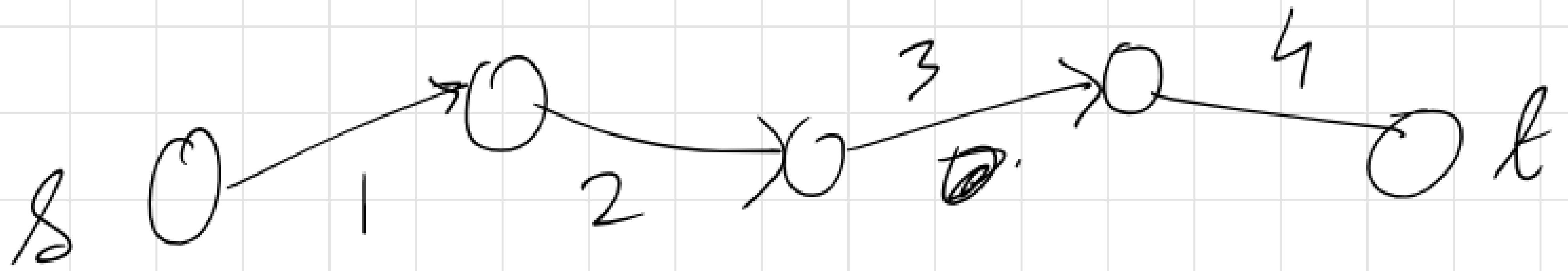
Single Source Shortest Path

<u>Problem</u> $G=(V, E)$, s, each edge e has length l_e associated to it

length of path P: $l(P) = \sum_{e \in P} l_e$



$l(P) = 10$

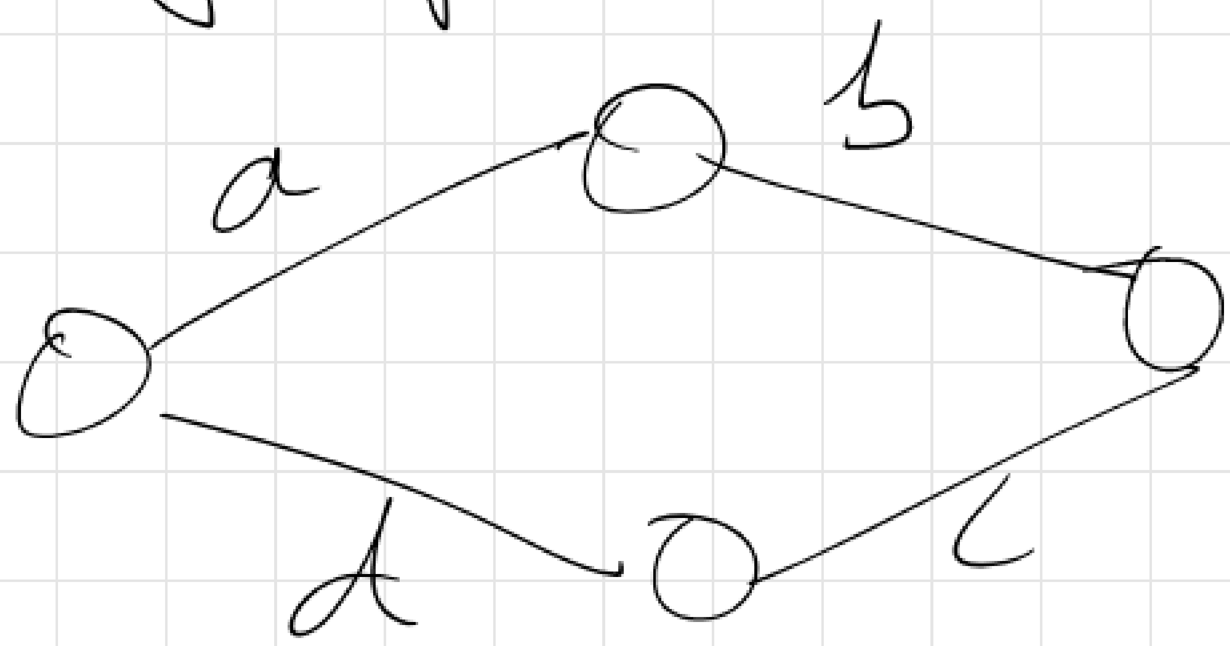
<u>Obj</u>: Determine the length of short path from s to every other node in G.

<u>Assumption</u>: ① Every node is reachable from s.

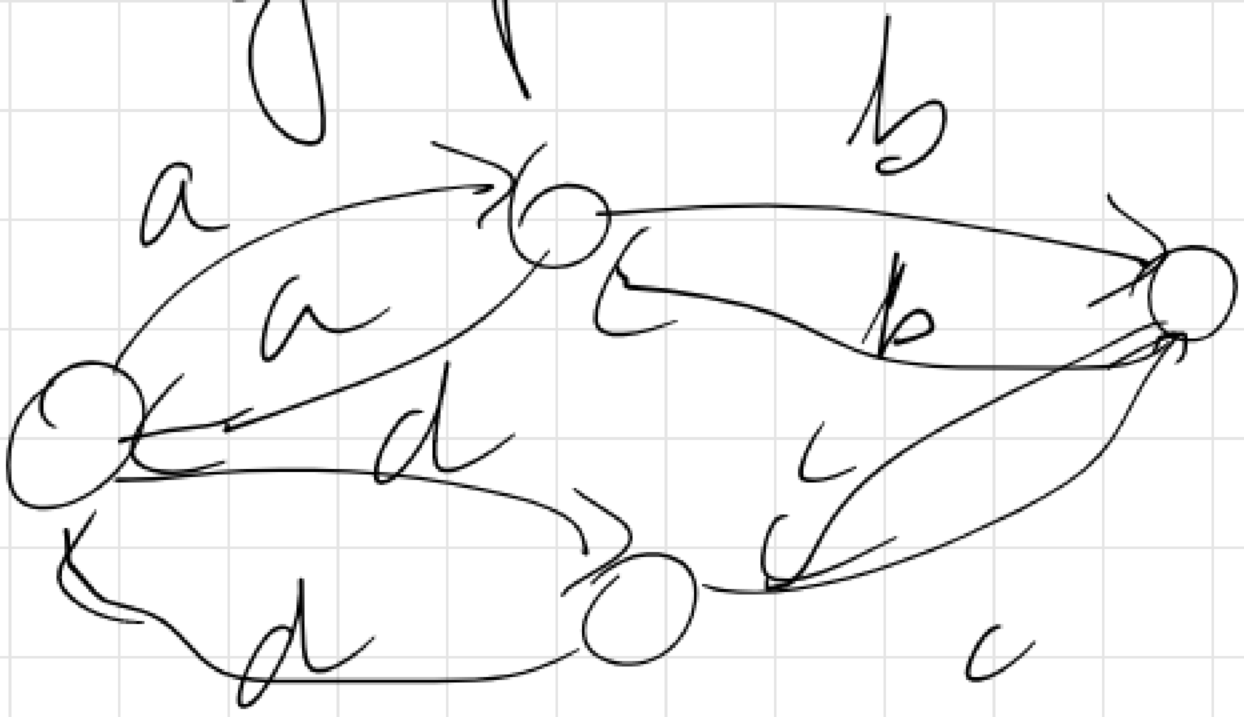
(Check this condition by doing BFS(s))

