PRIM-DIJKSTRA’S MST ALGORITHM

Input: $G = (V, E, W)$
Output: $T = (V_T, E_T)$

$E_T = \emptyset$

Select a vertex $v \in V$ and move $v$ to $V_T: V_T = v, V = V - v$
For $i = 1$ to $n - 1$ do

Let $\{v, w\}$ be an edge such that $v \in V_T$, $w \in V_1$, and for all such edges, $\{v, w\}$ has the minimum weight.

\[
V_T = V_T \cup \{w\} \\
E_T = E_T \cup \{\{v, w\}\}
\]

\[
V = v - \{w\}
\]

endfor
\[ w(n) = O(n^3) \]

**Theorem:** Let \( G = (V, E, w) \) be a weighted connected graph and \( T = (V, E_T) \) be a MST of \( G \). Let \( T' = (V', E') \) is a subtree of \( T \). If \( \{x, y\} \) is the minimum weight edge such that \( x \in V' \) and \( y \in V - V' \) then \( T'' = (V' \cup \{y\}, E' \cup \{\{x, y\}\}) \) is a subtree of a MST of \( G \).

**Proof:**

1. If \( \{x, y\} \in E_T \), done
2. Let \( \{x, y\} \notin E_T \)
   
   Then \( E_T \cup \{\{x, y\}\} \) has a cycle:

By the choice of \( \{x, y\} \)

\[ W(\{x, y\}) \leq w(\{v, w\}) \]

Consider \( E_T \cup \{\{x, y\}\} - \{\{v, w\}\} \)

Its weight is no more than the weight of \( T \) and it is a spanning tree.

\[ \Rightarrow E' \cup \{\{x, y\}\} \text{ is a subtree of a MST of } G \]
2.1 Data Structure for Prim’s Algorithm

\[ NEAR[1 \ldots n] \]
\[ NEAR[i] = 0 \text{ if } i \in V_T \]
else \( = j \) if \( w(i, j) \) is minimum among all \( j \in V_T \)

\[ NEAR[i] = j \]
\[ NEAR[j] = 0 \]
\[ NEAR[l] = 0 \]
2.2 Modified Prim’s Algorithm

**Input:** $G = (V, E, w)$, a connected weighted graph

**Output:** $T = (V_T, E_T)$, a MST

1. $E_T = \phi$

2. $NEAR[1] = 0$ /* $v_T = \{1\} */

   $NEAR[2 \ldots n] = 1$ /* $V = V - \{1\} */

3. For $i = 1$ to $n - 1$ do

   Find $j$ such that $NEAR[j] \neq 0$ and $w(j, NEAR[j])$ is min.

   $NEAR[j] = 0$ /* $V_T = V_T \cup \{j\} */$

   $E_T = E_T \cup \{\{j, NEAR[j]\}\}$

   /* update $NEAR[j] = 0$ /* $V_T = V_T\{j\} */$

   for $k = 1$ to $n$ do
if $\text{NEAR}[k] \neq 0$ AND $w(k, \text{NEAR}(k)) > w(k, j)$
then $\text{NEAR}[k] = j$

endfor

4. endfor

$W(n) = O(n^2)$
3 KRUSKAL’S ALGORITHM

Input: $G = (V, E, W)$, a connected graph

Output: $T = (V, E_T)$, a MST of $G$

$T \leftarrow \phi$

While $|T| < n - 1$ do

Let $\{v, w\}$ be the least cost edge in $E$

$E \leftarrow E - \{\{v, w\}\}$

if $\{v, w\}$ does not create a cycle in $T$

then add $\{v, w\}$ to $T$

end while
3.1 Kruskal’s Algorithm (Refined)

**Operations:**

1. UNION($i, j$), of sets $i$ and $j$, contains elements of sets $i$ and $j$

2. FIND($v$) = $i$ iff $v \in \text{set } i$

a) Construct a min-heap $E$ of edges in $E$

b) Each $v \in V$ forms singleton set by itself, such that FIND($v$) = $v$.

c) $E_T = \emptyset \{$Tree is empty$\}$

d) while $|E_T| < n - 1$ do

- Delete the root edge $\{v, w\}$ from min-heap $E$ and restore heap $E$
if \text{FIND}(v) \neq \text{FIND}(w) \quad / \ast \quad \{E_T \cup \{\{v, w\}\}\} \text{ has no cycle} / \\

\text{then } E_T = E_T \cup \{\{v, w\}\} \\
\quad / \ast \text{now, combine the components of } E_T \text{ joined by } \{v, w\} \text{ into one} / \\

\text{UNION ( \text{FIND}(v), \text{FIND}(w))} \\

\text{end if} \\

\text{end while}
Let $T$ be the spanning tree for $G$ generated by Kruskal’s algorithm. Let $T'$ be a minimum cost spanning tree for $G$. Show that both $T$ and $T'$ have the same cost.

**Proof:** If edges of $T$ and $T'$ are the same, we are done.

Let $E(T) \neq E(T')$.

Let $e$ be a minimum cost edge such that $e \in E(T)$ and $e \notin E(T')$.

i.e. $e$ is minimum cost edge in $E(T) - E(T')$.

Inclusion of $e$ in $T'$ creates a cycle. Let $e e_1 e_2 \cdots e_k$ be this cycle.

At least one of $e_1$ through $e_k$ is not in $T$ else $e$ will form cycle in $T$ as well.

Let that edge be $e_j \notin E(T)$.

If $w(e_j) < w(e)$ then $e_j$ would be selected by Kruskal’s al-
algorithm first and included into $T$. But $e_j \notin E(T)$.
$\Rightarrow w(e_j) \geq w(e)$.

