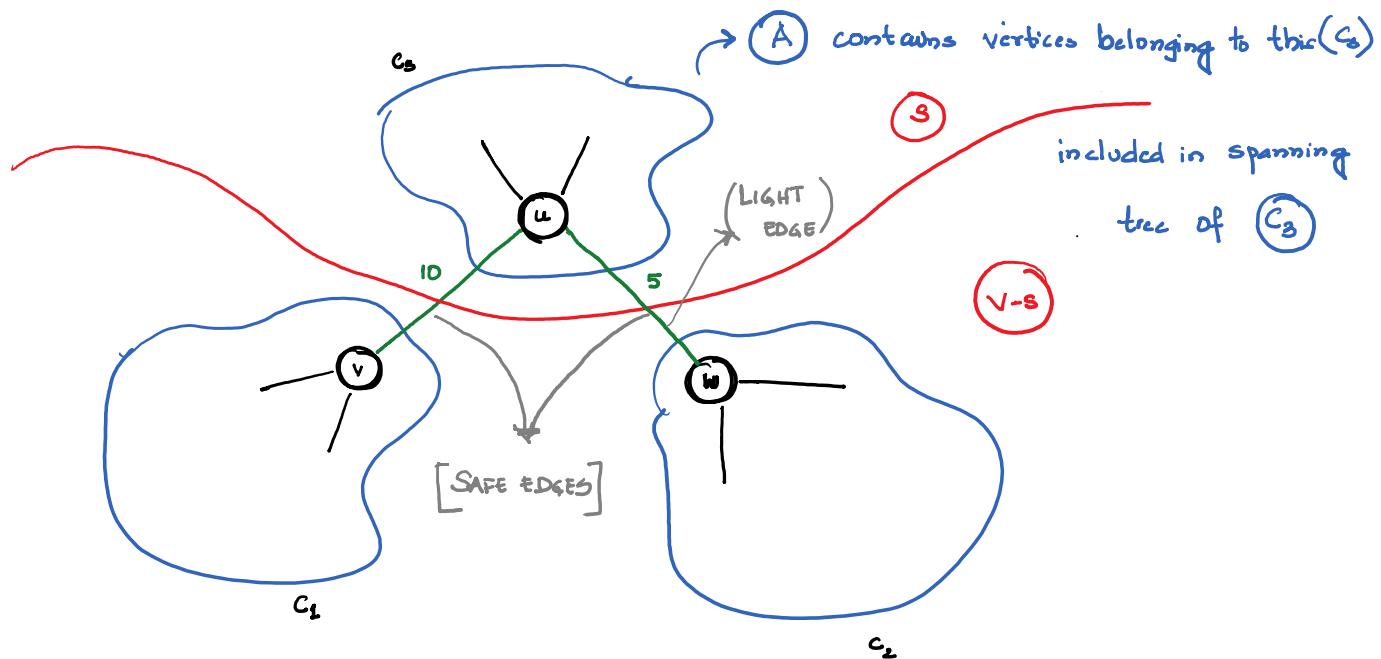


23.1-2) Professor Sabatier conjectures converse of Theorem 23.1. Let $G = (V, E)$ be a connected undirected graph with a real valued weight function w defined on E . Let A be a subset of E that is included in some minimum spanning tree for G , let $(S, V-S)$ be any cut of G that respects A , and let (u, v) be a safe edge for A crossing $(S, V-S)$. Then (u, v) is a light edge for the cut. Show that professor's conjecture is incorrect by giving a counter example.

THEOREM 23.1 : Let $G(V, E)$ be a connected undirected graph with a real valued weight function "w" defined on E . Let A be a subset of E that is included in some minimum spanning tree. Let $(S, V-S)$ be any cut of G that respects A . Let (u, v) be light edge crossing $(S, V-S)$ then edge (u, v) is safe for A .

INTUITION : Connecting different connected components by single edges (E)



C_1, C_2, C_3 represent three connected components (sub) joined to main component C .