② $l_e \geq 0$ (all edge length positive)

Undirected graph



directed graph



Can we apply any known algoritum to solve this problem (SSSP)??

→ BFS

→ Explaination (Board)

Dijkstra Algorithm

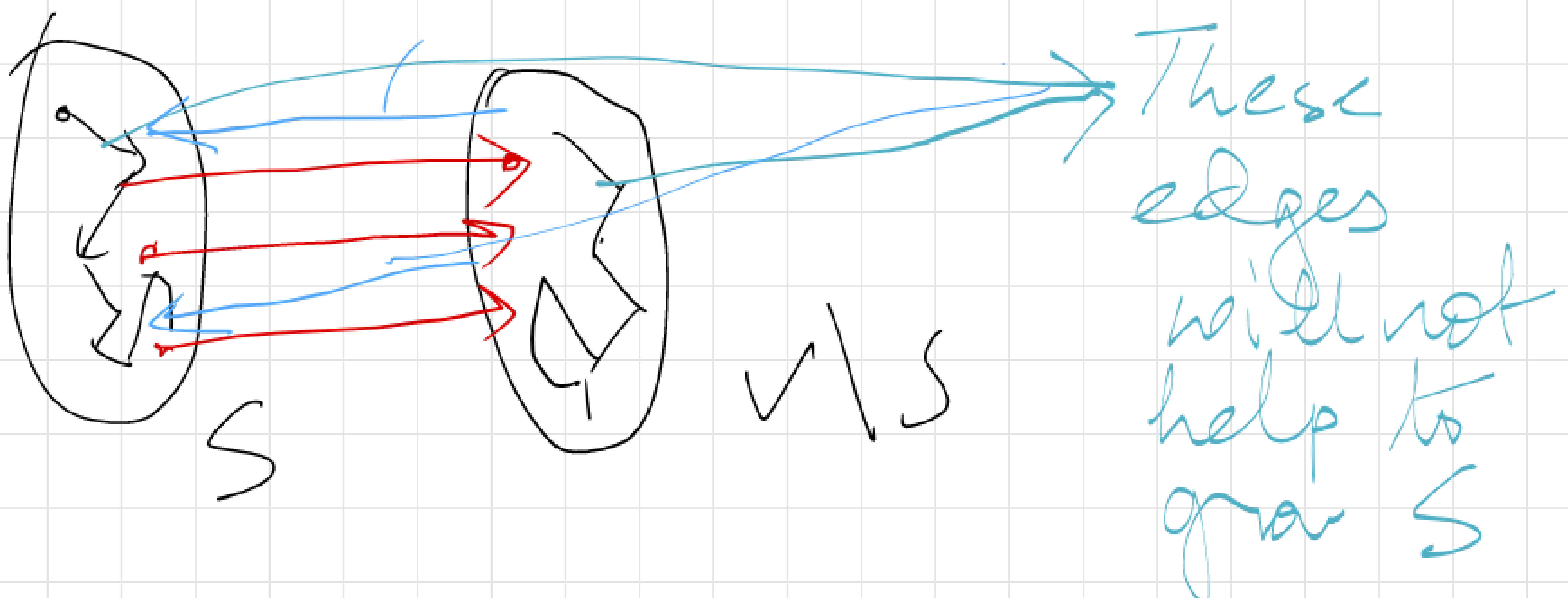
▷ Algo maintain a set S: Processed vertices

