Steiner Tree

Simran Gangwani
(M190359CA)

What is a Steiner Tree ?

-Jakob Steiner

Given a graph and a subset of vertices in a graph, a Steiner Tree spans through the given subset. The Steiner Tree may include some vertices other than in the given subset, so as to connect the vertices of subset.

The given subset is called Terminal Vertices and the other vertices that are used to construct Steiner Tree are called Steiner Vertices.

Source: https://www.geeksforgeeks.org/steiner-tree/

Application

- Steiner trees arise in network design problems, because the mini- mum Steiner tree describes how to connect a given set of sites using the smallest amount of wire.
- Typical Steiner tree problems in electronic circuit design require connecting a set of sites to (say) ground under constraints such as material cost and signal propagation delay.
- VLSI
- Perfect phylogeny in bioinformatics

Optimisation Version

Instance : Given an undirected graph G = (V,E) with non negative edge weights and a subset of vertices T.

Question : Find the smallest tree connecting all the vertices of T(minimum weight Steiner Tree)

Classical Complexity: NP- Hard

Decision Version

* **Instance:** Given an undirected graph G = (V,E) with non negative edge weights, a subset of vertices T, and a natural number K.

* **Question:** Does there exist a Steiner Tree spanning T that contains at most K edges.

Classical Complexity: NP- Complete

The problem is well known under Karp's 21 NP Complete Problem.

NP Completeness

To prove that Steiner Tree(ST) is NP Complete, we need to show the following:

1. ST is in NP.
2. Select any existing NP Complete problem(P) such that:
   - $R(I_p)$ Transform instance of P into instance of Steiner Tree.
   - I_P is an YES instance of P iff $R(I_P)$ is an YES instance of ST.

Resources: Steiner Tree NP-completeness Proof by Alessandro Santuari May 7, 2003

.

(i) STEINER TREE IS IN NP

Assuming <G,T,k> is a YES instance, given an hypothetic positive solution $H \subseteq G$

We need to check following:

1. H is really a tree: it contains no cycles and it is connected.
2. H touches all the terminal vertices in set T.
3. The number of edges in H does not exceed k.

(ii) SELECT EXISTING NP COMPLETE PROBLEM

Appropriate Problem : Exact Cover by 3 Sets(X3C)

Exact Cover Problem Statement

Input: a family S, subset of a set X.

Question: Does there exist a sub family S* of S such that each element in X is contained in exactly one subset in S*.

(ii) SELECT EXISTING NP COMPLETE PROBLEM

EXAMPLE OF EXACT COVER

Let $X = \{a, b, c, d\}$ and $S = \{X_1, X_2, X_3, X_4\}$ where,

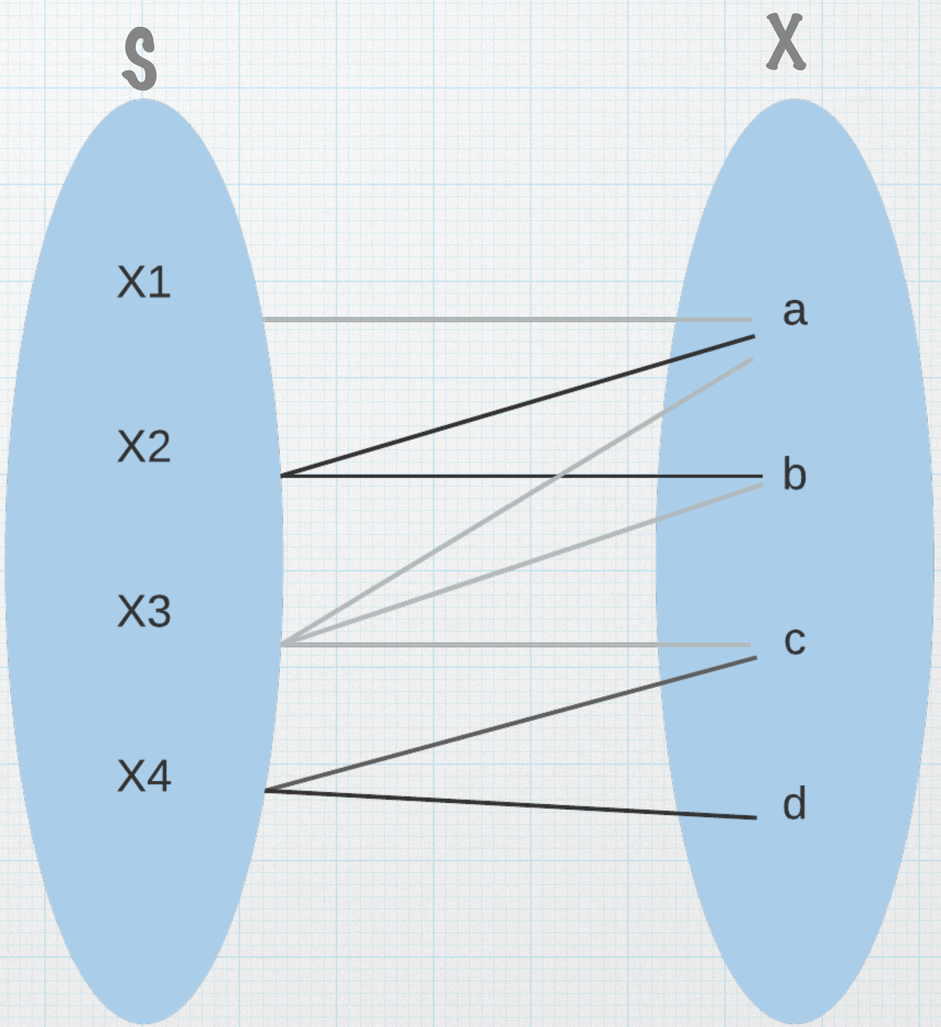
$X_1 = \{a\}$

$X_2 = \{a, b\}$

$X_3 = \{a, b, c\}$

$X_4 = \{c, d\}$

Then $S^* = \{X_2, X_4\}$



(ii) SELECT EXISTING NP COMPLETE PROBLEM

Exact cover by 3 sets

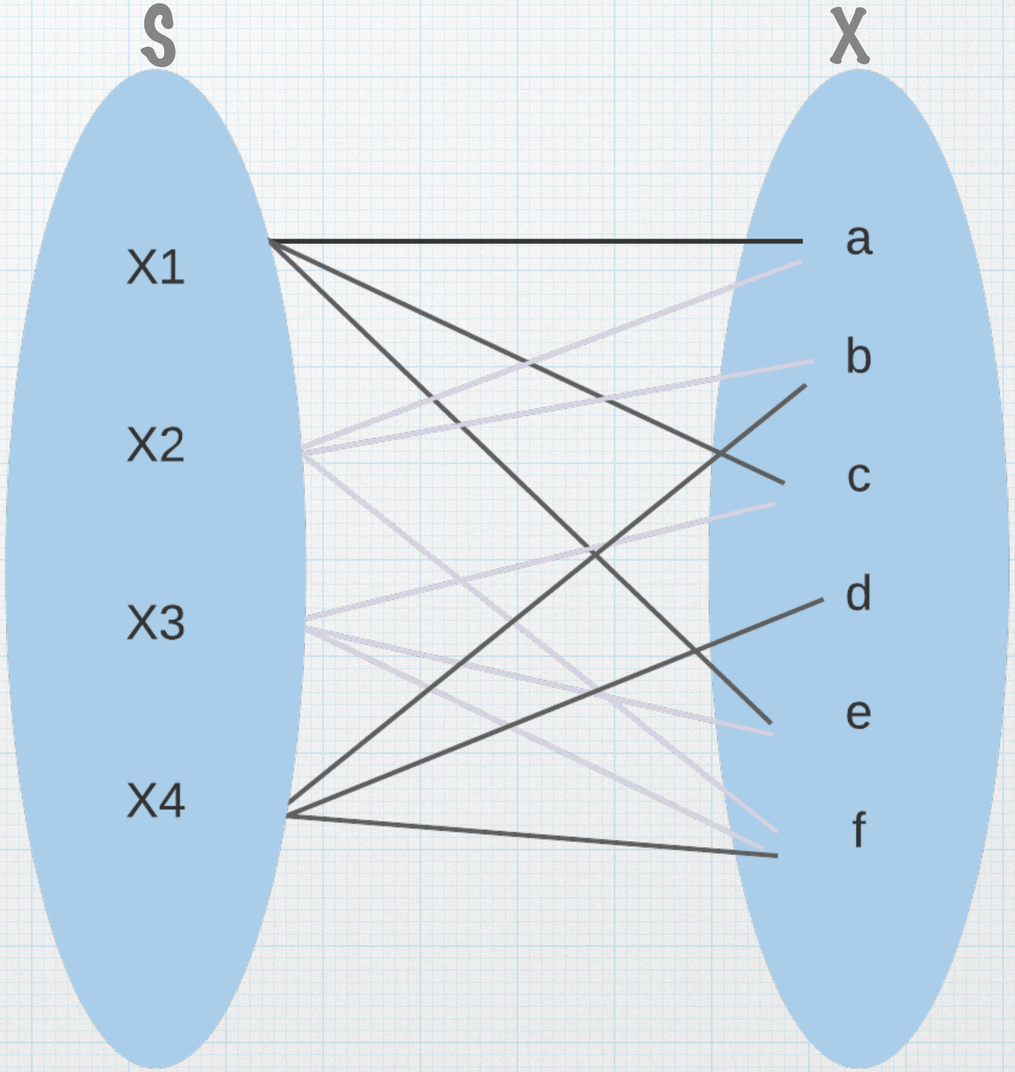
Input: Given a set X, with |X| = 3q and S of 3-element subsets of X.
Question: Does there exist a sub family S* of S such that each element in X is contained in exactly one subset in S*.