Here minimum spanning tree of the whole graph formed by component C includes both the edges (u,v) , (u,w)

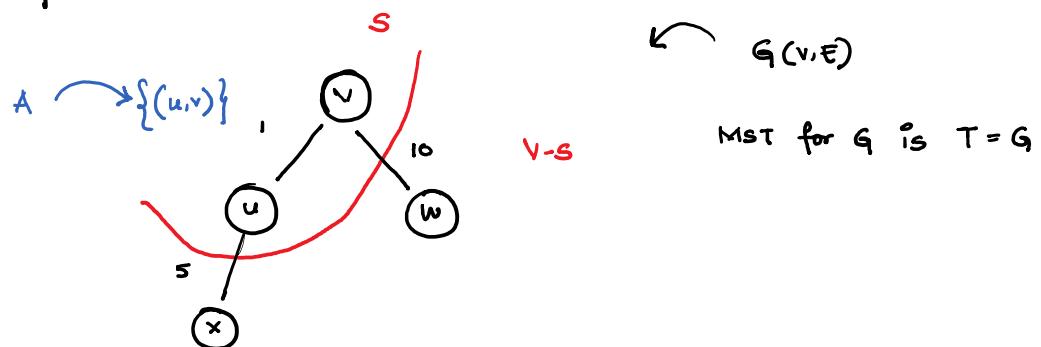
Now define a cut as shown in red to include

c_1 in S and c_2, c_3 in V-S

It can be observed that both edges (u,v) , (u,w) are safe but only one of it i.e (u,w) is a light edge

Thus proving professor's statement false:

Trivial example:



A is subset of MST - (T)

Edges (u,x) , (v,w) are safe but only (u,x) is LIGHT EDGE

Proving professor's converse false

23.1-6) Show that a graph has a unique minimum spanning tree if, for every cut of the graph, there is a unique light edge crossing the cut. Show that the converse is not true by giving a counter example.

PROOF BY CONTRADICTION :

Let T, T' be two minimum spanning trees corresponding to graph $G(V, E)$ with property, for every cut of the graph there is unique light edge

For T, T' to be distinct they should differ by atleast two edges

Let the differed edges be $(u, v), (a, b)$ such that

$$\left\{ (u, v) \notin T', (u, v) \in T \right\} \text{ & } \left\{ (a, b) \in T', (a, b) \notin T \right\}$$

Let us define our cut $S = \{u, a\}$. & $V - S$ contains $\{v, b\}$.

As there exists a UNIQUE LIGHT EDGE spanning any cut.

Now consider T, T' only one of $(u, v), (a, b)$ (or something else) can be a light edge spanning this cut.

If (u, v) is light edge of this cut then

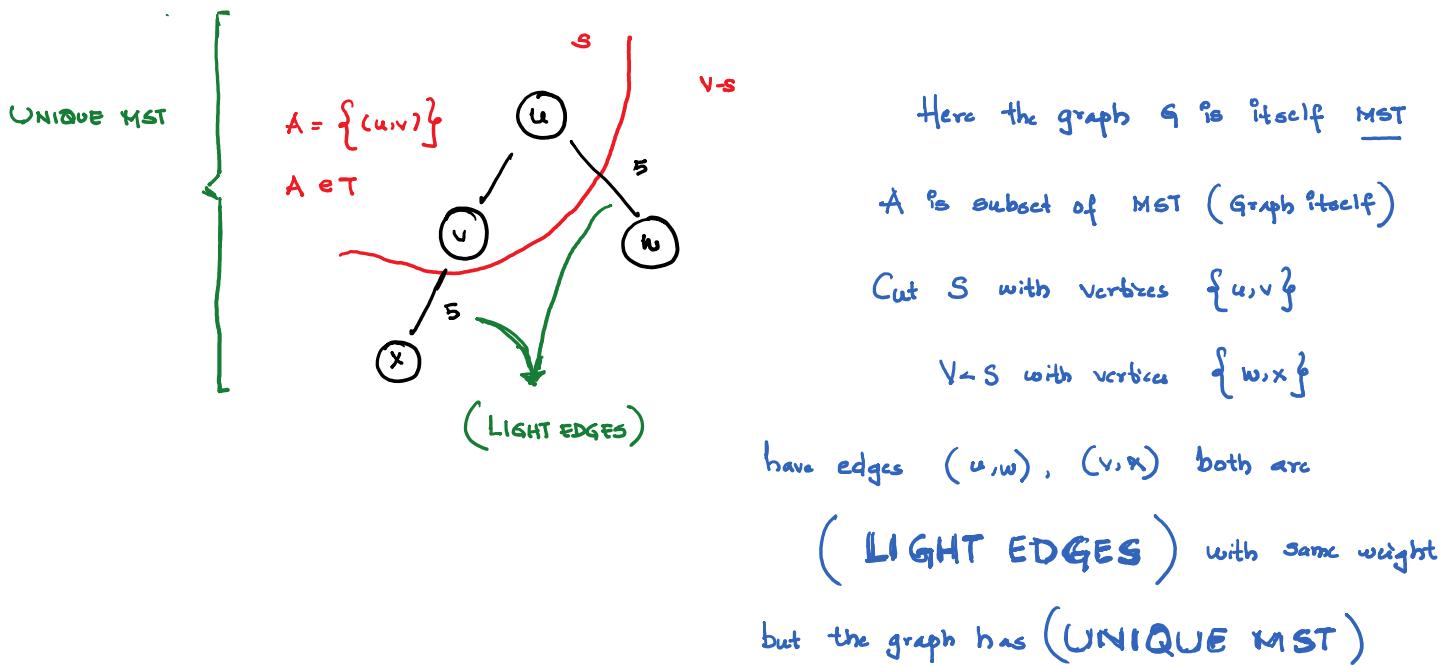
$(a, b) \in T'$ can be replaced with (u, v) forming a more minimum SPT

