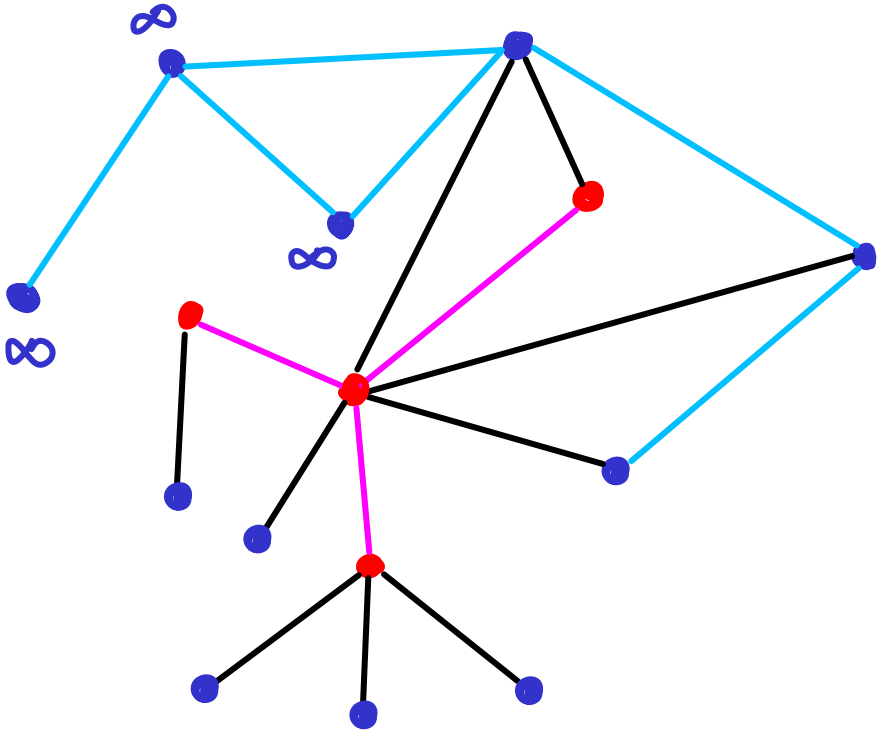
modified

PRIM'S ALGORITHM ~~for MST~~



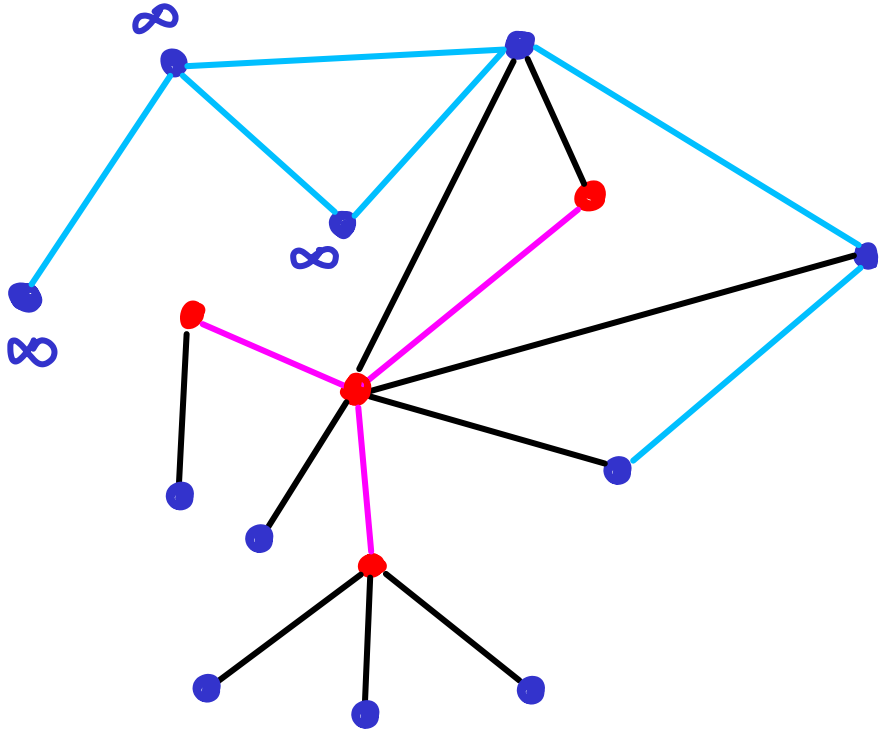
(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T
| mark x → in T.
| for each ~~unmarked~~ neighbor v of x
| ~~if w(v) > w(v,x) then decrease.~~
| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP

(1956 ~ 1959)