(ii) SELECT EXISTING NP COMPLETE PROBLEM

EXAMPLE OF EXACT COVER by 3 Set

Let $X = \{a, b, c, d, e, f\}$, $|X| = 3x2$
and $S = \{X_1, X_2, X_3, X_4\}$ where,
$X_1 = \{a, c, e\}$
$X_2 = \{a, b, f\}$
$X_3 = \{c, e, f\}$
$X_4 = \{b, d, f\}$
Then $S^* = \{X_1, X_4\}$

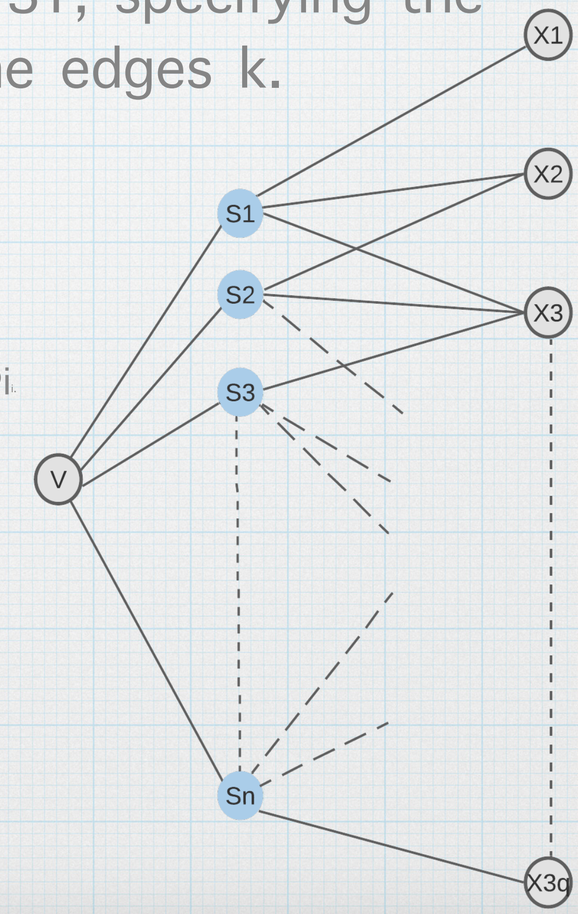
$|S^*| = q$



(ii.a) TRANSFORM INSTANCE OF X3C TO INSTANCE OF ST

Given an instance of X3C, defined by the set $X = \{x_1, x_2, x_3 \ldots, x_{3q}\}$ and a collection of 3 element sets $S = \{S_1, S_2, S_3, \ldots, S_n\}$, we have to build the instance of ST, specifying the graph $G = (V,E)$, the set of terminals T, and the upper bound on the edges k.

- $V(G) = \{v\} \cup \{S_1, S_2, S_3, \ldots, S_n\} \cup \{x_1, x_2, x_3 \ldots, x_{3q}\}$
- $E(G) = \{vS_1, vS_2, vS_3, \ldots, vS_n\} \cup (\cup \{S_i x_j\})$, where x_j belongs to S_i
- $T = \{v, x_1, x_2, x_3 \ldots, x_{3q}\}$
- $K = 4q$



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.

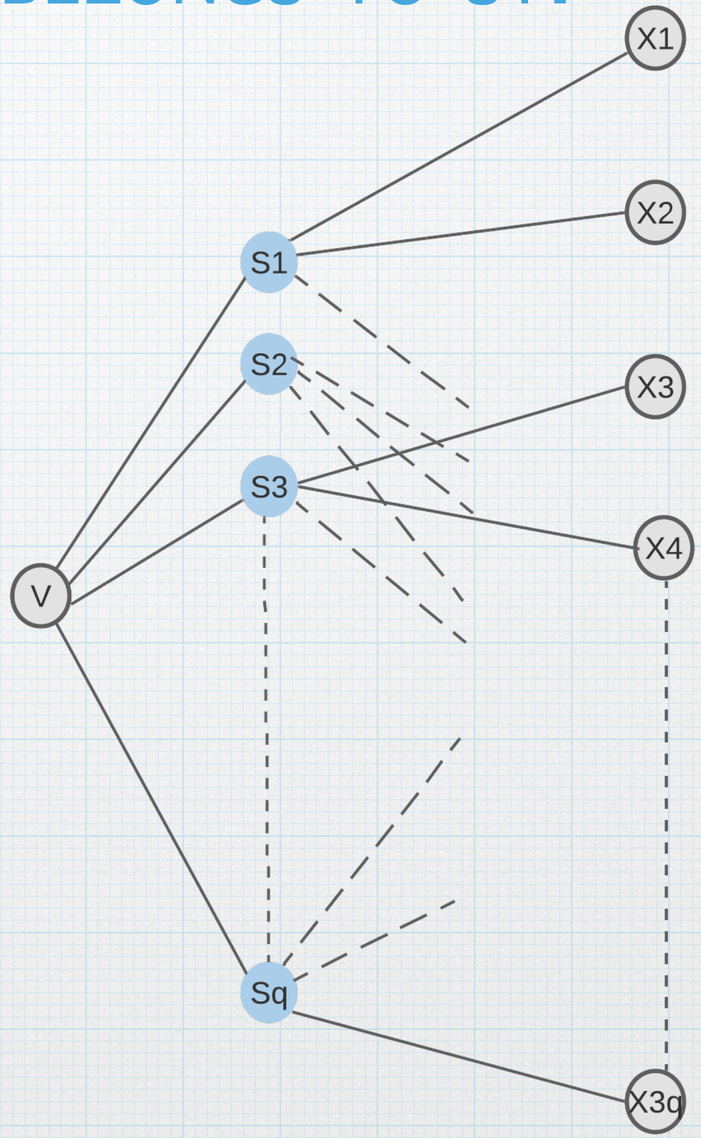
X3C => ST

If there exist an exact 3-cover, then $|S^*| = q$.

Let $S^* = \{S_1, S_2, S_3, \ldots, S_q\}$

We will get the tree consisting edges,

- vS_1, vS_2, $vS_3 \cdots$, vS_q
- $S_i x_j$, if $x_j \in S_i$ and $1 \leq i \leq q$



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.

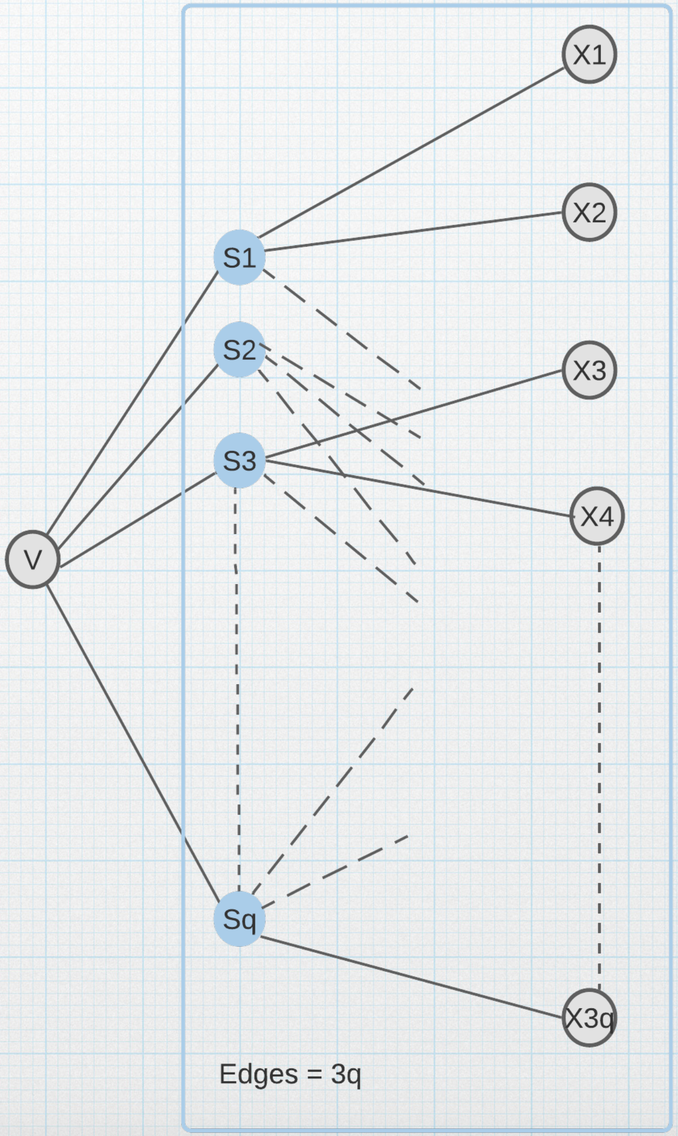
X3C => ST

If there exist an exact 3-cover, then $|S^*| = q$.

Let $S^* = \{S_1, S_2, S_3, \ldots, S_q\}$

We will get the tree consisting edges,

- $vS_1, vS_2, vS_3 \cdots, vS_q$
- $S_i x_j$, if $x_j \in S_i$ and $1 \leq i \leq q$



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.

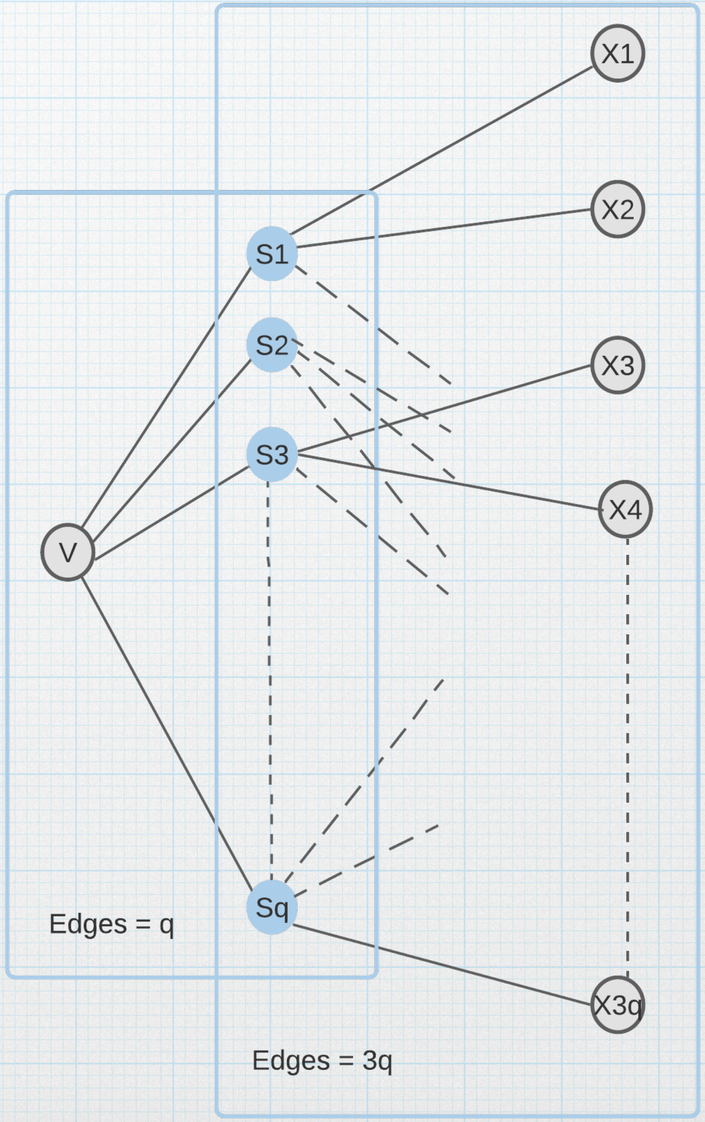
X3C => ST

If there exist an exact 3-cover, then $|S^*| = q$.

