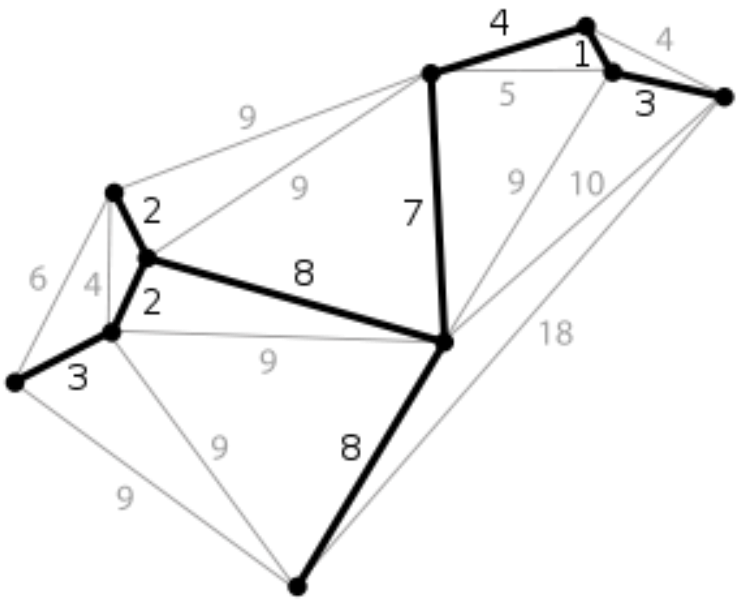
THE MINIMAL SPANNING TREE PROBLEM



Minimum spanning trees prove important for several reasons

- **They can be computed quickly and easily, and they create a *sparse sub graph* that reflects a lot about the original graph.**
- **They provide a way to *identify clusters in sets of points.* Deleting the long edges from a minimum spanning tree leaves connected components that define natural clusters in the data set.**
- **They can be used to give approximate solutions to hard problems such as *Steiner tree and traveling salesman.***
- **As an educational tool, minimum spanning tree algorithms provide graphic evidence that *greedy algorithms* can give provably optimal solutions.**

1.1 IP Formulation

The decision variables for the IP formulation of MST are:

$$x_e = \begin{cases} 1 & \text{if edge } e \in E_T \\ 0 & \text{otherwise} \end{cases}$$

The constraints of the IP formulation need to enforce that the edges in E_T form a tree. Recall from Lecture 1 that a tree satisfies the following tree conditions: have $n-1$ edges, be connected, be acyclic. Also remember that if any two of these conditions imply the third one. A possible IP formulation of MST is given in problem 1.

$$\min \quad \sum_{e \in E} w_e x_e \tag{1a}$$

$$\text{s.t.} \quad \sum_{e \in E} x_e = n - 1 \tag{1b}$$

$$\sum_{e \in (S,S)} x_e \leq |S| - 1 \quad \forall\, S \subseteq V \tag{1c}$$

$$x_e \in \{0, 1\} \quad \forall\, e \in E \tag{1d}$$

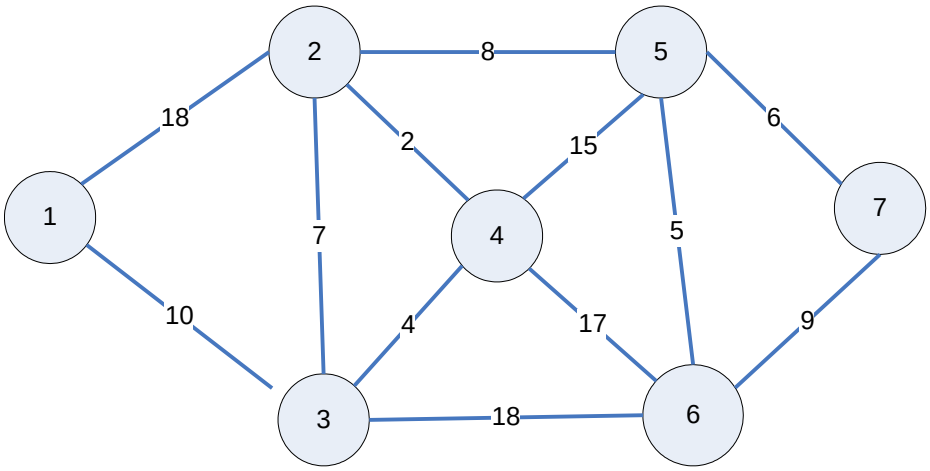
where (S, S) denotes all edges that go from a node in the set S to another node in the set S. Equation (1c) enforce the constraint that the edges in E_T can't form cycles.

