KRUSKAL'S & PRIM'S MST

HOW IT WORKS ?

KEY OBSERVATIONS AND FACTS

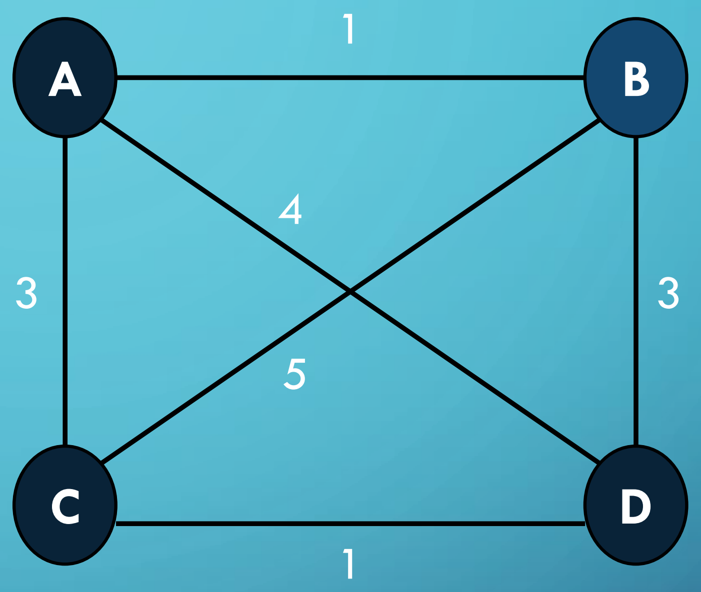
COMPARATIVE RUNTIME ANALYSIS FOR SPARSE AND DENSE GRAPHS

A CLOSER LOOK AT THE RESULTS

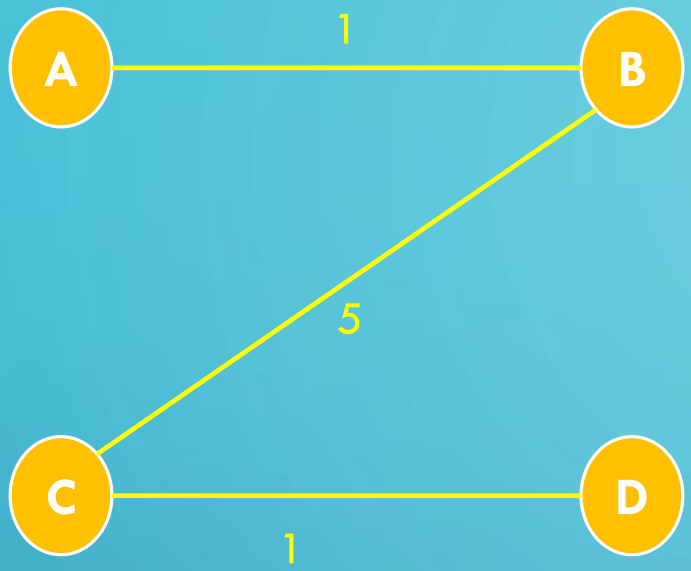
Presented by : Archana(#1001415817) & Ganesh (#1001452475)
Date : 11/29/2016

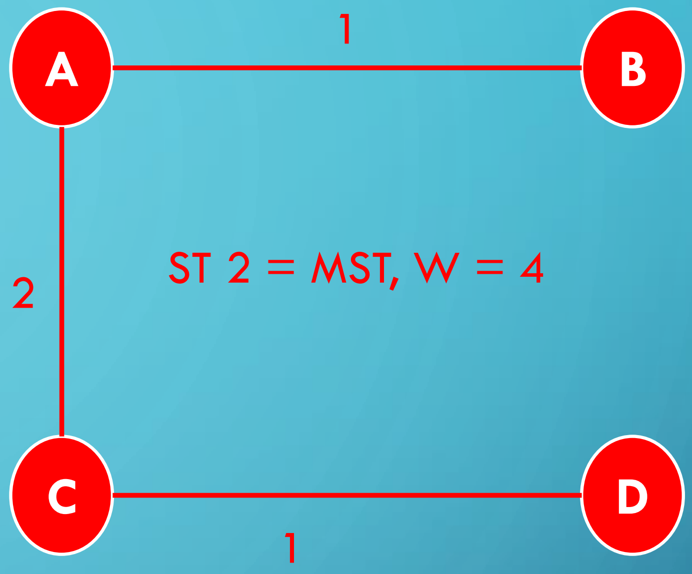
MINIMUM SPANNING TREE (MST)