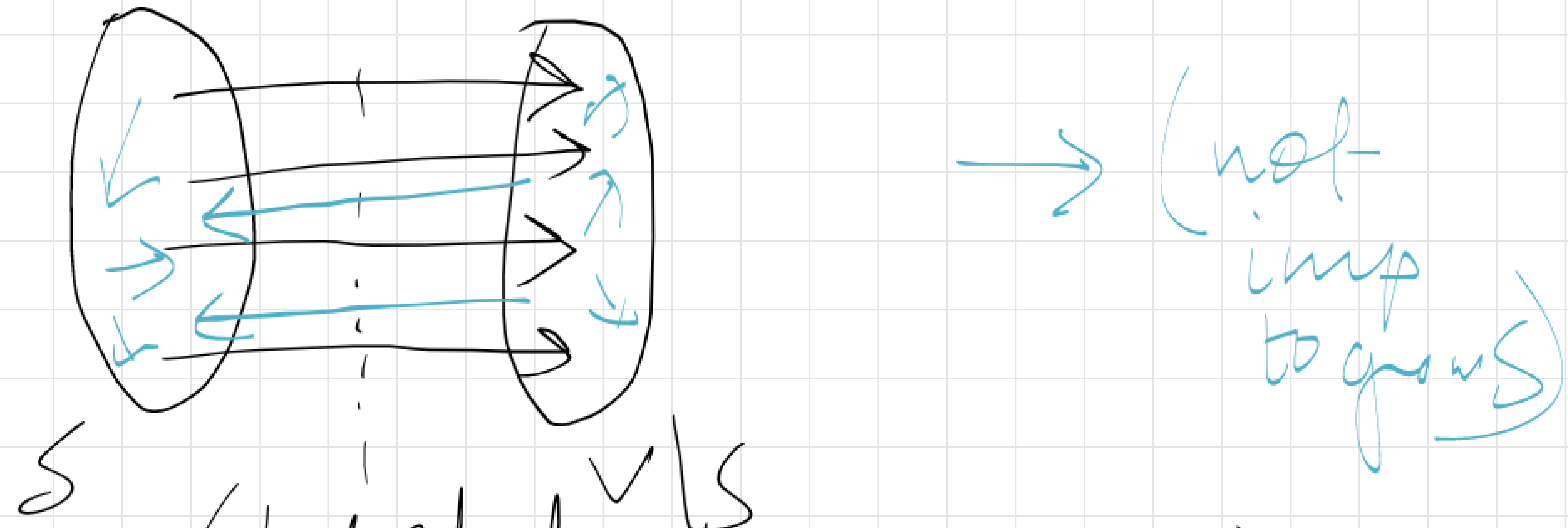
[explored set] — Computed shortest path lengths from s to each node in S

▷ $d(u)$: length of the shortest path from s to u

→ Initially $S = \{s\}$; $d(s) = 0$

→ Idea: Grow S by adding to it one node in each iteration until $S = V$

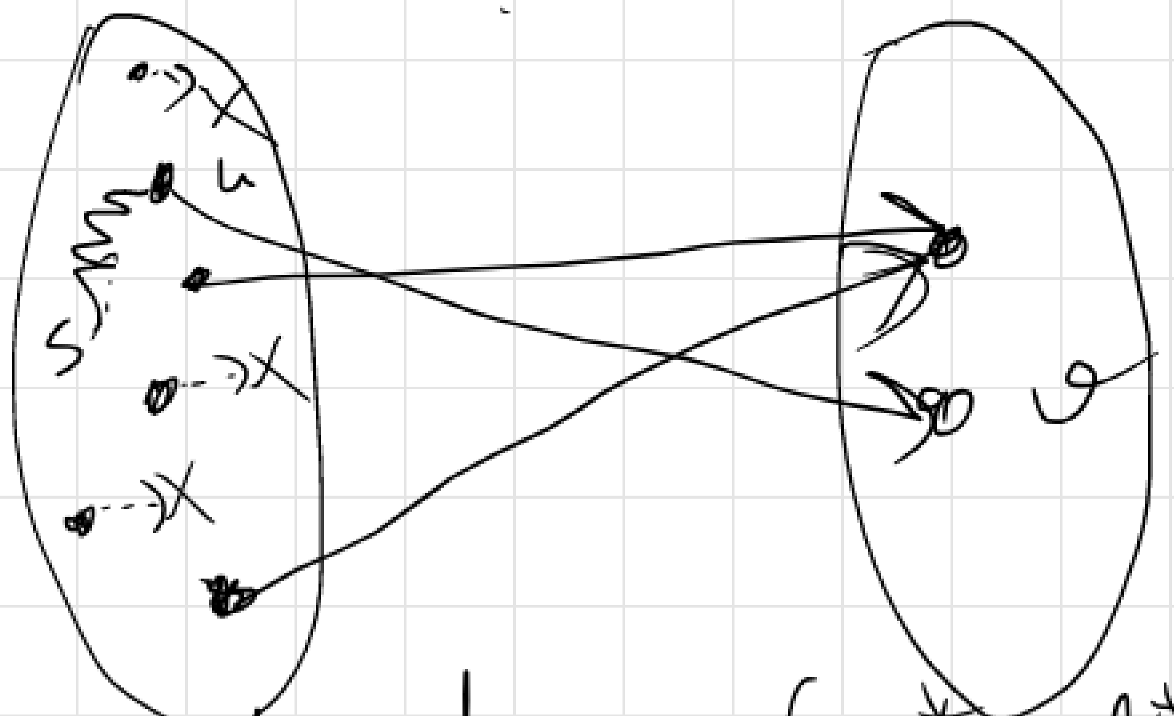




loop until $S = V$

▷ among all edges $(u, v) \in E$ s.t. $u \in S$ & $v \notin S$, pick the one that minimizes

$$\underbrace{d(u)}_{l(s \rightsquigarrow u)} + \underbrace{l(u, v)}_{\text{(edge length)}} \quad \text{[greedy]}$$