We have the following observations.

1. The constraint matrix of problem (1) does not have a network flow structure, however Jack Edmonds proved that the LP relaxation has integral extreme points.
2. Even though the LP relaxation has integral extreme points, this does not imply that we can solve the MST problem in polynomial time. This is because the formulation has an exponential number of constraints. Nevertheless, we can use the following strategy to solve problem (1).
   Relax the set of constraints given in (1c), and solve the remaining problem. Given the solution to such relaxed problem, find which of the relaxed constraints are violated (this process is called separation) and add them. Resolve the problem including these constraints, and repeat until no constraint is violated.
   It is still not clear that the above algorithm solves in polynomial time MST. However, in light of the equivalence between separation and optimization (one of the central theorems in optimization theory), and since we can separate the inequalities (1c) in polynomial time, it follows that we can solve problem (1) in polynomial time.

- **Greedy Algorithm**
- **Kruskal Algorithm**
- **Prim Algorithm**

In a minimal spanning tree problem, we seek the tree interconnects all the nodes in a network at minimum total distance. *The Greedy Algorithm* is an easy, efficient way to find this minimal spanning tree.



The Greedy Algorithm

Initialization Step

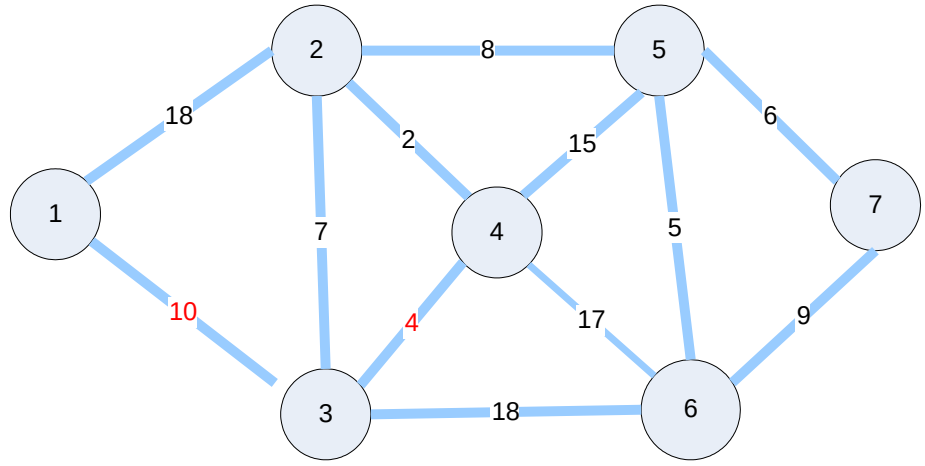
Select the minimum arc out of node 1 to start the tree.

