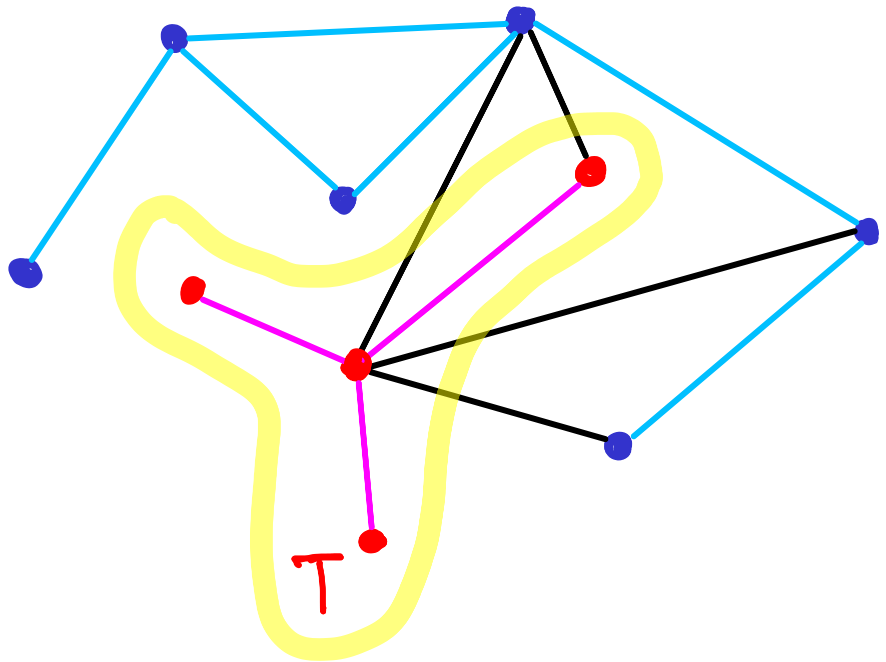
PRIM'S ALGORITHM for MST

R. PRIM - 1957

(V. JARNIK - 1930)

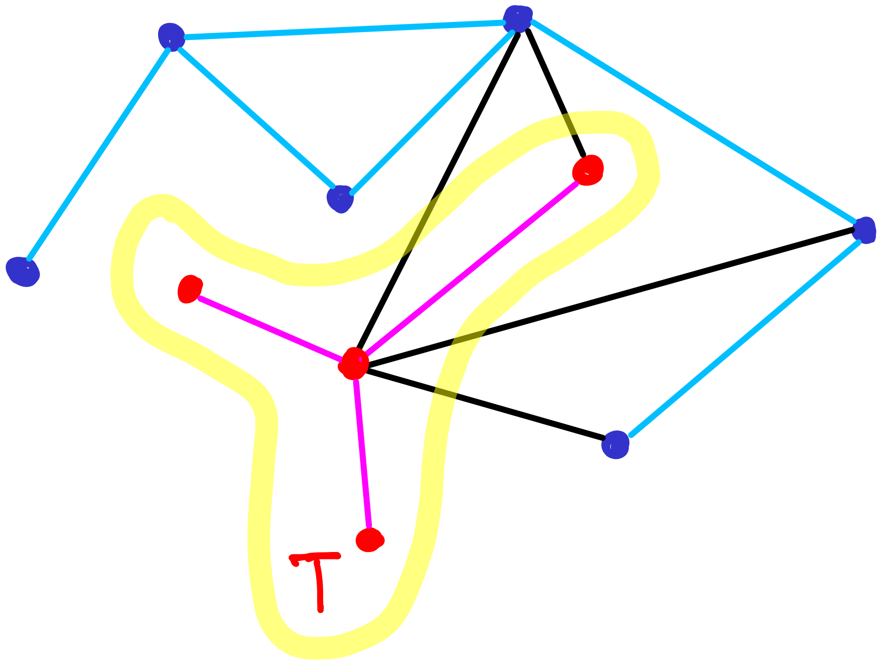
PRIM'S ALGORITHM for MST



Uses basic principle:

Given a subtree T of MST, the "lightest" edge connecting to a vertex not in T can be added to T.

PRIM'S ALGORITHM for MST

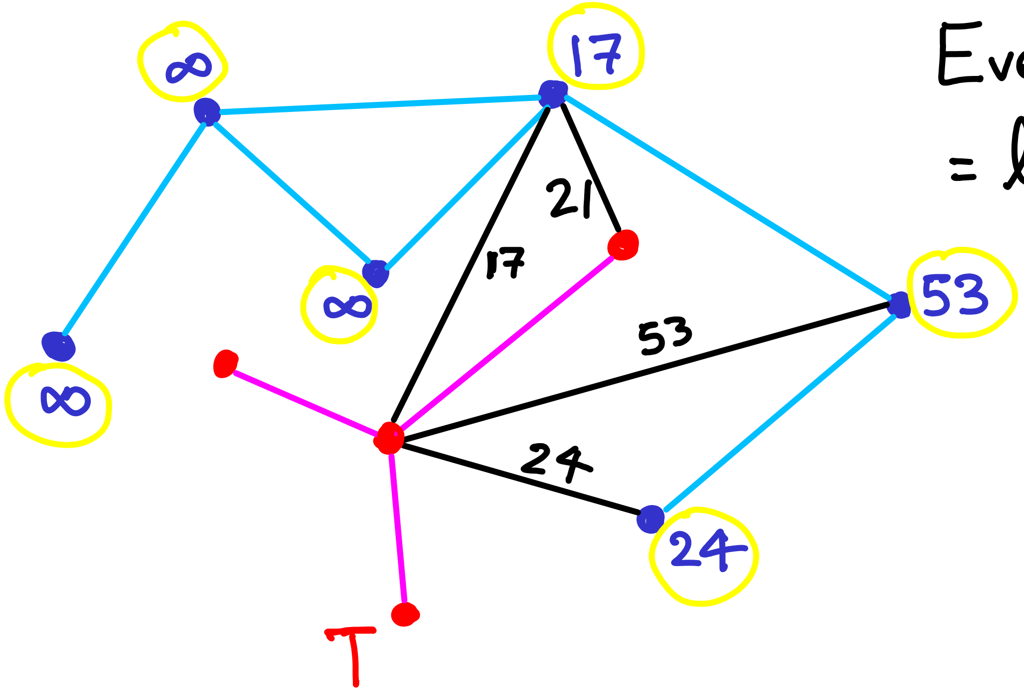


Uses basic principle:

Given a subtree T of MST, the "lightest" edge connecting to a vertex not in T can be added to T.

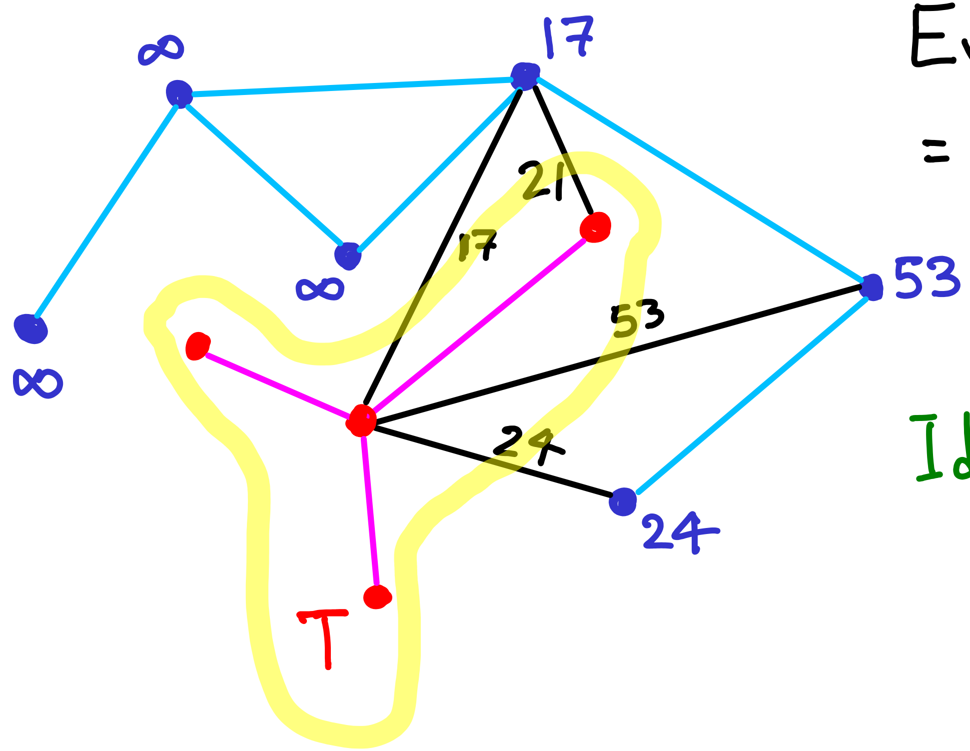
Grow one tree, incrementally adding one edge (& vertex)

PRIM'S ALGORITHM for MST



Every vertex not in T has a score =
= lightest edge weight connecting it to T

PRIM'S ALGORITHM for MST

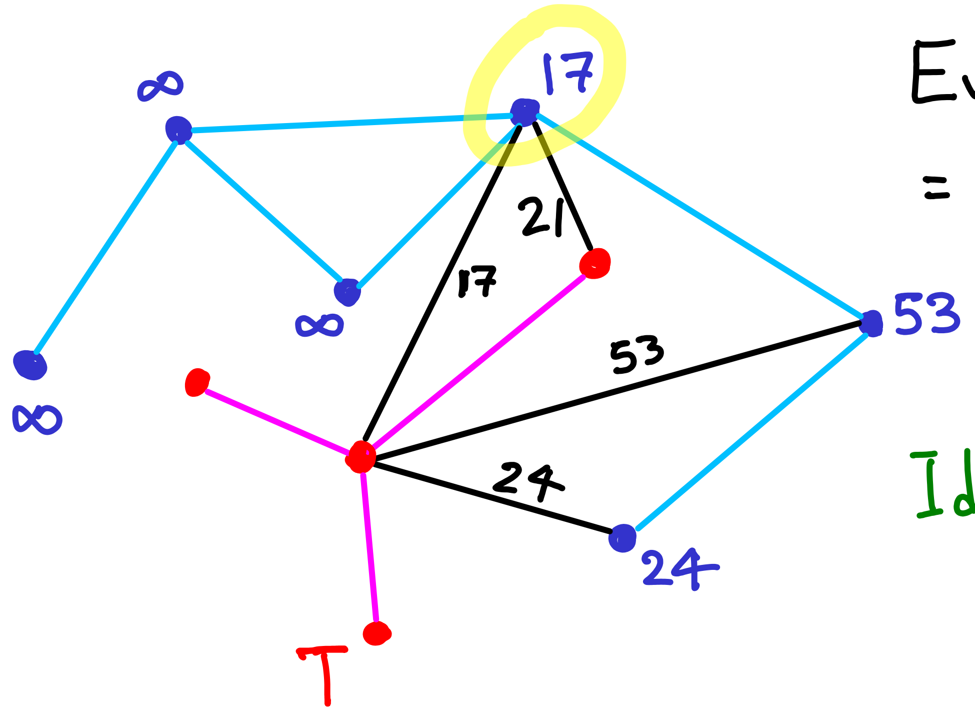


Every vertex not in T has a score =
= lightest edge weight connecting it to T

Identify lightest edge crossing cut:

How?

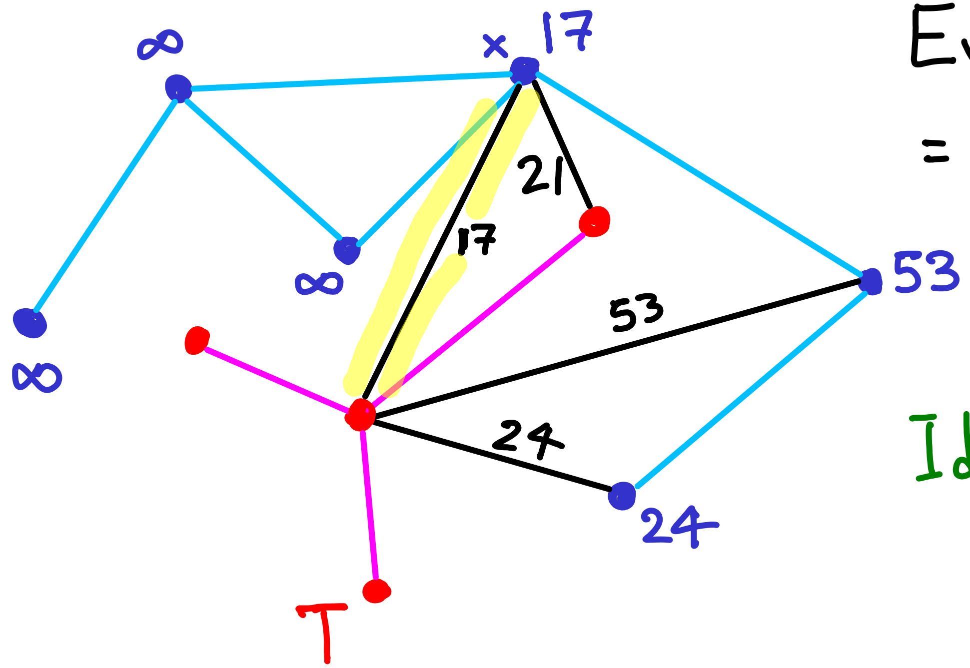
PRIM'S ALGORITHM for MST



Every vertex not in T has a score =
= lightest edge weight connecting it to T

Identify lightest edge crossing cut:
1) identify min-score vertex, x

PRIM'S ALGORITHM for MST

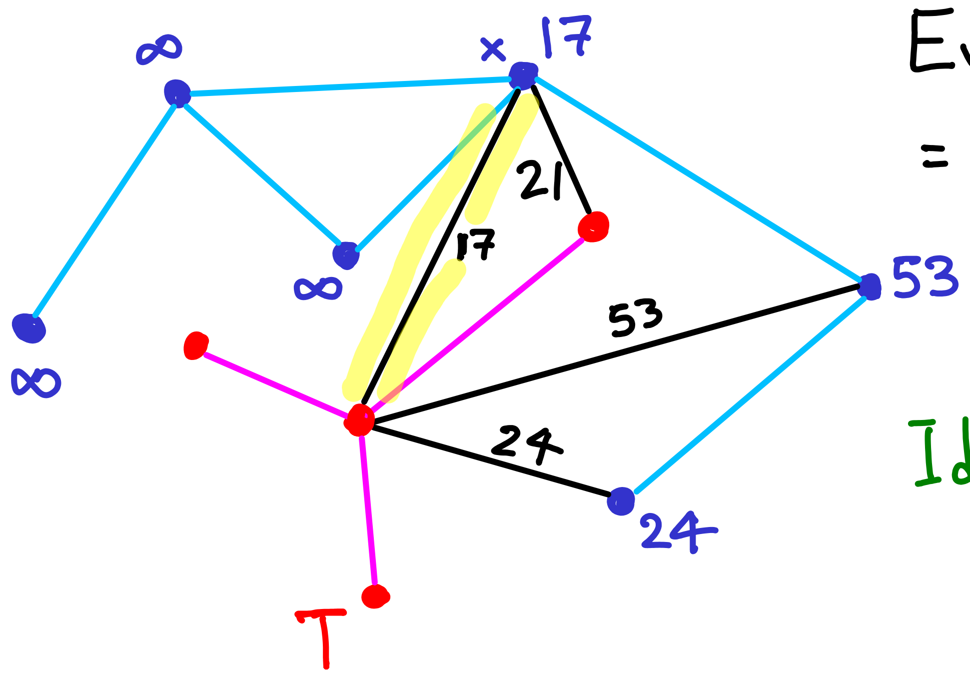


Every vertex not in T has a score =
= lightest edge weight connecting it to T

Identify lightest edge crossing cut:

1) identify min-score vertex, x
2) identify lightest edge from x to T

PRIM'S ALGORITHM for MST



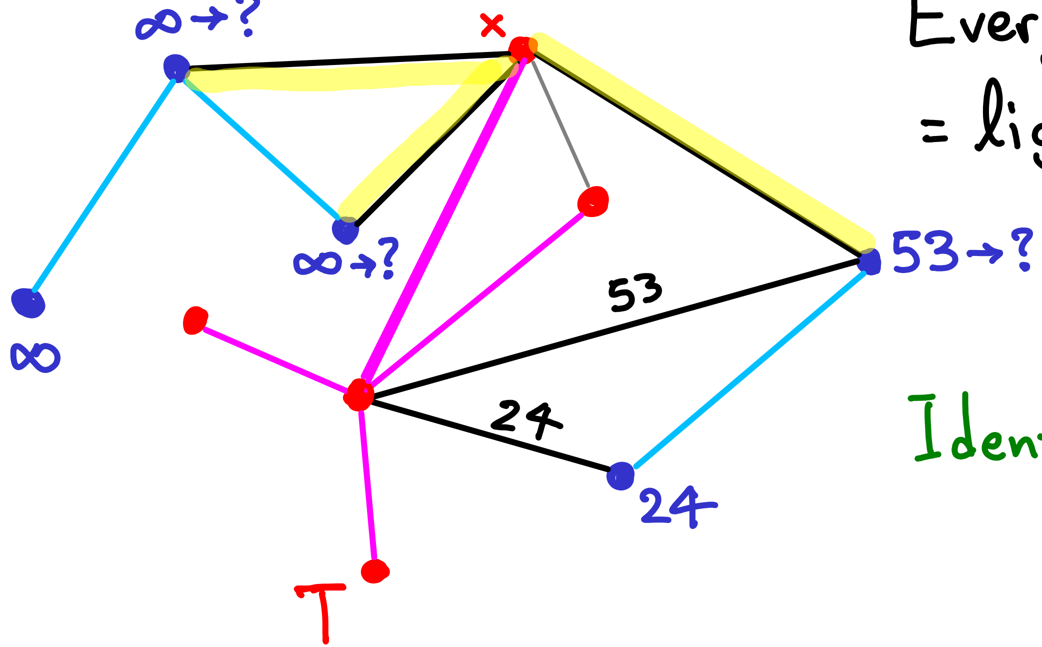
Every vertex not in T has a score =
= lightest edge weight connecting it to T

Identify lightest edge crossing cut:

1) identify min-score vertex, x
2) identify lightest edge from x to T

Brute force: O(V) per MST edge

PRIM'S ALGORITHM for MST



Every vertex not in T has a score =
= lightest edge weight connecting it to T

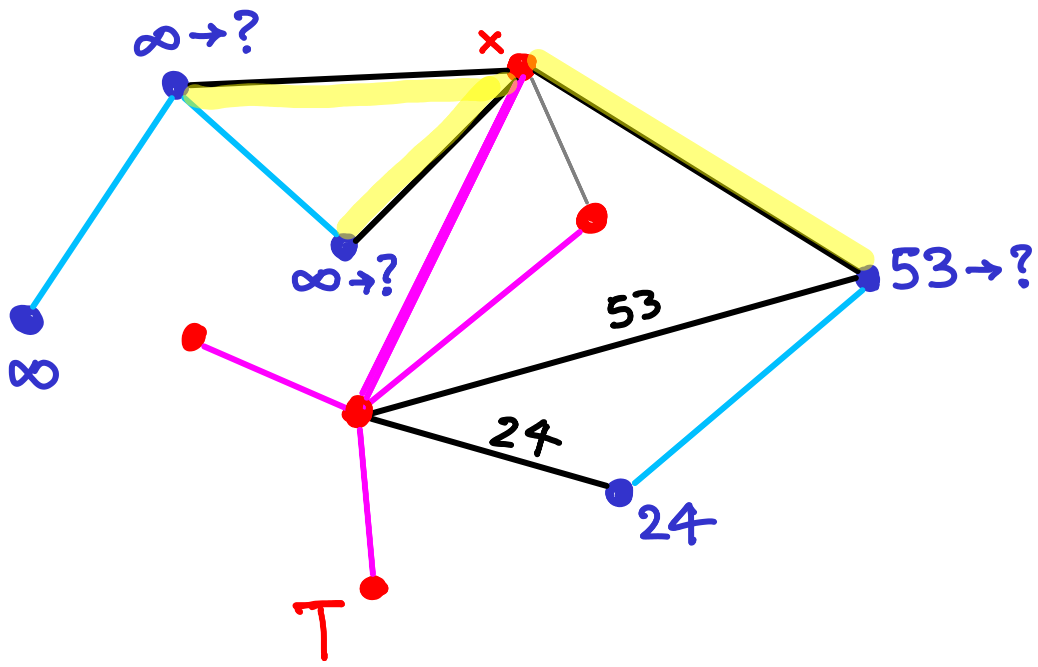
Identify lightest edge crossing cut:

1) identify min-score vertex, x
2) identify lightest edge from x to T

Brute force: O(V) per MST edge

... but we must still update scores after adding x to T

PRIM'S ALGORITHM for MST



Update scores when x joins T:

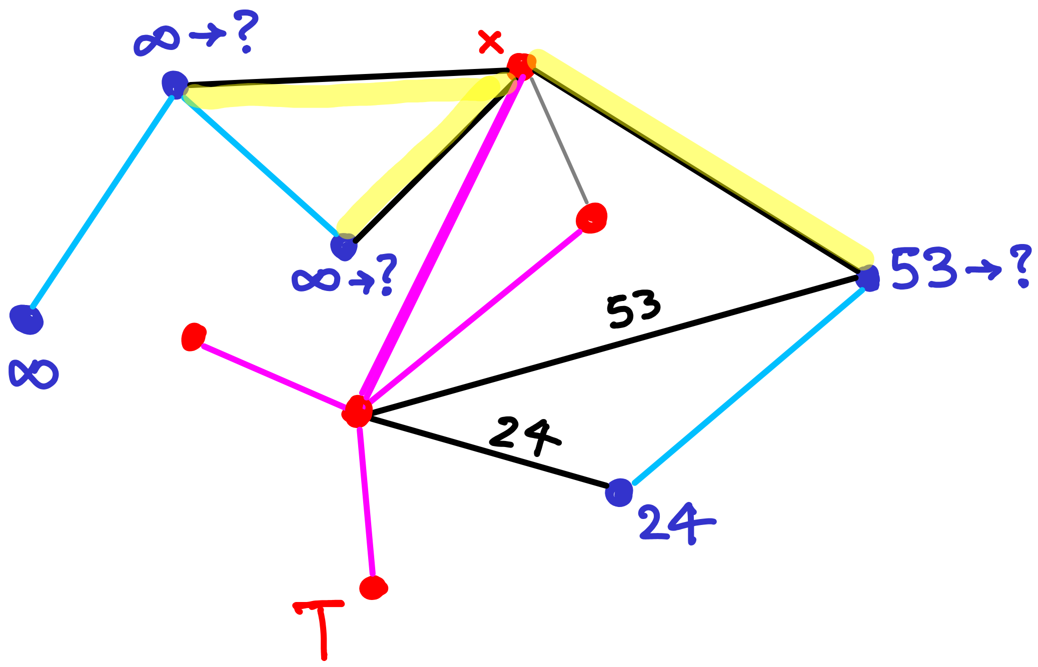
For each neighbor v_i of x

  if v_i not in T

    $c = \text{score}(v_i)$

    $\text{score}(v_i) \leftarrow \min\{c, \underbrace{w(x, v_i)}_{\text{new option}}\}$

PRIM'S ALGORITHM for MST



Update scores when x joins T:

For each neighbor v_i of x

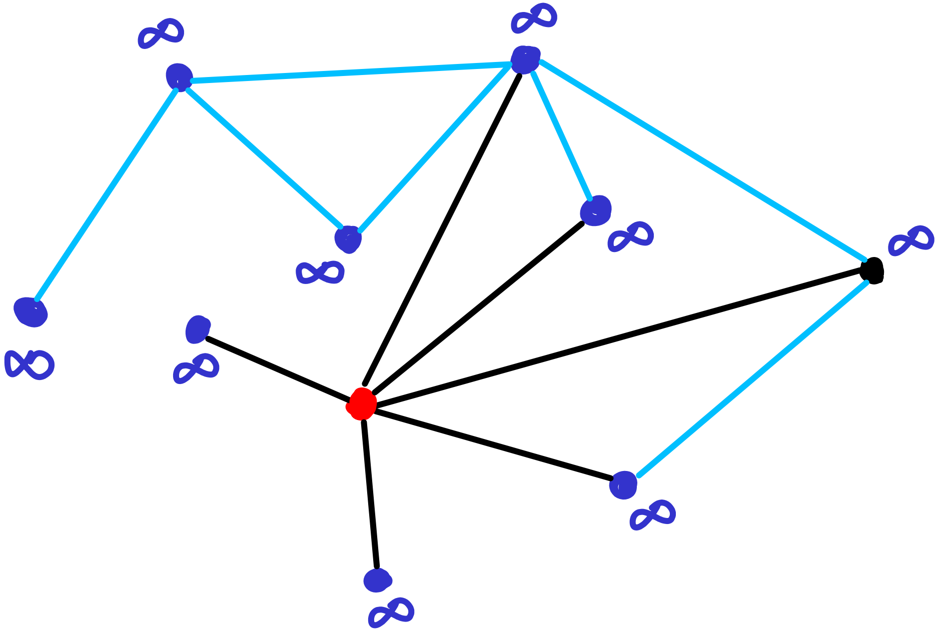
if v_i not in T

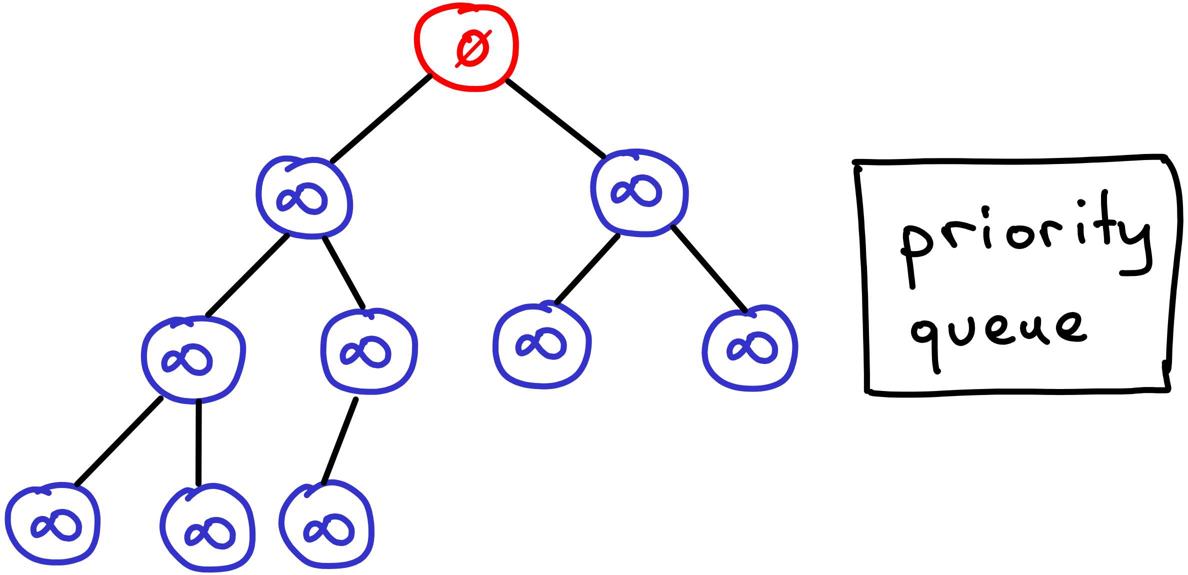
$c = \text{score}(v_i)$

$\text{score}(v_i) \leftarrow \min\{c, \underbrace{w(x, v_i)}_{\text{new option}}\}$

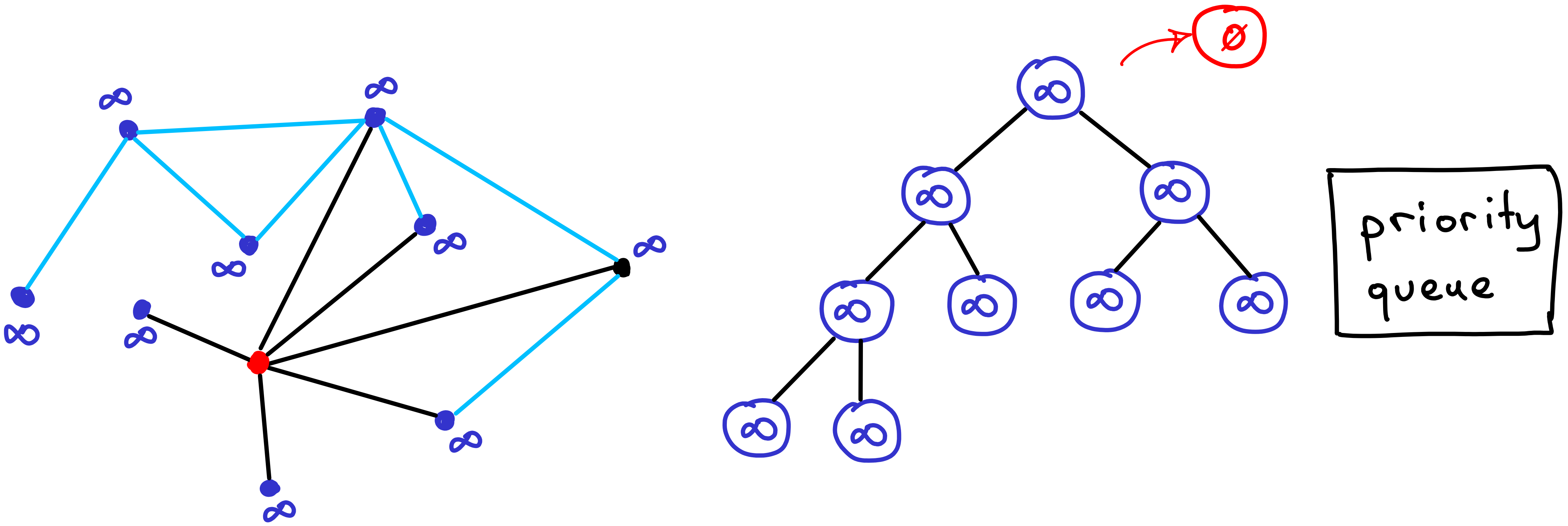
Need to extract min score & decrease scores. How?