(after initializing)

while pr.queue not empty

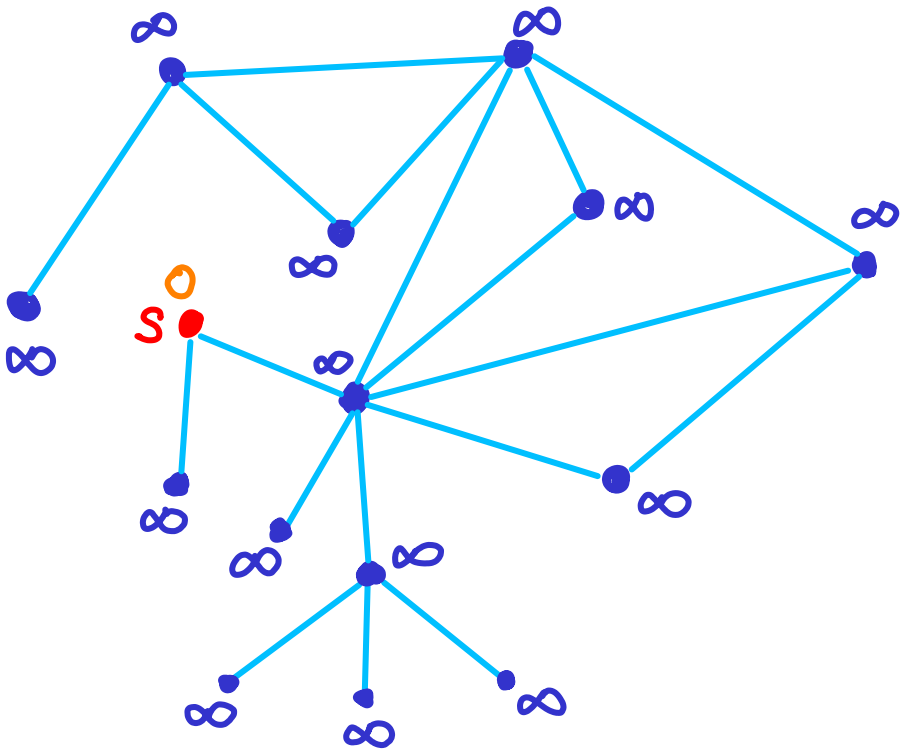
| x: extract-min & add edge to T

| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP



Initialize

(after initializing)

while pr.queue not empty

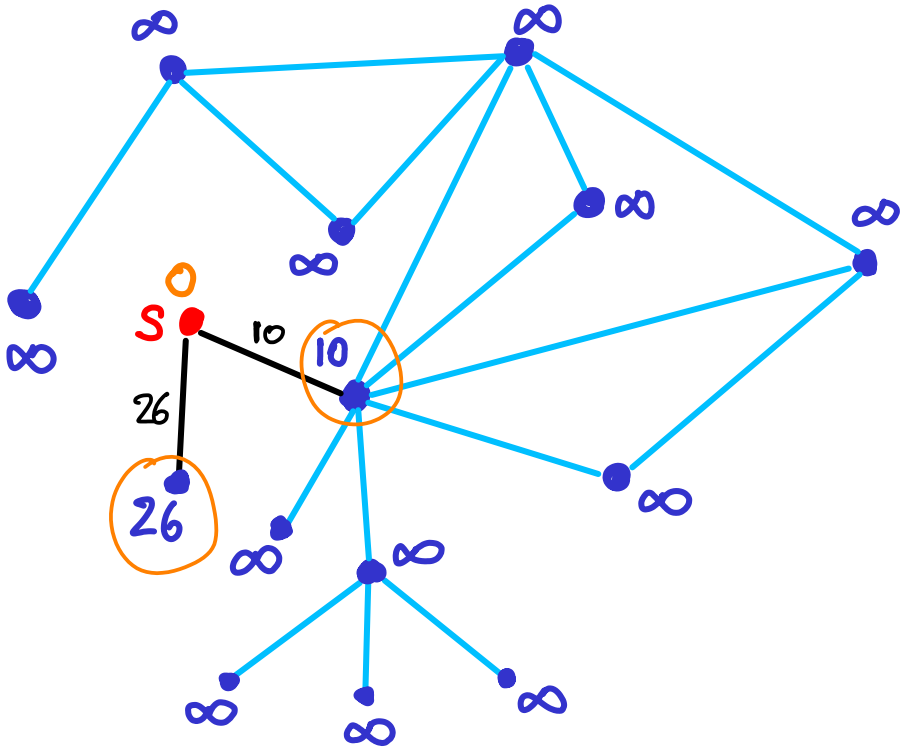
| x: extract-min & add edge to T

| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP



(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T

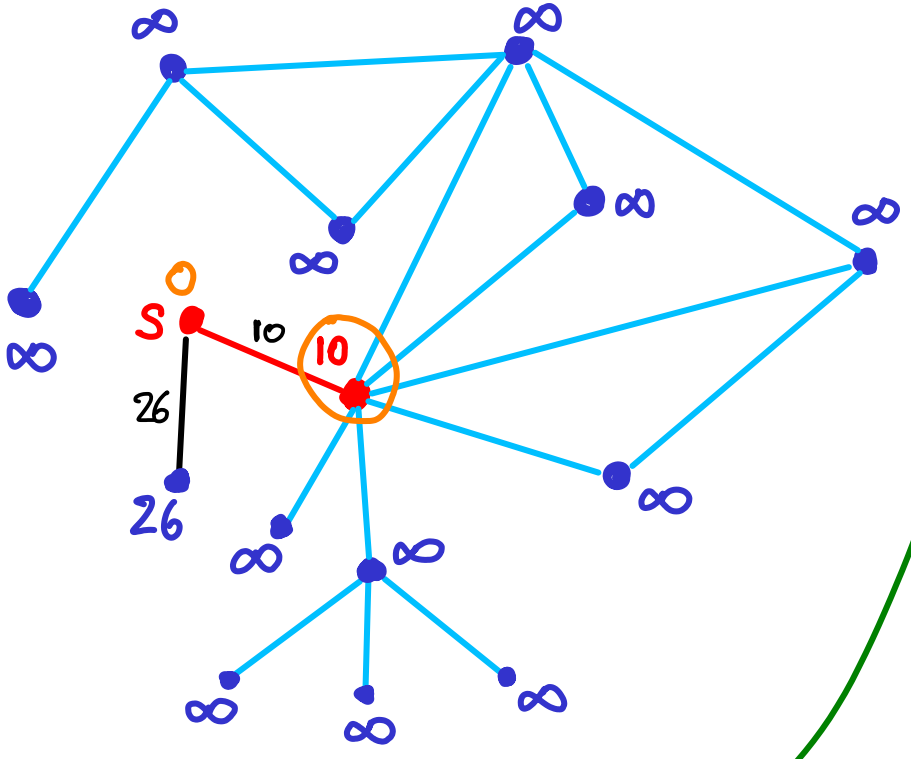
| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