Hly if (a, b) is light edge (u, v) in T can be replaced.

So, T, T' are no longer minimum spanning trees \rightarrow [CONTRADICTION !!]

COUNTER EXAMPLE FOR CONVERSE :

Consider this trivial MST for graph $g(V, E)$ defined as follows:



28.1-10) Given a graph G and a minimum spanning tree T , suppose we decrease the weight of one of the edges in T . Show that T is still a minimum spanning tree for G . More formally, let T be a minimum spanning tree for G with edge weights given by weight function w . Choose one edge $(x,y) \in T$ and a positive number k and the weight function w' by.

$$w'(u,v) = \begin{cases} w(u,v) & \text{if } (u,v) \neq (x,y) \\ w(x,y) - k & \text{if } (u,v) = (x,y) \end{cases}$$

Show that T is a minimum spanning tree for G with edge weights given by w'

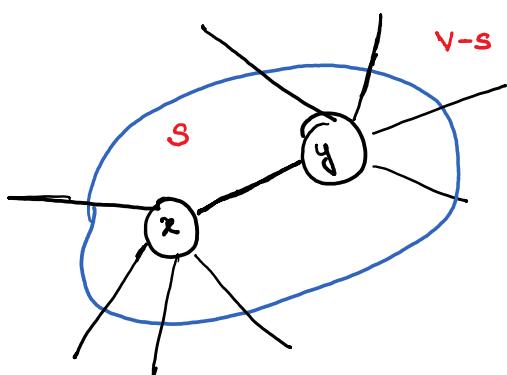
For graph G , $(x,y) \in$ minimum spanning tree (MST) — T

Weight of edge (x,y) is decreased

T.B.P : T is still the minimum spanning tree (MST)

Consider the following cut in G where

$$S = \{x, y\} \quad V-S = V \setminus \{x, y\}$$



Given edge (x,y) is part of MST

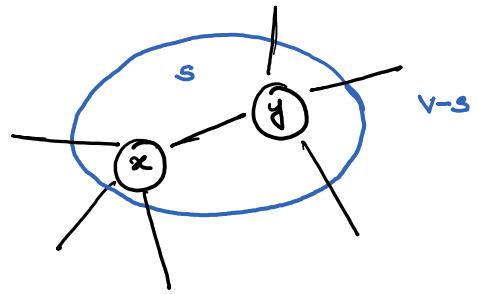
To include (x,y) in MST

$$[(w(x,y)) + (\text{light edge spanning this cut})]$$

$$< [(\text{light edge spanning this cut}) + (\text{light edge spanning the cut formed after including one of } (x,y) \text{ in } V-S)] \rightarrow ①$$