PRIM'S ALGORITHM for MST

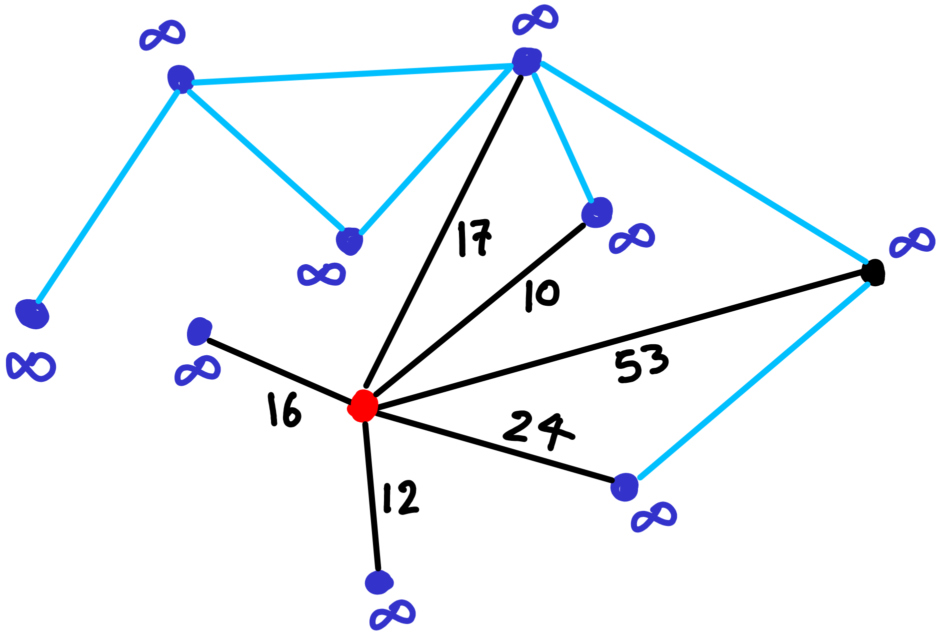


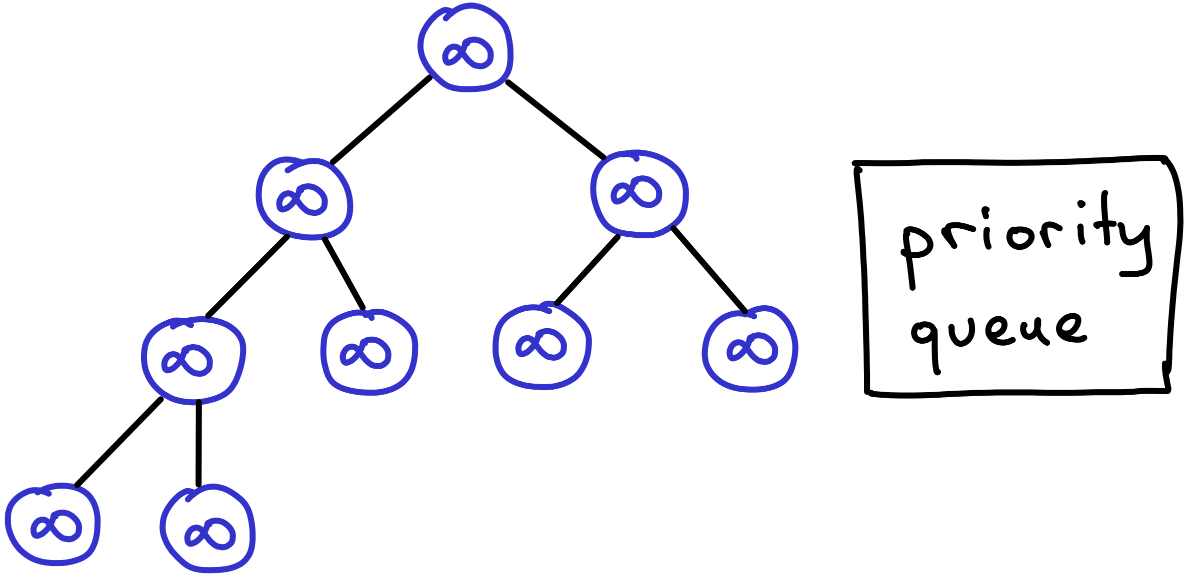


PRIM'S ALGORITHM for MST

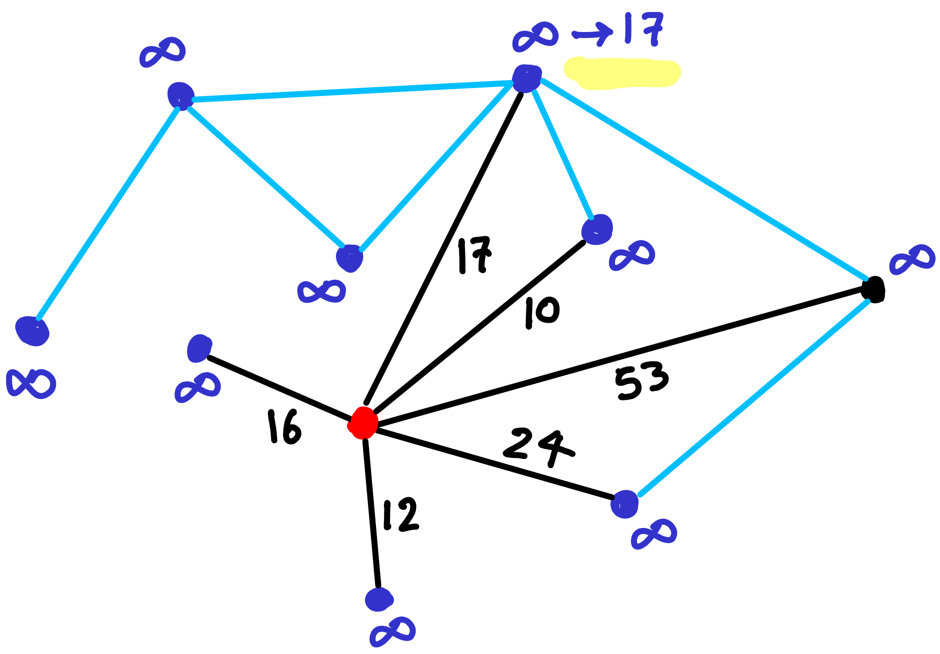


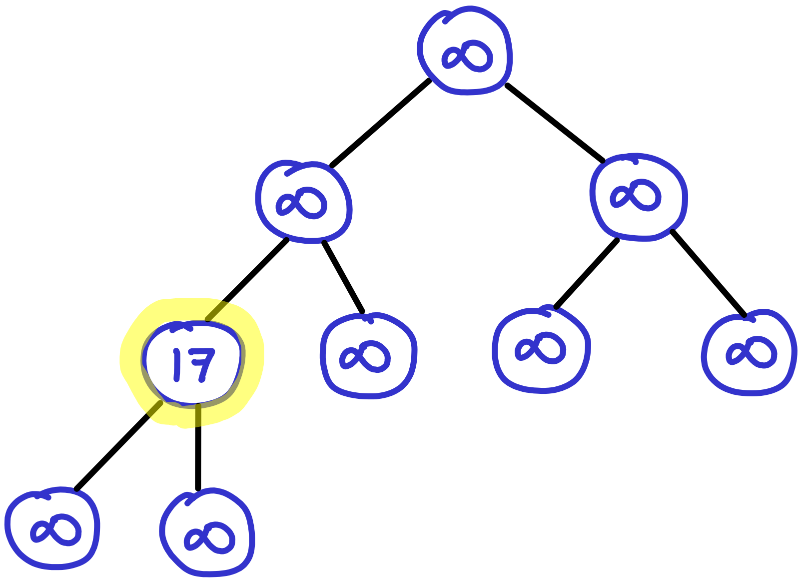
PRIM'S ALGORITHM for MST





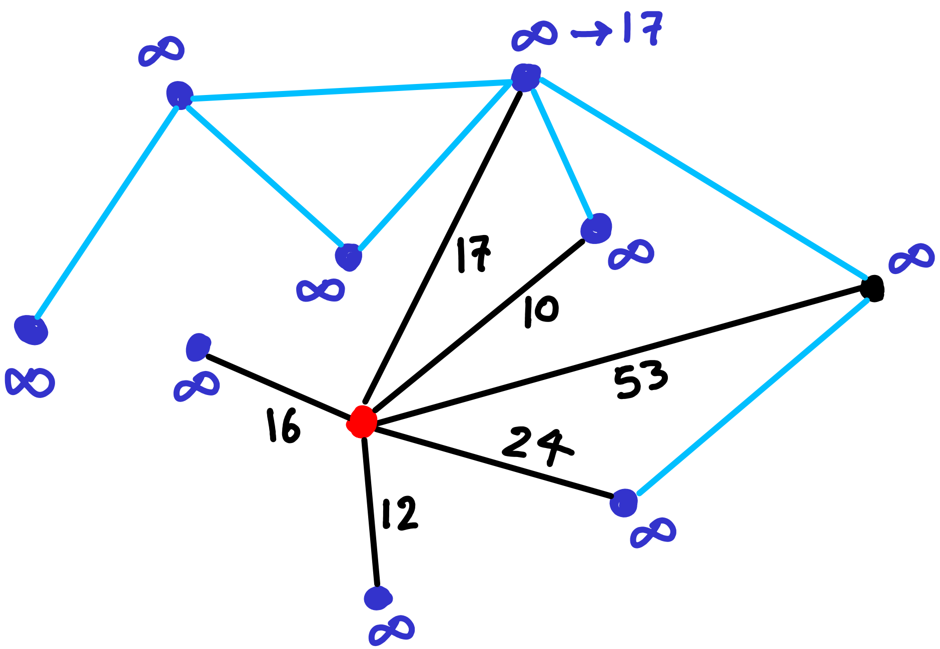
PRIM'S ALGORITHM for MST

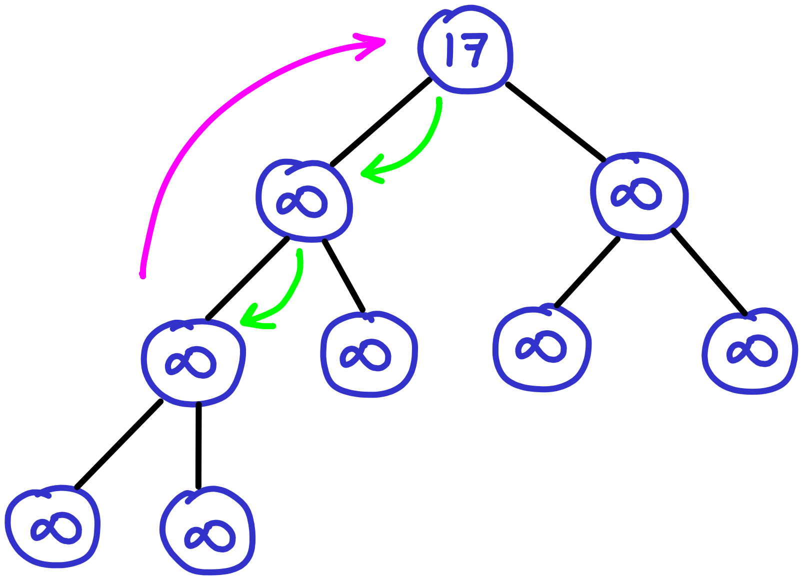




priority queue

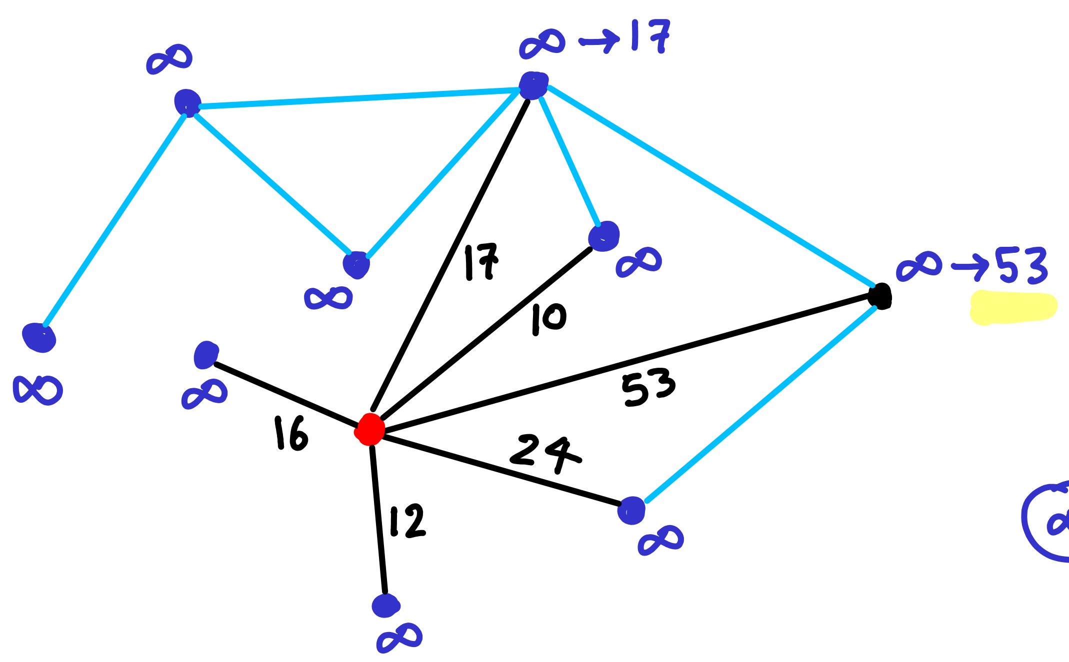
PRIM'S ALGORITHM for MST

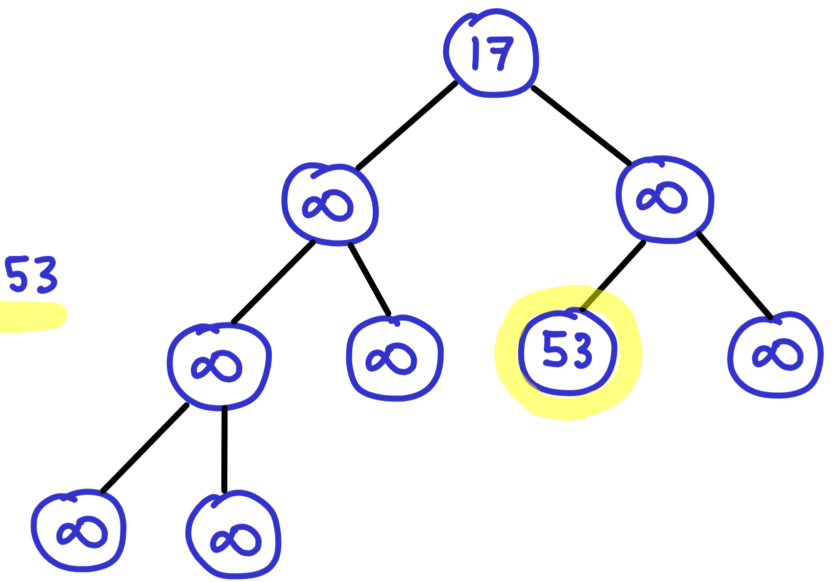




priority queue

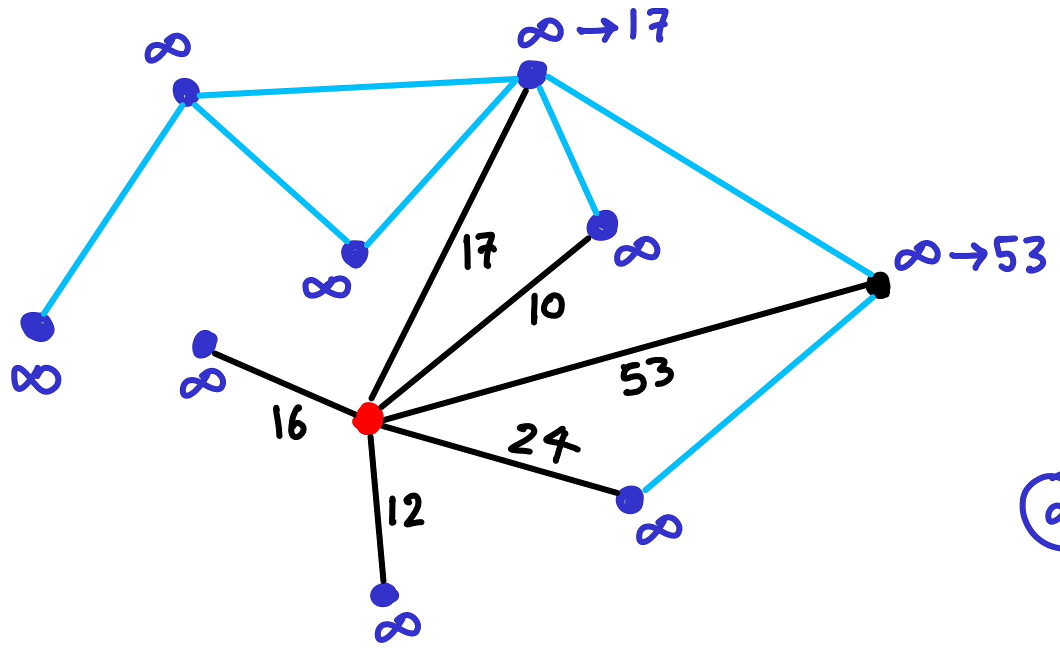
PRIM'S ALGORITHM for MST

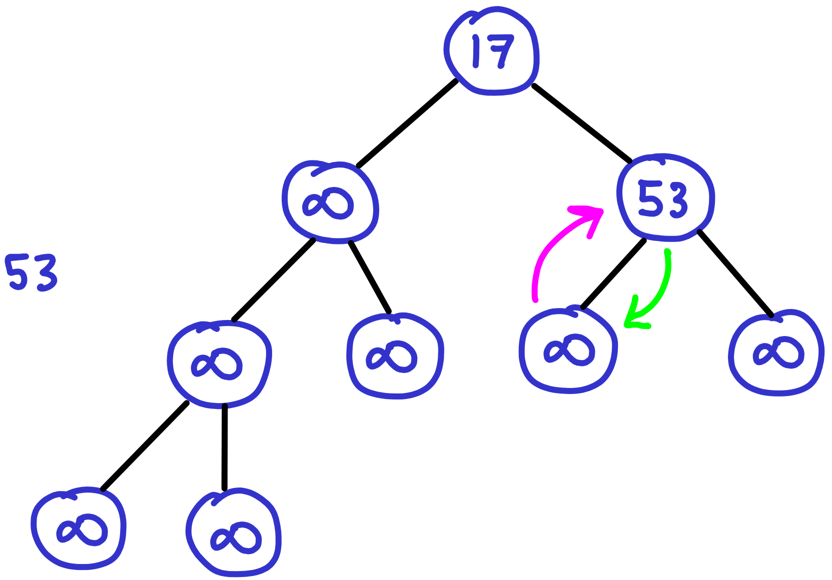




priority queue

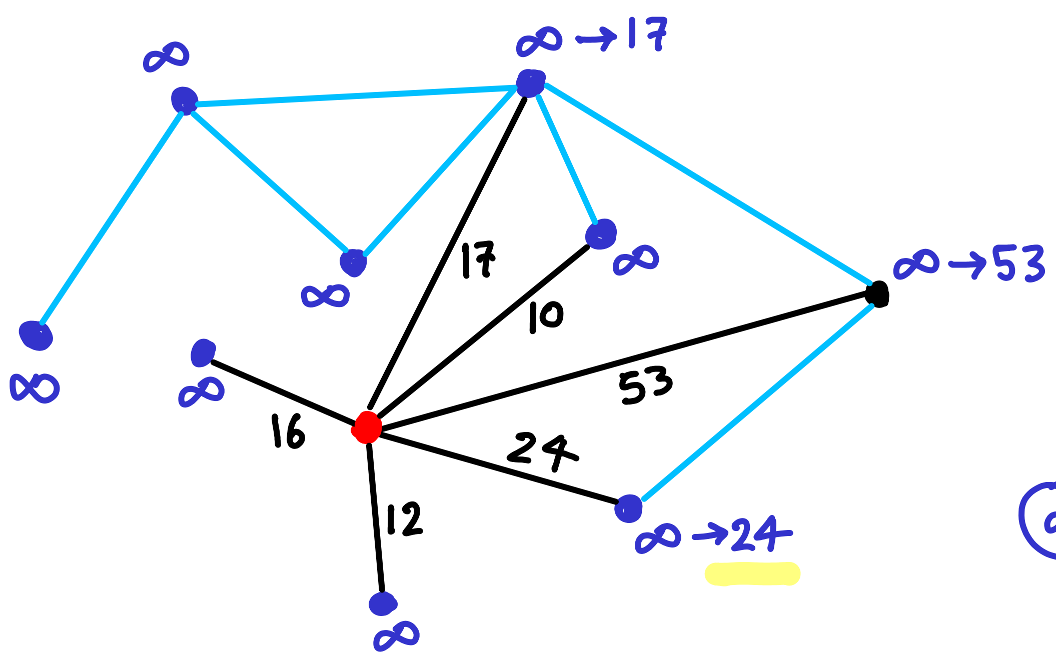
PRIM'S ALGORITHM for MST

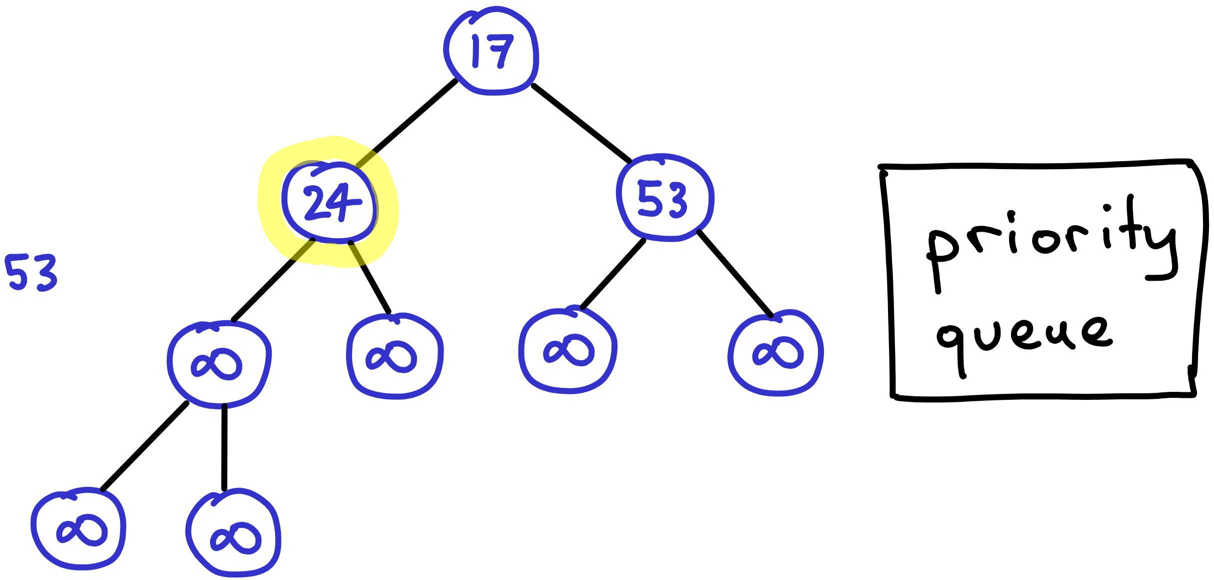




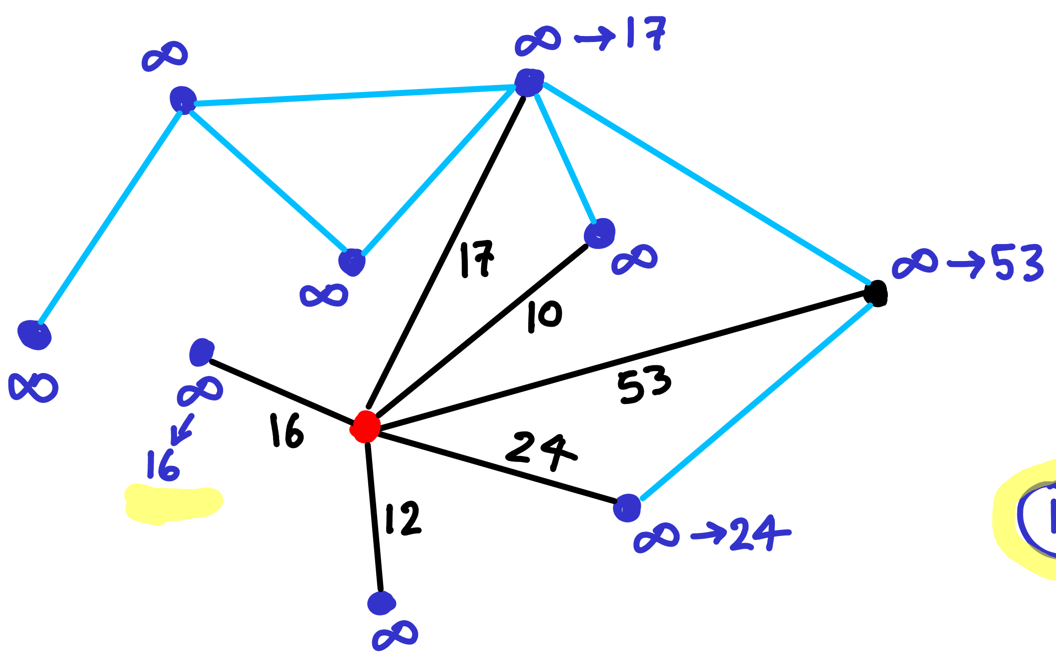