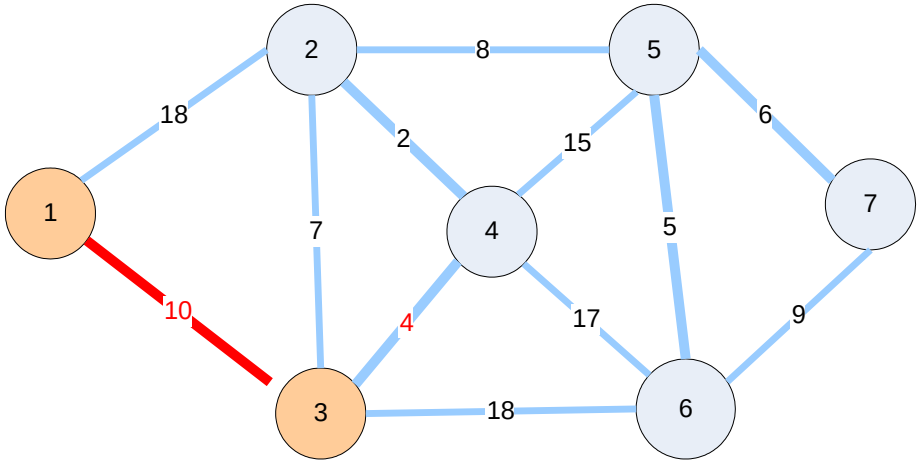
Iterative Steps

Add to the current tree the connecting arc of minimum distance that does not form a cycle. If all nodes are connected, STOP; we have the minimal spanning tree.

We repeat the above iterative steps until all nodes have been connected.
If there are n nodes, the tree consists of n-1 arcs.

Example: (Greedy Algorithm)



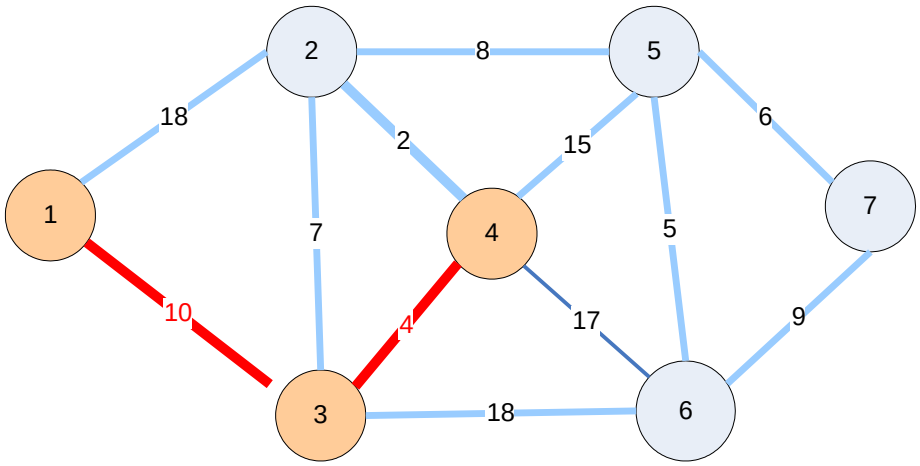


INITIALIZATION

Choose the minimum arc out of node-1- arc (1,3) –distance 10.

Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10

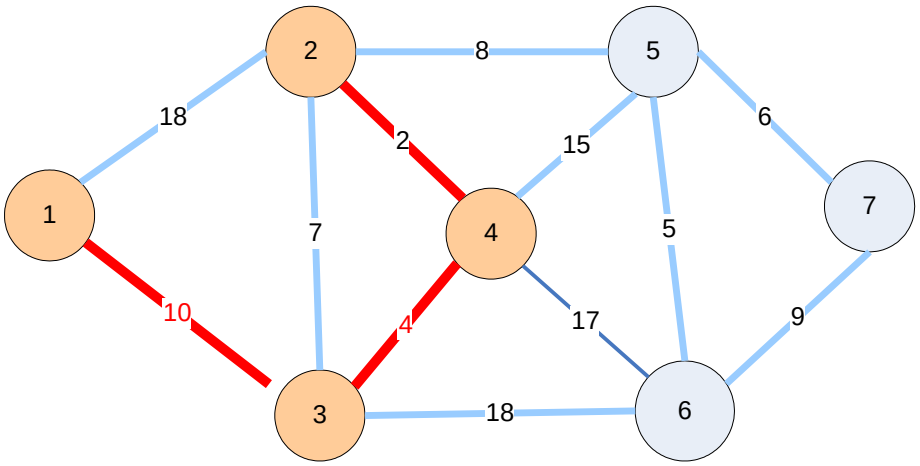
1.st Iteration



Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10
1	(3,4)*	4	Yes	14

*** Min {(3,4);(3,2);(1,2);(3,6)}=(3,4)**

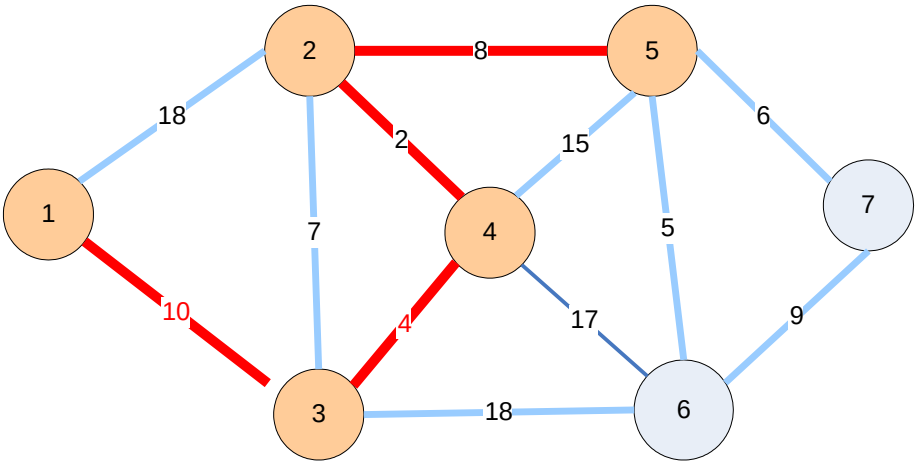
2.nd Iteration



Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10
1	(3,4)	4	Yes	14
2	(4,2)*	2	Yes	16

***Min{(1,2);(3,2);(3,6);(4,2);(4,5);(4,6)}=(4,2)**

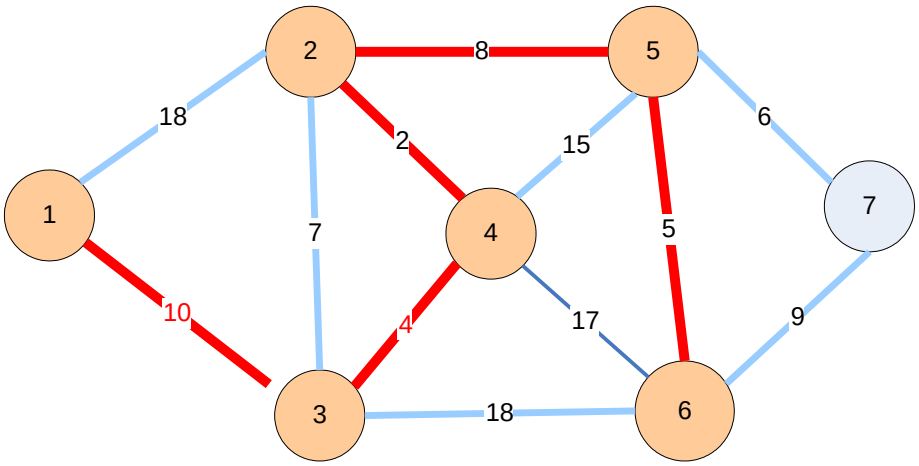
3.rd Iteration



Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10
1	(3,4)	4	Yes	14
2	(4,2)	2	Yes	16
3	(2,3)	7	No (Cycle)	
3	(2,5)*	8	Yes	24