extract source & relax two edges

DIJKSTRA'S ALGORITHM for SSSP



extract min : 10

(after initializing)

while pr.queue not empty

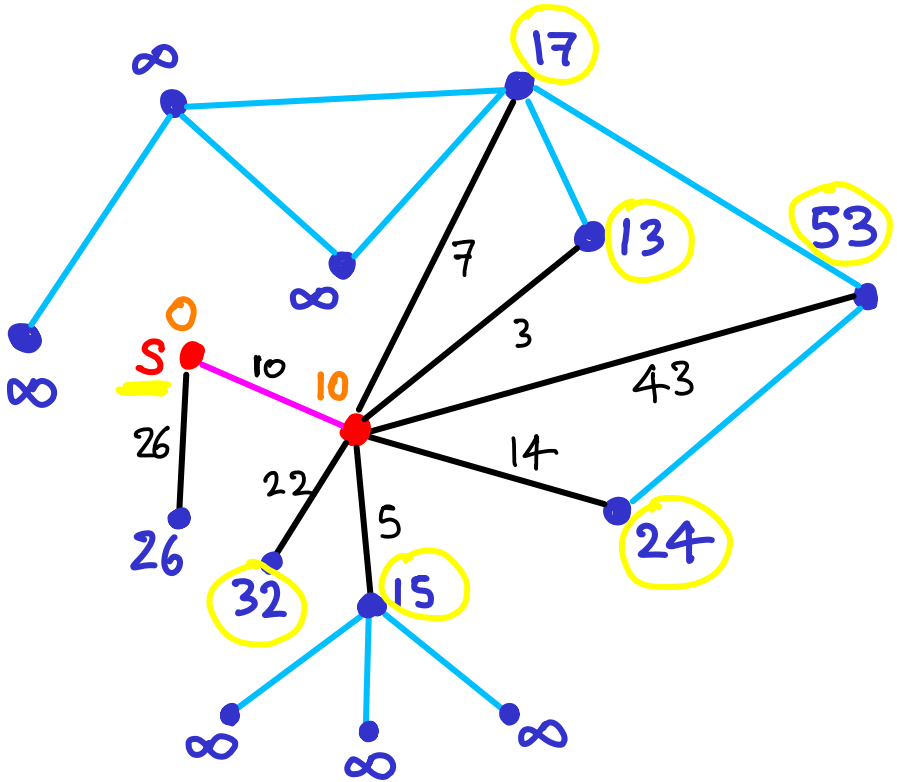
| x: extract-min & add edge to T

| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP



(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T

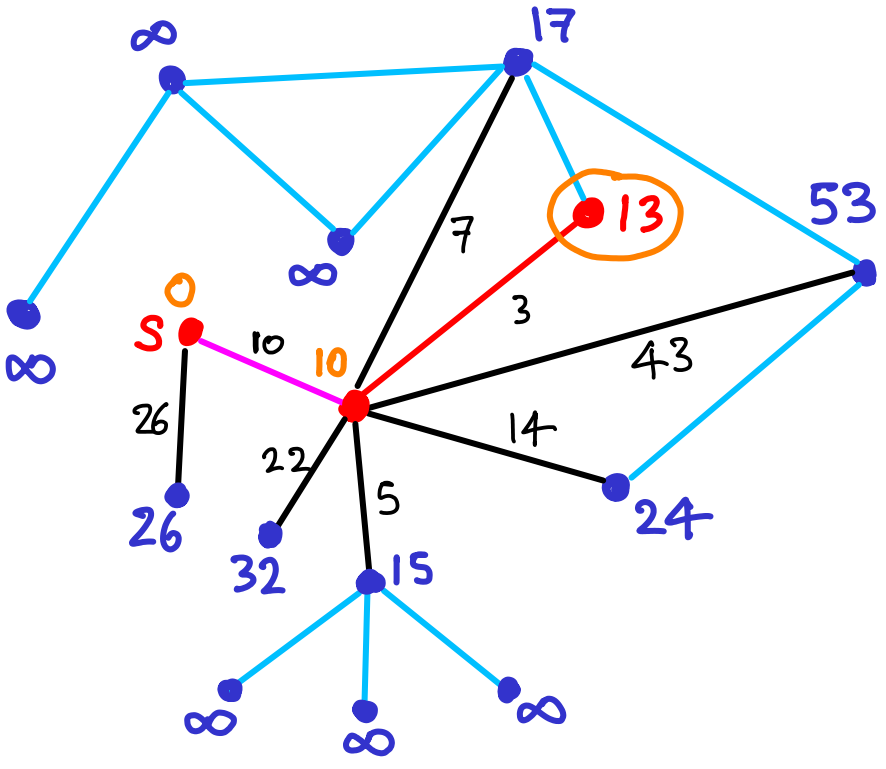
| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