- For a connected weighted undirected graph, G = (V, E).
- Possible spanning trees for G :
  { ST 1, ST 2, .. ST N }
- Possible MSTs for G :
  Minimum { ST 1, ST 2, .. ST N}

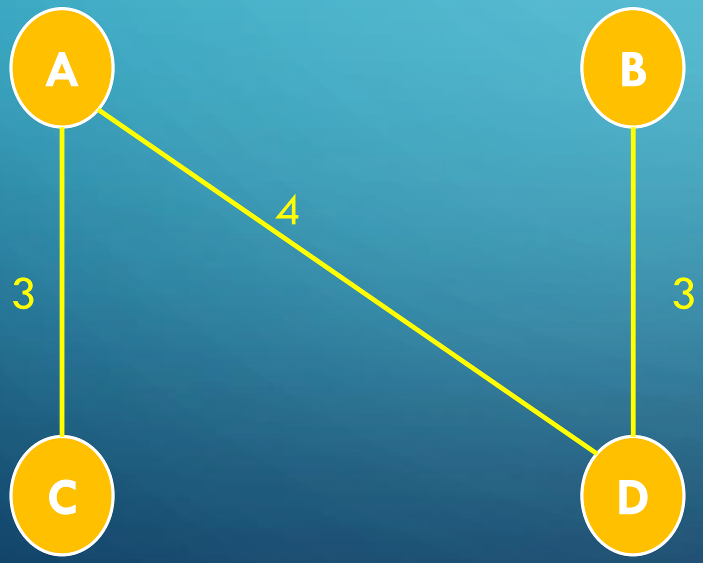


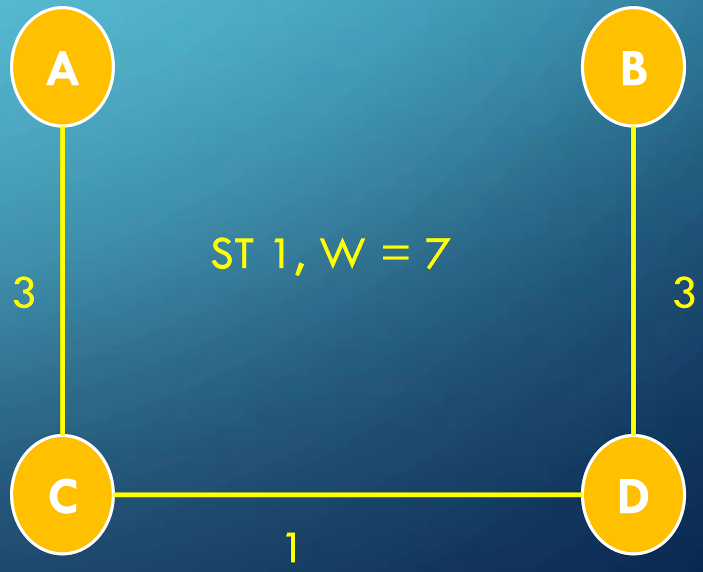
ST 1, W = 7





ST 3, W = 10





KRUSKAL'S ALGORITHM

- Initialize empty set A and the set for each vertex in a graph - O(v)
- Sort all the edges in an increasing order - O(E lg E)
- For each edge, add it to A if it connects two distinct trees. Is it safe?- O(E lg E)
- Data structure : Disjoint Set with Union by Rank and Path Compression
- Total Running Time : O(E lg E)