***Min{(2,5);(4,5);(4,6);(3,6)}=(2,5)**

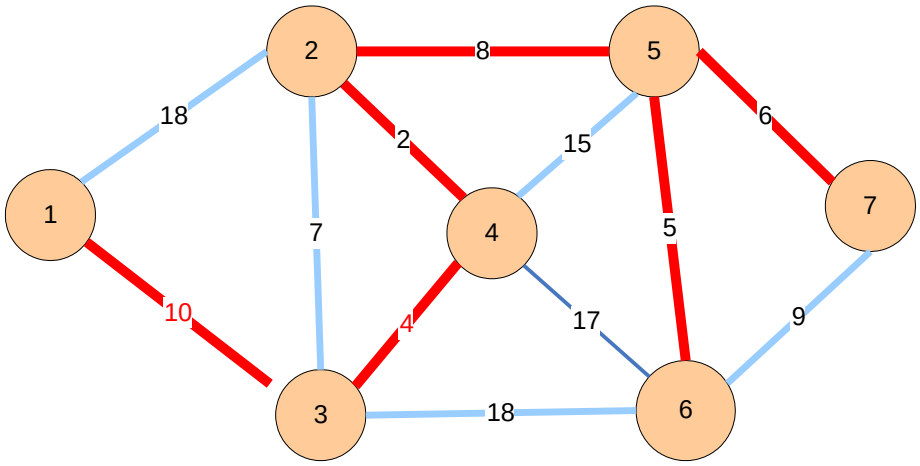
4. th Iteration



Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10
1	(3,4)	4	Yes	14
2	(2,4)	2	Yes	16
3	(2,3)	7	No(Cycle)	
3	(2,5)	8	Yes	24
4	(5,6)*	5	Yes	29

***Min{(5,6);(4,6);(3,6)}=(5,6)**

5.th Iteration



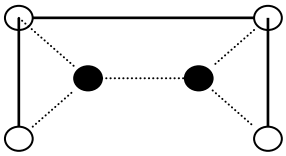
ITERATIVE STEPS

Iteration	Minimum Distance Connecting Arc	Distance	Add Arc to Tree?	Cumulative Tree Distance
0	(1,3)	10	Yes	10
1	(3,4)	4	Yes	14
2	(2,4)	2	Yes	16
3	(2,3)	7	No(Cycle)	
3	(2,5)	8	Yes	24
4	(5,6)	5	Yes	29
5	(5,7)*	6	Yes	35

***Min{(5,7);(6,7)}=(5,7)**

Thus the minimum spanning tree has a total distance of 35 and consists of arcs (1,3), (3,4), (2,4), (2,5) and (5,7).