$d(u) + l(u,v)$

$[s \rightsquigarrow u \rightarrow v]$

Call selected edge (u^*, v^*) add v^* to S

set $d(v^*) = d(u^*) + \ell(u^*, v^*)$

Summary of the algorithm

S: set of explored vertices

$d(\cdot)$: distance from s

$\boxed{P(s) = \emptyset}$

Initializes $S = \{s\}$ $d(s) = 0$

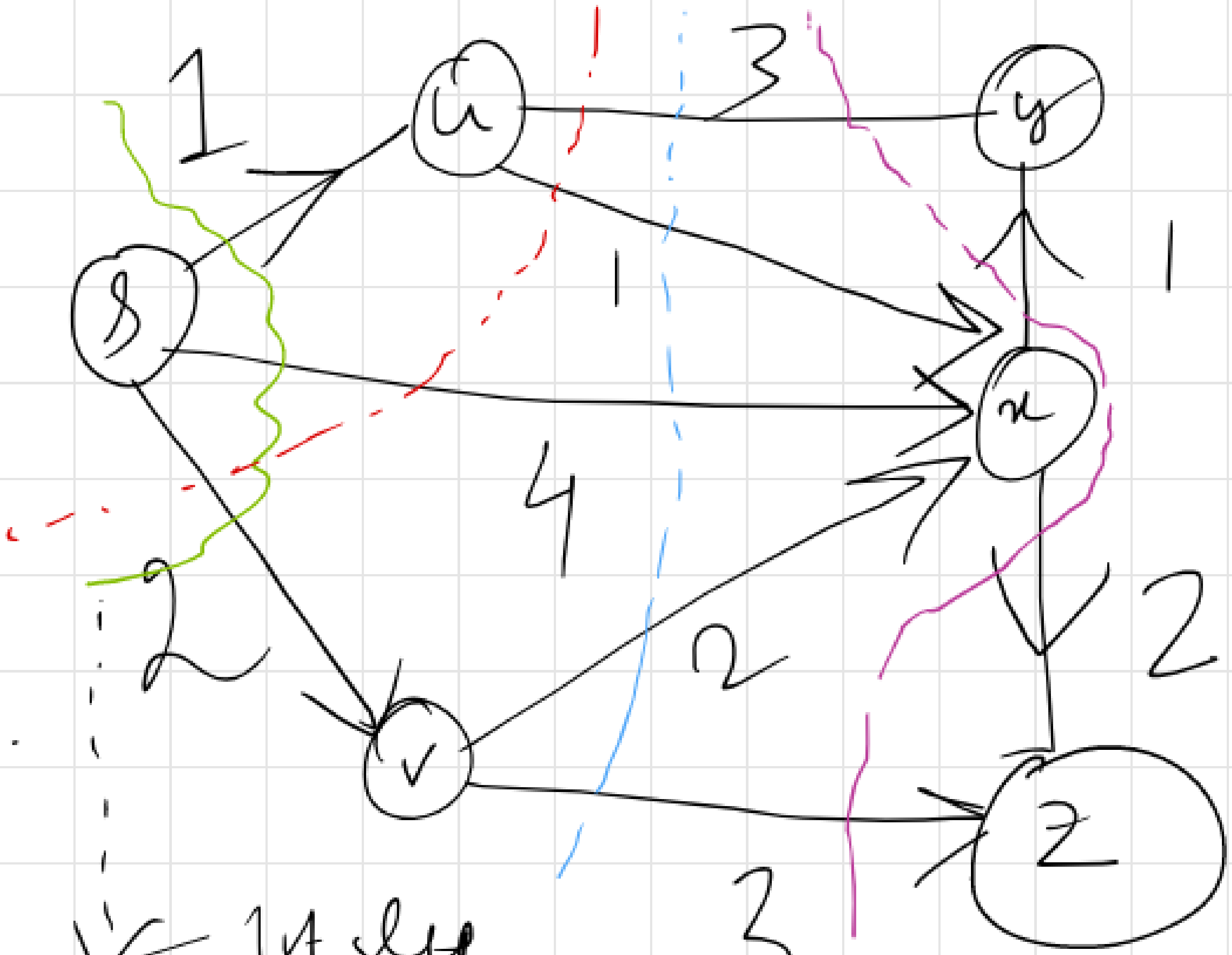
while $S \neq V$

let $(u^*, v^*) = \min\limits_{\substack{e=(u,v) \\ u \in S, \\ v \notin S}} \{d(u) + \ell(u,v)\}$

$S = S \cup \{v^*\}$

for path

$\boxed{P(v^*) = P(u^*) \cup \{(u^*, v^*)\}}$



$d(s) = 0 \; ; \; S = \{s\}$

1st step

$d(s) + l(s,u) = 0 + 1 = 1 \longrightarrow$ min

$d(s) + l(s,v) = 0 + 2 = 2$

$d(s) + l(s,x) = 0 + 4 = 4$

$S = \{s, u\}$

$d(u) = 1$

2nd step

$d(s) + l(s,v) = 2 \dashrightarrow$ min

$d(s) + l(s,x) = 4$

$d(u) + l(u,y) = 4$

$d(u) + l(u,x) = 2$

$S = \{s, u, v\}$

$d(v) = 2$

3rd step

$d(v) + l(v,z) = 2 + 3 = 5$

$d(v) + l(v,x) = 2 + 2 = 4$

$d(s) + l(s,x) = 0 + 4 = 4$

$d(u) + l(u,x) = 2 \dashrightarrow$ min

$d(u) + l(u,y) = 1 + 3 = 4$

$S = \{s, u, v, x\}$

$d(x) = 2$

4th step

$$\left.\begin{array}{l} d(u)+l(u,y) = 4 \\ d(v)+l(v,z) = 5 \\ d(x)+l(x,y) = 2+1 = 3 \\ d(x)+l(x,z) = 2+2 = 4 \end{array}\right\} \rightarrow \min$$

$S = \{s, u, v, x, y\}$

$d(y) = 3$

5th step

$$\left.\begin{array}{l} d(v)+l(v,z) = 2+3 = 5 \\ d(x)+l(x,z) = 2+2 = 4 \end{array}\right\} \rightarrow \min$$

$S = \{s, u, v, x, y, z\}$

$d(z) = 4$

(Correctness of Dijkstra Algorithm)

Claim: When Dijkstra algo adds a node v^* to S, we get the shortest path length from s to v^*. So for every node $u \in S$, $d(u)$ is the shortest path length from s to u & $P(u)$ is a shortest path from s to u.