Let $S^* = \{S_1, S_2, S_3, \ldots, S_q\}$

We will get the tree consisting edges,

- $vS_1, vS_2, vS_3 \cdots, vS_q$
- $S_i x_j$, if $x_j \in S_i$ and $1 \leq i \leq q$
- Total number of edges => $3q + q$ => $4q = K$



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.

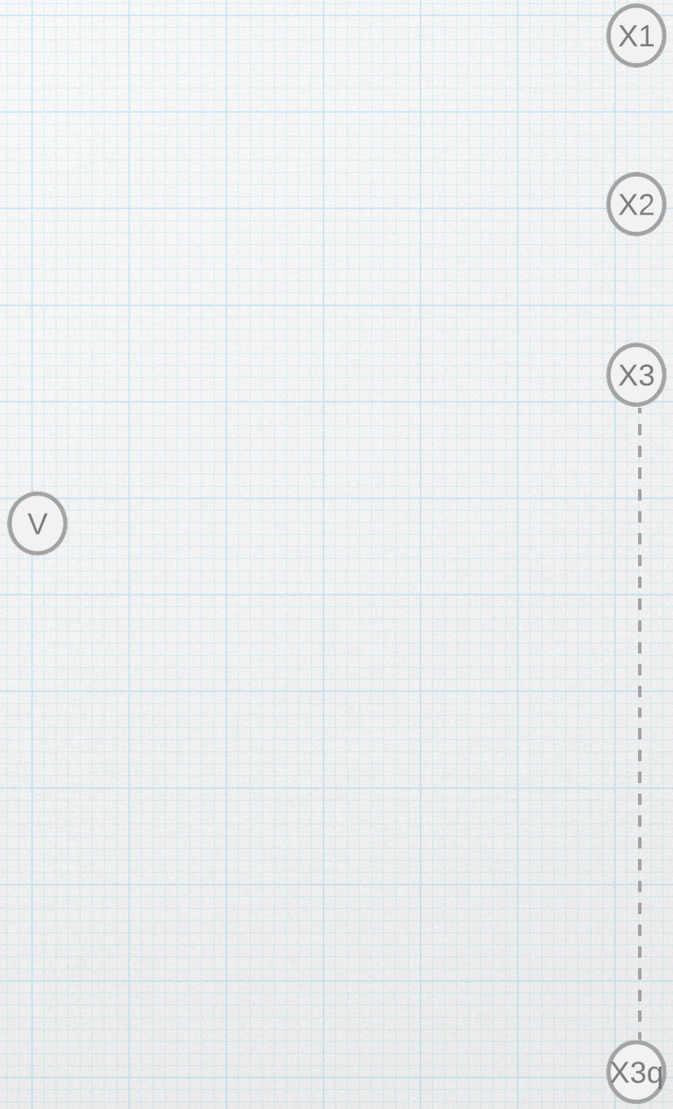
X3C <= ST

If there exist a Steiner Tree , then k = 4q.

|E| <= 4q, |v| = 4q + 1

ST should touch all the terminal vertices

i.e v + $\{x_1, x_2, x_3 \ldots, x_{3q}\}$ = 3q + 1



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.

X3C <= ST

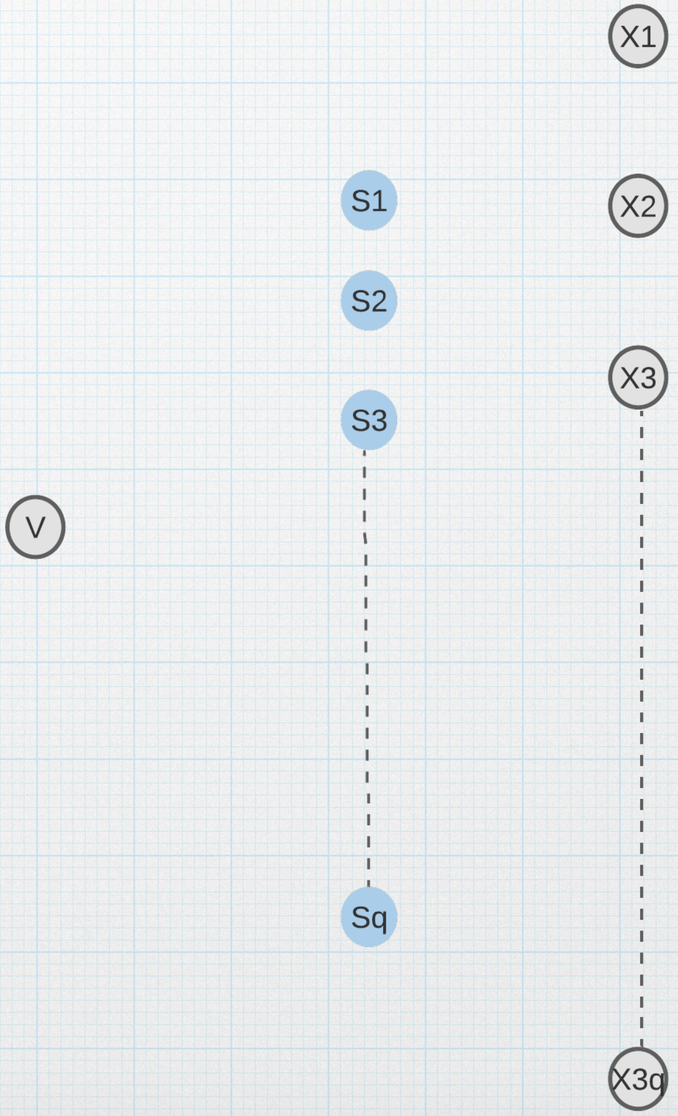
If there exist a Steiner Tree , then k = 4q.

|E| <= 4q, |v| = 4q + 1

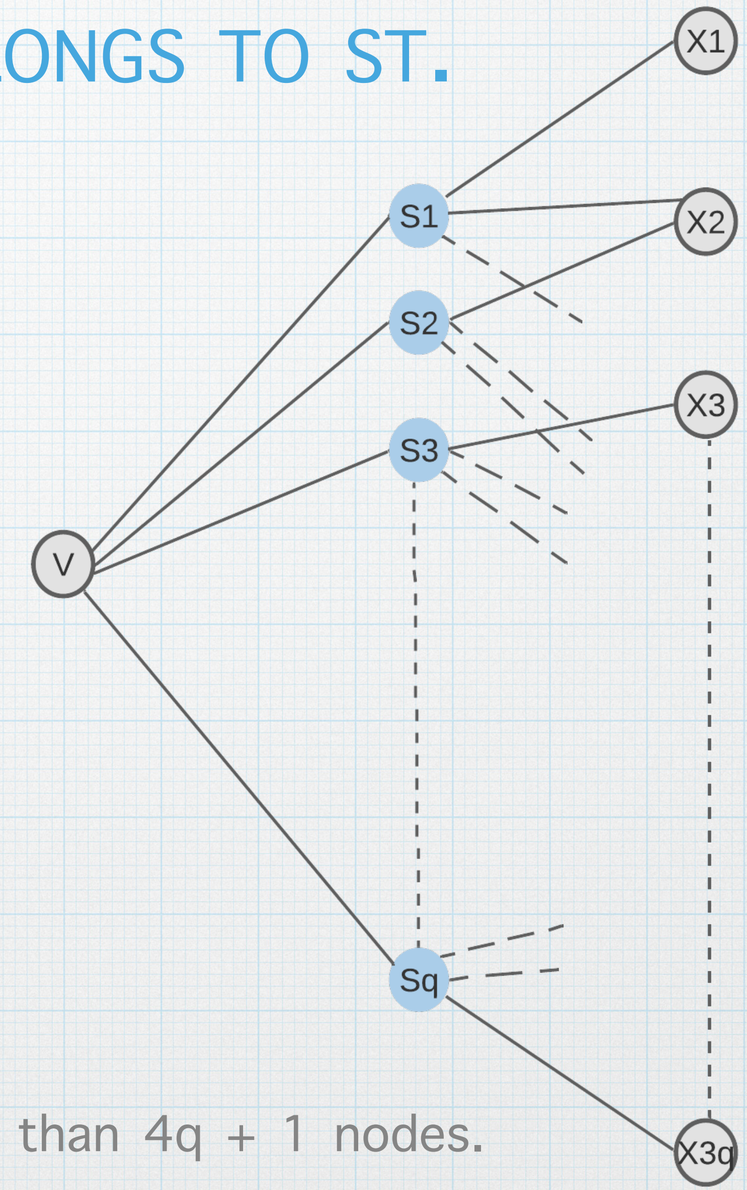
ST should touch all the terminal vertices

i.e v + $\{x_1, x_2, x_3 \ldots, x_{3q}\}$ = 3q + 1

Steiner Vertices <= q.



(ii.b)⟨X, S⟩ BELONGS TO X3C IFF ⟨G, T, k⟩ BELONGS TO ST.



X3C <= ST

If there exist a Steiner Tree , then k = 4q.

|E| <= 4q, |v| = 4q + 1

ST should touch all the terminal vertices

i.e $v + \{x_1, x_2, x_3 \ldots, x_{3q}\} = 3q + 1$

Steiner Vertices <= q.

It is impossible to hit all the 3q x-nodes if the tree contains less than 4q + 1 nodes.