priority queue

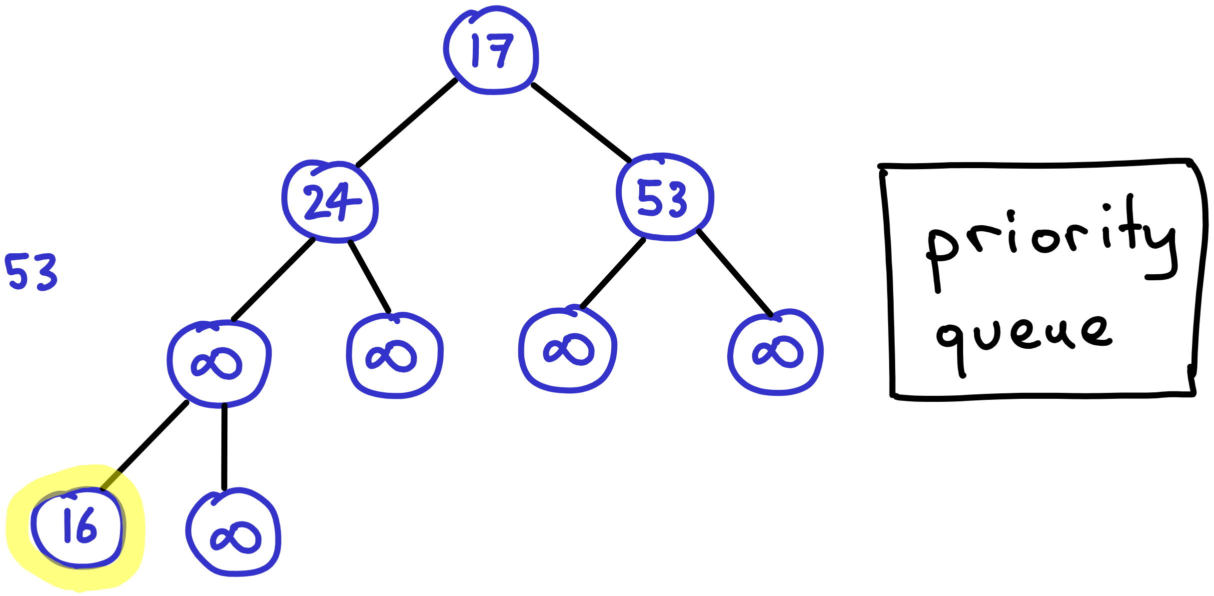
PRIM'S ALGORITHM for MST



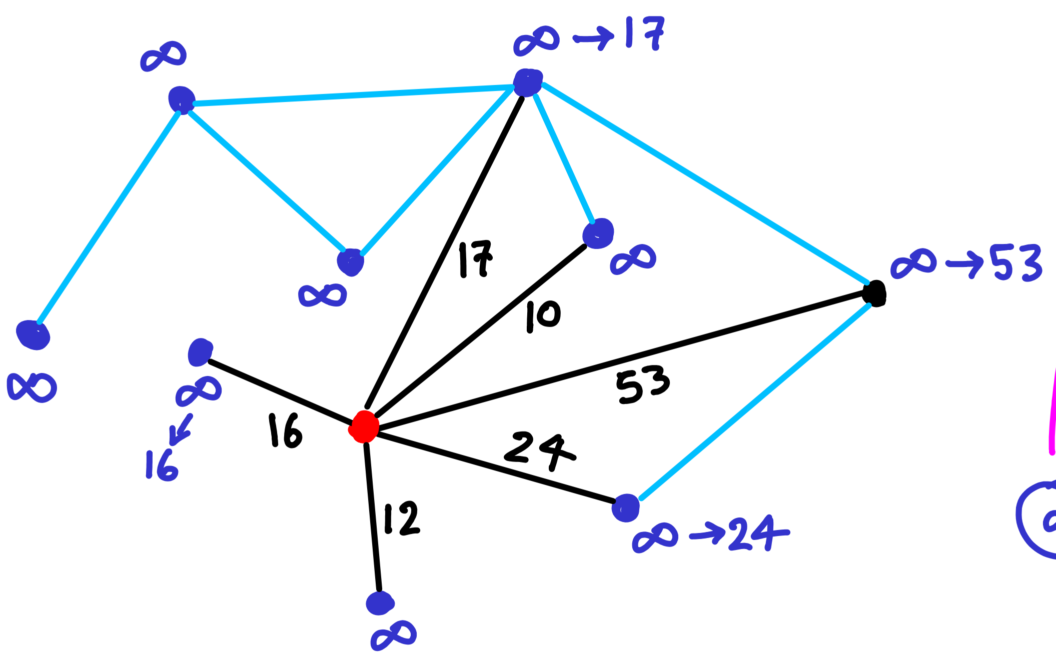


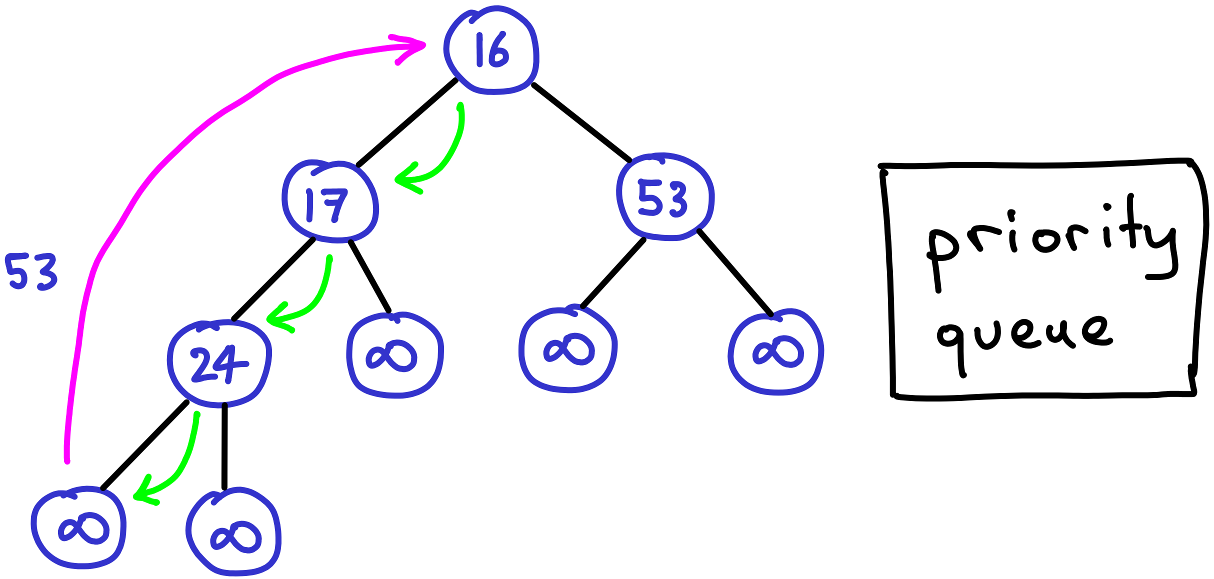
PRIM'S ALGORITHM for MST



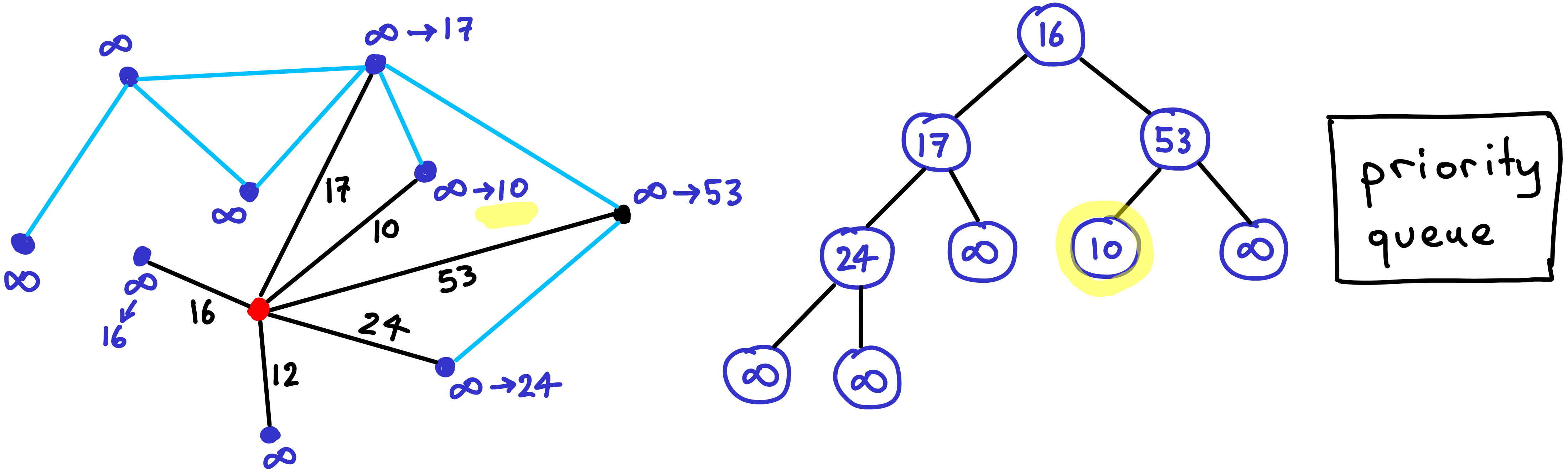


PRIM'S ALGORITHM for MST

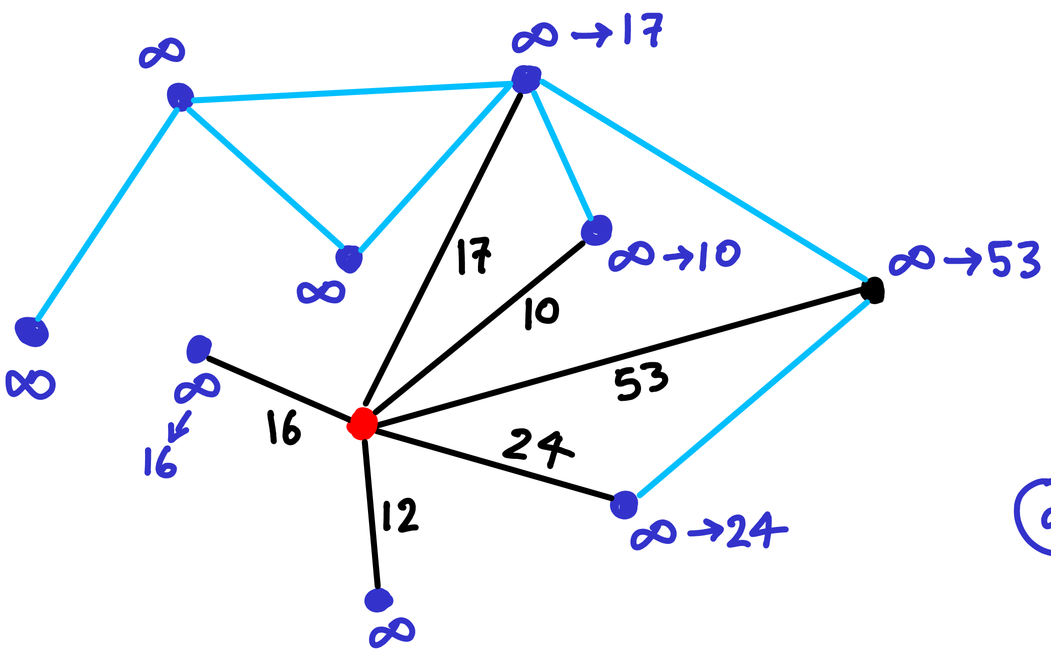


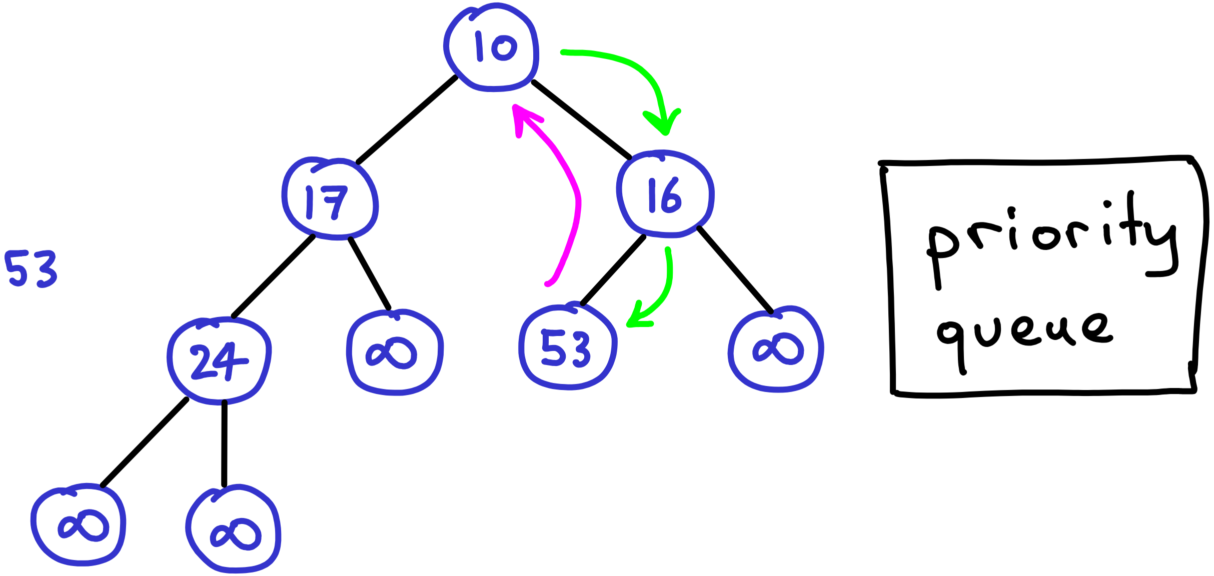


PRIM'S ALGORITHM for MST

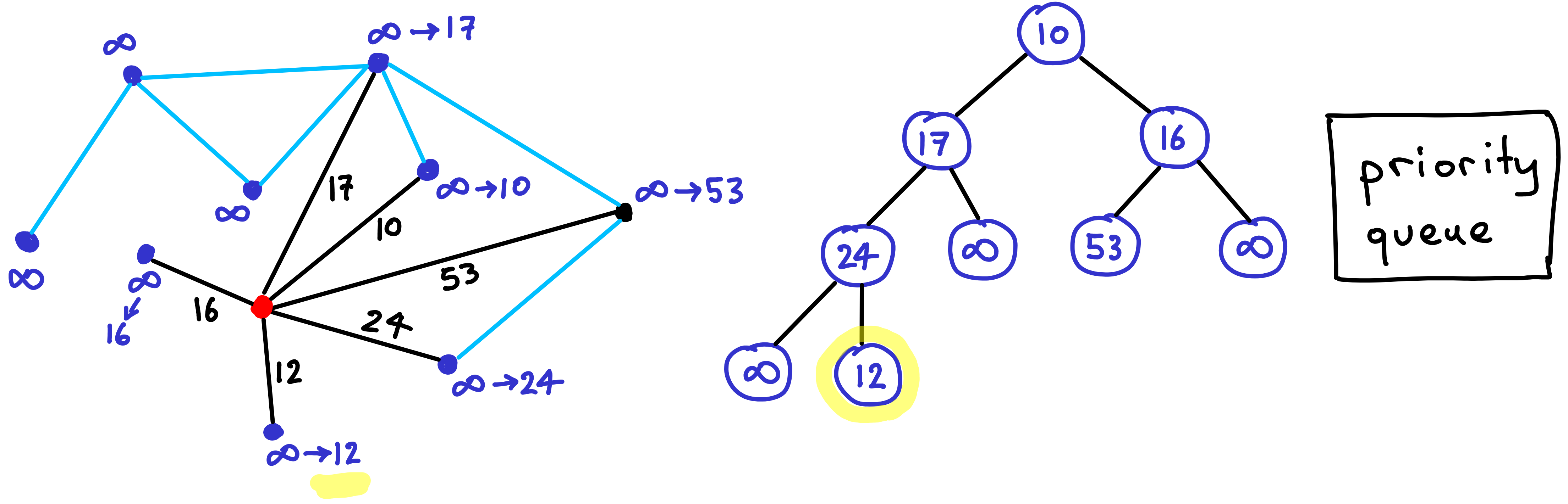


PRIM'S ALGORITHM for MST

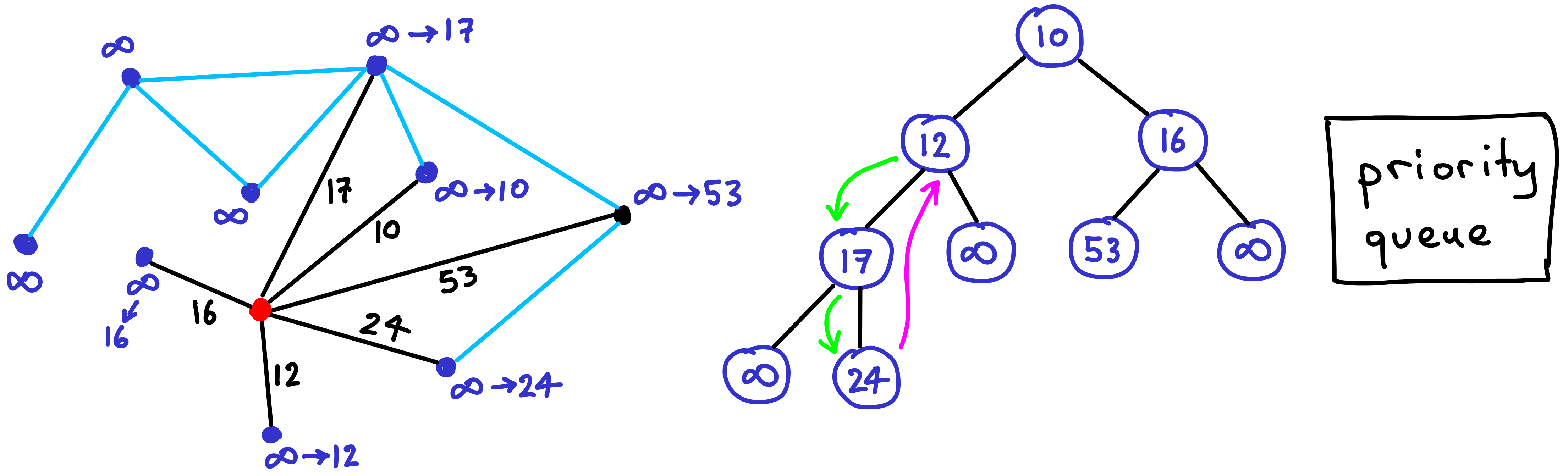




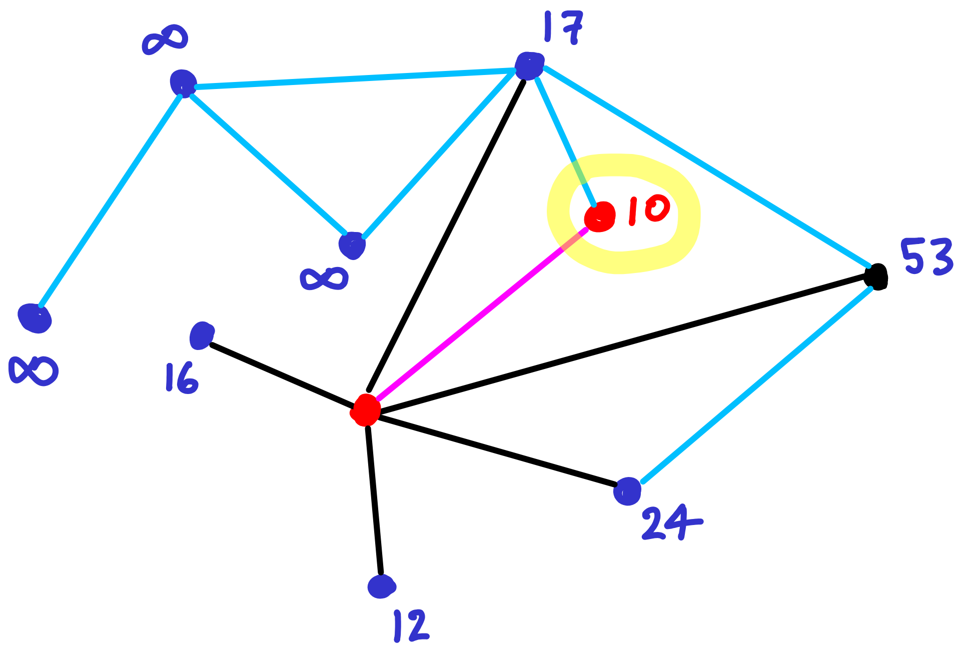
PRIM'S ALGORITHM for MST

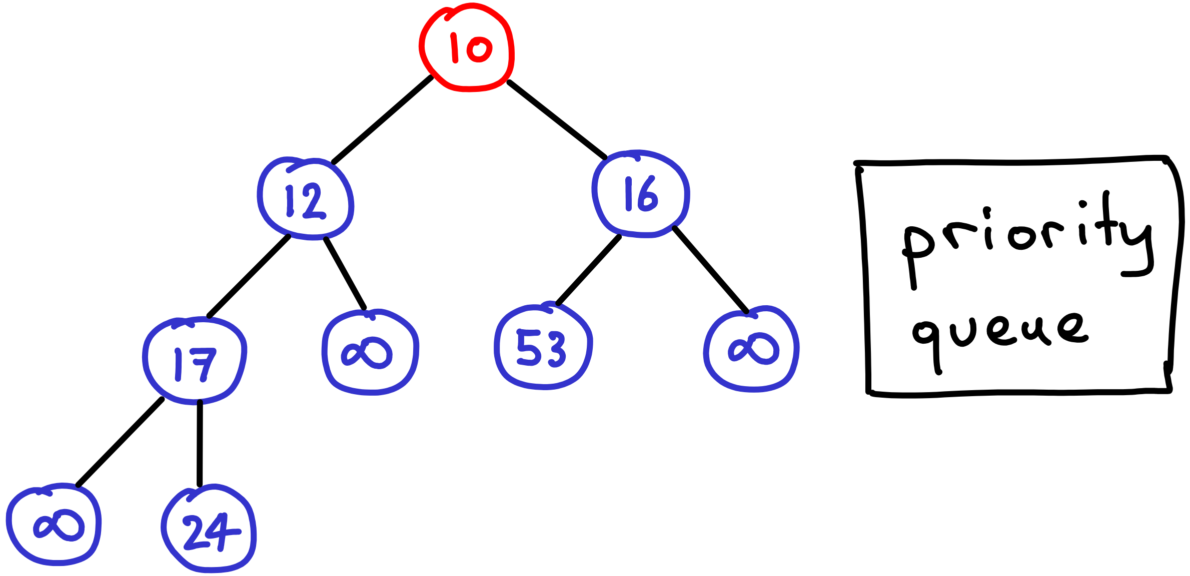


PRIM'S ALGORITHM for MST

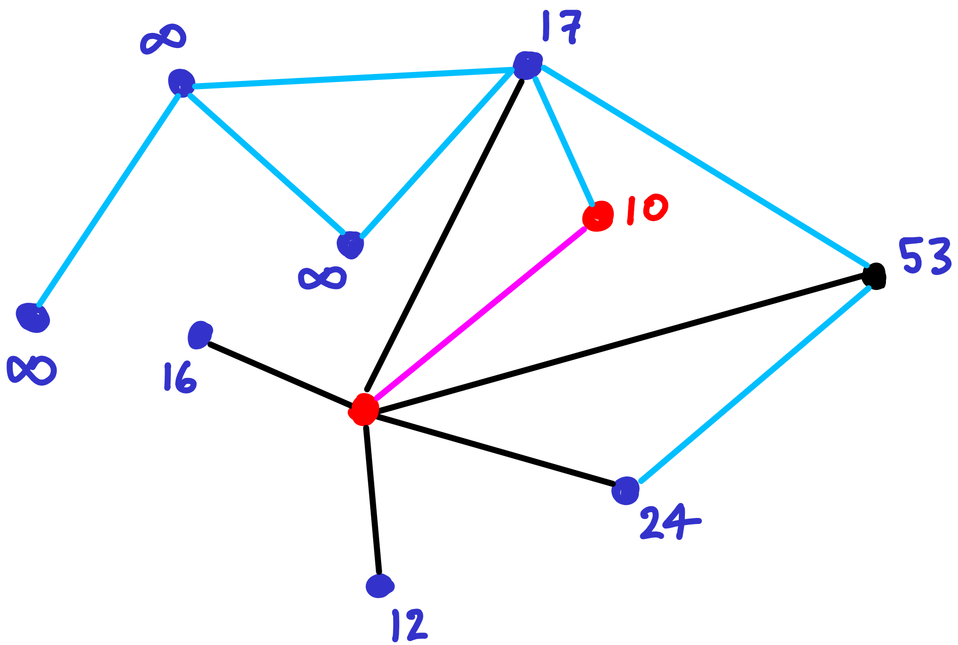


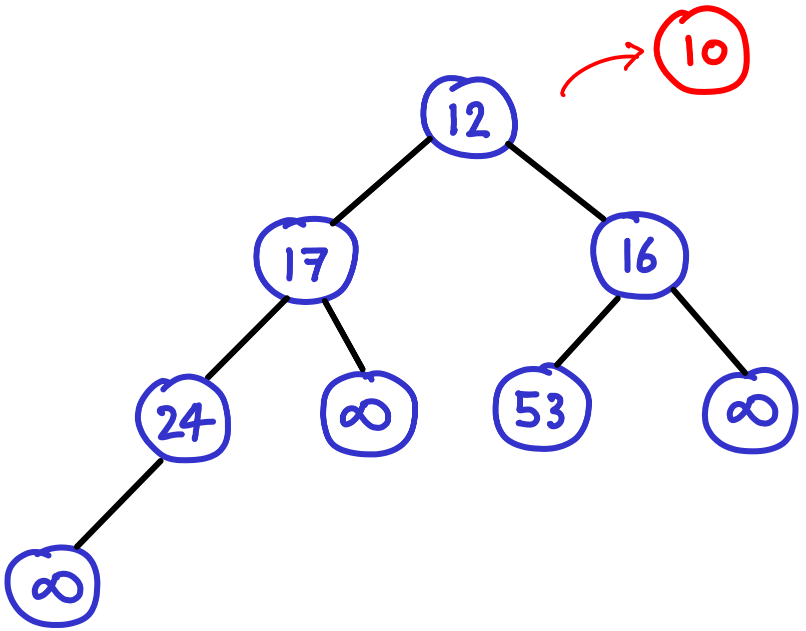
PRIM'S ALGORITHM for MST





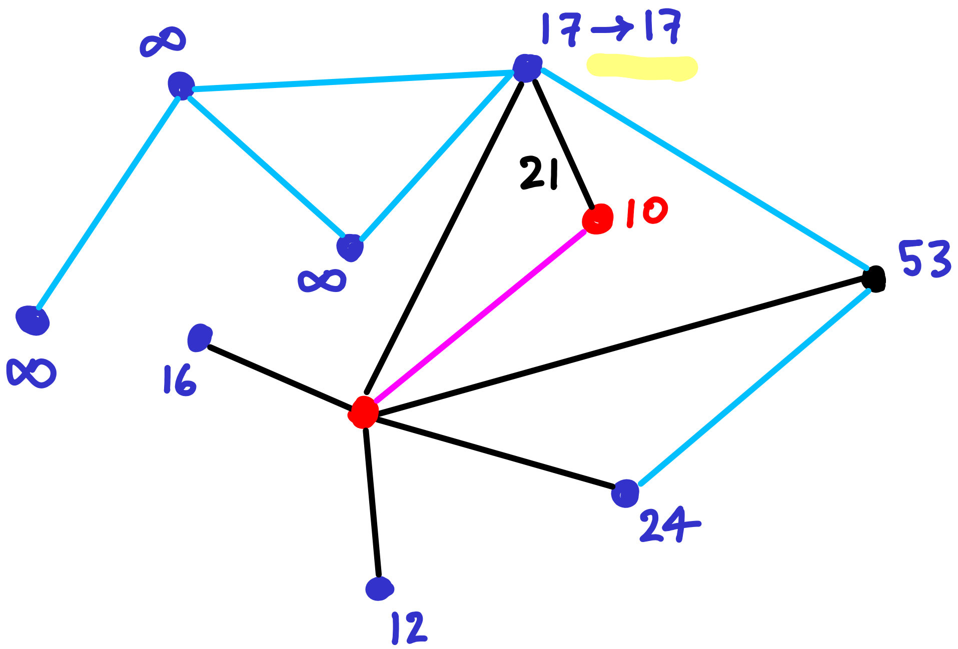