Related Problems:
Steiner Minimal Tree Problem

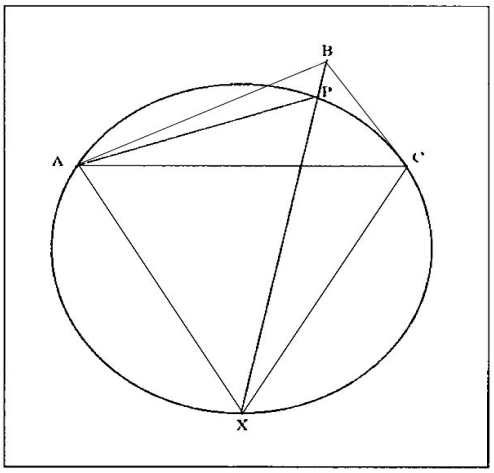


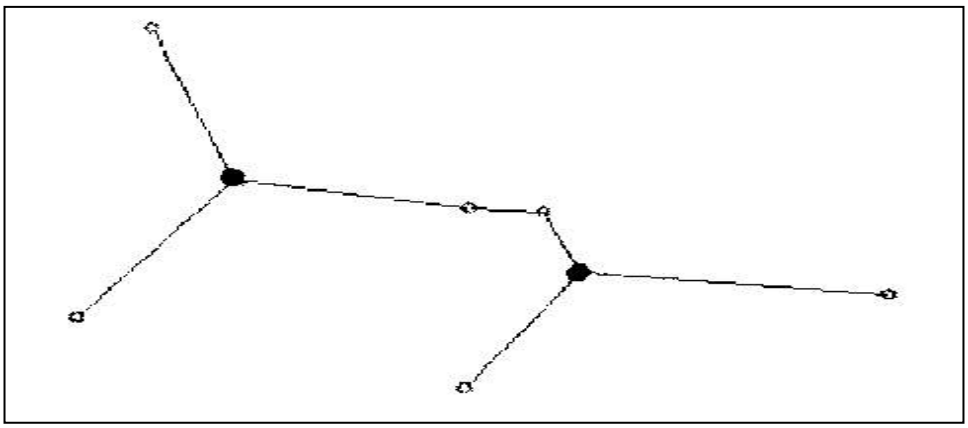
NODES

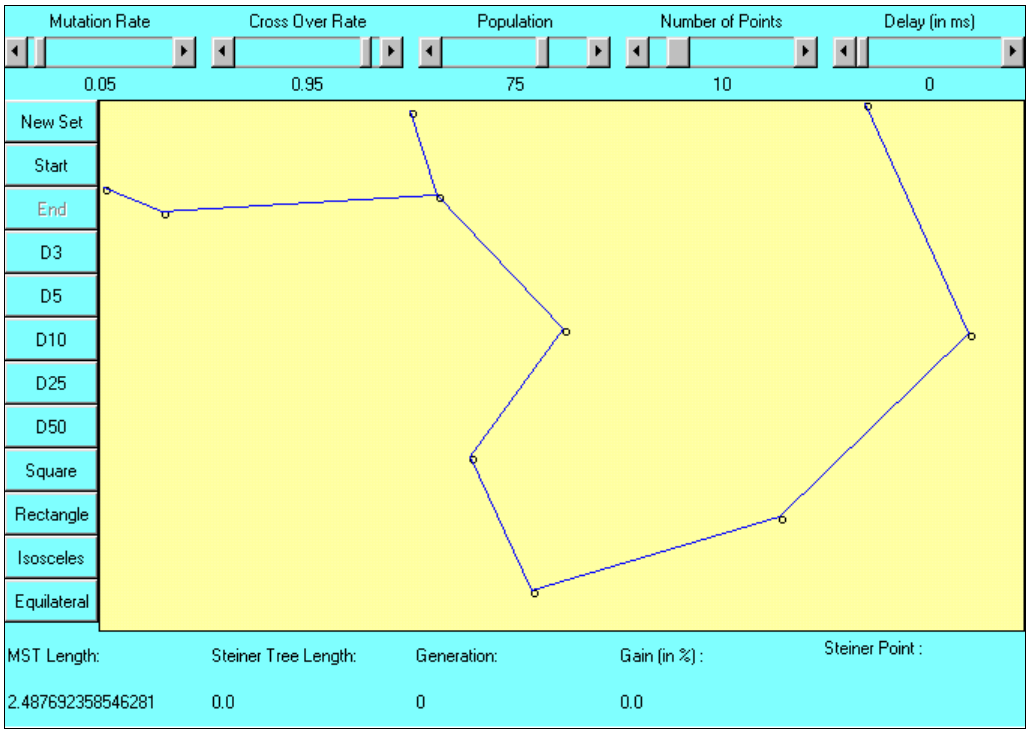
STEINER POINTS

MST

SMT






Mutation Rate
Cross Over Rate
Population
Number of Points
Delay (in ms)
0.05
0.95
75
10
0
New Set
Start
End
D3
D5
D10
D25
D50
Square
Rectangle
Isosceles
Equilateral
MST Length:
Steiner Tree Length:
Generation:
Gain (in %) :
Steiner Point :
2.487692358546281
0.0
0
0.0