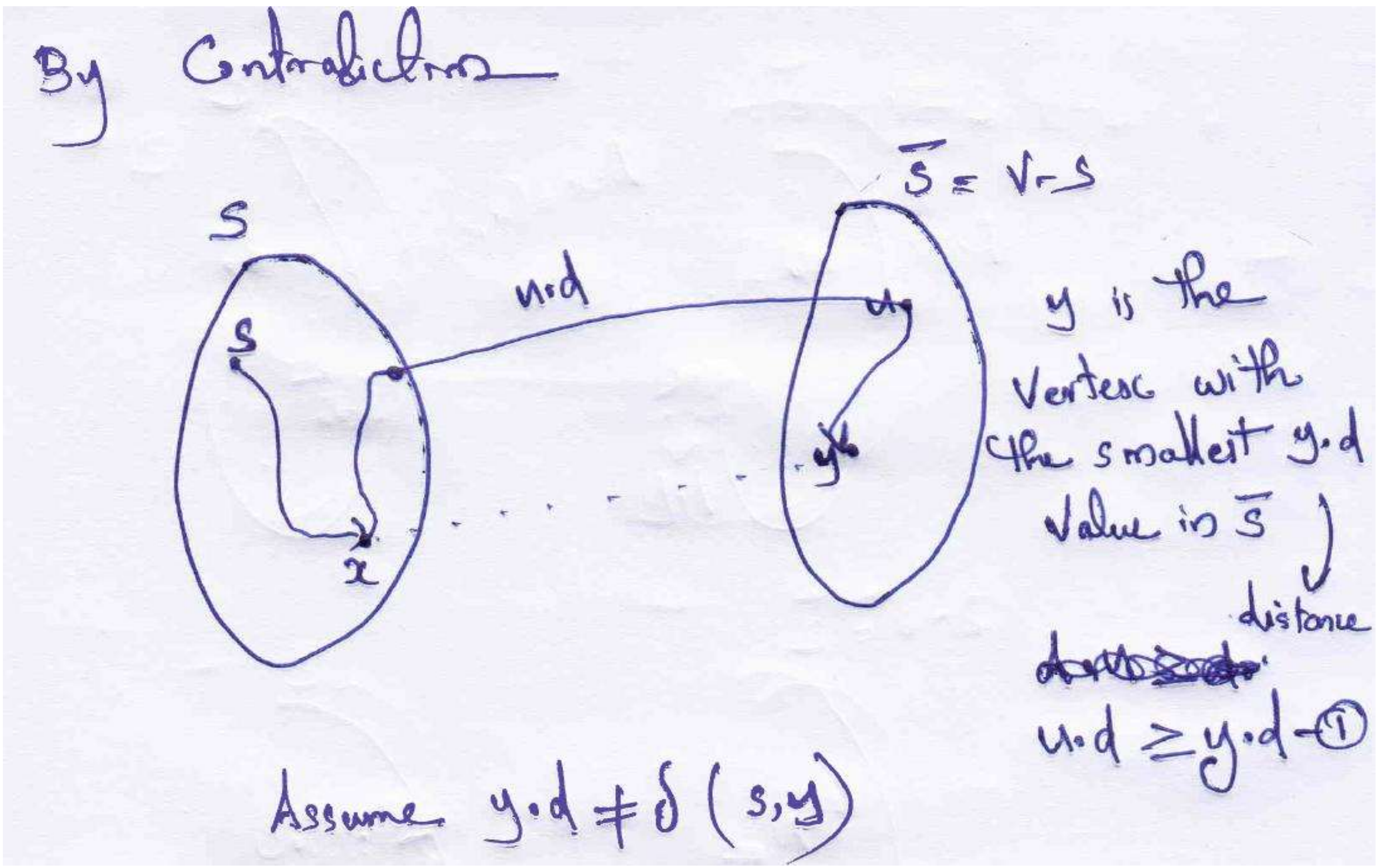
```
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

u.d=min[u.d, y.d + w(y,u)]

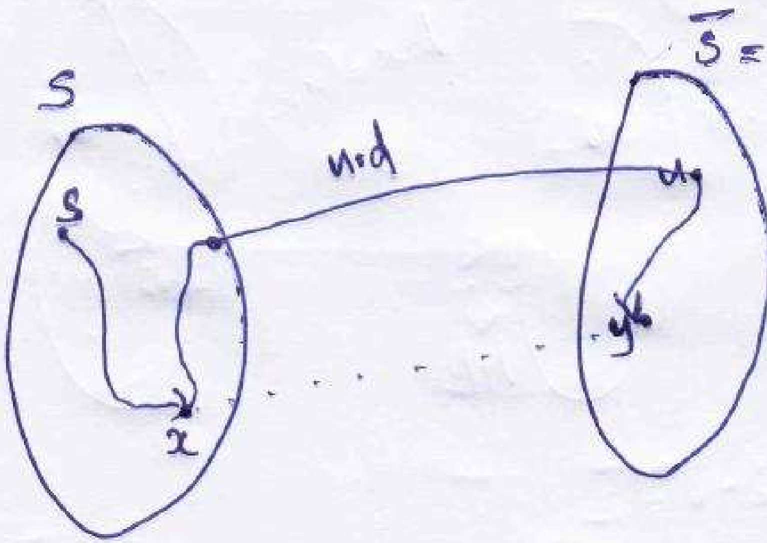
Theorem (Correctness of Dijkstra's algorithm)

We claim that $y.d = \delta(s, y)$ when u is added to S.

Claim: y.d is the shortest path starts from s to y.



Theorem (Correctness of Dijkstra's algorithm)



y is the vertex with the smallest y.d value in $\bar{S}$

distance

~~[illegible]~~

$u.d \geq y.d$ — ①

Assume $y.d \neq \delta(s, y)$

$y.d \geq u.d$ — ②

$y.d =$
$u.d + (u,y)w$

$\Downarrow$

① & ② $\underline{u.d = y.d}$

Negative weighted graph