PRIM'S ALGORITHM for MST

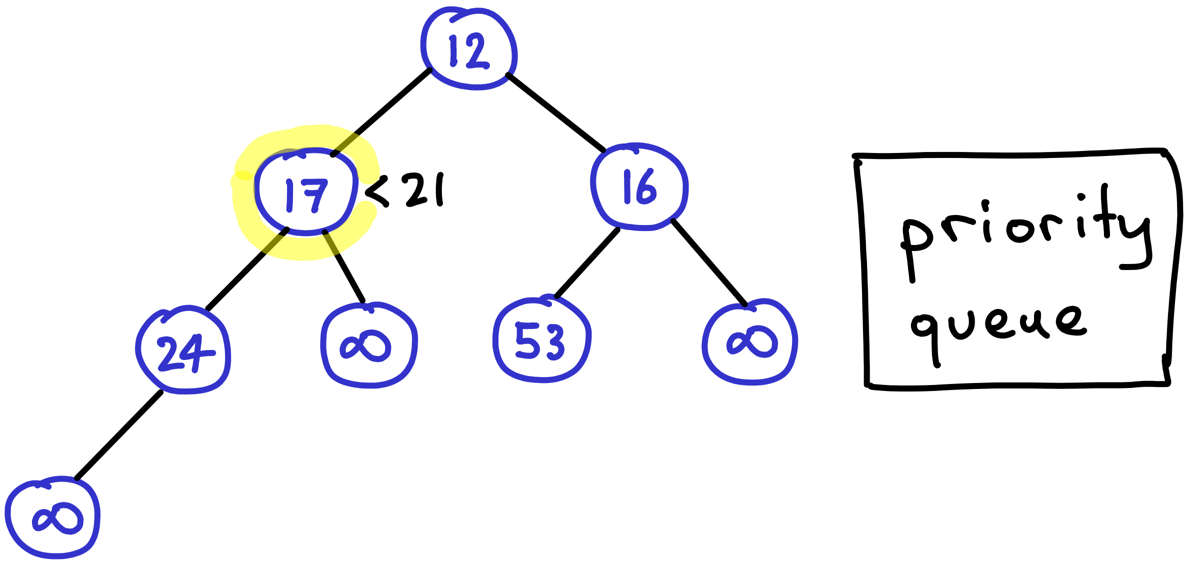




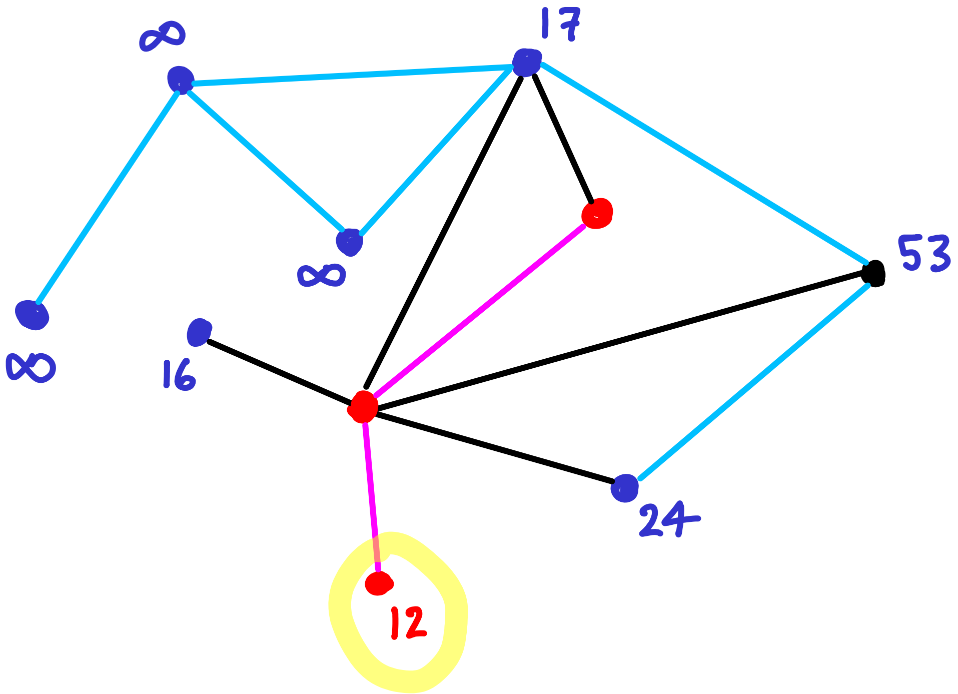
priority queue

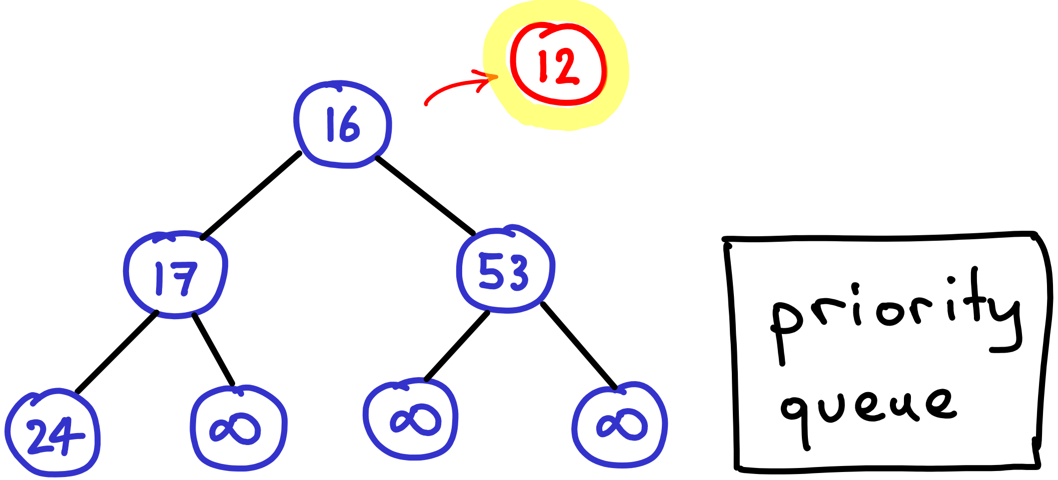
PRIM'S ALGORITHM for MST



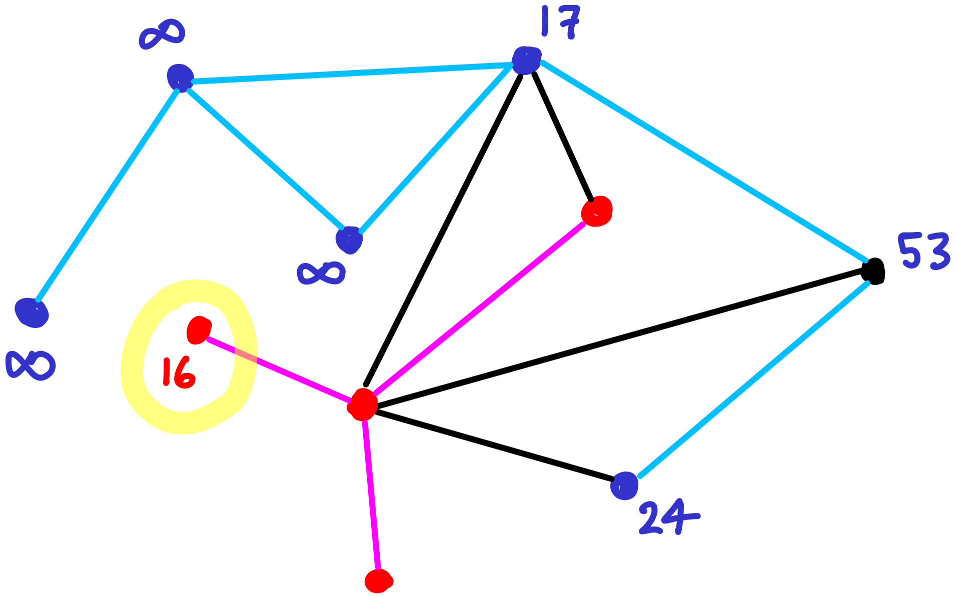


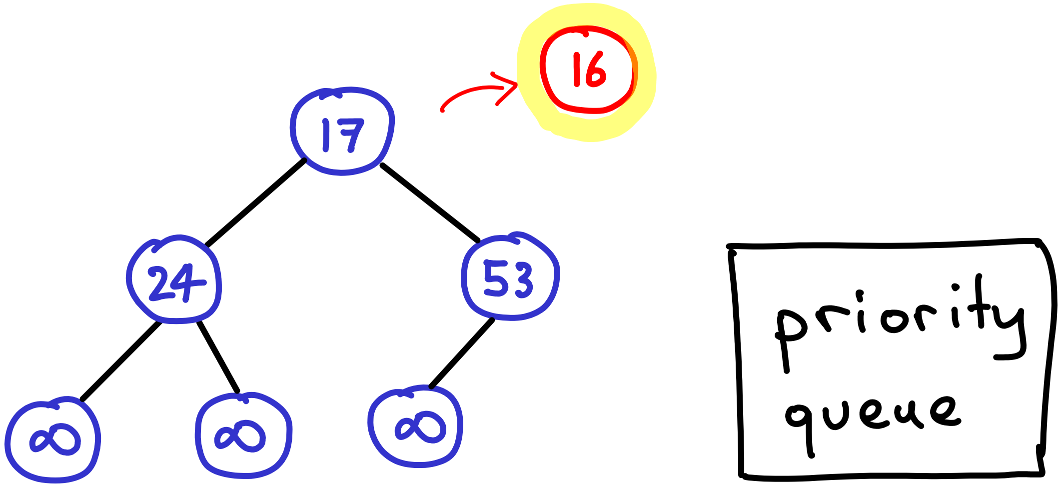
PRIM'S ALGORITHM for MST



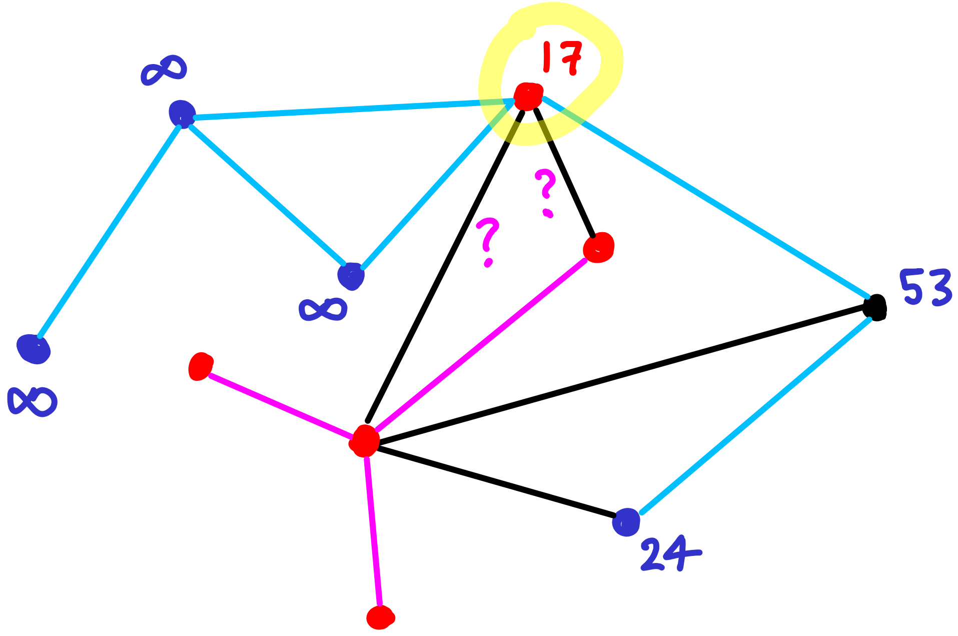


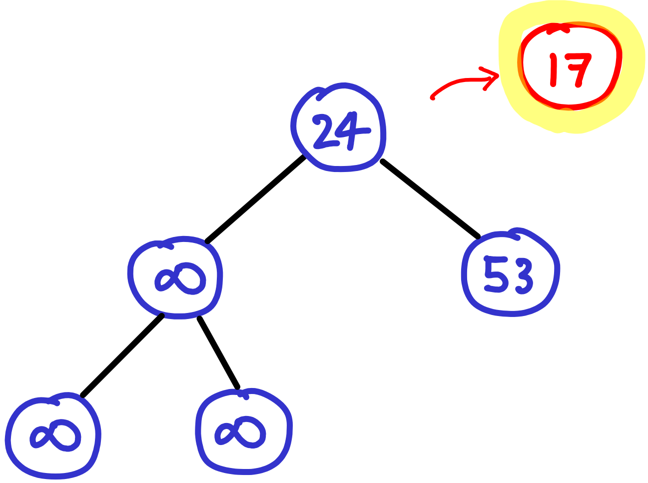
PRIM'S ALGORITHM for MST





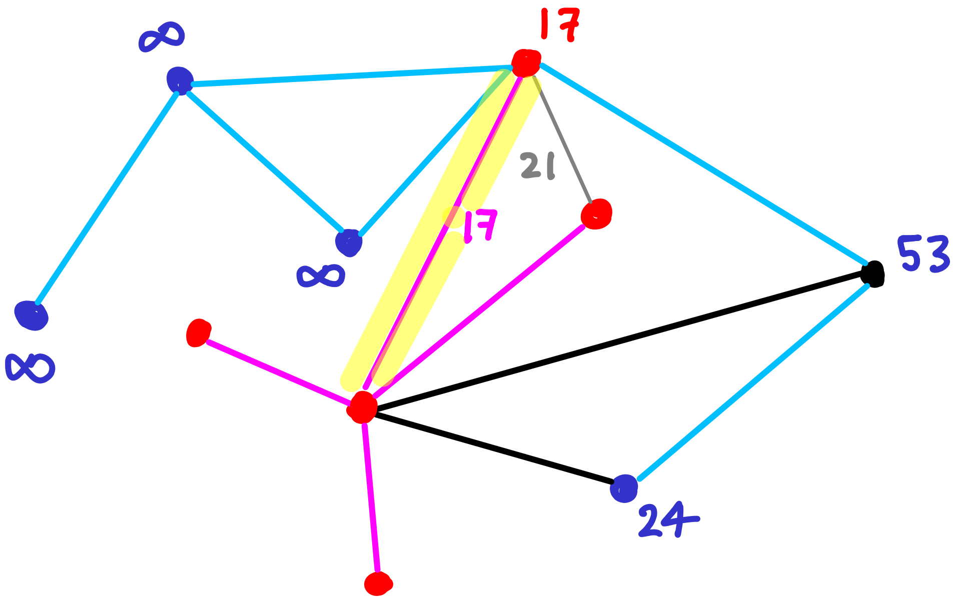
PRIM'S ALGORITHM for MST

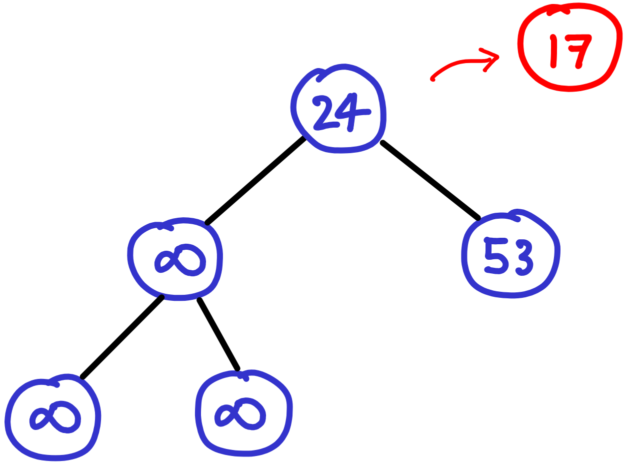




priority queue

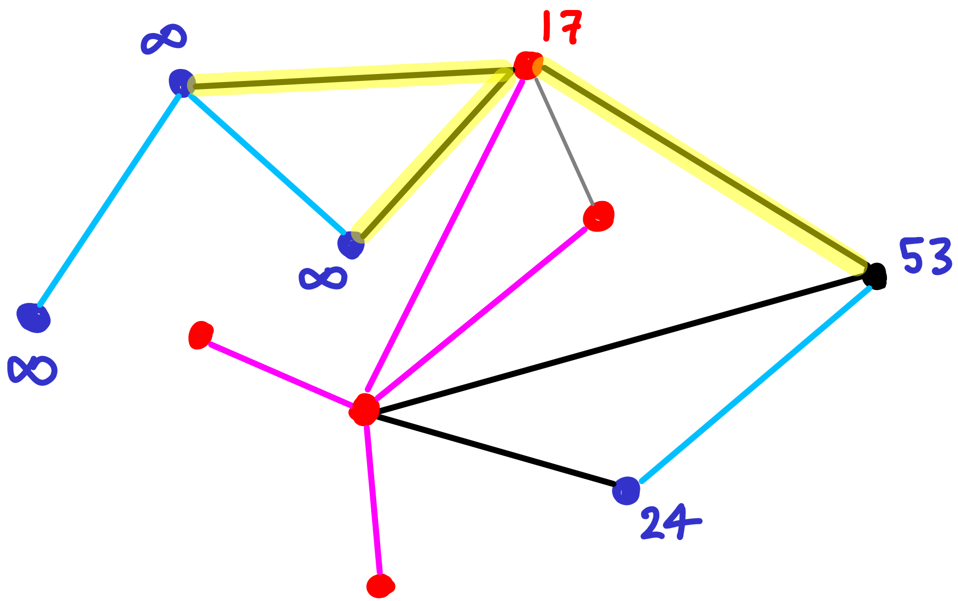
PRIM'S ALGORITHM for MST

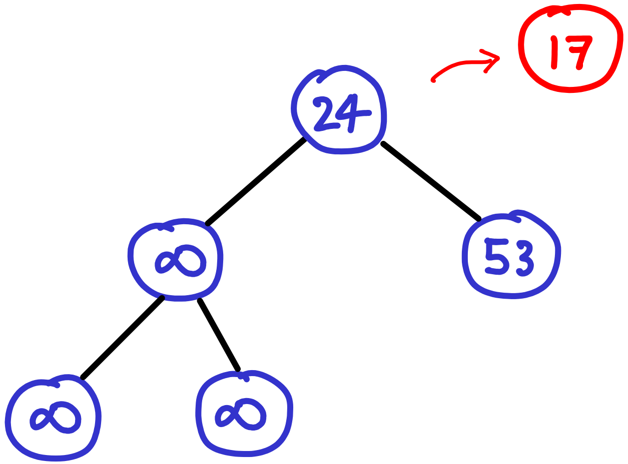




priority queue

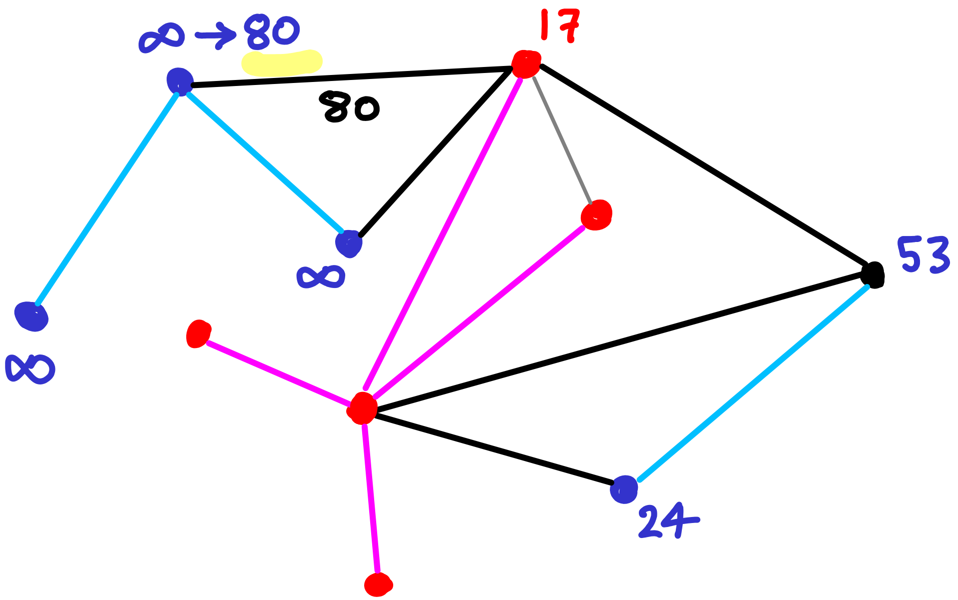
PRIM'S ALGORITHM for MST