[Length given by algo $\leq$ length of any other path from s to v^*]

PBI: Base Case $|S| = 1$ $S = \{s\}$
$d(s) = 0$

Hyp: The claim holds true for $|S| = K$ $[K \geq 1]$

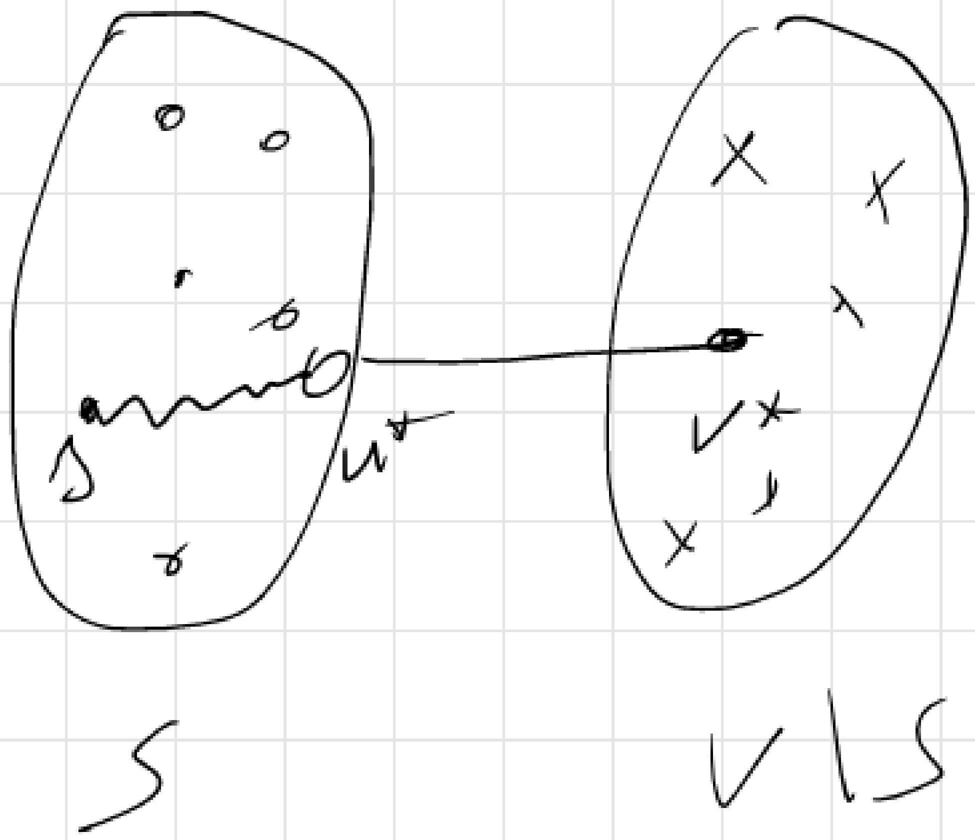
Ind Step: To show that the claim holds when we grow $|S|$ to $K+1$ by adding v^*.

let (u^*, v^*) be the last edge on $P(v^*)$ [v^* determined by our algo)

$$[s \rightsquigarrow u^* \longrightarrow v^*] \equiv P(v^*)$$

$\ell(P(u^*)) = d(u^*)$

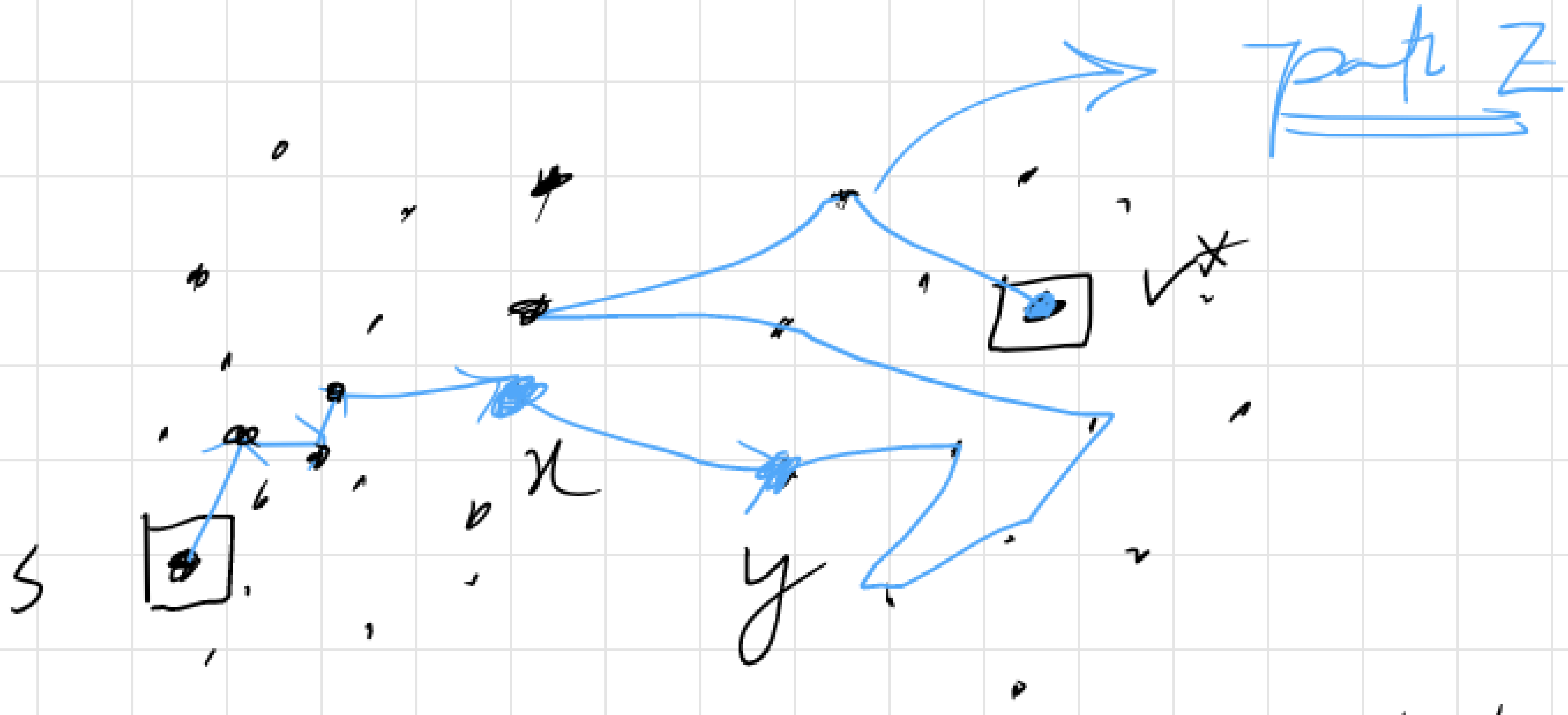
edges in $P(u^*) \in S$



Consider any other $s \rightsquigarrow v^*$ path Z

<u>To show</u> $\ell(Z) \geq \ell(P(v^*))$

<u>Idea:</u> [look at the nodes (edges) of path Z.] (in the realm of our partition $S, V \setminus S$)