$S^* = \{S_1, S_2, S_3, \ldots, S_q\}$

FPT Version

Instance : Given an undirected graph G(V,E), subset of vertices T.

Parameter : K = |T|

Question : A tree, subgraph of G, spanning T, which minimizes the sum of its edge-weights.

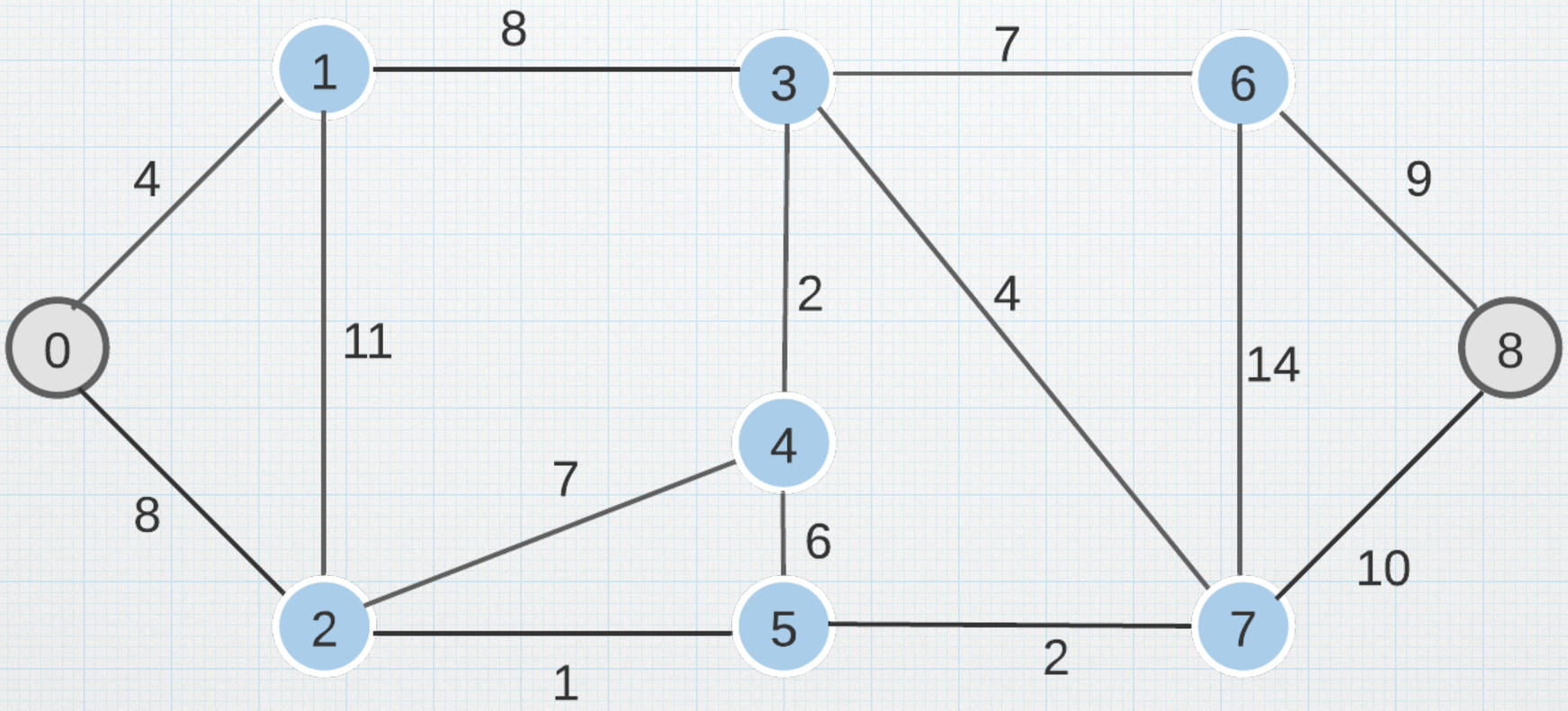
Classical Complexity: $O^*(3^k \cdot n^{O(1)})$

Special Cases

1. The Steiner tree of a pair of vertices is simply the shortest path between them.

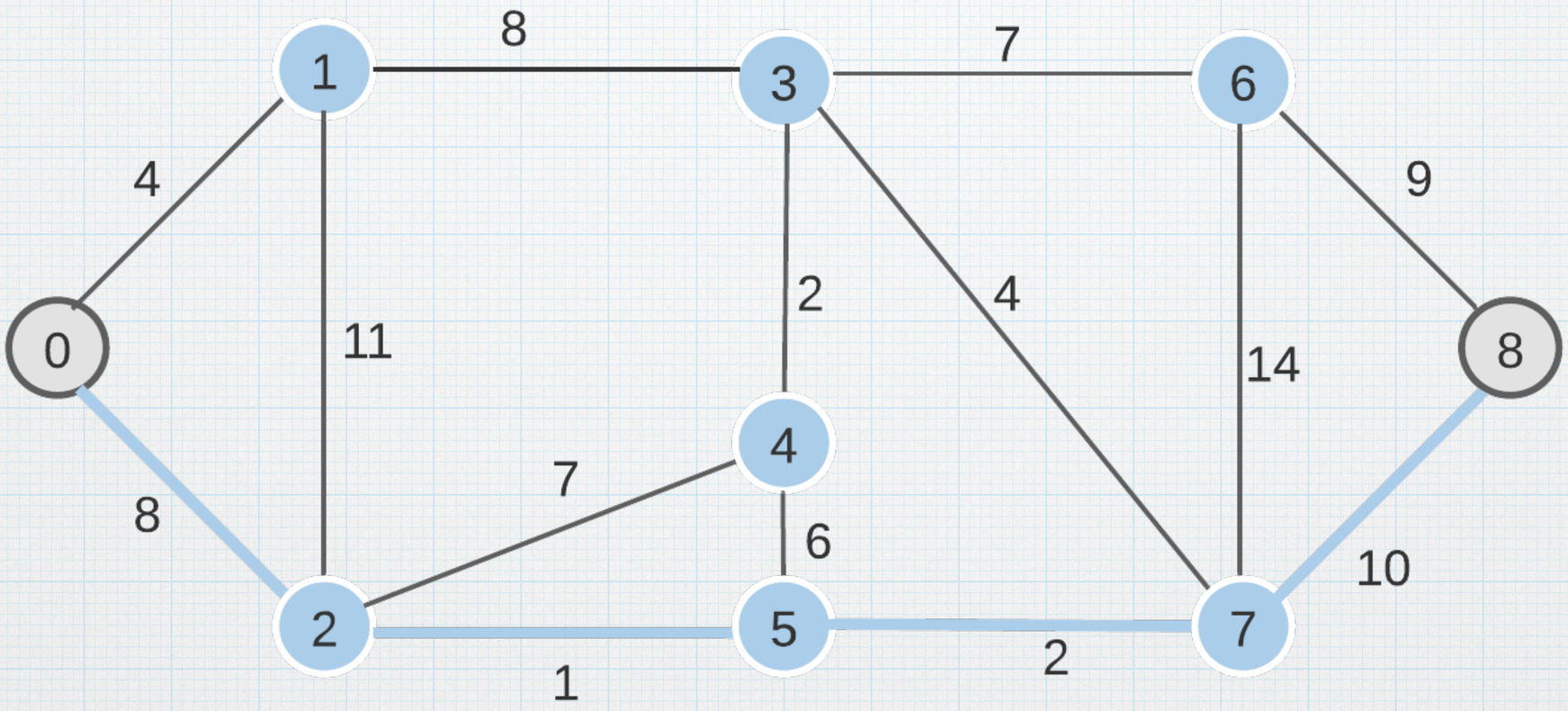
2. The Steiner tree of all the vertices, when $T = V$, simply defines the minimum spanning tree of G.