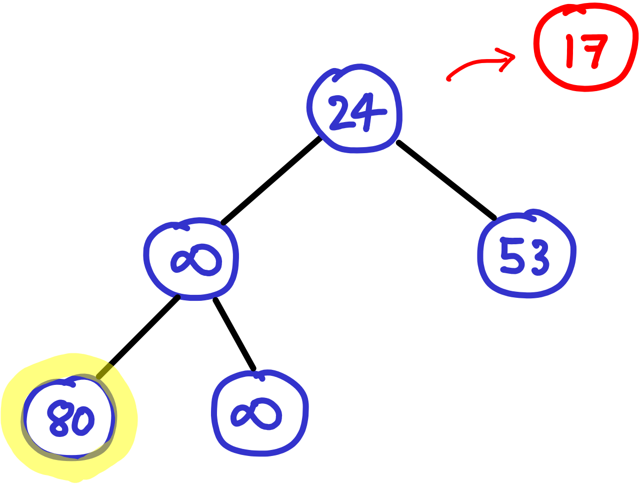




priority queue

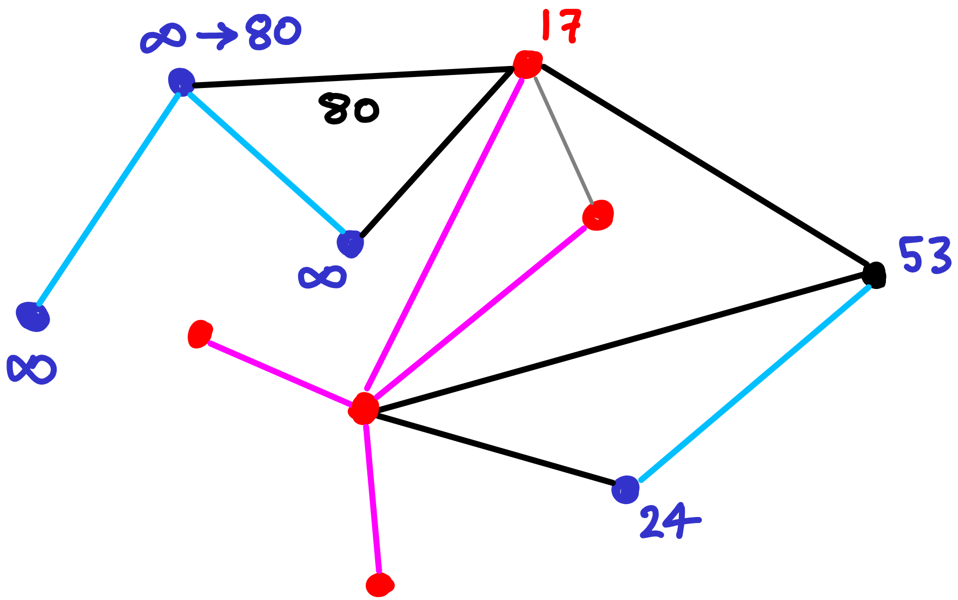
PRIM'S ALGORITHM for MST

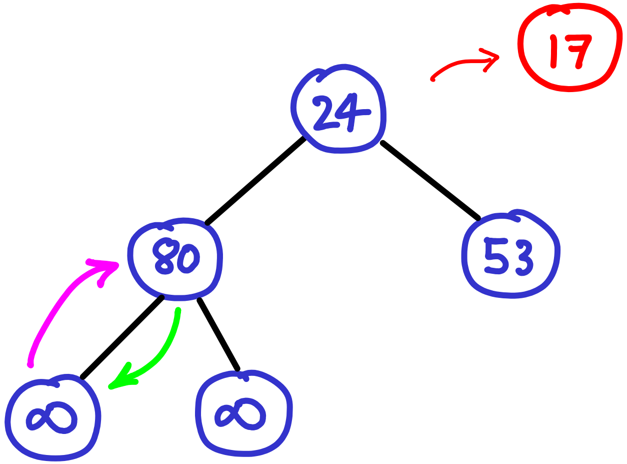




priority queue

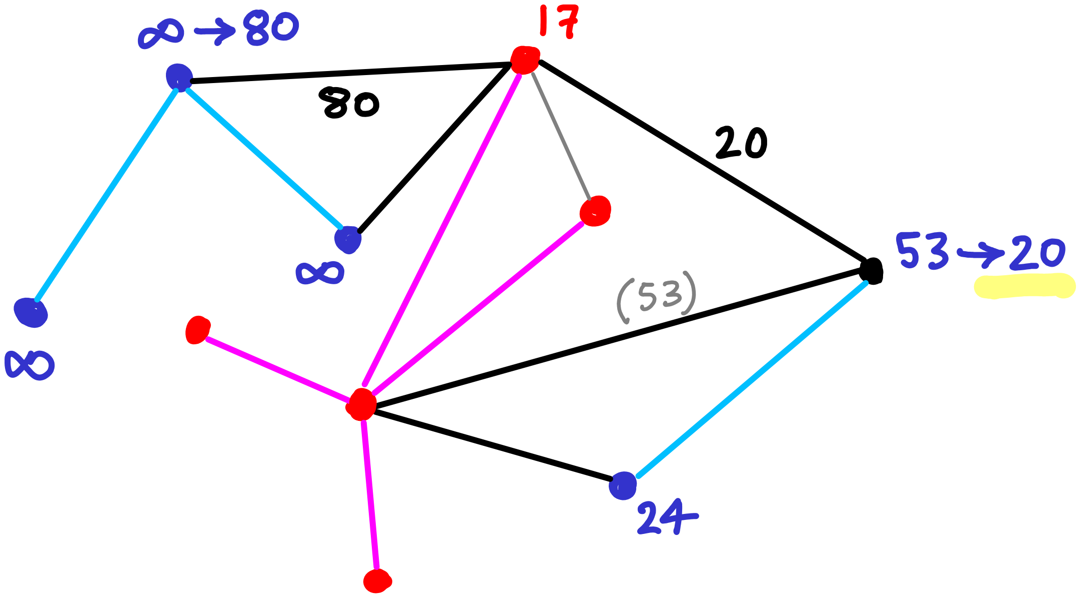
PRIM'S ALGORITHM for MST

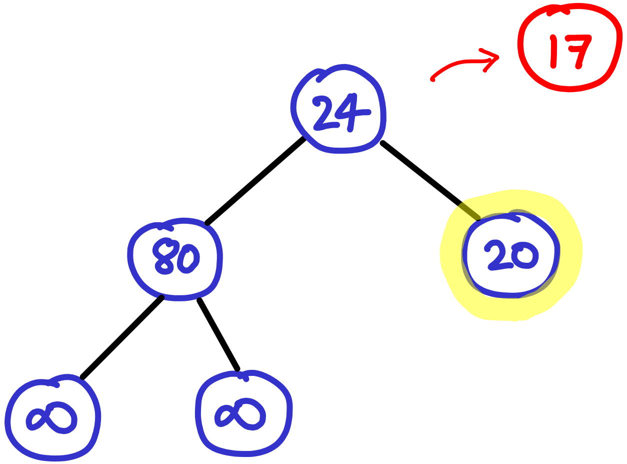




priority queue

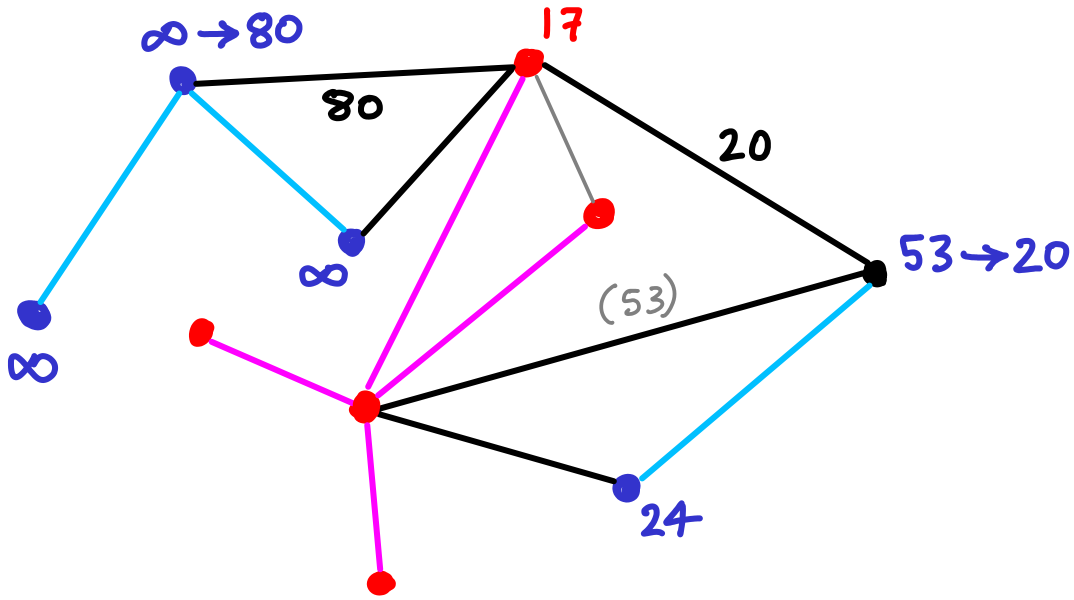
PRIM'S ALGORITHM for MST

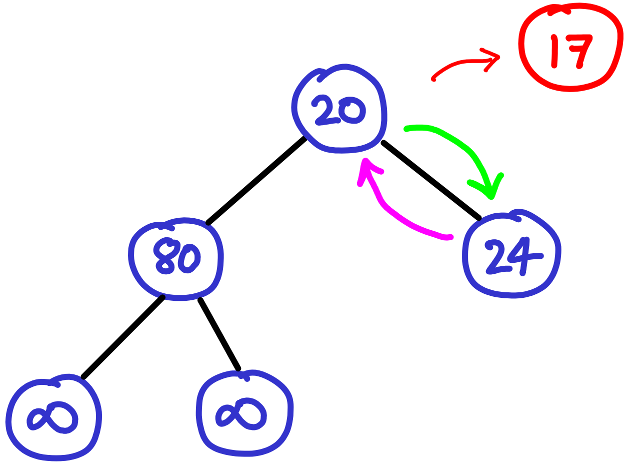




priority queue

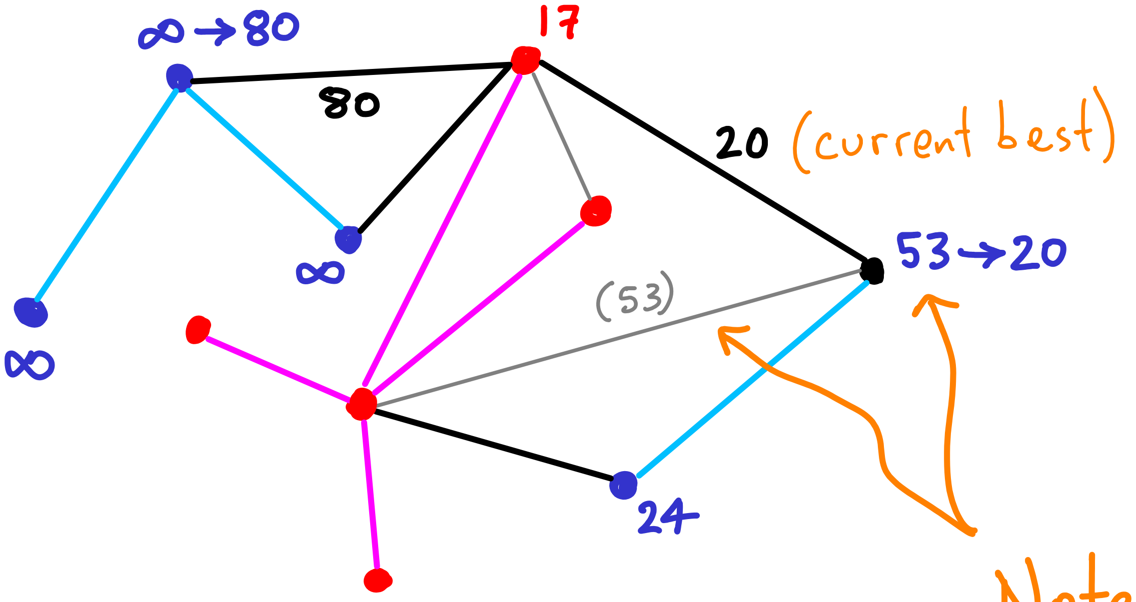
PRIM'S ALGORITHM for MST

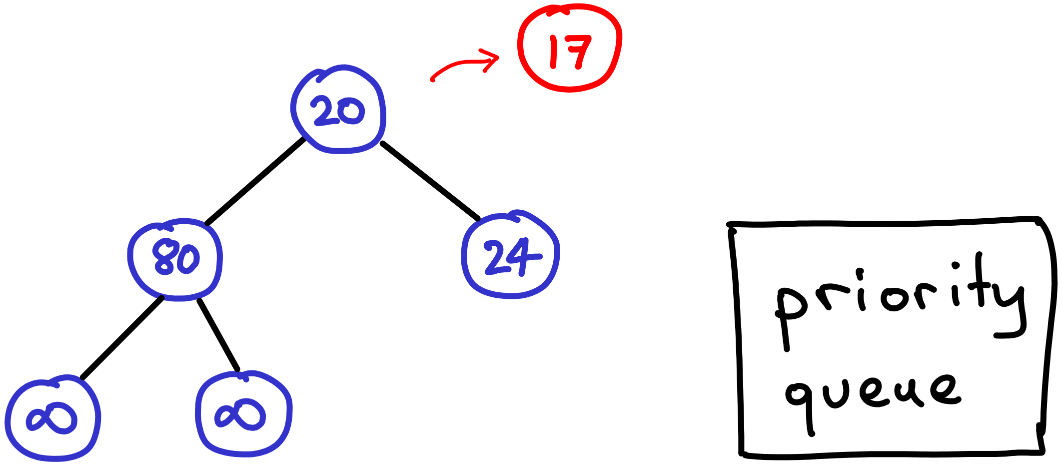




priority
queue

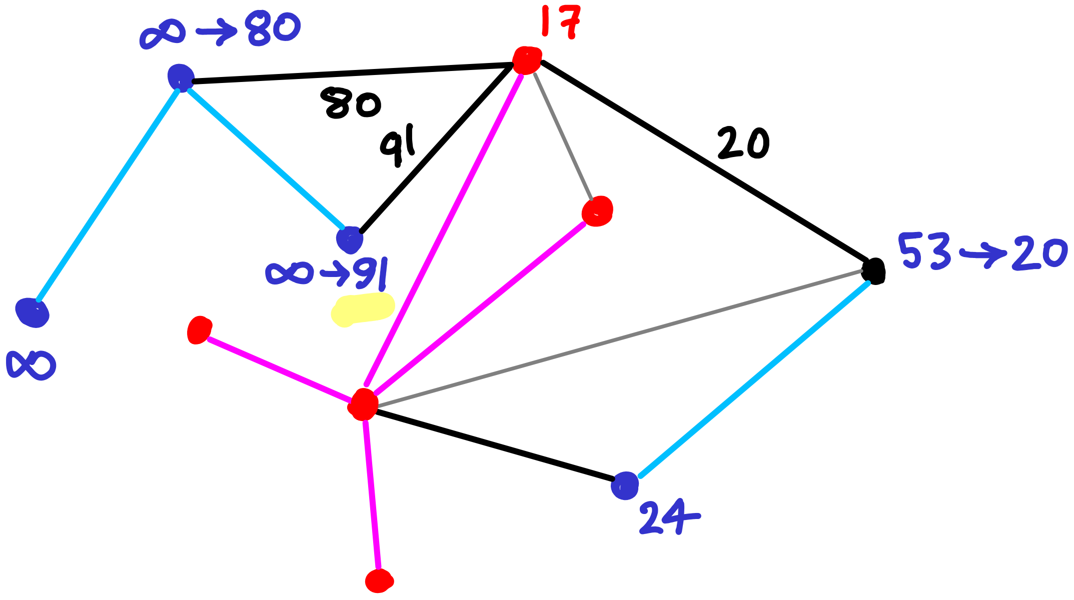
PRIM'S ALGORITHM for MST

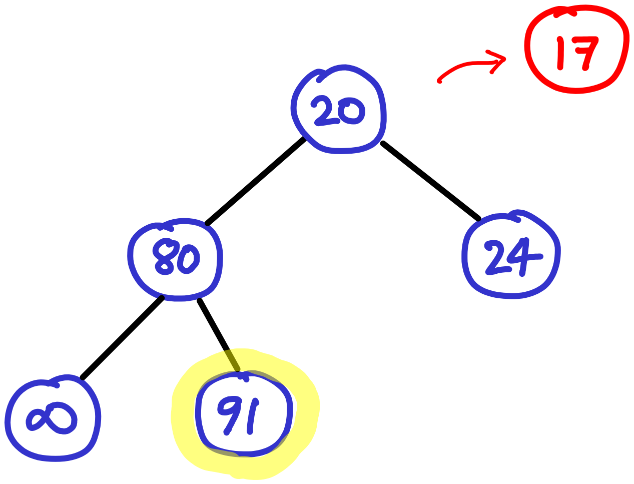




Note: a vertex can keep track of its "best" edge, so when it is added to T, we don't need to find min.

