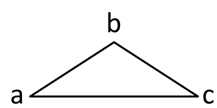
Approximation Algorithms:

- An algorithm that returns near optimal solutions in polynomial time.
- A way of dealing with NP-Completeness.
- Work only if the problem instance satisfies Triangle-Inequality.

Recap:

- P (Polynomial) complexity class consists of problems that can be solved with known polynomial-time algorithms using deterministic machines.
- NP (Nondeterministic Polynomial) complexity class involves the concept of a nondeterministic computer.
- NP-hard complexity class consists of problems with exponential time complexity.
- NP-Complete complexity class contains the NP-Hard problems that have been solved using nondeterministic machine in such a way that the implementation works in polynomial time.

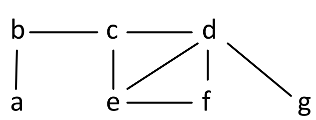
Triangle-Inequality:



w(a,b) + w(b,c) >= w(a,c)

Vertex Cover Problem:

- Vertex Cover of a graph is a subset of vertices which covers every edge.
- It is a set of vertices incident to every edge.



Algorithm:
1) Initialize the result as {}
2) Consider a set of all edges in given graph. Let the set be E.
3) Do following while E is not empty
   a) Pick an arbitrary edge (u, v) from set E and add u and v to result.
   b) Add u and v to the result.
   c) Remove all edges from E which are either incident on u or v.
4) Return result

Option 1:

R = {}		E = {(a,b),(b,c),(c,d),(c,e),(d,e),(d,f),(d,g),(e,f)}
R = {a,b}	Selected (a,b)	E = {(c,d),(c,e),(d,e),(d,f),(d,g),(e,f)}
R = {a,b,e,f}	Selected (e,f)	E = {(c,d),(d,g)}
R = {a,b,e,f,d,g}	Selected (d,g)	E = {}

Option 2:

R = {}		E = {(a,b),(b,c),(c,d),(c,e),(d,e),(d,f),(d,g),(e,f)}
R = {a,b}	Selected (a,b)	E = {(c,d),(c,e),(d,e),(d,f),(d,g),(e,f)}
R = {a,b,c,d}	Selected (c,d)	E = {(e,f)}
R = {a,b,c,d,e,f}	Selected (e,f)	E = {}

Option 3:

R = {}		E = {(a,b),(b,c),(c,d),(c,e),(d,e),(d,f),(d,g),(e,f)}
R = {d,e}	Selected (d,e)	E = {(a,b),(b,c)}
R = {d,e,a,b}	Selected (a,b)	E = {}

Optimal Solution: Result set with minimum number of vertices.
Complexity: O(V+E)

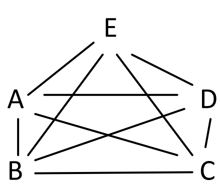
Travelling Salesperson Problem:

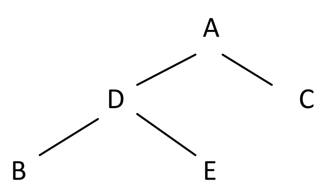
- No known polynomial time solution.
- n cities.
- Travel to ALL cities exactly ONCE.
- Start and End at the same city.
- Hamiltonian Cycle with minimum cost.

Algorithm (Undirected Complete Graph):

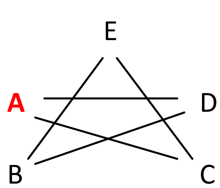
1. Computer MST.
2. Perform Pre-Order/DFS Walk.
3. Join in this order to obtain an approx. tour walk.

EXAMPLE: Starting and Ending point is A

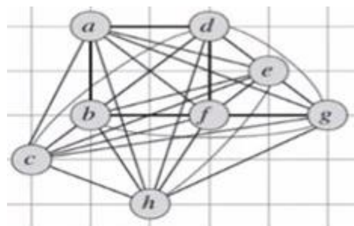


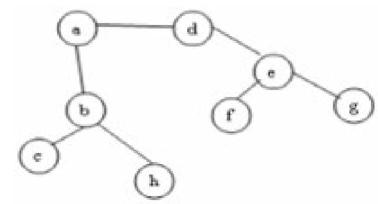


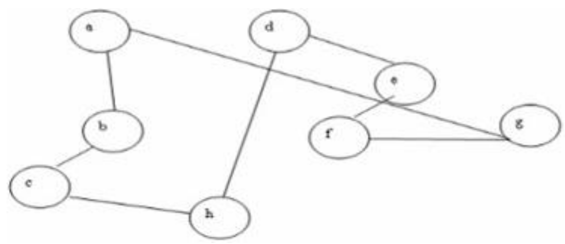
Pre-Order/DFS: A,D,B,D,E,D,A,C,A
Remove repeated nodes: A,D,B,E,C,A



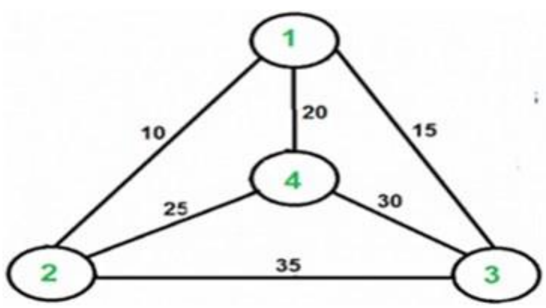
EXAMPLE:







TASK:



Edge	1-2	1-3	1-4	2-4	3-4	2-3
Weight	10	15	20	25	30	35

Start and End: Node 1
Pre-Order: 1,2,1,4,1,3,1
Remove Repetition: 1,2,4,3,1