(I)SPECIAL CASE, |T| = 2



Terminal Vertices = { 0, 8}

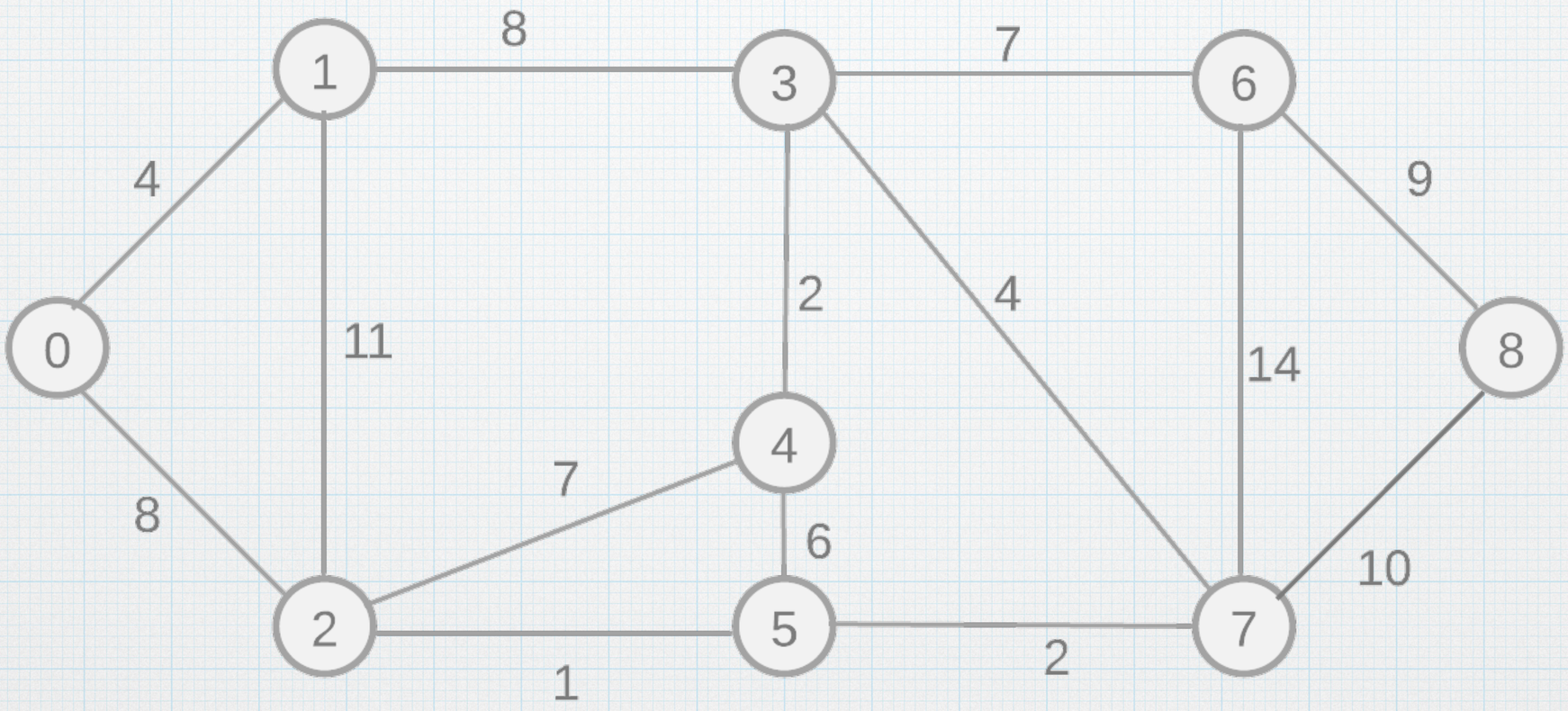
(I)SPECIAL CASE, |T| = 2



Terminal vertices = {0, 8} Steiner vertices = {2, 5, 7}

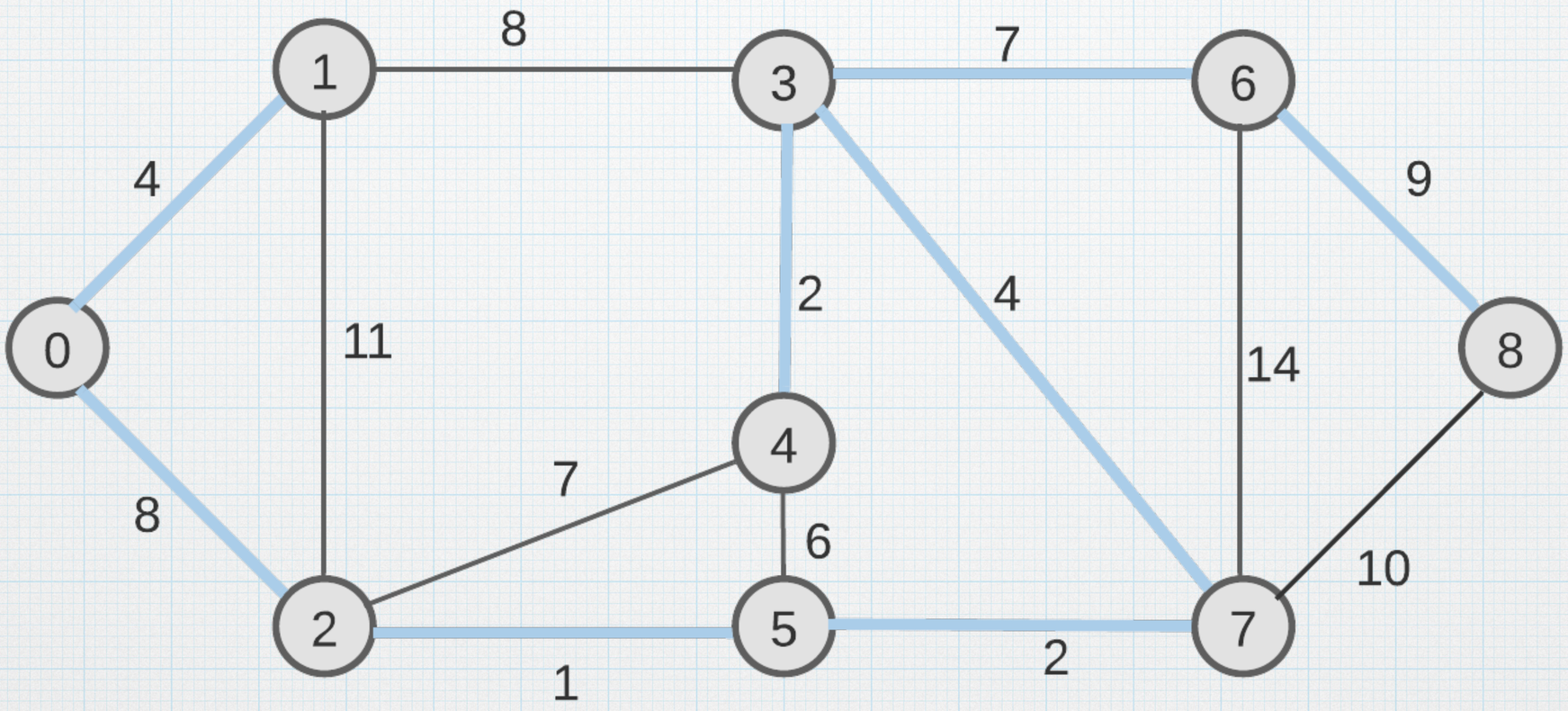
Minimum cost of Steiner Tree = 21

(II)SPECIAL CASE, |T| = |V|



Terminal Vertices = { 0, 1, 2, 3, 4, 5, 6, 7, 8}

(II)SPECIAL CASE, |T| = |V|



Terminal vertices = {0, 1, 2, 3, 4, 5, 6, 7, 8} Steiner vertices = {}

Minimum cost of Steiner Tree = 37

Dynamic Programming for Minimum Steiner Trees

Proposed By: Dreyfus and Wagner

Resources Used:

1. New Developments in Exact Algorithms and Lower Bounds, IIT Delhi, 13th December 2014. Exact Algorithm for Steiner Tree by Ondra Such´y
2. Marek Cygan, Fedor V. Fomin, ukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michaa Pilipczuk and Saket Saurabh Parameterized Algorithms.

Assumptions

- $|T| > 1$
- G is connected : A Steiner Tree for T exists in G only if all terminal vertices belong to same connected component of G.
- No two terminal nodes are adjacent to each other, and degree of each terminal vertices is one.
- all the leaves in the Steiner tree must be terminals. Otherwise, one could simply delete the non-terminal leaves, yielding a feasible solution with less cost.

Problem Statement + Notation

Instances

1. Given an undirected graph G = (V, E)
2. Edge cost w : E -> R+
3. A subset T of V, and K = |T|

Question

Obtain a Tree H connecting all terminals such that H is a subset of E.

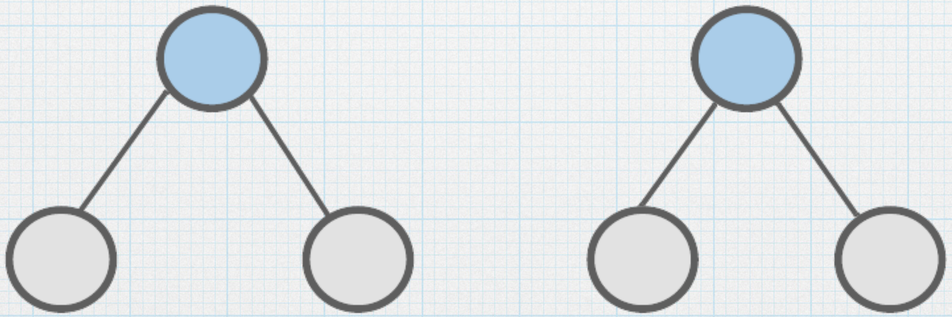
So an optimal Steiner Tree for T is a tree H = H(T) that minimises w(H) subject to T is a subset of V(H).

D is a subset of T.