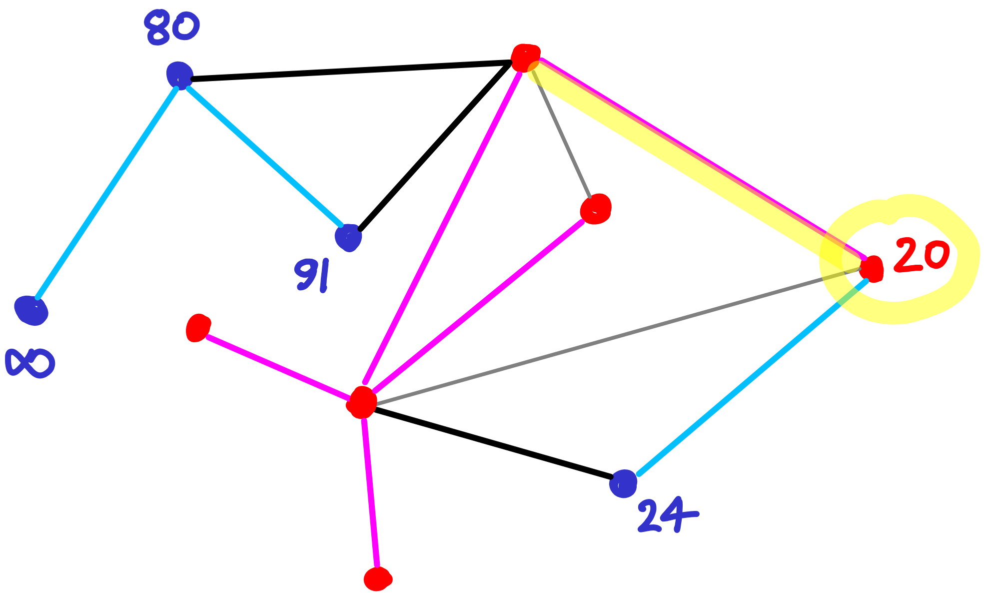
PRIM'S ALGORITHM for MST

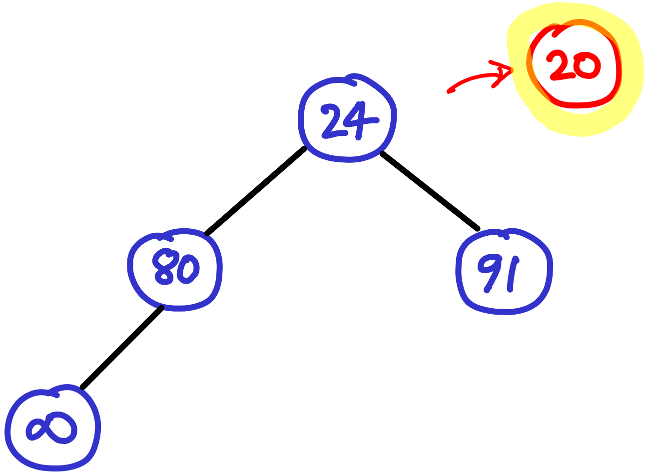




priority queue

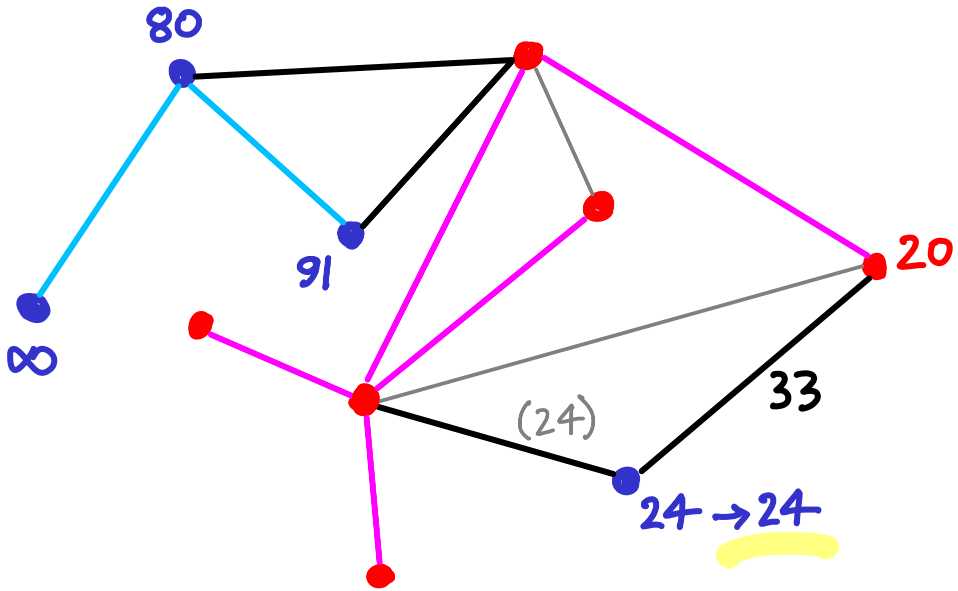
PRIM'S ALGORITHM for MST

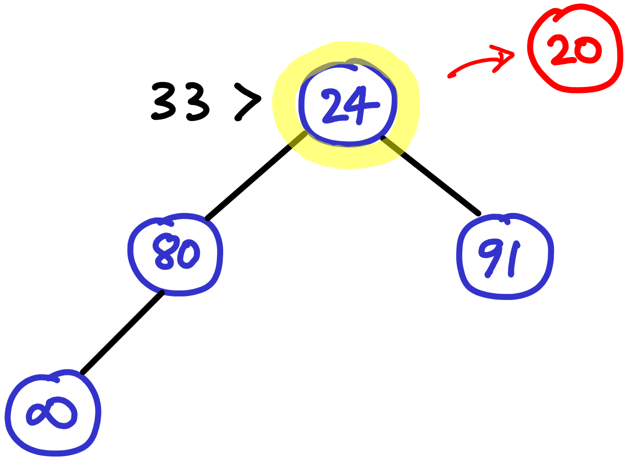




priority queue

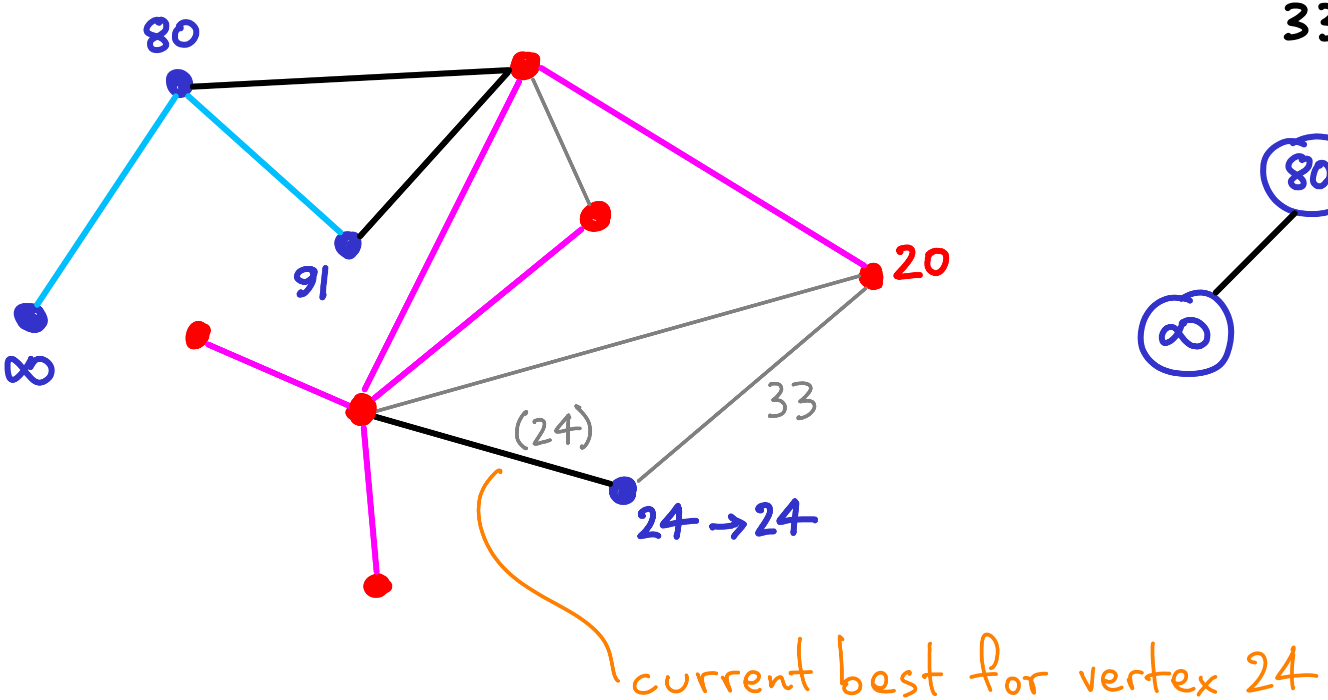
PRIM'S ALGORITHM for MST

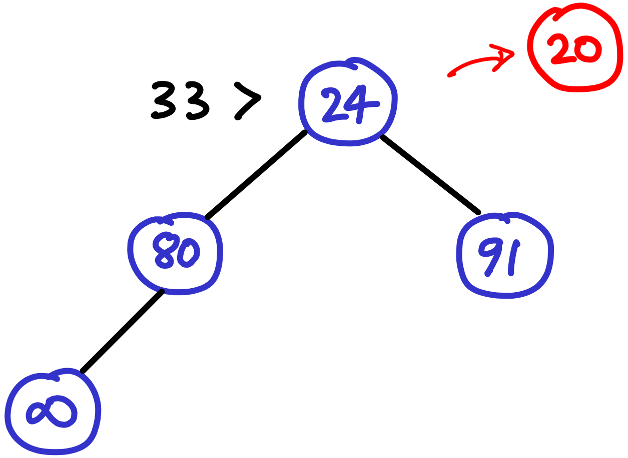




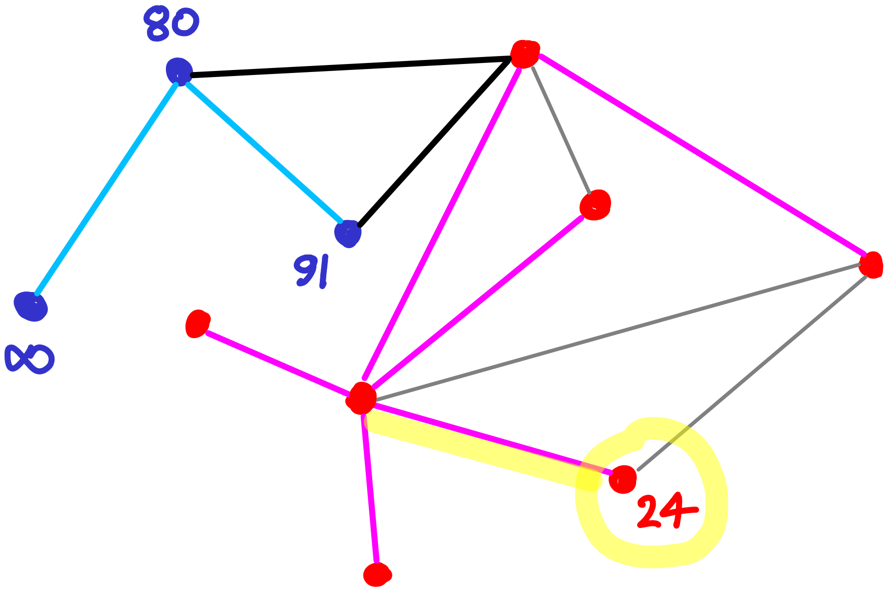
priority queue

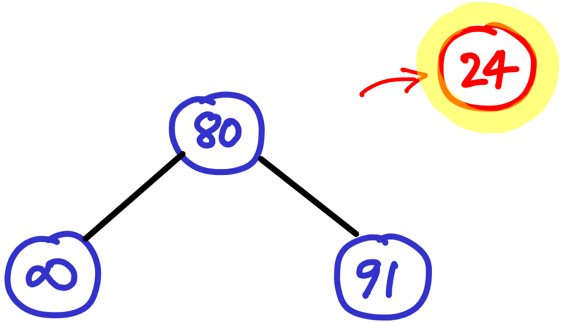
PRIM'S ALGORITHM for MST





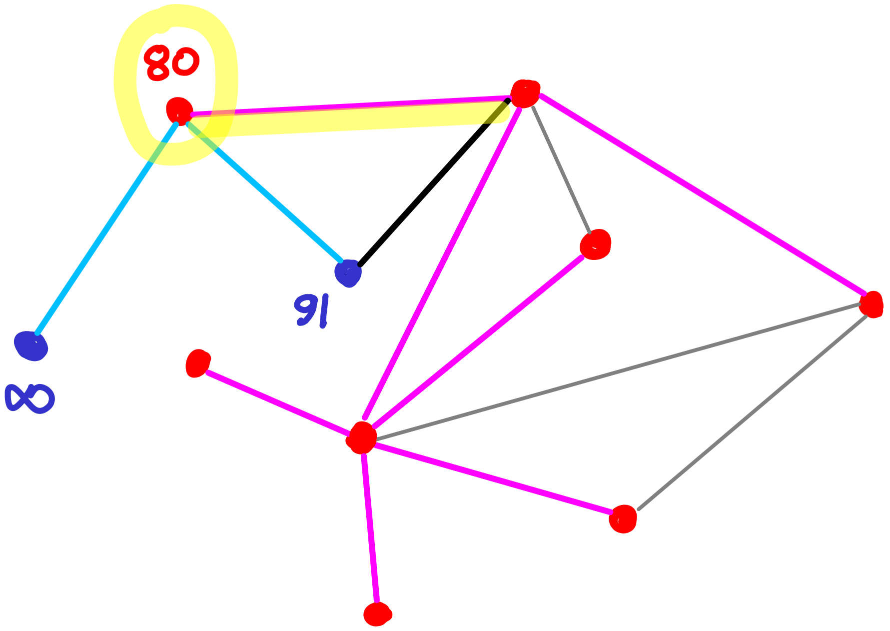
PRIM'S ALGORITHM for MST

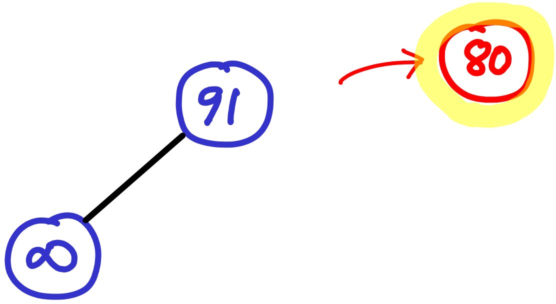




priority queue

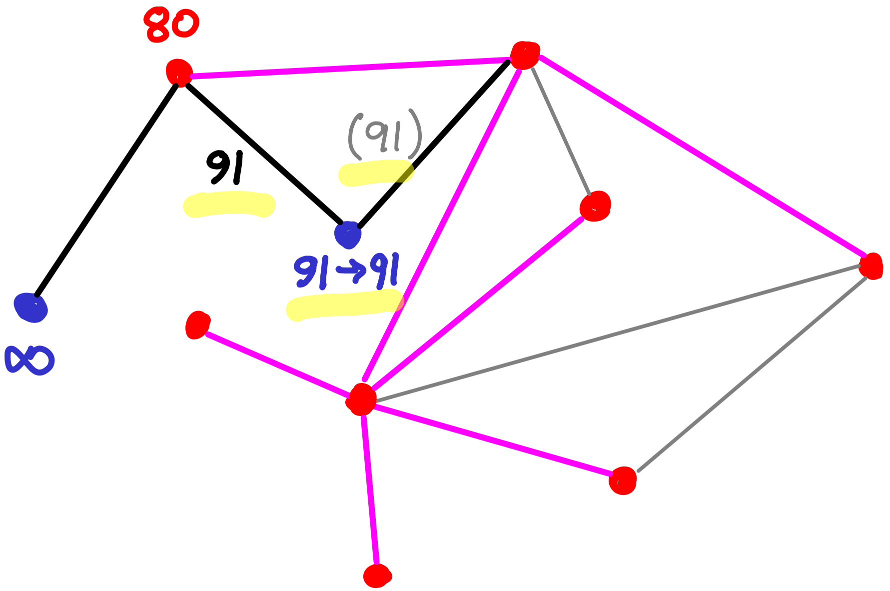
PRIM'S ALGORITHM for MST

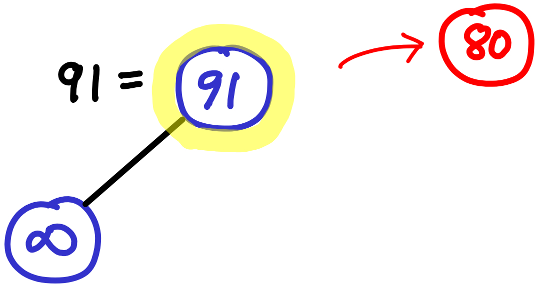




priority queue

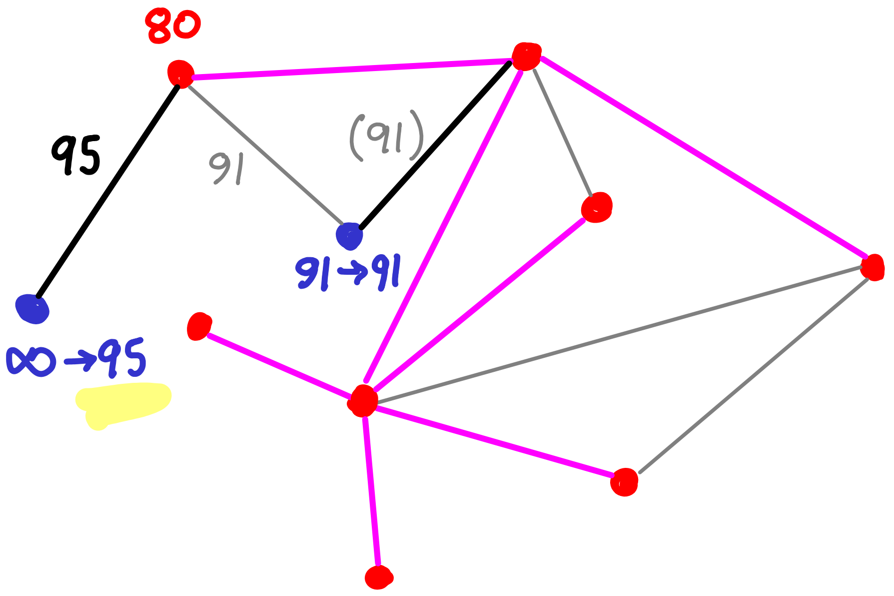
PRIM'S ALGORITHM for MST

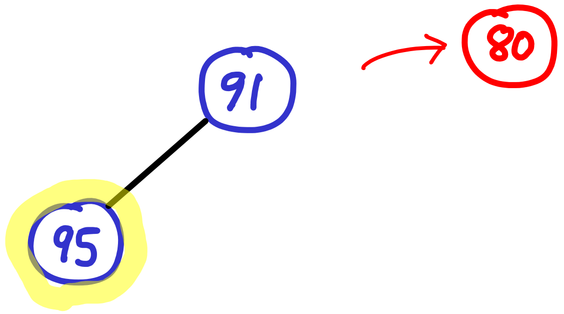




priority queue

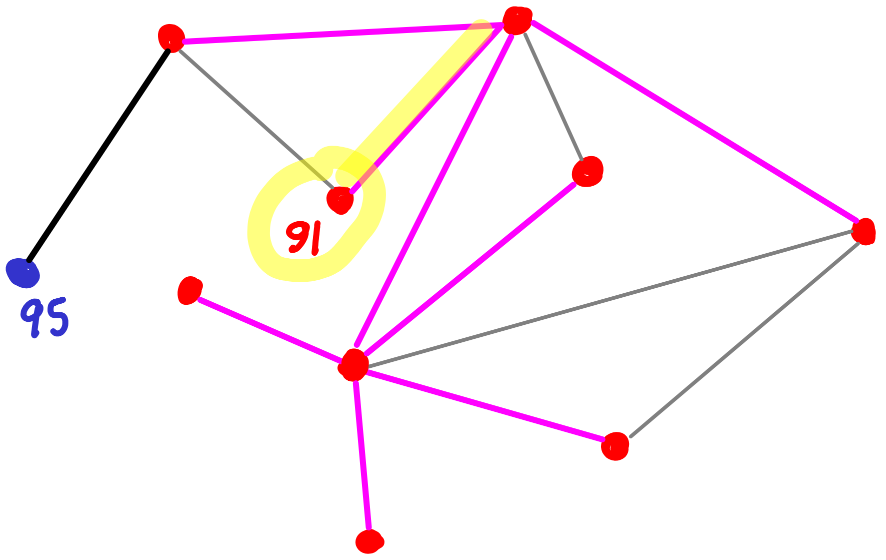
PRIM'S ALGORITHM for MST

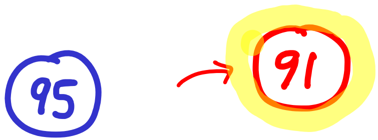




priority queue

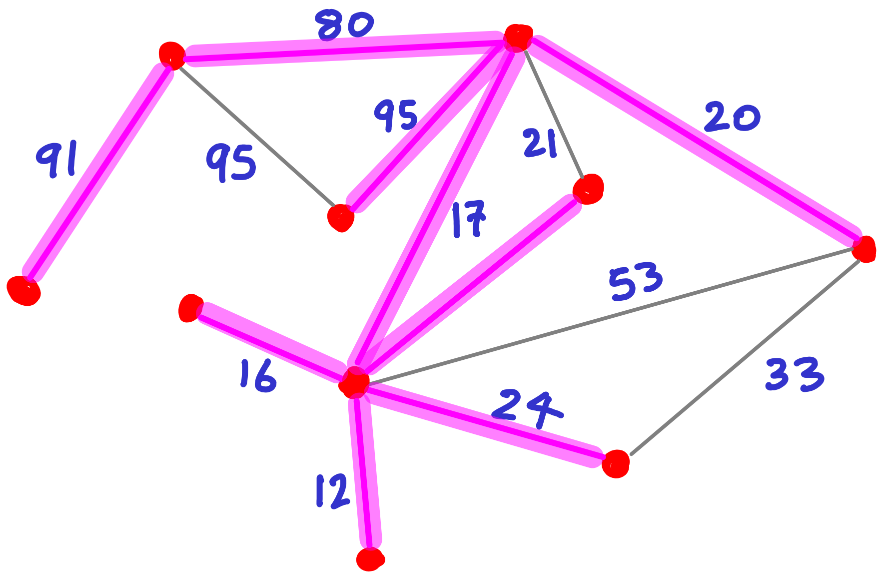
PRIM'S ALGORITHM for MST



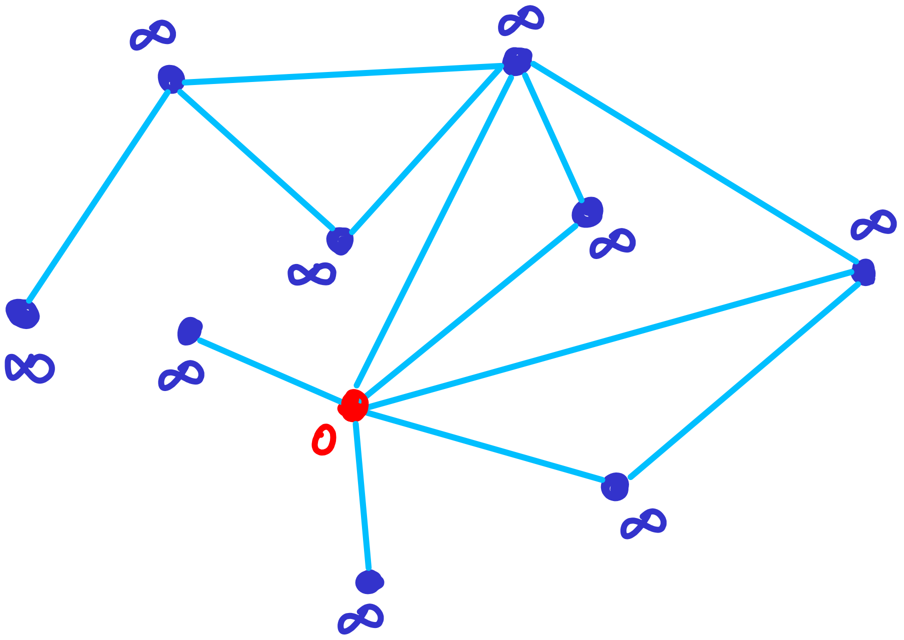


priority queue

PRIM'S ALGORITHM for MST

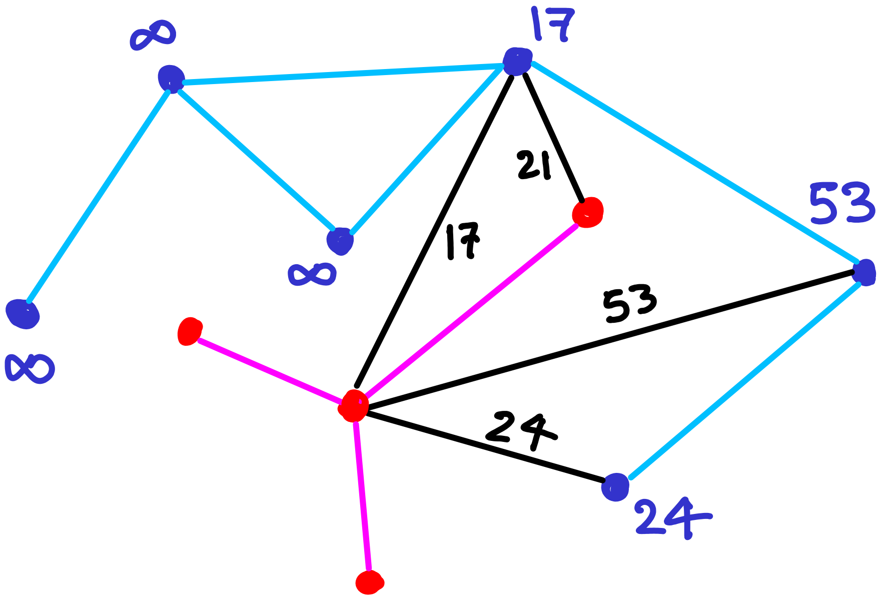


PRIM'S ALGORITHM for MST



1) start w/ any vertex s; set w(s)=0
2) set w(≠s) = ∞ & put all in pr.queue

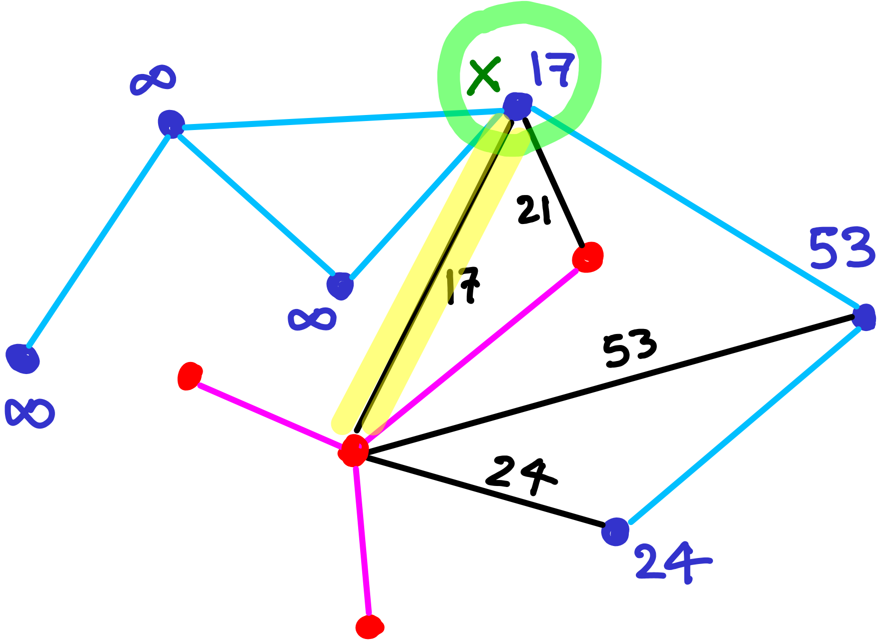
PRIM'S ALGORITHM for MST



1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

$|V|$ rounds

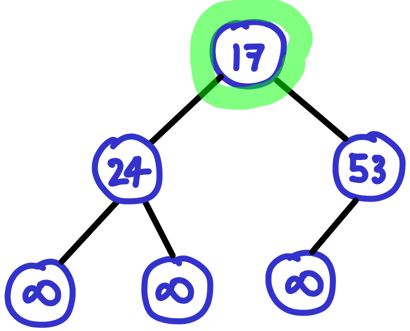
PRIM'S ALGORITHM for MST



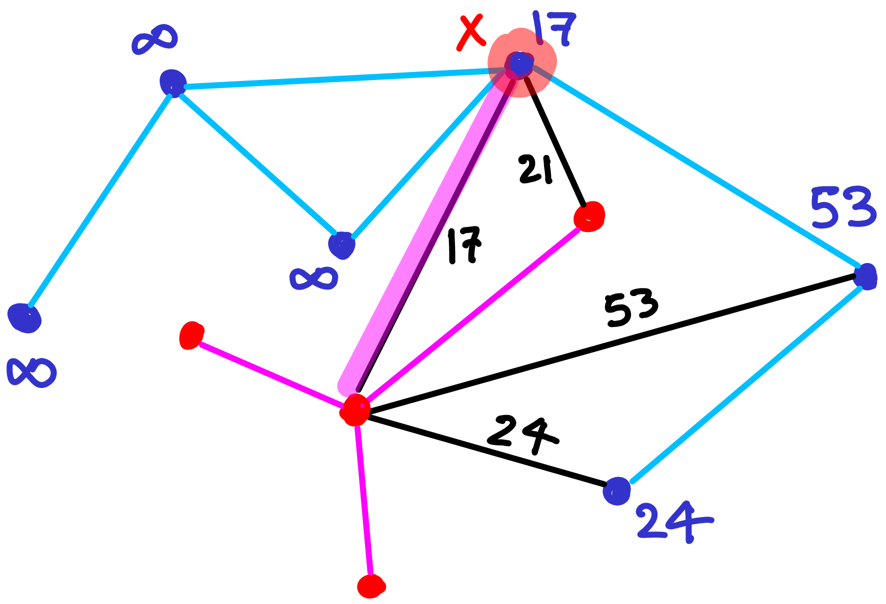
1) start w/ any vertex s; set w(s)=0
2) set w(≠s) = ∞ & put all in pr.queue
3) while pr.queue not empty
   x: extract-min & add edge to T
   mark x → in T.

("add edge" → find an edge from x to T, w/ min weight)

(if x=s, no edge to add)

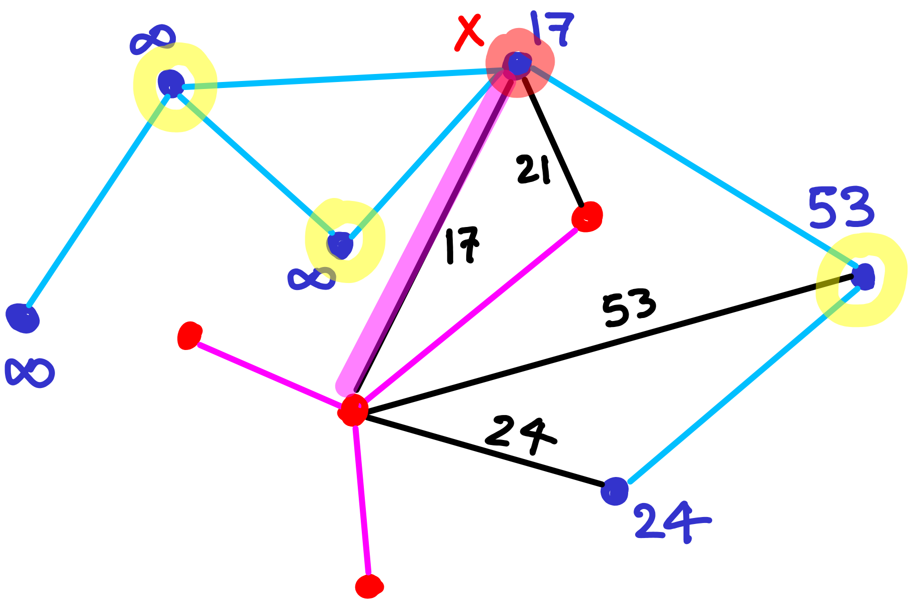


PRIM'S ALGORITHM for MST



1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty
   x: extract-min & add edge to T
   mark x → in T.

PRIM'S ALGORITHM for MST



1) start w/ any vertex s; set w(s)=0
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty
   - x: extract-min & add edge to T
   - mark x → in T.
   - for each unmarked neighbor q of x
     - if $w(q) > w(q,x)$ then decrease.

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in pr.queue

$|V|$ rounds — 3) while pr.queue not empty

x: extract-min & add edge to T

mark x → in T.

for each unmarked neighbor q of x

if $w(q) > w(q,x)$ then decrease.

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue

|V| rounds

3) while pr.queue not empty

cost for one round?

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

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in pr.queue

|V| rounds ——— 3) while pr.queue not empty

$O(\log V) + O(\text{degree}(x))$

x: extract-min & add edge to T

mark x → in T.

for each unmarked neighbor q of x

if $w(q) > w(q,x)$ then decrease.

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue

$|V|$ rounds

3) while pr.queue not empty

   x: extract-min & add edge to T

   mark $x \rightarrow$ in T.

   for each unmarked neighbor q of x

   if $w(q) > w(q,x)$ then decrease.

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right)$$

$= ?$

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

    x: extract-min & add edge to T

    mark x → in T.

    for each unmarked neighbor q of x

        if $w(q) > w(q,x)$ then decrease.

|V| rounds

$$\sum_{x \in V} \left(O(\log V) + O(\mathrm{degree}(x)) \right)$$