(because of choice of $e$, $e_j$ would have been considered for inclusion in a subset $T^*$ of $T \cap T'$. Hence, $e_j$ would not form a cycle in $T^*$ because $T^* \subseteq T'$.)

Consider $T'' = E(T') \cup \{e\} - \{e_j\}$

Here the cycle created by $E(T') \cup \{e\}$ is broken by $\{e_j\}$

Cost of $T''$ is no more than that of $T'$ which does not have $e$ in it.

Repeat this process until all such $e'$s in $T$ not in $T''$ to eventually convert $T'$ into $T$ without increasing cost.

$\Rightarrow T$ is a minimum spanning tree.

Thus greedy strategy which is locally optimal decision leads to a global optimal solution for spanning tree problem.
3.3 Weighted Union

**Lemma:** With W-UNION, any tree that has \( k \) nodes has depth at most \( \lceil \log k \rceil \).

Let \( k_1 \geq k_2, k = k_1 + k_2 \)
\[
d = d_1, \text{ if } d_1 > d_2
\]
\[
= d_2 + 1, \text{ if } d_1 \leq d_2
\]

**Basis:** \( k = 1 \quad \lceil \log 1 \rceil = 0 \)

**Hypothesis:** for \( k' < k \), depth \( \leq \lceil \log k' \rceil \)

**Induction:**

I) \( d = d_1 \leq \lceil \log(k_1) \rceil \leq \lceil \log(k_1 + k_2) \rceil = \lceil \log k \rceil \)

II) \( d = d_2 + 1 \leq \lceil \log k_2 \rceil + 1 = \lceil \log 2k_2 \rceil \leq \lceil \log(k_1 + k_2) \rceil \)
Lemma: A UNION-FIND PROGRAM of size $n$ does $\Theta(n \log n)$ link operations in the worst case if the weighted union is used.
3.4 Compressed Find

**Lemma:** The number of link operations done by a UNION-FIND program of length $n$ implemented with W-UNION and C-FIND is $O(n G(n))$.

\[
G(n) = \log^* n \\
= \text{SMALLEST } i \text{ SUCH THAT } \log^i n \leq 1
\]

where $\log^i n = \log(\log^{i-1} n)$
and $\log^0 n = n$

$G(65536) = \log^*(2^{16}) = 4$
$\log 2^{16} = 16$
$\log 16 = 4$
$\log 4 = 2, \log 2 = 1$

$G(2^{65536}) = 5, \Rightarrow G(n) \leq 5, \text{ for all reasonable } n.$
4 BELLMAN-FORD ALGORITHM

1. For each $v \in V$
   
   do $d[v] \rightarrow \infty$

2. for $i \rightarrow 1$ to $|V| - 1$
   
   do for each edge $(u, v) \in E$
      
   do if $d[v] > d[u] + w(u, v)$
      
      then $d[v] \rightarrow d[u] + w(u, v)$

3. for each edge $(u, v) \in E$
   
   do if $d[v] > d[u] + w(u, v)$
      
      then no solution
Lemma: \(d[v] \geq (s, v)\) always.
First part of Lemma 25.5, section 25.1

Initially true

Let \(v\) be first vertex for which \(d[v] < (s, v)\), and let \(u\) be vertex that cause \(d[v]\) to change:

\[
d[v] = d[u] + w(u, v)
\]

Then

\[
d[v] < (s, v)
\]

\[
\leq (s, u) + w(u, v) \text{ (Triangle inequality)}
\]

\[
\leq d[u] + w(u, v) \text{ (} v \text{ is first violation)}
\]

contradicts \(d[v] = d[u] + w(u, v) \text{ (above)}\).
5 DIJKSTRA’s ALGORITHM

DIJKSTRA(G)

for each $v \in V$
do $d[v] \to$
d[$s$] $\to 0$
$S \to 0$
$Q \to V$
while $Q \neq 0$
do $u \to \text{EXTRACT-MIN}(Q)$
  $S \to S \cup \{u\}$
for each $v \in \text{Adj}[u]$
do if $d[v] > d[u] + w(u, v)$
  then $d[v] \to d[u] + w(u, v)$

Correctness: Prove that whenever $u$ is added to $S$, $d[u] = (s, u)$
Theorem 25.10
Proof: Basically same as in book, but derives a different contradiction.
Figure 25.6
Note that $s[v] \geq (s, v)$
(because lemma proved for Bellman-Ford above was just about relaxation, didn’t depend on order of relaxing edges)
Let $u$ be first vertex picked such that shorter path than $d[u] \leftarrow d[u] > (s, u)$
Let $y$ be first vertex $\in V - S$ on actual shortest path from $s$ to $u \leftarrow d[y] = (s, y)$
Because:

d\[x\] is set correctly for y’s predecessor \(x \in S\) on the shortest path (by choice of \(u\) as first choice for which that’s not true)

when put \(x\) into \(S\), relaxed \((x, y)\), giving \(d[y]\) correct value,

\[
d[u] > (s, u)
\]

\[
= (s, y) + (y, u) \quad \text{(optimal substructure)}
\]

\[
= d[y] + (y, u)
\]

\[
\geq d[y] \quad \text{(no negative weights)}
\]

But \(d[u] > d[y]\) \iff algorithm would have chosen \(y\) to process next, not \(u\).
Contradiction.
1. UNION (1,2)
2. UNION (2,3)

1. UNION (N/2, N/2+1)

n/2. UNION (N/2, N/2+1)
n/2+1. FIND (1)

FIND (1)
. 
. 
n. FIND (1)

TREE WITH UNWEIGHTED UNION

TREE WITH WEIGHTED UNION
Before C-FIND (v)

COMPRESSING -FIND(v)

AFTER C-FIND(v)