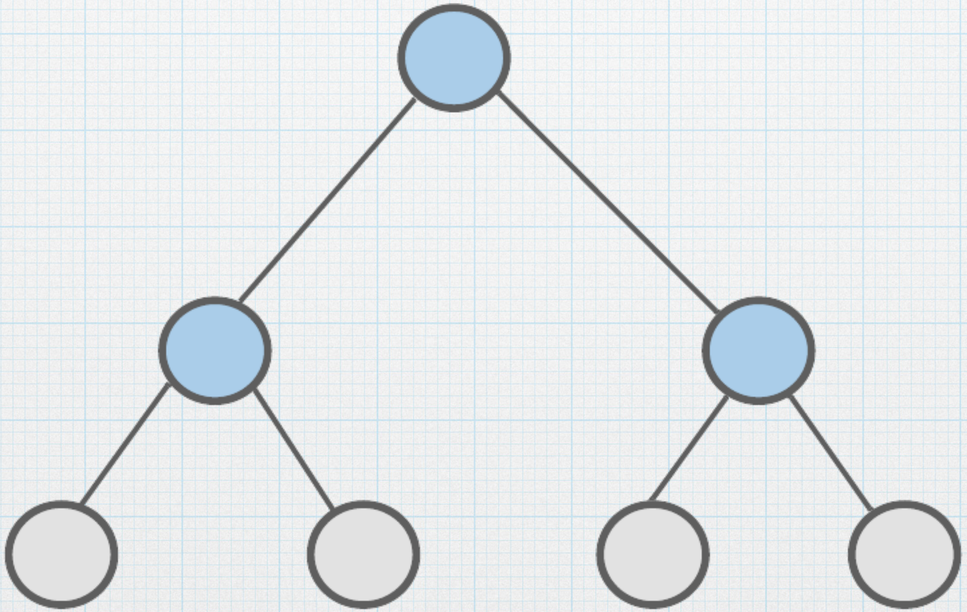
D' is a subset of D

Idea

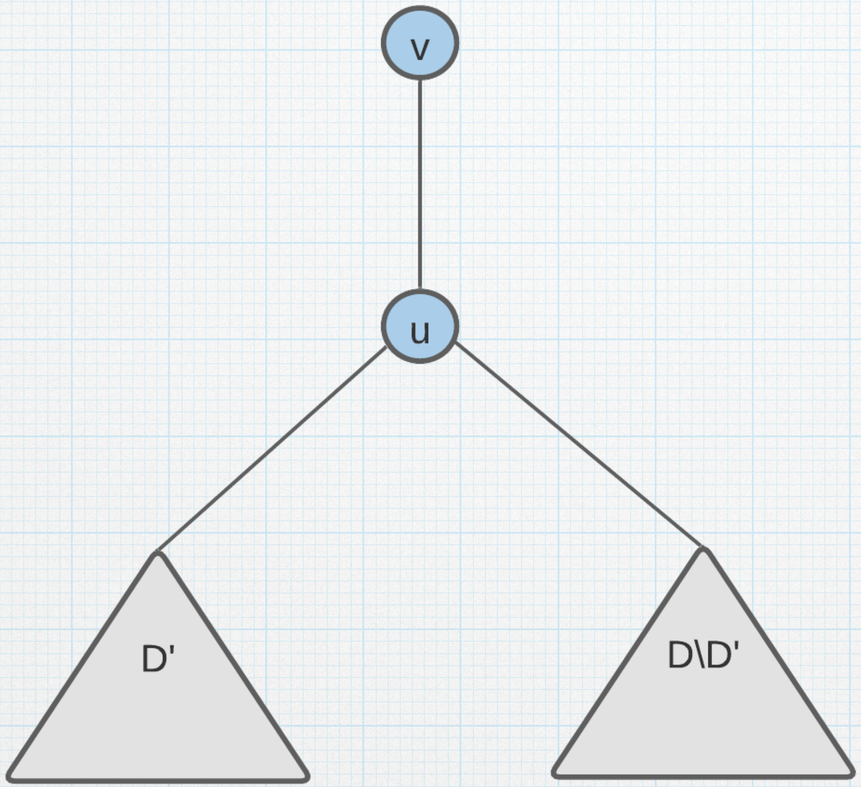
Idea



Idea



Idea



For every non empty subset D of T and every vertex v belongs to V(G)\ T, consider the possibility that in the optimal Steiner Tree H for T, there is a subtree of H that contains D and is connected to rest of the tree H through the vertex v. For |D| > 1, such a subtree decomposes into two smaller subtrees rooted at some vertex u(possibly u = v), and a shortest path between u and v.

Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

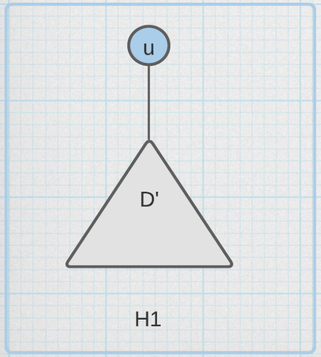
$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

Where D' is a non empty subset of D and $u \in V(G)\backslash T$

==>

Suppose we have $u \in V(G)\backslash T$ and D' as a non empty subset of D.

Let H1 be the Steiner Tree for $(D' \cup \{u\})$ in G.



Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

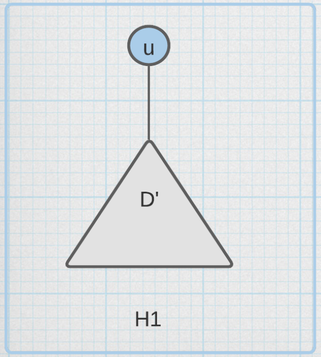
Where D' is a non empty subset of D and $u \in V(G)\backslash T$

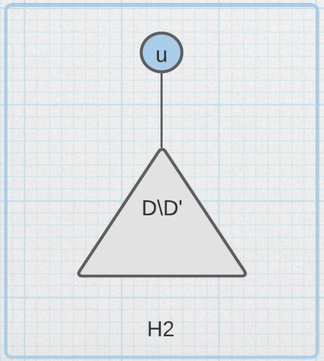
==>

Suppose we have $u \in V(G)\backslash T$ and D' as a non empty subset of D.

Let H1 be the Steiner Tree for $(D' \cup \{u\})$ in G.

Let H2 be the Steiner Tree for $(D\backslash D' \cup \{u\})$ in G.





Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

Where D' is a non empty subset of D and $u \in V(G)\backslash T$

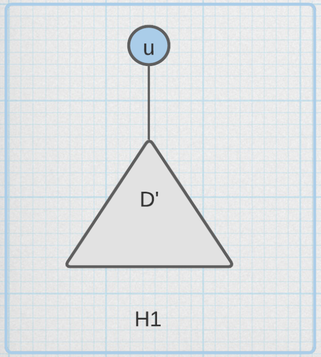
==>

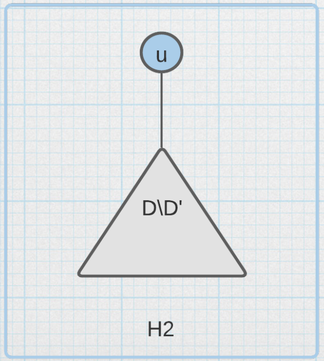
Suppose we have $u \in V(G)\backslash T$ and D' as a non empty subset of D.

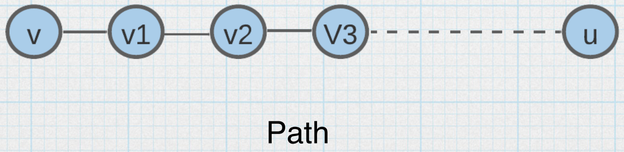
Let H1 be the Steiner Tree for $(D' \cup \{u\})$ in G.

Let H2 be the Steiner Tree for $(D\backslash D' \cup \{u\})$ in G.

Let P be the path between u and v in G.







Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

Where D' is a non empty subset of D and $u \in V(G)\backslash T$

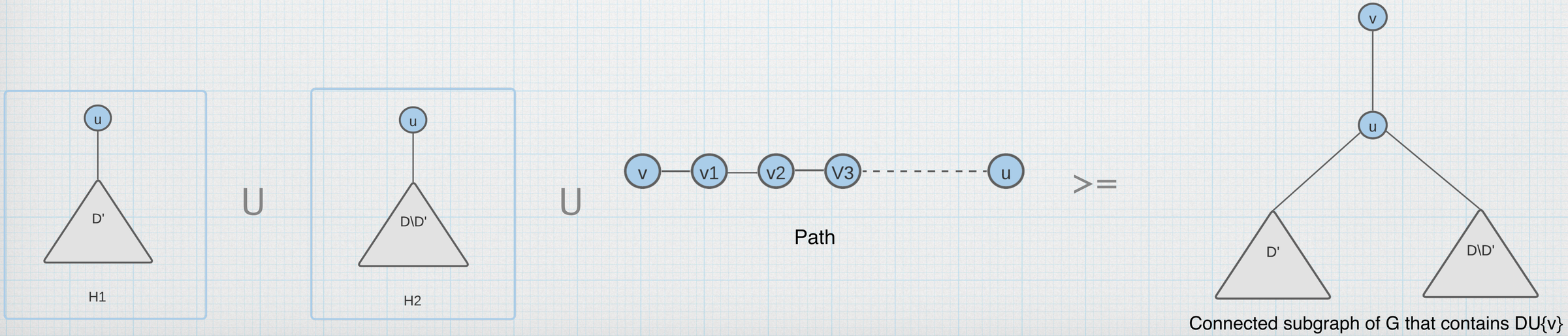
==>

Suppose we have $u \in V(G)\backslash T$ and D' as a non empty subset of D.

Let H1 be the Steiner Tree for $(D' \cup \{u\})$ in G.

Let H2 be the Steiner Tree for $(D\backslash D' \cup \{u\})$ in G.

Let P be the path between u and v in G.



Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

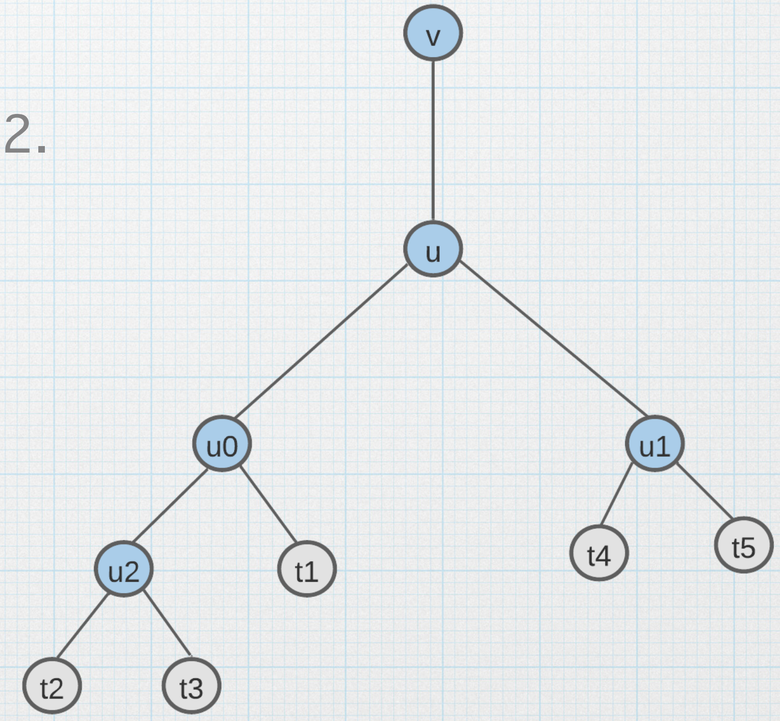
Where D' is a non empty subset of D and $u \in V(G)\backslash K$

<==

Let H be a Steiner Tree for $D \cup \{v\}$ in G, where v is the root.

Let u be the vertex with two steiner subtrees namely H1 and H2.

Here $u \notin T$.