Relax neighbors of 10

DIJKSTRA'S ALGORITHM for SSSP

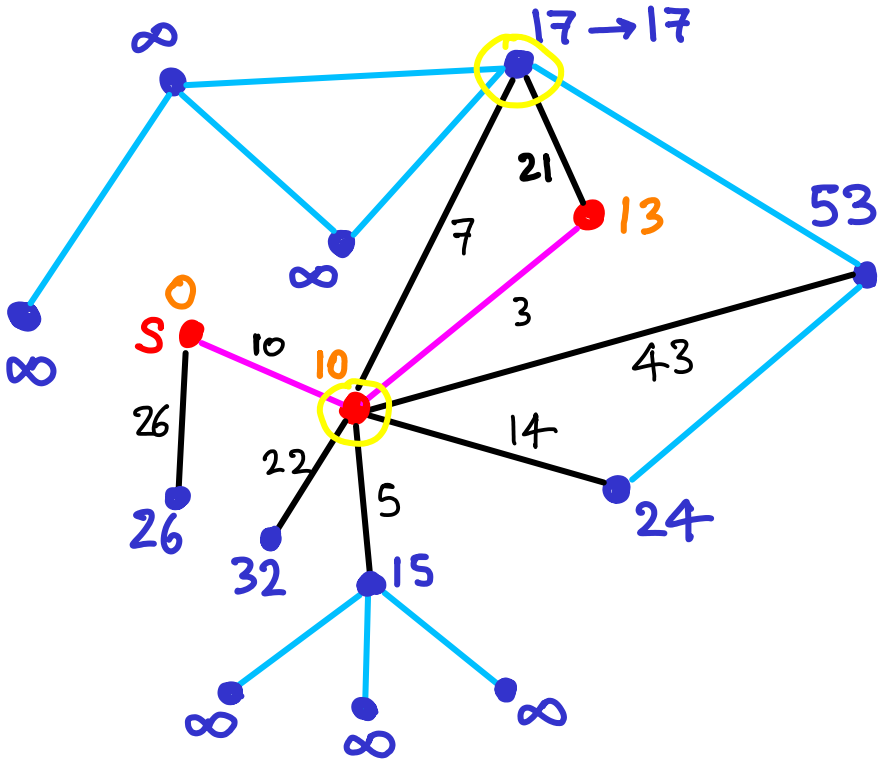


(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T
| mark x → in T.
| for each neighbor v of x
| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP



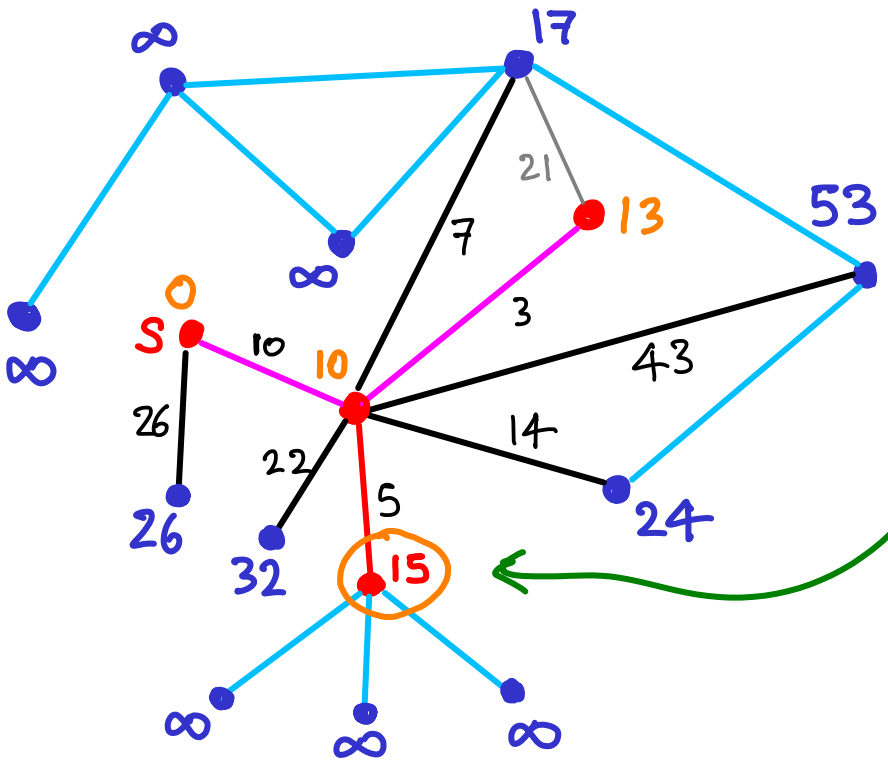
(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T
| mark x → in T.
| for each neighbor v of x
| RELAX (x,v)