<u>Observation</u> All nodes in path Z will not be the part of S [$\because v^* \notin S$]

$$\begin{aligned} Z &= s \rightsquigarrow x \quad (\in S) \\ & \quad x \longrightarrow y \quad \to \text{Edge} \\ & \quad y \rightsquigarrow v^* \quad (\text{arbitrary}) \end{aligned}$$

$$\begin{aligned} P(v^*) &= s \rightsquigarrow u^* \quad [P(u^*) \in S] \\ & \quad u^* \longrightarrow v^* \quad [\text{Edge}] \end{aligned}$$

The algo chose (u^*, v^*), at that time point (x, y) was also a candidate.

$$\Rightarrow d(x) + l(x, y) \geq d(u^*) + l(u^*, v^*) \quad (1)$$

$$l(Z) = l(s \rightsquigarrow x) + l(x \longrightarrow y) + l(y \rightsquigarrow v^*)$$

$$\ell(Z) = \ell(S \rightsquigarrow x) + \ell(x,y) + \ell(y \rightsquigarrow v^*)$$

$$= L_1 + \ell(x,y) + L_2$$

$d(x) \longrightarrow$ shortest length path by induction hyp.

$\therefore L_1 \geq d(x)$

Why $L_2 \geq 0$? $\Rightarrow$ [no -ve lengths of e]

$$\ell(Z) \geq d(x) + \ell(x,y) + L_2$$

$$\geq d(x) + \ell(x,y)$$

$$\geq d(u^*) + \ell(u^*, v^*) \quad (\text{by } 1)$$

$$= \ell(P(v^*))$$

[length of arbitrary path is atleast the length of path given by our algo]

[Implementation]

S : set of explored nodes

$d(\cdot)$: distance from s

while $S \neq V$ $\xrightarrow{\hspace{3em}}$ $n-1$

$$\text{let } (u^*, v^*) = \min_{\substack{u \in S \\ v \notin S}} d(u) + \ell(u, v)$$

$$S = S \cup \{v^*\}$$

$$d(v^*) = d(u^*) + \ell(u^*, v^*)$$

<u>$n-1$ iterations</u>

You might do linear scan for all edges $\Rightarrow O(m)$

for each edge $\Rightarrow$ constant

$O(m)$

In each iteration

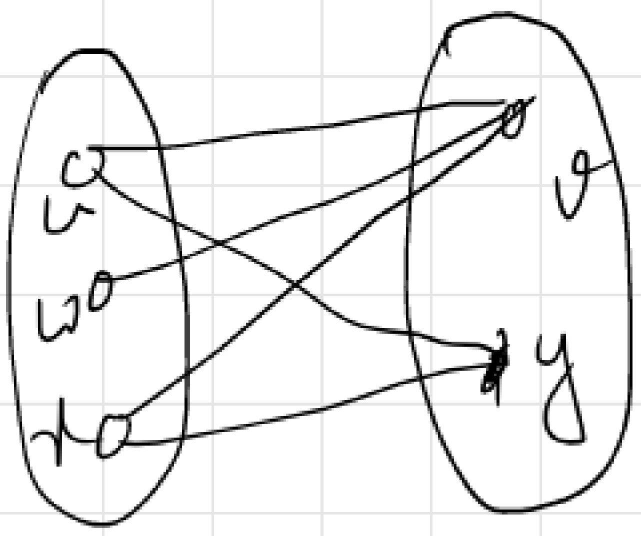
$\boxed{O(mn)}$

(Using better Data Structure)

[Repeated minimum computations]

Heap: $\left[\begin{array}{l} \text{insert} \\ \text{Extract-min} \\ \text{Decrease key} \end{array}\right] O(\log n)$

What we will store in a heap?



$$d'(v) = \min \begin{Bmatrix} d(u) + l(u,v) \\ d(w) + l(w,v) \\ d(x) + l(x,v) \end{Bmatrix}$$

$$d'(y) = \min \begin{Bmatrix} d(u) + l(u,y) \\ d(x) + l(x,y) \end{Bmatrix}$$

["insert $d'(\cdot)$ of node $\in V \setminus S$" in a heap]

[$d'(v) = \infty$ for node $v \in V \setminus S$ s.t. there doesn't exist any edge from S to v]

(1) elements in heap = nodes in $V \setminus S$

(2) for $v \in V \setminus S$,

$d'(v) = key[v]$ = smallest value of

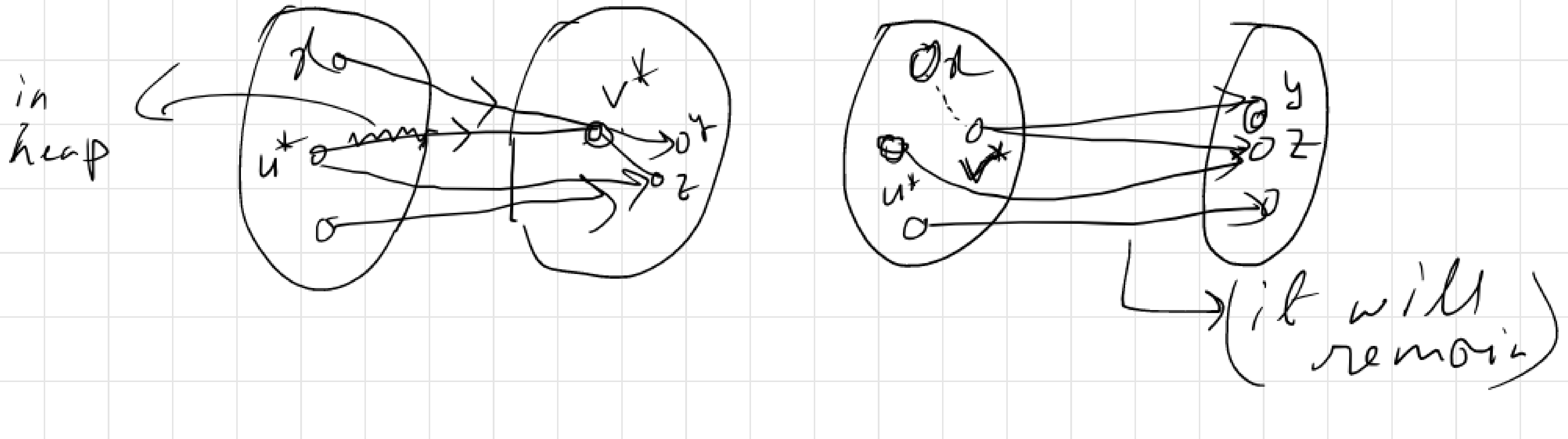
local min $\left\{ \begin{array}{l} \text{greedy parameter of} \\ \text{an edge } (u, v) \text{ s.t.} \end{array} \right.$