The above equation shows pictorially. \rightarrow

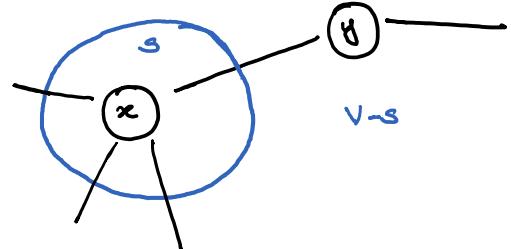
①



without loss of generality assume light edge

corresponds to one of edges of vertex y

②



Now, for this picture edge (x,y) should be one of the light edges corresponding to the defined cut $S = \{z\}$, $V-S = V \setminus \{z\}$ for it to be part of MST.

OPERATION : $[w'(x,y) = w(x,y) - k]$ and keeping all the other edges same.

For the graph G consider ② cut which establishes for (x,y) to be part of the MST (x,y) should be one amongst light edges

So, all other edges spanning the cut $\geq w(x,y)$

Now modified graph has $w'(x,y)$ instead of $w(x,y)$.

$w'(x,y) < w(x,y) \leq$ All other edges spanning the cut

So, $w'(x,y)$ shall be the only light edge spanning the cut now
by T which includes (x,y) shall remain the minimum spanning tree.

Akhilesh

Consider $MST \rightarrow T$ for original Graph G (\underline{w})

In this MST for edge (x,y) weight has been decreased by k

Let T' be the new MST for graph G' with reduced weight $\underline{w'}$

For T' to be different than T

$$[\text{NEW MST } T'] \leftarrow [\text{ORIGINAL MST}] \quad w'(T') < w(T) - k \quad [\text{REDUCED WEIGHT}] \longrightarrow \textcircled{1}$$

Now take this $\textcircled{T'}$ \rightarrow for original graph \textcircled{G}

here (x,y) may (so may not be) in T'

$$\Rightarrow \begin{cases} w(T') \leq w'(T') + k & \xrightarrow{\text{REDUCTION}} \textcircled{2} \\ \begin{array}{l} \text{WEIGHT IN ORIGINAL} \\ \text{GRAPH FOR NEW MST } T' \end{array} & \begin{array}{l} \text{WEIGHT IN} \\ \text{MODIFIED GRAPH FOR} \\ \text{NEW MST} \end{array} \end{cases}$$

If (x,y) is not in $T' \rightarrow w(T') = w'(T')$

If present $\rightarrow w(T') = w'(T') + k$

From $\textcircled{1}$ & $\textcircled{2}$ $w'(T') < w(T) - k, w(T') \leq w'(T') + k$

$$\Rightarrow \boxed{w(T') < w(T)}$$

CONTRADICTION!! that T is not MST under w

BOTTLENECK SPANNING TREE :

A bottleneck spanning tree T of an undirected graph G is a spanning tree of G whose largest edge weight is minimum over all spanning trees of G . We say that bottleneck spanning tree is the weight of the maximum-weight edge in T .

- Argue that a minimum spanning tree is a bottleneck spanning tree
- Give a linear time algorithm that given a graph G and an integer b , determines whether the value of the bottleneck spanning tree is atmost (b)

a) Bottleneck spanning tree : SPANNING TREE of G whose largest weight is MINIMUM over all spanning trees of G .

Value of BOTTLENECK SPANNING TREE maximum weight of edge in (T)

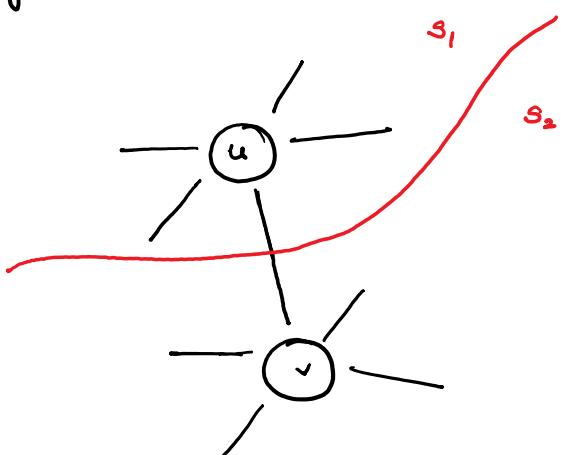
MINIMUM SPANNING TREE \Rightarrow BOTTLENECK SPANNING TREE

Let T be MST of G (u,v) be some edge of MST whose weight is larger than the value of bottleneck spanning tree.