No update from relaxing

DIJKSTRA'S ALGORITHM for SSSP

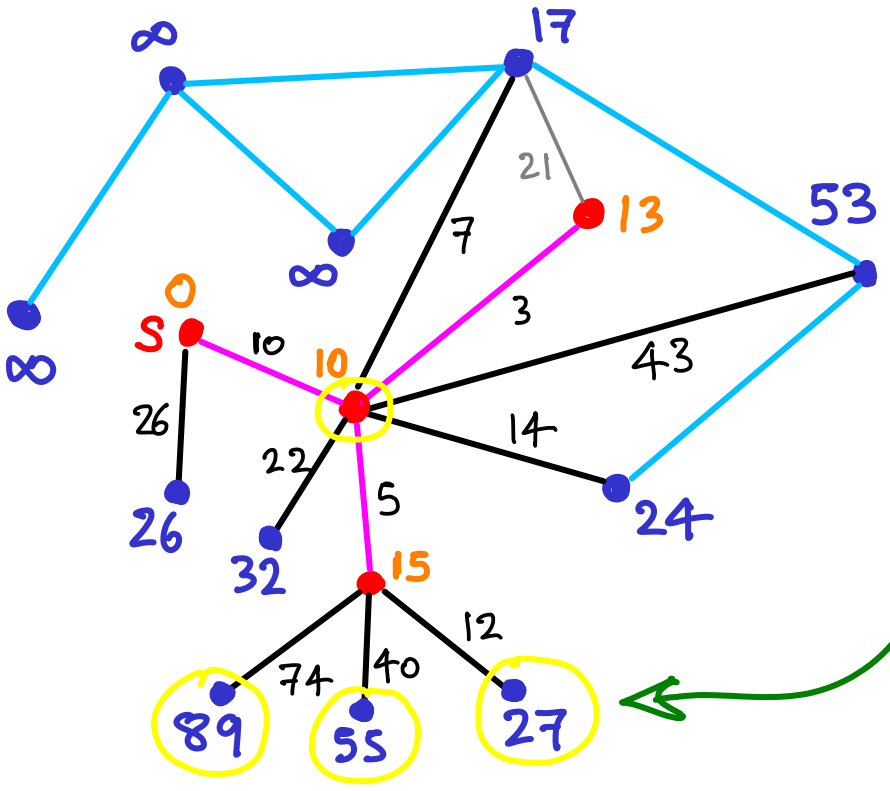


(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T
| mark x → in T.
| for each neighbor v of x
| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP

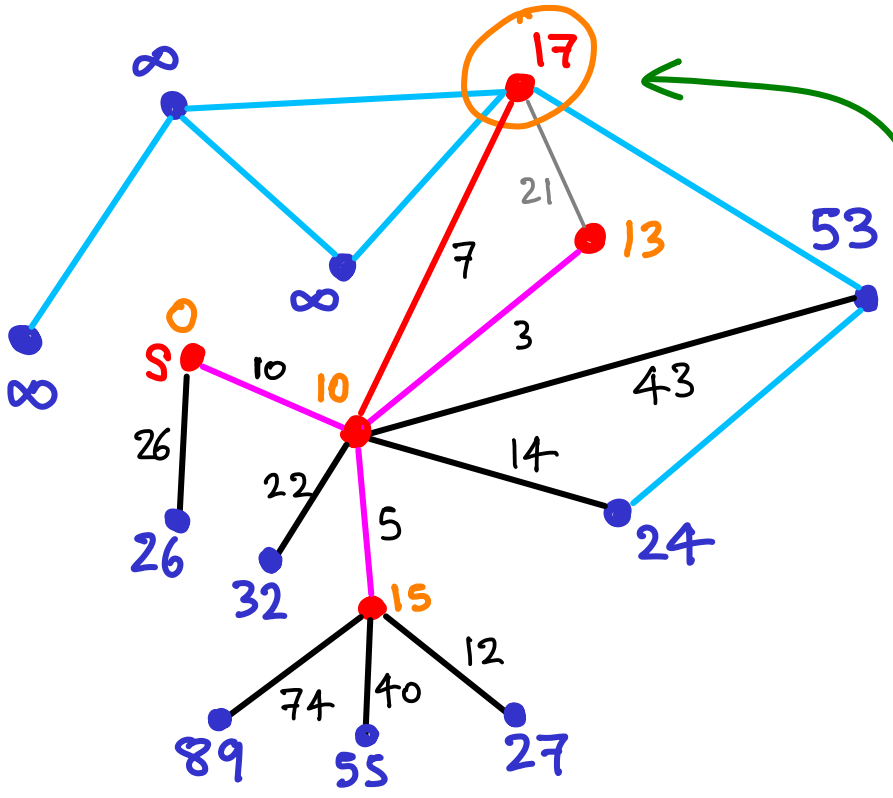


(after initializing)

while pr.queue not empty

| x: extract-min & add edge to T
| mark x → in T.
| for each neighbor v of x
| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP



(after initializing)

while pr.queue not empty

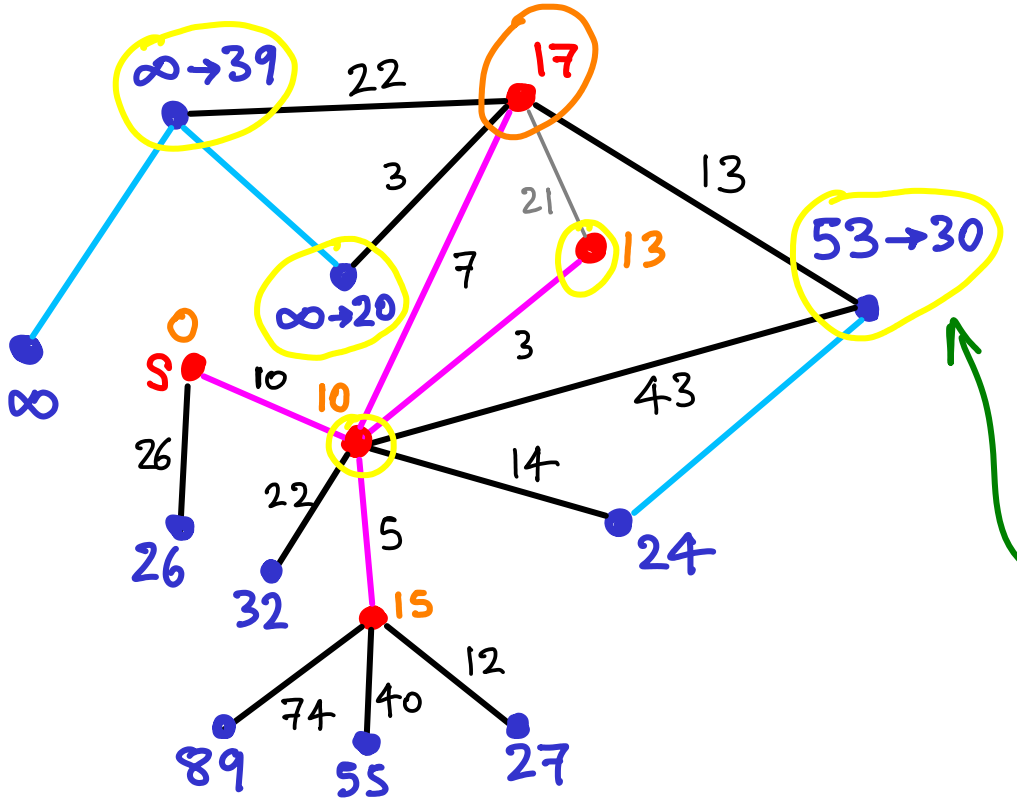
| x: extract-min & add edge to T

| mark x → in T.

| for each neighbor v of x

| RELAX(x,v)

DIJKSTRA'S ALGORITHM for SSSP



(after initializing)

while pr.queue not empty

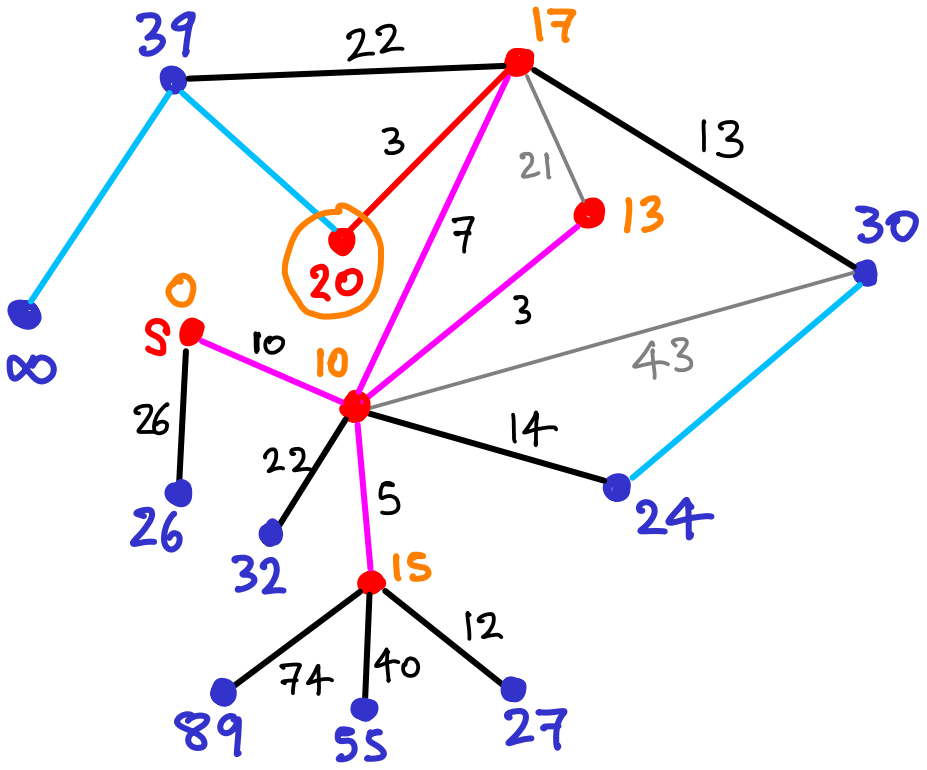
| x: extract-min & add edge to T

| mark x → in T.