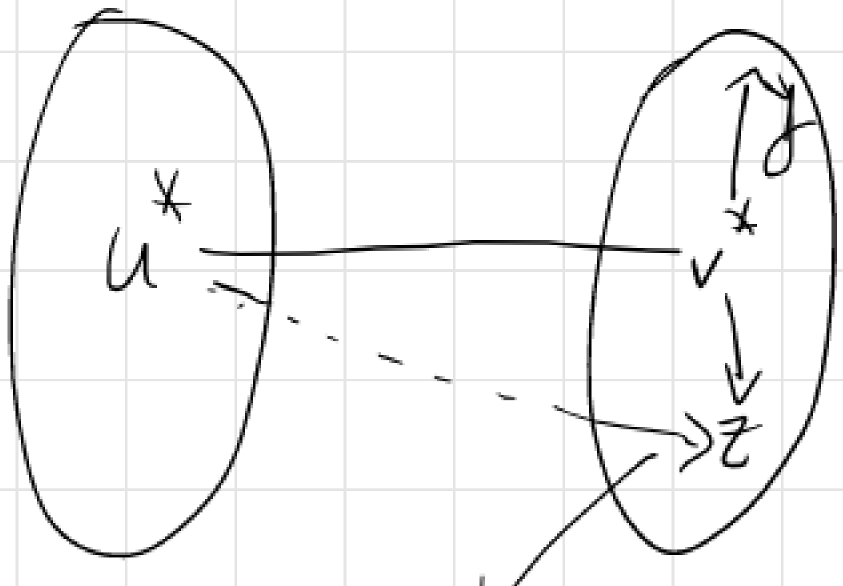
$u \in S$ and $v \in V \setminus S$

$= \infty$, otherwise

(3) Extract min : will choose global min from the local min

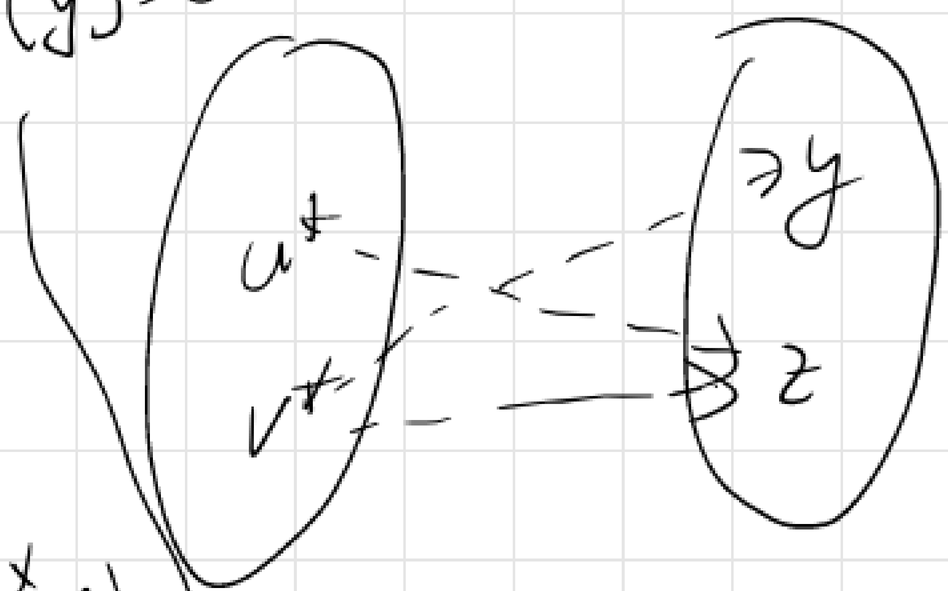
(4) Maintenance when we identify v^*, delete v^* from the heap and add it to S;





$key(y) = 0$

$key(z) = d(u^*) + l(u^*, v)$



$key(y) = d(v^*) + l(v^*, y)$

$key(z)$ might decrease due to v^*

for each edge $(v^*, w^*) \in E$

if $w^* \in V \setminus S$ (i.e. $w^* \in$ Heap)

$newkey[w^*] = \min\{key[w^*], d[v^*] + l(v^*, w^*)\}$

$decreasekey(w^*, newkey[w^*])$

Runtime $\boxed{n-1} \rightarrow$ Extract Min

$\boxed{\text{Maintenance}} \rightarrow O(\log n)$ for decrease key

$\#(v^*, w^*)$ can trigger ≤ 1 update operations

$$\#\text{ heap operation} = O(\underbrace{m}_{\text{update key}} + \underbrace{n}_{\text{Extract Min}})$$

$(m \geq n)$

$$\text{Total time} = O(m \underline{\underline{\log n}})$$

1) $d(s) \leftarrow 0 \longrightarrow O(1)$

2) for all $v \in V/s$ $\quad d(v) \leftarrow \infty \longrightarrow O(n)$

3) $S \leftarrow \emptyset \longrightarrow O(1)$

4) $Q \leftarrow V$ // Q is a heap maintaing $V \setminus S \longrightarrow O(n)$

5) while $Q \neq$ NULL

do $u \leftarrow$ ExtractMin$(Q) \longrightarrow O(\log n)$

$S \leftarrow S \cup \{u\} \longrightarrow O(1)$

6) for all $v \in adj(u) \longrightarrow O(deg(u))$

Relaxation
do if $d(v) > d(u) + l(u,v)$
then $d(v) \leftarrow d(u) + l(u,v)$
$\longrightarrow (\log n)$

Time $= O(n) \cdot T_{\text{Extract Min}} + \sum deg(u) \cdot T_{\text{Decrease Key}}$

$= O(n \log n) + O(m \log n)$

$= m \log n$

(Graph with negative weights)

[What will happen if there exists a negative cycle ??]