PRIM'S ALGORITHM

- Start with the random vertex, V which is the root. (key = 0, parent = null)
- For each vertex in adjacent vertices, find out the light edge, update the key & parent attributes and add it to A. – O(E lg E)
- Repeat above by taking vertex with minimum key till all the nodes are visited. – O(V lg V)
- Data structure : Priority Queue using Binary Min Heap
- Total Running Time : O(V lg V + E lg V)

KEY OBSERVATIONS AND FACTS

- Kruskal's: Growing forest, Prim's: Maintains a single tree
- Kruskal's: Similar to CC algorithm, Prims: Similar to DIJ Shortest path Algorithm
- Kruskal's: Avoids a cycle by not adding the light edge to the same CC
- Both: Running time depends on how we implement supporting data structure
- Both: Implementation of Generic MST method
- Both: No guarantee that it will produce a globally optimal solution
- Both: O(E lg E) Vs O(E lg V), E is the most dominating factor

ANALYSIS OF KRUSKAL’S AND PRIM’S

RUNNING TIME COMPARISON WITH SPARSE AND DENSE GRAPHS

INPUT GRAPHS FOR ANALYSIS

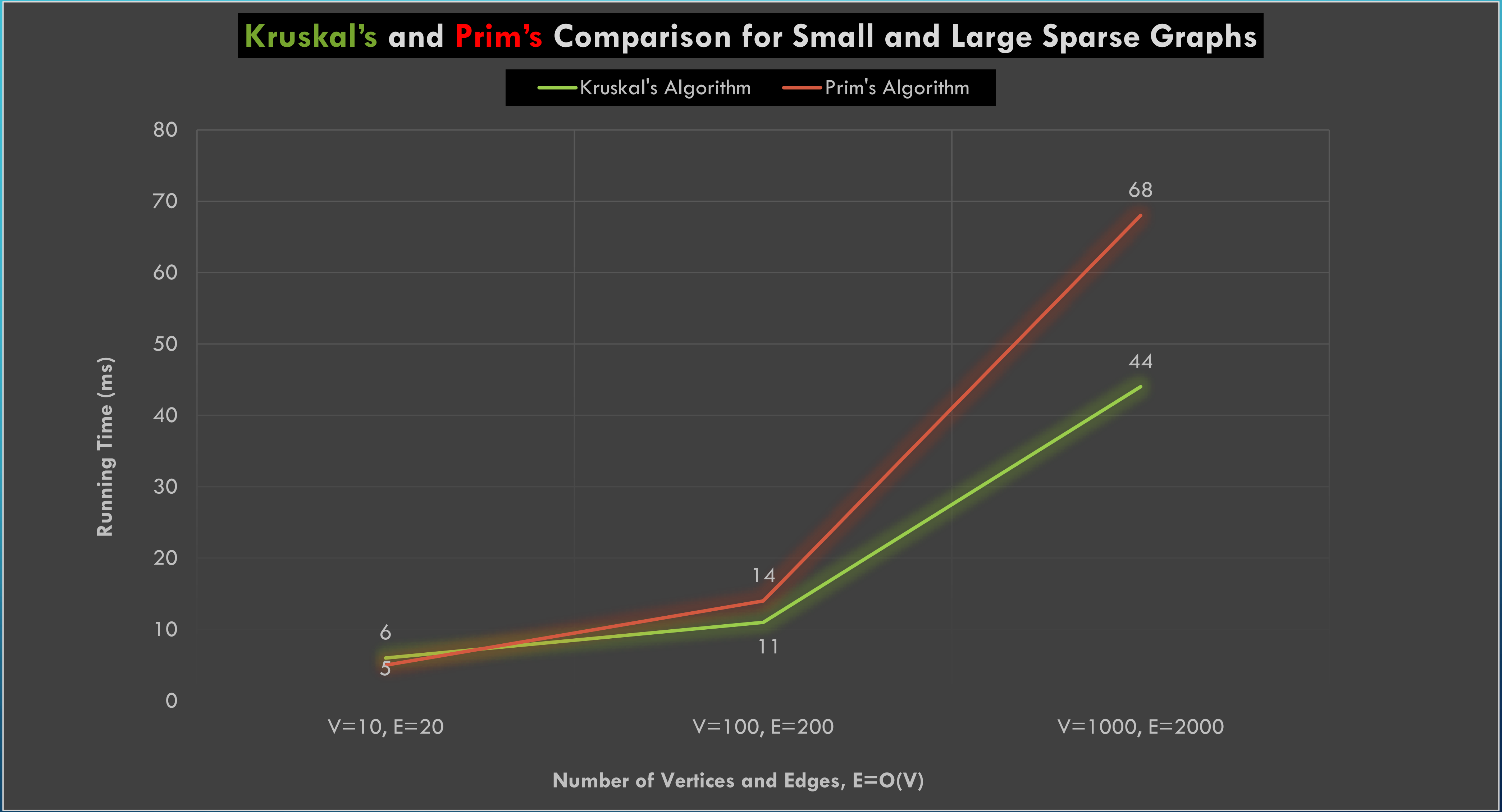
- Sparse Graph
  - G1 = (10,20), G2 = (100,200), G3 = (1000,2000)
- Dense Graph
  - G1 = (10,90), G2 = (100,9900), G3 = (1000,999000)
- All input graphs are represented using Adjacency List.

