CSCE 750
11/16/2023

Prim's Algo for MST
Dijkstra's Shortest Path algo

Prim: Fix a source vertex s.

Input: Given connected, undirected graph G with weight function $w : G.E \to \mathbb{R}$

Output: An MST $T \subseteq G.E$

```
Prim(s)                    // s ∈ G.V  "the source"
Empty min-heap H
s.d := 0 ; s.π := nil
Insert(H, s)                          // d-values are the keys
for all v ∈ G.V s.t. v ≠ s, do
        v.d := ∞
        v.π := nil
        Insert(H, v)
while H not empty, do
        u := ExtractMin(H)        // s comes out first
        for each v ∈ G.V such that (u,v) ∈ G.E &
                    v ∈ H, do
            if w(u,v) < v.d then
                    v.π := u
                    v.d := w(u,v)        // actually DecreaseKey(H, v, w(u,v))
```

end foreach
end-while

return $\{(v.\pi, v) : v \in G.V \ \& \ v.\pi \neq nil\}$

Idea: Pull from H vertex whose edge has min weight for all edges from G.V − H to H

s leaves H

t leaves H

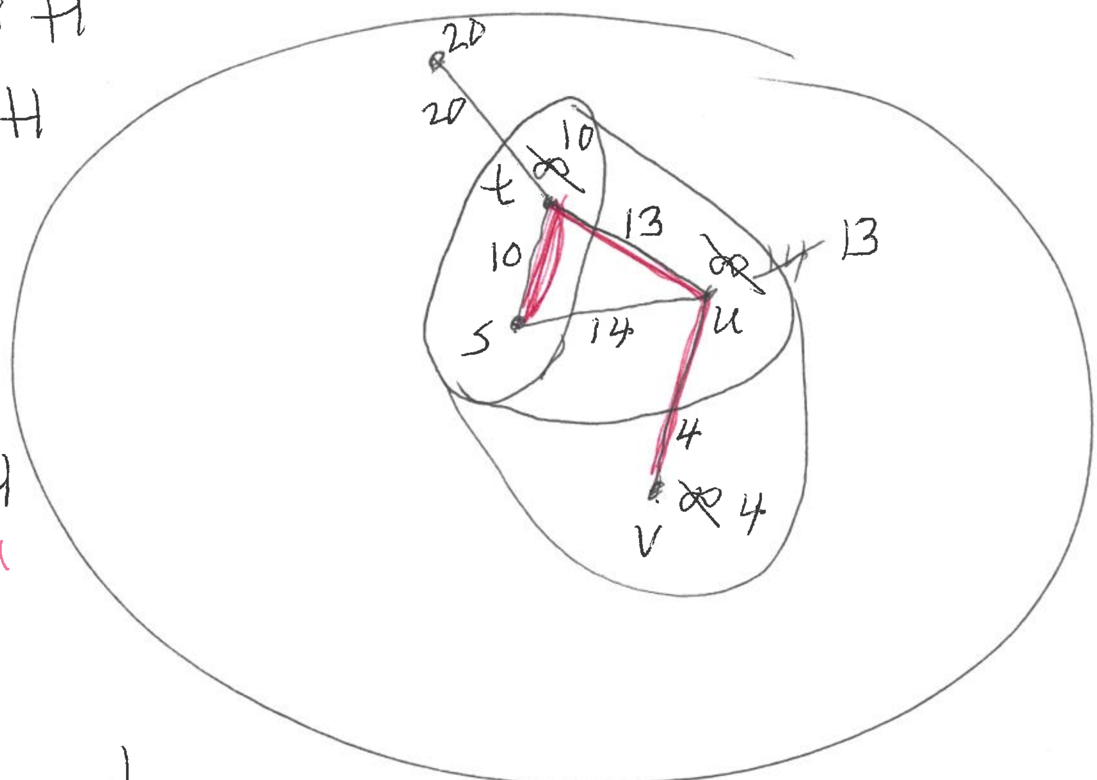
$t.\pi := s$

u leaves H

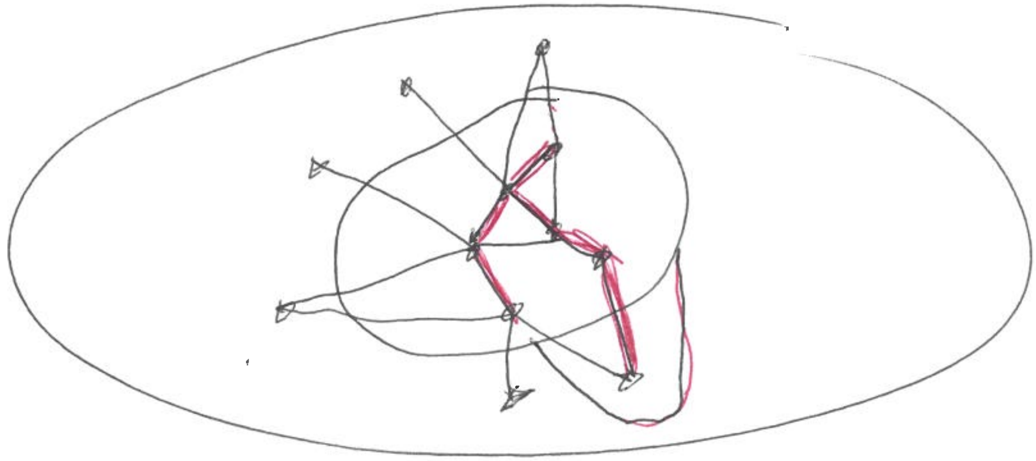
$u.\pi := t$

v leaves H

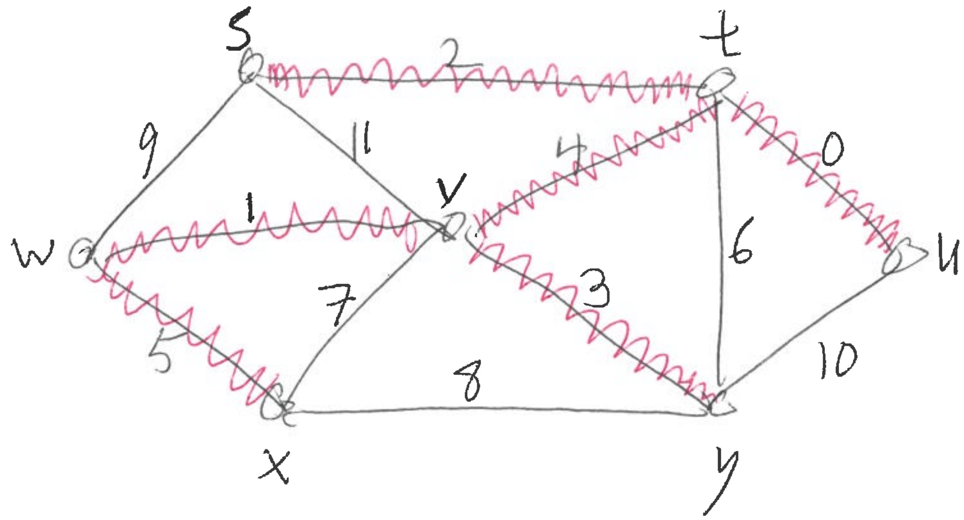
$v.\pi := u$



In general:



Concrete example;



Exercise: choose a different source.

Correctness: Building a tree (connected component) one edge at a time, starting with s.