| for each neighbor v of x

| RELAX (x,v)

DIJKSTRA'S ALGORITHM for SSSP

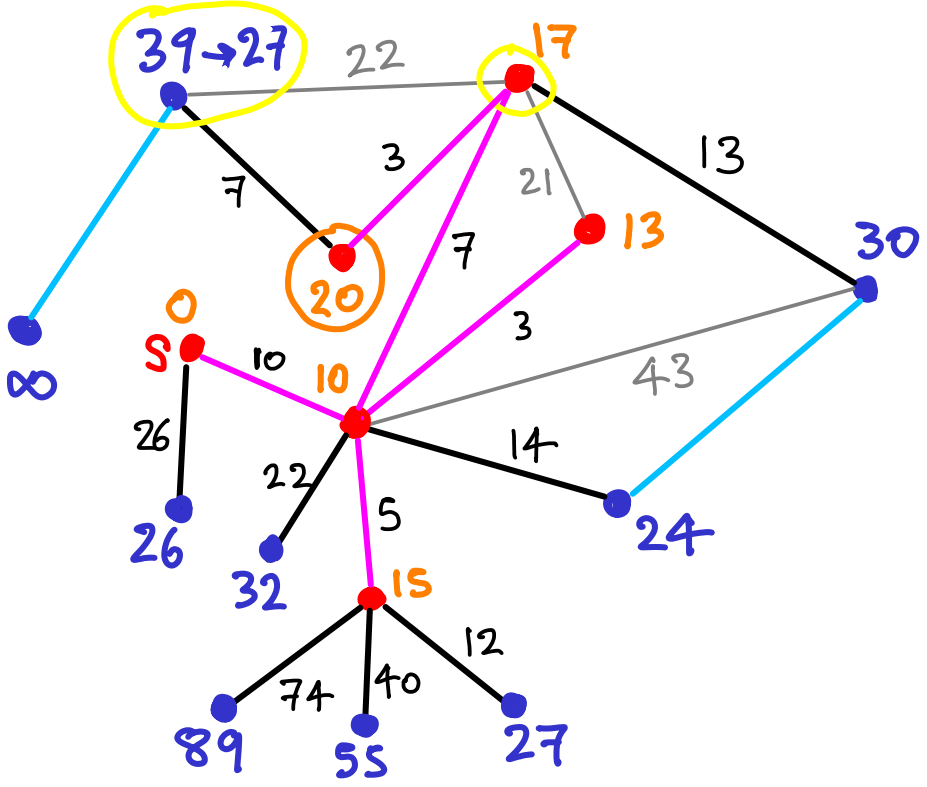


(after initializing)

while pr.queue not empty

- x: extract-min & add edge to T
- mark x → in T.
- for each neighbor v of x
  - RELAX(x,v)

DIJKSTRA'S ALGORITHM for SSSP

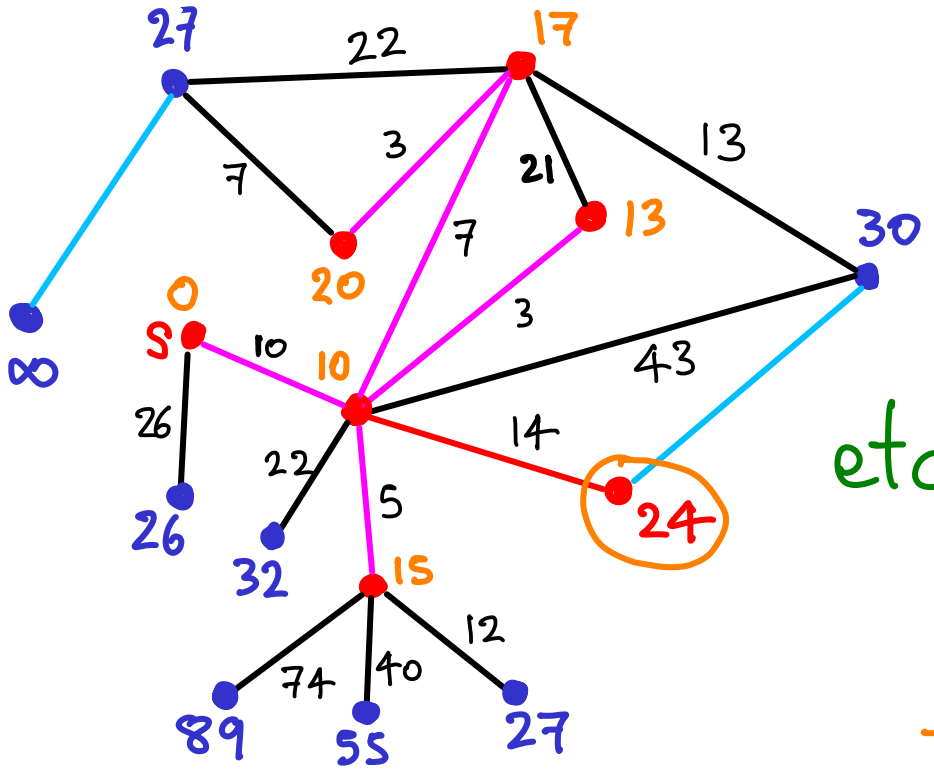


(after initializing)

while pr.queue not empty

- x: extract-min & add edge to T
- mark x → in T.
- for each neighbor v of x
  RELAX(x,v)