$$= O(V \log V) + \underbrace{O(E)}_{\text{adjacency list}}$$

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

   x: extract-min & add edge to T

   mark x → in T.

   for each unmarked neighbor q of x

   if $w(q) > w(q,x)$ then decrease.

|V| rounds

$$\sum_{x \in V} \left(O(\log V) + O(\mathrm{degree}(x)) \right)$$

$$= O(V \log V) + O(E)$$

cost ?

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in pr.queue

$|V|$ rounds

3) while pr.queue not empty

x: extract-min & add edge to T

mark x → in T.

for each unmarked neighbor q of x

if $w(q) > w(q,x)$ then decrease.

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right)$$

$$= O(V \log V) + O(E)$$

$$O(\text{degree}(x)) \cdot O(\log V)$$

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

    x: extract-min & add edge to T

    mark x → in T.

    for each unmarked neighbor q of x

        if $w(q) > w(q,x)$ then decrease.

|V| rounds

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right)$$

$$= O(V \log V) + O(E)$$

$$\sum_{x \in V} O(\text{degree}(x)) \cdot O(\log V)$$

$$= ?$$

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

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

$|V|$ rounds

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right)$$

$$= O(V \log V) + O(E)$$

$$\sum_{x \in V} O(\text{degree}(x)) \cdot O(\log V)$$

$$= O(E) \cdot O(\log V)$$

TOTAL COST ?

PRIM'S ALGORITHM for MST

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty

   x: extract-min & add edge to T

   mark x → in T.

   for each unmarked neighbor q of x

   if $w(q) > w(q,x)$ then decrease.

|V| rounds

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right) = O(V \log V) + O(E)$$

$$\sum_{x \in V} O(\text{degree}(x)) \cdot O(\log V) = O(E) \cdot O(\log V)$$

dominates

Using adjacency list

TOTAL = $O(E \log V)$

PRIM'S ALGORITHM for MST

with Fibonacci heap

(beyond scope of COMP160)

$|V|$ rounds

amortized

$$\sum_{x \in V} \left(O(\log V) + O(\text{degree}(x)) \right)$$

$$= O(V \log V) + O(E)$$

$$\sum_{x \in V} O(\text{degree}(x)) \cdot O(\cancel{\log} V)$$

$$= O(E) \cdot O(\cancel{\log} V)$$

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in pr.queue

3) while pr.queue not empty

x: extract-min & add edge to T

mark x → in T.

for each unmarked neighbor q of x

if $w(q) > w(q,x)$ then decrease.

$O(1)$ amortized

Using adjacency list

TOTAL $= O(E + V\log V)$

PRIM'S ALGORITHM for MST

What if graph is in adjacency matrix?

1) start w/ any vertex s; set $w(s)=0$
2) set $w(\neq s) = \infty$ & put all in pr.queue
3) while pr.queue not empty
   - x: extract-min & add edge to T
   - mark $x \rightarrow$ in T.
   - for each unmarked neighbor q of x
     - if $w(q) > w(q,x)$ then decrease.

PRIM'S ALGORITHM for MST

Using adj. matrix
w/ weighted entries

& no pr. queue

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in ~~pr.queue~~ array

3) while ~~pr.queue not empty~~ $\exists\, v$ not in T

scan array

- x: extract-min & add edge to T
- mark x → in T.

scan row(x) in matrix

- for each unmarked neighbor q of x
  - if $w(q) > w(q,x)$ then decrease.

PRIM'S ALGORITHM for MST

Using adj. matrix
w/ weighted entries

& no pr. queue

1) start w/ any vertex s; set $w(s)=0$

2) set $w(\neq s) = \infty$ & put all in ~~pr.queue~~ array

$|V|$ rounds 3) while ~~pr.queue not empty~~ $\exists$ v not in T

scan array $O(V)$ {
x: extract-min & add edge to T
mark x → in T.

scan row(x) in matrix $O(V)$ {
for each unmarked neighbor q of x
if $w(q) > w(q,x)$ then decrease.

$O(V^2)$ time & space