"Explaination on board"

Bellmon Ford Algorithm

$\hookrightarrow$ Can detect negative Cycle;

Dijkstra Algo Cannot detect it

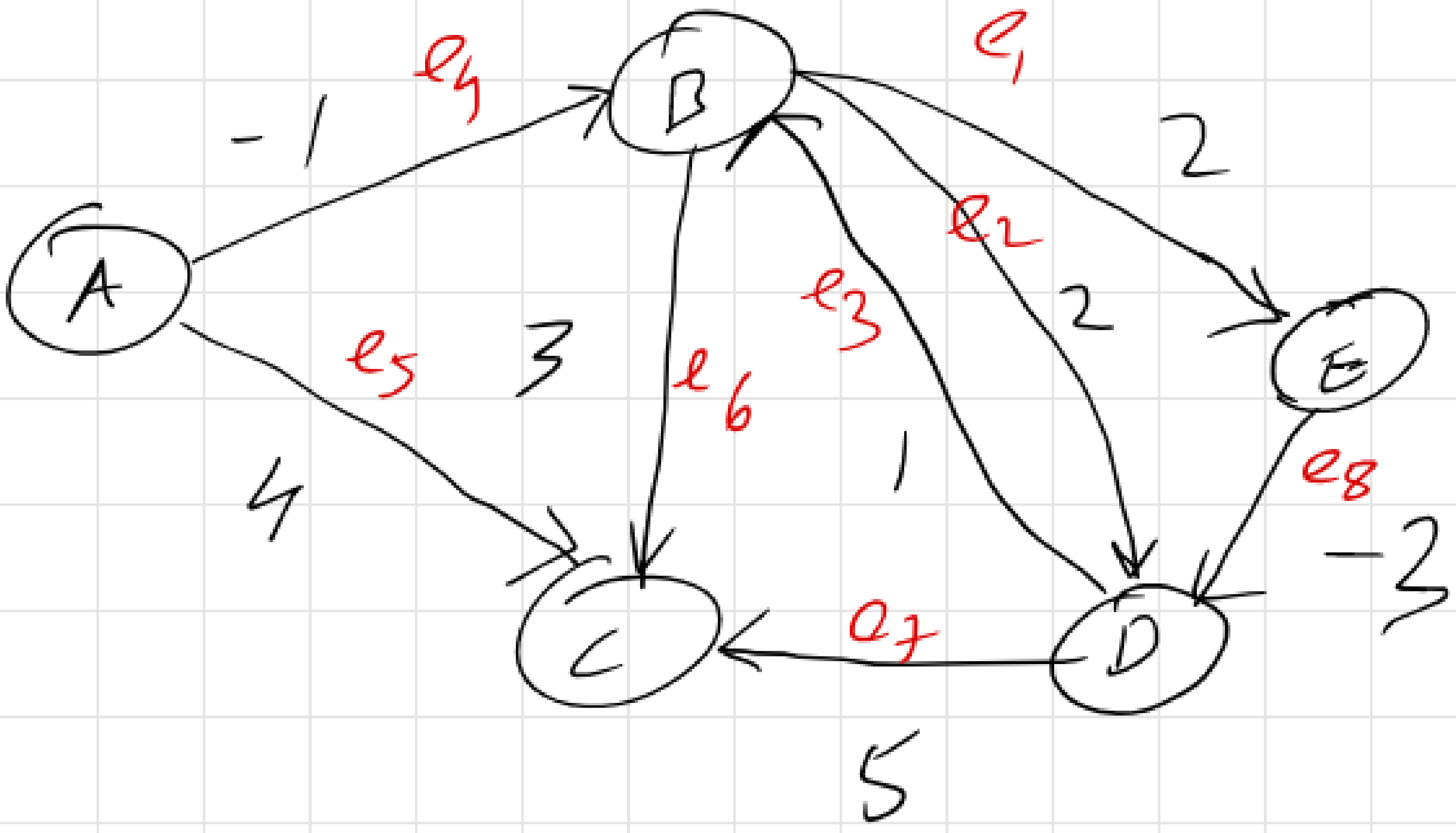
$$\left[\begin{array}{ll} \underline{I/P} & (G = (V, E) \quad l : E \to \mathbb{R},\ s \in V) \\ \underline{OP} & \delta(s, v) \quad \forall v \in V \end{array}\right]$$

---

1. $d(s) \leftarrow 0$
2. $d(v) \leftarrow \infty$
3. for $i \leftarrow 1$ to $|V| - 1$ $[O(n)]$
4. for <u>each edge</u> $(u, v) \in E$ $[O(m)]$
5. if $d(v) > d(u) + w(u, v)$
6. [Relax] then $d(v) \leftarrow d(u) + w(u, v)$
7. for each edge $(u, v) \in E$
   do if $d(v) > d(u) + w(u, v)$
   then Report neg cycle

Example

(iterate edges in any order)



A	B	C	D	E	
0	∞	∞	∞	∞	
during one iteration { 0	-1	~~4~~ 2	∞	∞	← i = 1
0	-1	2	1	~~1~~ -2 (e8)	← i = 2
0	-1	2	1	-2	i = 3
0	-1	2	1	-2	i = 4

(If there is a neg cycle, then it will not converge for at least one node)