Each step, the lightest edge from the component to outside is added to the component

[light edge crossing the cut]

Running time:

$V := |G.V|$

$E := |G.E|$

bulk of the time

- V many inserts $V \lg V$
- V many ExtractMins $V \lg V$
- E many DecreaseKey's $E \lg V$

$+ O(V+E)$ additional stuff

worst-case time

Binary or Binomial heap: $\Theta(E \lg V)$ $[E \geq V-1]$

Fibonacci Heap $\Theta(V \lg V + E)$

Dijkstra's Algo — single source shortest path algo [changes from Prim in red]

Input: G is a digraph (not necessarily connected)

Given a source vertex $s \in G.V$

$w : G.E \to \mathbb{R}^{\geq 0}$

Dijkstra(s)

Empty min-heap H

$s.d := 0$; $s.\pi := nil$

Insert(H, s) // d-values are the keys

for all $v \in G.V$ s.t. $v \neq s$, do

$v.d := \infty$

$v.\pi := nil$

Insert(H, v)

while H not empty, do

$u :=$ ExtractMin(H) // s comes out first

for each $v \in G.V$ such that $(u,v) \in G.E$ & $v \in H$, do

if $u.d + w(u,v) < v.d$ then

$v.\pi := u$

$v.d := u.d + w(u,w)$ // actually DecreaseKey

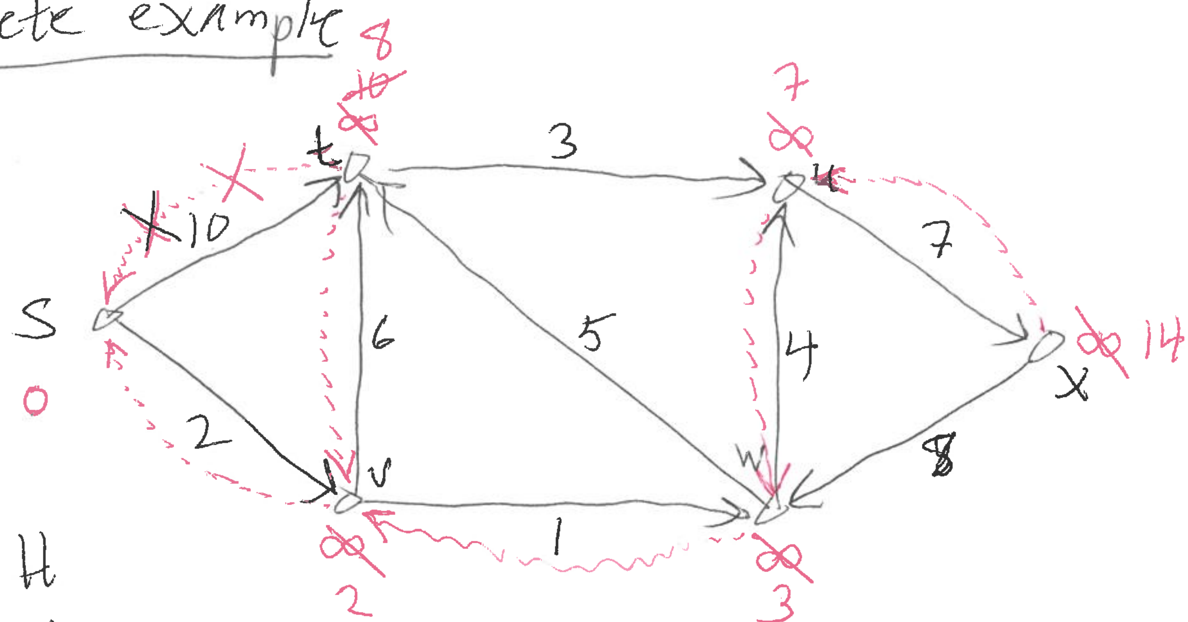
end foreach

end while

return $\{(v, v.\pi, v.d) : v \in G.V\}$

<u>Correctness</u>: <u>Claim</u>: When a vertex u is removed from H, ~~th~~ u.d = shortest length of any s ⟶ u path, and u.π is the predecessor to u along such a path.

<u>Concrete example</u>



s leaves H

v leaves H

w leaves H

u leaves H

t leaves H

x leaves H

d-value hold shortest path lengths, follow π-values ~~fro~~ from any vertex v back to s, read edges in reverse gives a shortest path.