H

Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

Where D' is a non empty subset of D and $u \in V(G)\backslash K$

<==

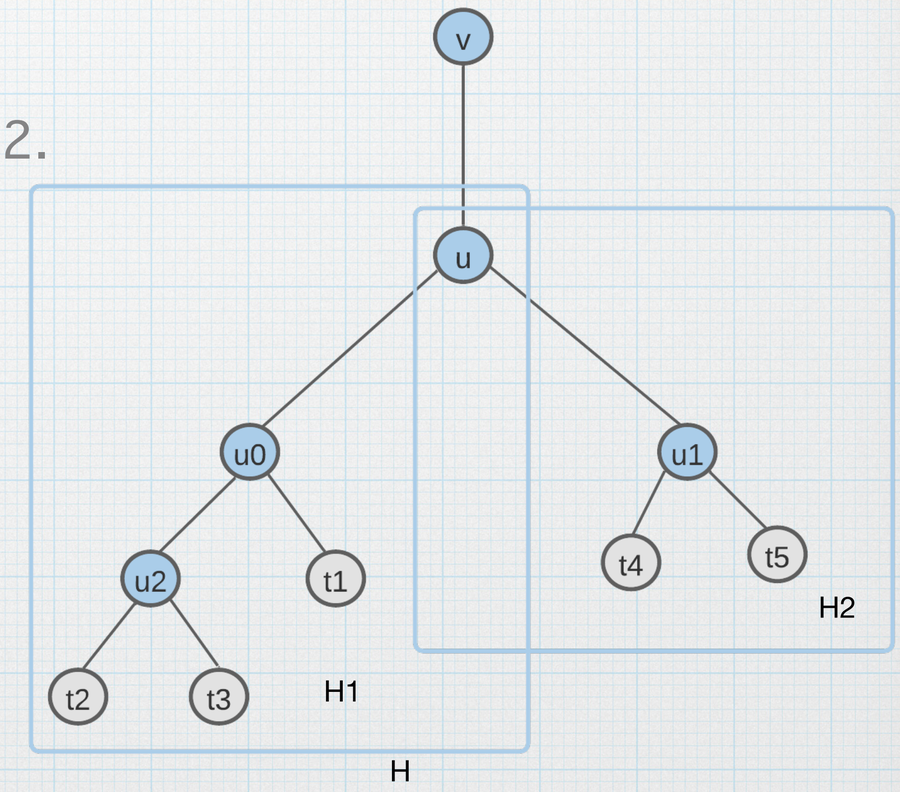
Let H be a Steiner Tree for $D \cup \{v\}$ in G, where v is the root.

Let u be the vertex with two steiner subtrees namely H1 and H2.

Here $u \notin T$.

We decompose H into following three:

1. Path from u -> v
2. H1 as a subtree rooted at u.
3. H2 as a subtree rooted at u.



Correctness of recurrence relation

Statement: For every $D \subseteq T$, when $|D| > 1$, and every $v \in V(G)\backslash T$, following holds

$DP[D,v] = \min\{DP[D',u] + DP[D\backslash D',u] + dist(v,u)\}$

Where D' is a non empty subset of D and $u \in V(G)\backslash K$

<==

Let H be a Steiner Tree for $D \cup \{v\}$ in G, where v is the root.

Let u be the vertex with two steiner subtrees namely H1 and H2.

Here $u \notin T$.

$D' = V(H1) \cap T$

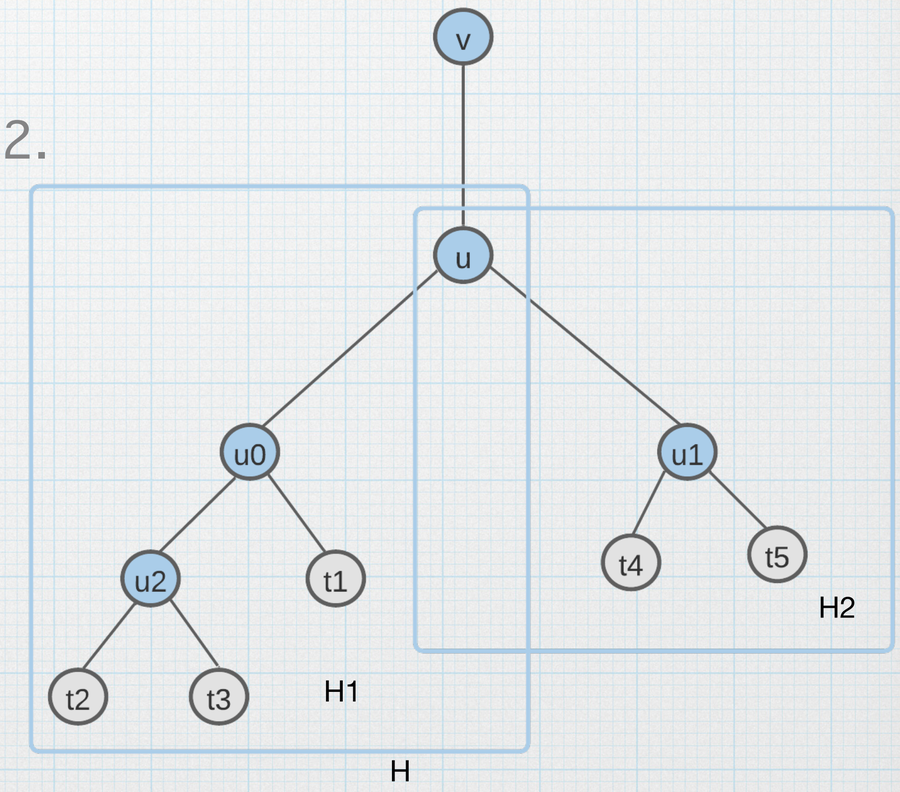
$D\backslash D' = V(H2) \cap T$

$w(H1) = DP[D',u]$

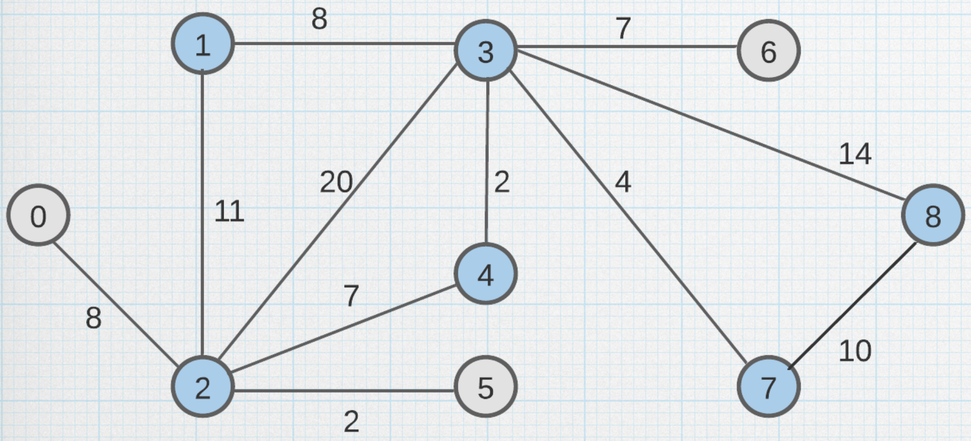
$w(H2) = DP[D\backslash D',u]$

$Path = dist(u,v)$

$DP[D,v] >= w(H1) + w(H2) + path$

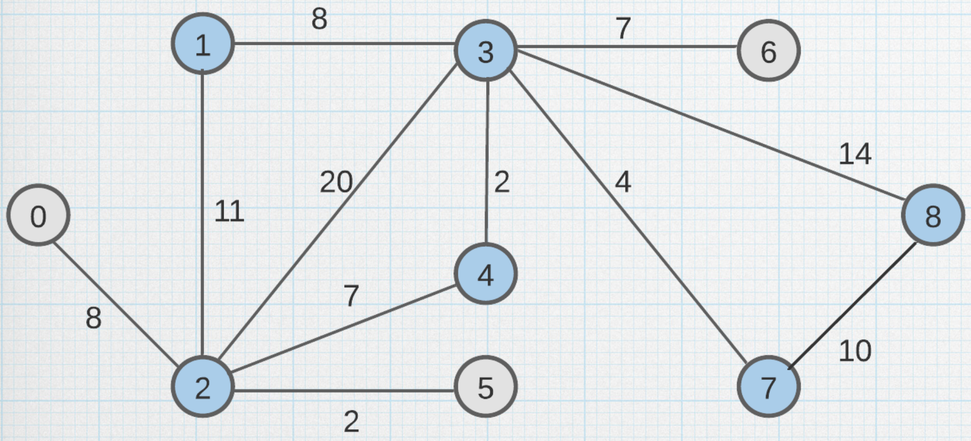


Example



Terminal Vertices : {0, 5, 6}

Example



T = {0, 5, 6}

v ∈ {1, 2, 3, 4, 7, 8}

Dynamic Programming

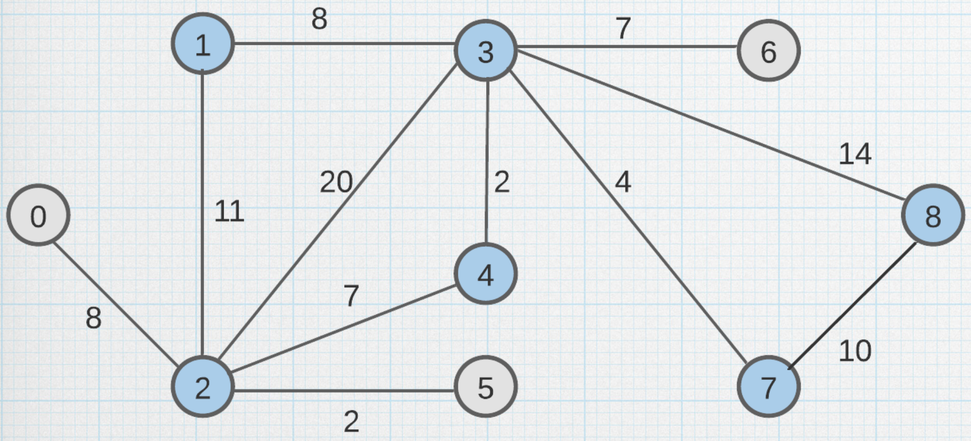
State = dp[D][v]

Base case, when |D| = 1.