Kruskal's Running Time (Sparse)

G = (V,E)	T_1 (ms)	T_2 (ms)	T_3 (ms)	T_4 (ms)	Average
V = 10 E = 20	5	6	8	5	6
V = 100 E = 200	11	8	14	13	11
V = 1000 E = 2000	48	38	47	42	44

Prim's Running Time (Sparse)

G = (V,E)	T_1 (ms)	T_2 (ms)	T_3 (ms)	T_4 (ms)	Average
V = 10 E = 20	4	8	5	4	5
V = 100 E = 200	12	16	17	13	14
V = 1000 E = 2000	63	65	71	72	68

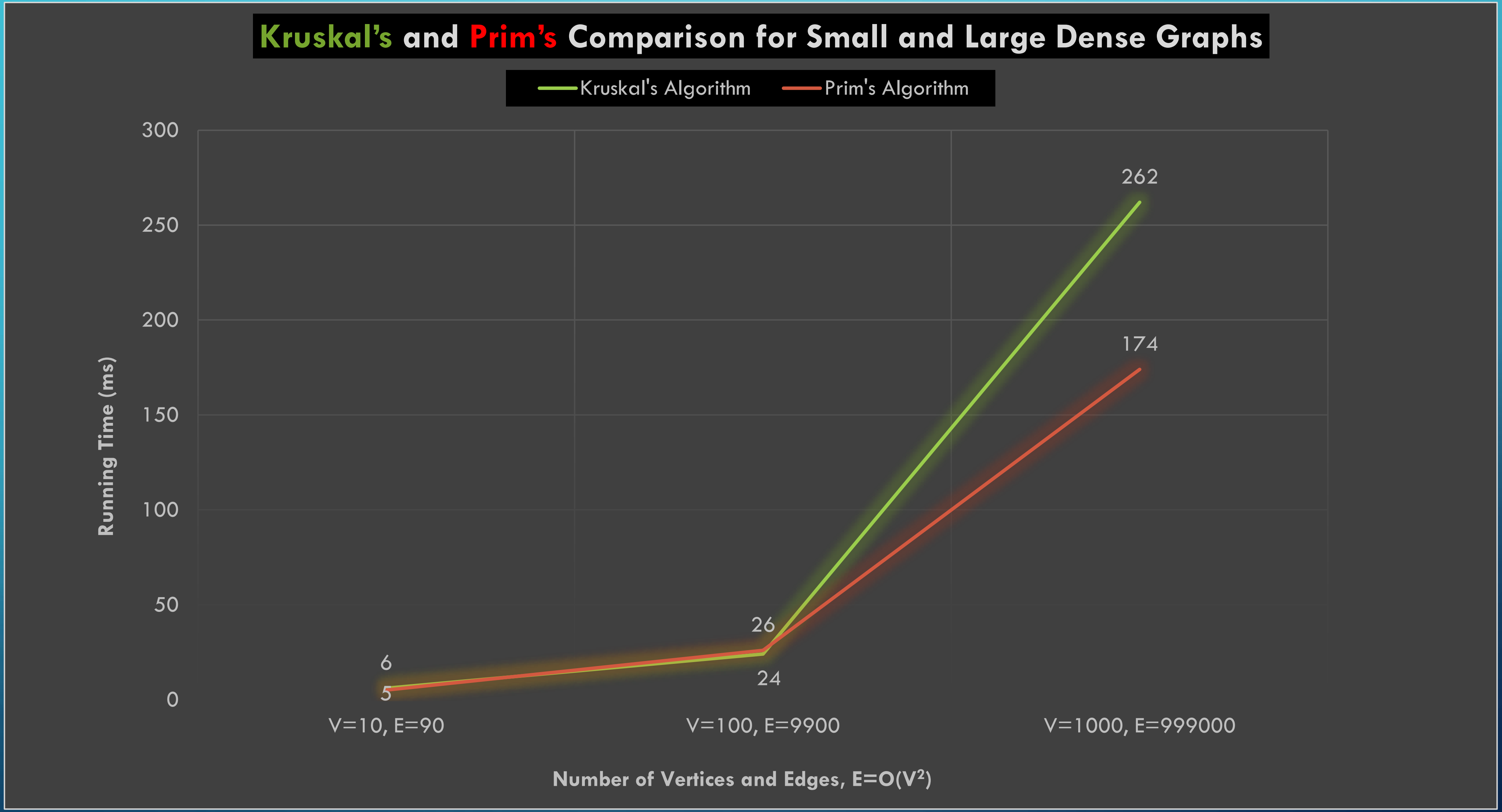

Kruskal's and Prim's Comparison for Small and Large Sparse Graphs
Kruskal's Algorithm
Prim's Algorithm
Running Time (ms)
80
70
60
50
40
30
20
10
0
6
5
14
11
68
44
V=10, E=20
V=100, E=200
V=1000, E=2000
Number of Vertices and Edges, E=O(V)

Kruskal's Running Time (Dense)

G = (V,E)	T_1 (ms)	T_2 (ms)	T_3 (ms)	T_4 (ms)	Average
V = 10 E = 90	7	6	5	6	6
V = 100 E = 9900	24	25	24	23	24
V = 1000 E = 999000	289	207	273	279	262

Prim's Running Time (Dense)

G = (V,E)	T_1 (ms)	T_2 (ms)	T_3 (ms)	T_4 (ms)	Average
V = 10 E = 90	6	4	6	4	5
V = 100 E = 9900	29	27	22	28	28
V = 1000 E = 999000	180	174	179	163	174


Kruskal's and Prim's Comparison for Small and Large Dense Graphs
Kruskal's Algorithm
Prim's Algorithm
Running Time (ms)
300
250
200
150
100
50
0
262
174
26
24
6
5
V=10, E=90
V=100, E=9900
V=1000, E=999000
Number of Vertices and Edges, E=O(V²)

“

SO, CAN WE SAY PRIM’S IS BEST FOR LARGE DENSE GRAPHS AND KRUSKAL’S IS GOOD FOR SPARSE GRAPHS ??
- - MAYBE/DEPENDS ☺

”

Thank you for your time. Have a pleasant day!

If you want a closer look at the results, download our output files here!

mailto: ganeshbabu.thavasimuthu@mavs.uta.edu