Consider graph $G(v,e)$ as follows

S_1 contains all vertices reachable from \underline{u}

S_2 contains all vertices reachable from \underline{v}



since (u,v) is part of MST of graph then

for the cut defined above formed by (S_1, S_2) , (u,v) should be one of the light edge.

As there exists some edge in bottleneck spanning tree

with lesser weight than (u,v) crossing the cut (S_1, S_2) .

→ This is a contradiction to our assumption that (u,v) is part of MST as the lighter edge of bottleneck spanning tree can be included here to form a much lighter minimum spanning tree.

b) Linear time algorithms to check for graph G and almost b

$O(E)$ {
 for each e in $G \cdot E$ do
 if $w(e) > b$ then
 $G \cdot \text{remove}(e)$ }
First let us remove all edges with weight greater than b

$O(V+E) \leftarrow$ if (IS_CONNECTED(G)): → BFS or DFS to check if the resulted graph after discarding edges is connected or not

Any path traced by DFS or BFS

shall be bottleneck spanning trees of weight almost b

RUNTIME:

$O(V+E)$ (linear)

24.3-6)

We are given a directed graph $G = (V, E)$ on which each edge $(u, v) \in E$ has an associated value $r(u, v)$, which is a real number in the range $0 \leq r(u, v) \leq 1$ that represents the reliability of a communication channel from vertex u to vertex v . We interpret $r(u, v)$ as the probability that the channel from u to v will not fail. and we assume that these probabilities are independent. Give an efficient algorithm to find the most reliable path between two given vertices.

Reliable paths consist of paths with maximum probability of reaching the other vertex

Let u, u_1, \dots, u_k be vertices in reliable path from u to v

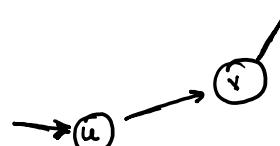
$$\text{Reliability} : r(u, u_1) r(u_1, u_2) \dots r(u_k, v)$$

OPTIMAL SUBSTRUCTURE

Reliable path consists of reliable sub-paths.

INTUITION

Similar to Dijkstra's algorithm



we need to modify RELAXATION procedure such that

reliability of reaching vertex v when passing through vertex u is increased

\nwarrow RELIABILITY OF CHANNEL (u, v)

if $\rightarrow R(u) (r(u, v)) > R(v) \rightarrow$ include (u, v) in the path.

RELIABILITY OF REACHING (u)

RELIABILITY OF REACHING (v)

Runtime also remains the same as DIJKSTRA's ALGORITHM

$O(V \log V + E)$ if FIBONACCI HEAPS are used.

MAX-RELIABILITY-ALGORITHM (G, r, s) :

INITIALIZE - SINGLE - SOURCE (G, s)

$$S = \emptyset$$

$$Q = G \cdot V$$

$O(v)$ ← while $Q \neq \emptyset$:

$O(lgv)$ ← $u = \text{EXTRACT-MAX-RELIABLE-CHANNEL}(Q)$
(FIBONACCI HEAPS)

$$S = S \cup \{u\}$$

$O(E)$
AGGREGATE
ANALYSIS

for each vertex $v \in G \cdot \text{adj}[u]$
 $\text{RELAX}(u, v, r)$

[Can be stopped once destination is reached]

INITIALIZE - SINGLE - SOURCE (G, s) :

for each vertex v in $G \cdot V$:

$$(\text{// Initial reliability of all } v) \quad R[v] = 0$$

$$R[s] = 1 \quad \text{// Reliability of reaching the source}$$

$$O(vlgv + E)$$

$\text{RELAX}(u, v, r) :$

$$\text{if } R[u] \cdot r(u, v) > R[v]$$

$$R[v] = R[u] \cdot r(u, v)$$

$$v \cdot \pi = u.$$

EXTRACT-MAX-RELIABLE-CHANNEL (Q) :