DIJKSTRA'S ALGORITHM for SSSP



etc

(after initializing)

while pr.queue not empty

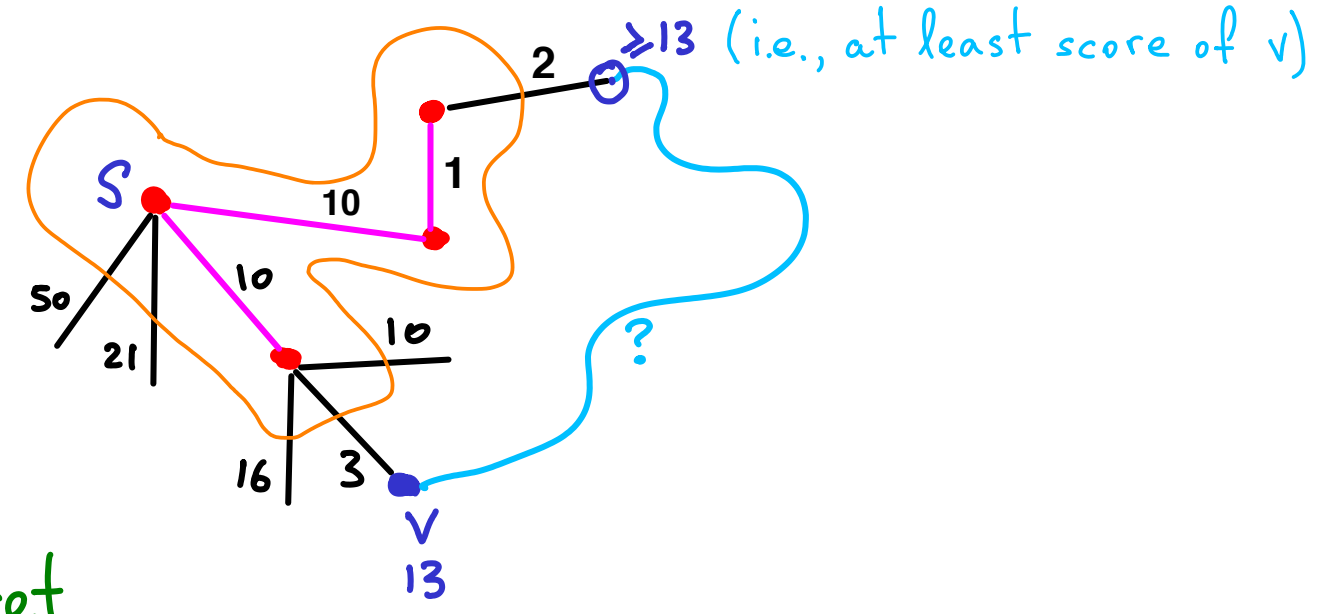
- x: extract-min & add edge to T
- mark x → in T.
- for each neighbor v of x
  - RELAX(x,v)

time : $O(V^2)$ or $O(E \log V)$

(like Prim's algo // for fancier see CLRS

Correctness :

assume
we have shortest path
to a set of red vertices

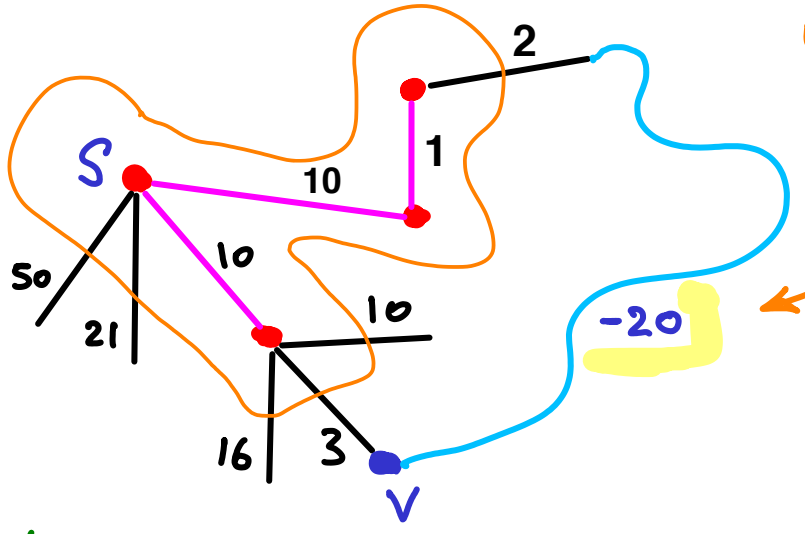


Somewhere outside this set
is a vertex v with shortest path =
= [a path in known set] + black edge

Any other path s ~~~ v
will cost more

Correctness :

assume
we have shortest path
to a set of red vertices



w/ weights ≤0
greedy choice
now invalid

Somewhere outside this set
is a vertex v with shortest path =
= [a path in known set] + black edge

Any other path s ~~~ ~ ~ v will cost more

} assuming weights ≥0