Let D = {t}

Minimum weight D ∪ {v} = dist(v,t)

Example



T = {0, 5, 6}
D = {{6}, {5}, {5,6}, {0}, {0,6}, {0,5}, {0,5,6}}
BitMasking of D = { 001, 010, 011, 101, 110, 111}
v ∈ {1, 2, 3, 4, 7, 8}

Dynamic Programming

State = dp[D][v]

Base case:

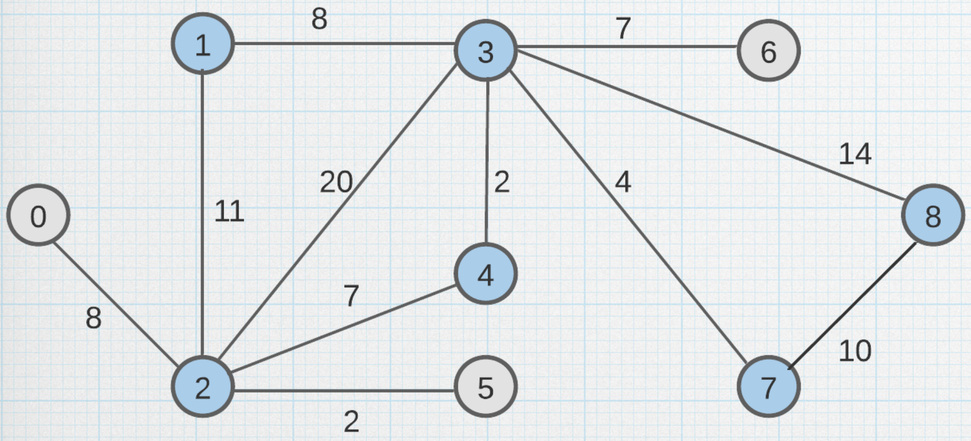
when |D| = 1.
Let D = {t}
Minimum weight D ∪ {v} = dist(v,t)

Recursive Relation:

When |D| > 1
DP[D,v] = min{DP[D',u] + DP[D\D',u] + dist(v,u)}
Where D' is a non empty subset of D and u ∈ V(G)\K

Example

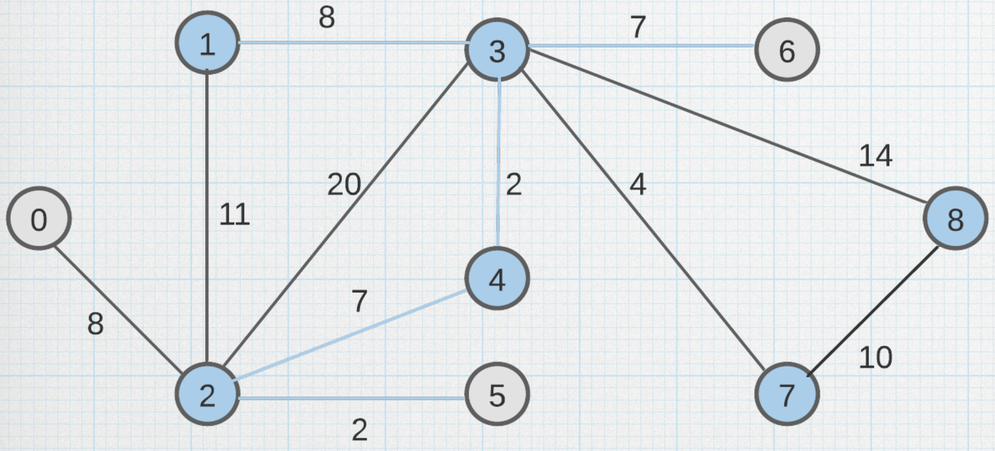
Base case |D| = 1



	1	2	3	4	7	8
{6}	15	14	7	9	11	21
{5}	13	2	11	9	15	25
{5,6}						
{0}	19	8	17	15	21	31
{0,6}						
{0,5}						
{0,5,6}						

Example

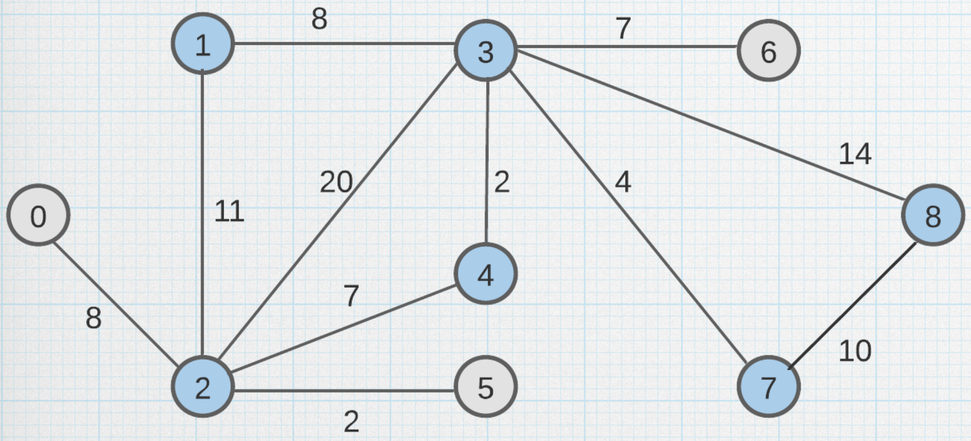
|D| > 1



	1	2	3	4	7	8
{6}	15	14	7	9	11	21
{5}	13	2	11	9	15	25
{5,6}	26					
{0}	19	8	17	15	21	31
{0,6}						
{0,5}						
{0,5,6}						

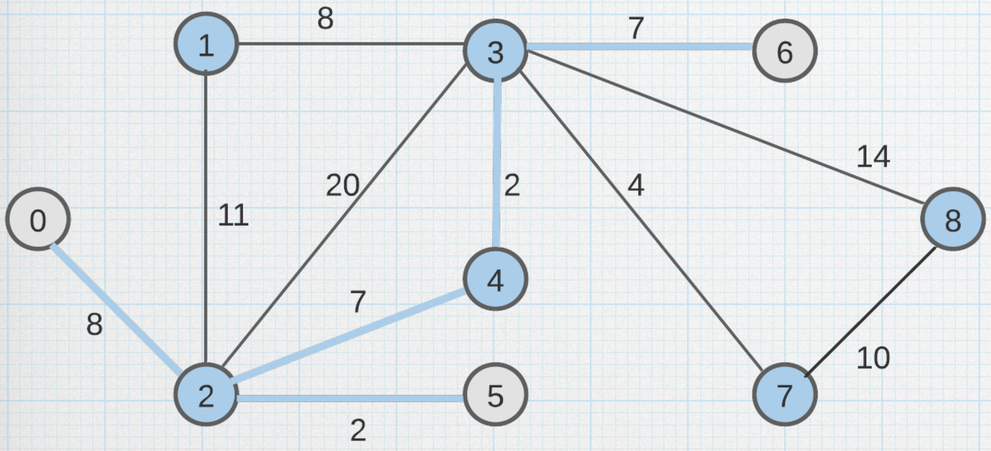
Example

|D| > 1



	1	2	3	4	7	8
{6}	15	14	7	9	11	21
{5}	13	2	11	9	15	25
{5,6}	26	18	18	18	22	32
{0}	19	8	17	15	21	31
{0,6}	32	24	24	24	28	38
{0,5}	21	10	19	17	23	33
{0,5,6}	34	26	26	26	30	40

Idea



	1	2	3	4	7	8
{6}	15	14	7	9	11	21
{5}	13	2	11	9	15	25
{5,6}	26	18	18	18	22	32
{0}	19	8	17	15	21	31
{0,6}	32	24	24	24	28	38
{0,5}	21	10	19	17	23	33
{0,5,6}	34	26	26	26	30	40

Minimum Cost = 26

Algorithm

```
//pre processing step: an algorithm to compute shortest path between two vertices (u, v)

If (|D| == 1)
DP[x][V] = dist(x,v)

For all subset  D of T :
    For all vertices v in V(G)\T :
        For all non empty subset D' of D :
            For all vertices u in V(G)\T :
                DP[D][v] = min(DP[D'][u], DP[D-D'][u] + dist(u,v)
```

Complexity Analysis

```
//pre processing step: an algorithm to compute shortest path between two vertices (u, v)

1. For all subset  D of T :                                          //O(2k)
2.     For all vertices v in V(G)\T :
3.         For all non empty subset D' of D :
5.             For all vertices u in V(G)\T :
6.                 DP[D][v] = min(DP[D'][u], DP[D-D'][u] + dist(u,v)
```

Complexity Analysis

//pre processing step: an algorithm to compute shortest path between two vertices (u, v)

```
For all subset  D of T :                                    //O(2k)
    For all vertices v in V(G)\T :                          //O(n.2^k)
        For all non empty subset D' of D :
            For all vertices u in V(G)\T :
                DP[D][v] = min(DP[D'][u], DP[D-D'][u] + dist(u,v)
```

Complexity Analysis

//pre processing step: an algorithm to compute shortest path between two vertices (u, v)

```
1. For all subset  D of T :                                  //O(2^k)
2.     For all vertices v in V(G)\T :                        //O(n.2^k)
3.         For all non empty subset D' of D :                //O(n.2^|D|)=O(4^k)
5.             For all vertices u in V(G)\T :
6.                 DP[D][v] = min(DP[D'][u], DP[D-D'][u] + dist(u,v)
```

D = 1
D' = 0 or 1
But
D = 0
D' = 0 but not 1

Complexity Analysis

//pre processing step: an algorithm to compute shortest path between two vertices (u, v)

```
For all subset  D of T :                                    //O(2^k)
    For all vertices v in V(G)\T :                          //O(n.2^k)
        For all non empty subset D' of D :                  //O(n.3^k)
            For all vertices u in V(G)\T :                  //O(n^2.3^k)
                DP[D][v] = min(DP[D'][u], DP[D-D'][u] + dist(u,v)
```

Time Complexity $= 3^k n^{O(1)} + n^3$

Implementation

```
Enter number of vertices and edges in the graph : 9 11
Enter vertices forming edges and their respective weight
0 2 8
1 2 11
1 3 8
2 3 20
2 4 7
2 5 2
3 4 2
3 6 7
3 8 14
3 7 4
7 8 10
Enter number of Terminal verlices : 3
Enter Terminal Vertices : 0 5 6
All Pair Shortest Path Matrix
   0   19    8   17   15   10   24   21   31  INF
  19    0   11    8   10   13   15   12   22  INF
   8   11    0    9    7    2   16   13   23  INF
  17    8    9    0    2   11    7    4   14  INF
  15   10    7    2    0    9    9    6   16  INF
  10   13    2   11    9    0   18   15   25  INF
  24   15   16    7    9   18    0   11   21  INF
  21   12   13    4    6   15   11    0   10  INF
  31   22   23   14   16   25   21   10    0  INF
 INF  INF  INF  INF  INF  INF  INF  INF  INF    0

-------DP Table-------
 INF  INF  INF  INF  INF  INF
  15   16    7    9   11   21
  13    2   11    9   15   25
  26   18   18   18   22   32
  19    8   17   15   21   31
  32   24   24   24   28   38
  21   10   19   17   23   33
  34   26   26   26   30   40
Cost of Minimum Steiner Tree = 26
Program ended with exit code: 0
```

Questions ?

Thank You!