↳ It maintains a priority queue and extracts the maximum reliability channel (edge) of all the edges spanning the cut $(S, V-S)$

RUNTIME

This modified DIJKSTRA's algorithm indeed computes the MOST RELIABLE PATH

between vertices (u, v) in $O(vlgv + E)$ (FIBONACCI HEAPS)

24.3-3) Let $G = (V, E)$ be a weighted, directed graph with non-negative weight function $w : E \rightarrow \{0, 1, \dots, w\}$ for some non-negative integer w . Modify Dijkstra's algorithm to compute shortest paths from a given source vertex s in $O(wv + E)$.

Given, $G \rightarrow$ Weighted, directed graph with non-negative weights

$$w : E \rightarrow \{0, 1, 2, \dots, w\}$$

ORIGINAL DIJKSTRA'S ALGORITHM :

INITIALISE - SINGLE-SOURCE (G, s) :

$$S = \emptyset$$

$$Q = G \cdot V$$

$$O(v) \leftarrow \text{while } Q \neq \emptyset$$

AMORTIZED FOR
FIBONACCI HEAPS

AGGREGATE
ANALYSIS

$$O(\lg v) \leftarrow u = \text{EXTRACT-MIN}(Q)$$

$O(v \lg v)$

$\{$ PRIORITY QUEUE \rightarrow INSERT, EXTRACT MIN
DECREASE-KEY $\}$

RELAX (u, v, w)

if $v \cdot d > u \cdot d + w(u, v)$

$$v \cdot d = u \cdot d + w(u, v)$$

$$v \cdot \pi = u \cdot$$

[IMPROVEMENT POSSIBILITY]

$\{$ Running time depends on implementation
of PRIORITY-QUEUE $\}$

$O(E)$

for each vertex $v \in G \cdot \text{Adj}[u]$

RELAX (u, v, w)

FIBONACCI HEAPS: $O(v \lg v + E)$.

Idea of improvement: As we now know the range of weights for edges.

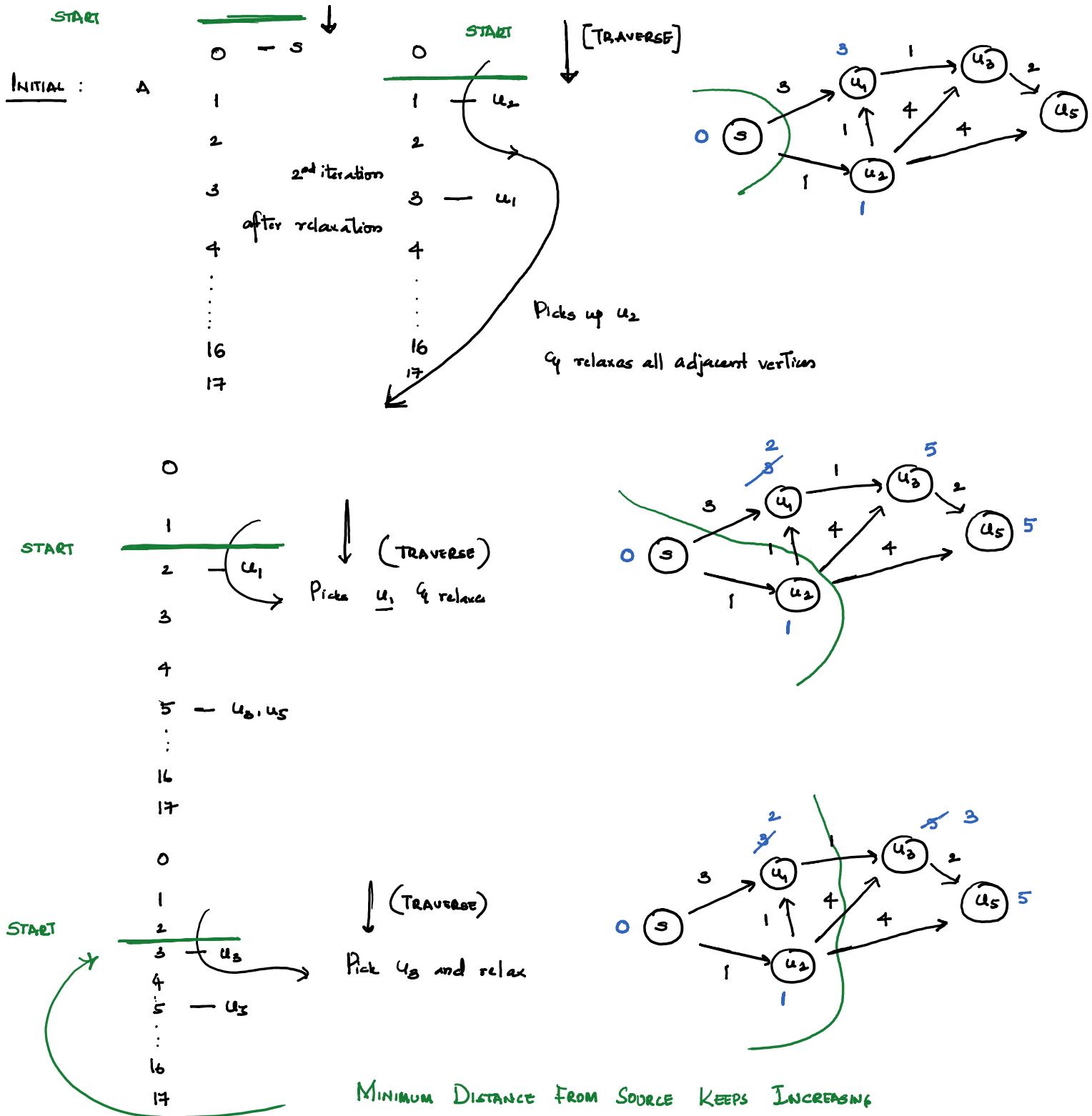
We can use a data structure to store vertices as values of

indexed by current shortest distance from the source

So, we maintain an array of size VW (worst case scenario) where

weights of all edges is W and lie on straight line

Always traverse from 0 to VW and fetch the vertex first reached. P, C



MODIFIED - ALGORITHM :

INITIALIZE - SINGLE - SOURCE (G, s)

$A[0] \cdot \text{insert}(s)$ // Initialize array

$s = \phi$ of size $(VW+2)$

$Q = G \cdot V$

$k = 0$

while $Q \neq \phi$:

// Maintain index

while $A[k] \neq \text{NULL}$ do

of minimum \boxed{k}

$k = k + 1$

$u = A[k] \cdot \text{head}()$

$A[k] \cdot \text{delete}(u)$.

// RELAXATION

for each vertex $v \in G \cdot \text{adj}[u]$ do

and updation of

if $v \cdot d > u \cdot d + w(u, v)$ then

data structure values.

$A[v \cdot d] \cdot \text{delete}(v)$

$v \cdot d = u \cdot d + w(u, v)$

$A[v \cdot d] \cdot \text{insert}(v)$

$v \cdot \pi = u$

INITIALIZE - SINGLE - SOURCE (G, s) :

for each v in $G \cdot V$

$v \cdot d = (VW+1) \rightarrow$ in place of $\boxed{\infty}$

$v \cdot \pi = \text{NULL}$

$s \cdot d = 0$.

$O(VW)$ [AGGREGATE ANALYSIS]

as k can go from 0 to $VW+1$

$O(E)$ [AGGREGATE ANALYSIS]

all of the edges are looked
only once.

Thus RUNTIME : $O(VW+E)$.

25.3 - 3)

Suppose $w(u,v) \geq 0$ for all edges $(u,v) \in E$. What is the relationship between the weight functions w and \hat{w} ?

All weights are non-negative:

$$w(u,v) = \hat{w}(u,v)$$

↳ Values computed as shortest distances on running

BELLMAN-FORD ALGORITHM will be ZERO



Computing G' of JOHNSON'S ALGORITHM

Involves placing edge weight of ZERO from s to all vertices



As there are no negative edges. Every path within the graph has no negative edges, its cost cannot be negative and does not beat the trivial path from s to any other vertex



As we have $b(u) = b(v)$, reweighting does not change weights of the graph. i.e adds or subtracts [ZERO]