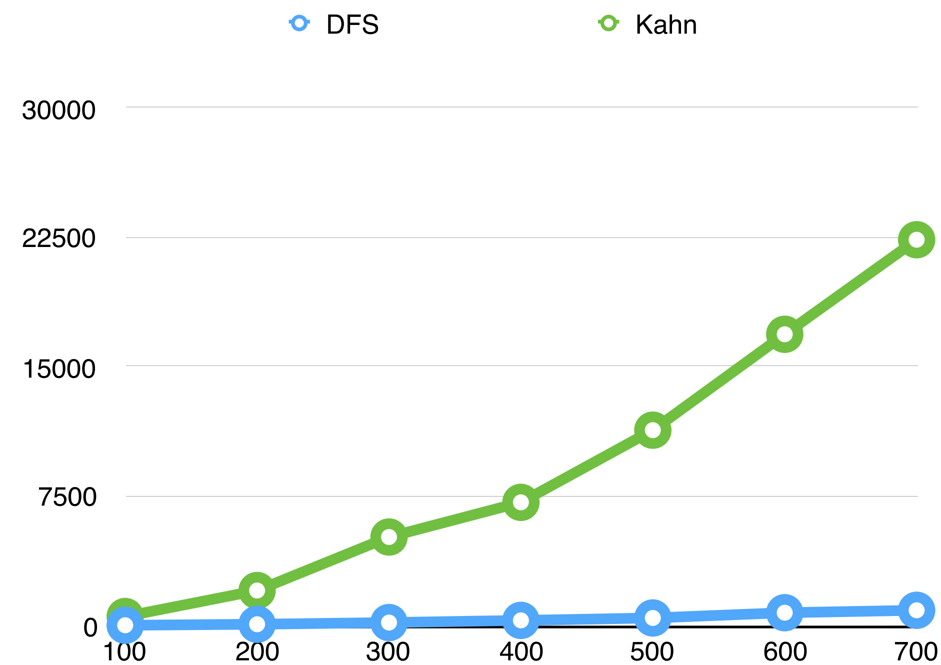
Topological Algorithms Analysis Report

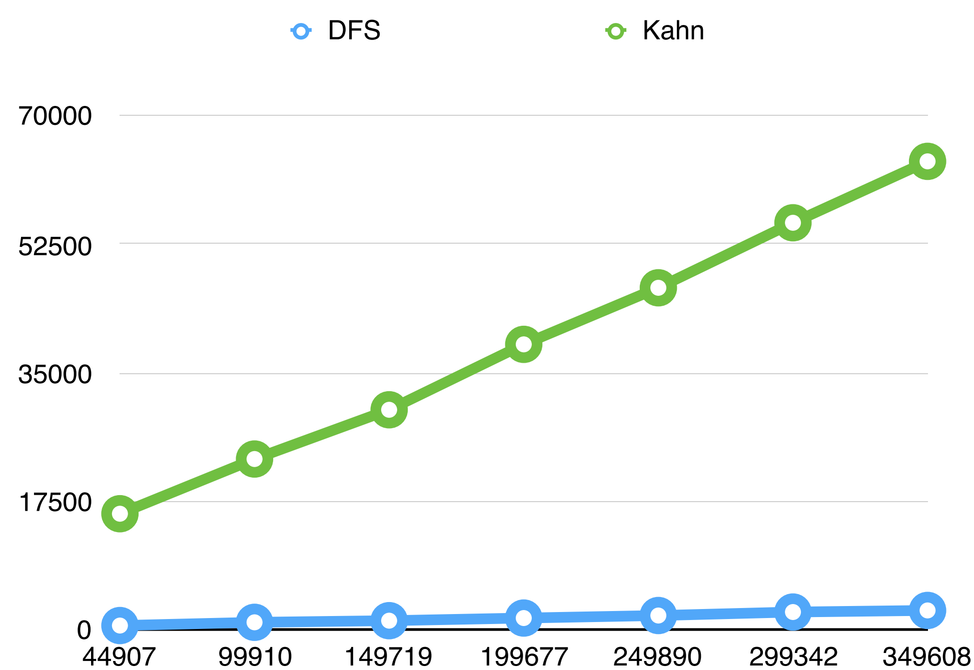
Fixed Edges Probability

Nodes(50% edges)	DFS	Kahn
100	32	539
200	91	2044
300	189	5144
400	314	7153
500	455	11334
600	759	16889
700	899	22366



Fixed Number of Nodes

Edges(1000 nodes)	DFS	Kahn
44907	470	15726
99910	911	23189
149719	1135	29908
199677	1494	38833
249890	1838	46551
299342	2314	55421
349608	2531	63801



ALGORITHMS ARE FUN!

DFS:

- For each node:
  - Check if node ***n*** is marked. If not, mark ***n*** temporarily: O(1)
  - Mark ***n*** permanently: O(1)
  - Prepend ***n*** to sorted list ***L***: O(1) for linked list
  - Total for |***V***| nodes: O(|***V***|)

- For each edge:
  - Visit edge: O(1)
  - Total for |***E***| edges: O(|***E***|)

- Total for graph ***G***(***V*** + ***E***):
  - Since each node and each edge is visited once, and each visit takes constant time, the total time complexity for graph ***G*** is O(|***V***|+ |***E***|). Linear growth.

Kahn:

- Get all nodes with no incoming edges in list ***S***:
  - Check each node for incoming edges: O(1) for linked list implementation of adjacency list.
  - Append source node to ***S***: O(1) for linked list
  - Total for |***V***| nodes: O(|***V***|)

- For each node in ***S***:
  - Pop node ***n*** off ***S***: O(1) for linked list
  - Append ***n*** to sorted list ***L***: O(1) for linked list
  - Total: since all nodes will eventually be in ***S***, time complexity is O(|***V***|)

- For each edge (***n***, ***m***):
  - Remove edge: O(2)
    - Maybe I should elaborate on how I got O(2). This is the step I had difficulty understanding the time complexity of. And the result may vary depending on the implementation in actual coding. <u>Assuming all |***E***| edges are from one node ***n***</u>, based on my implementation, pop node ***m*** from from the adjacency out list of ***n*** takes O(1). Since ***m*** only has one incoming edge, the edge from ***n***, removing ***n*** from the adjacency in list of ***m*** takes O(1). So that is O(2) in total. Otherwise I really have no idea how to figure out the time complexity of removing edges.
  - If ***m*** has no incoming edges, append ***m*** to ***S***: O(1)
  - Total for |***E***| edges: O(|***E***|)

- Total for graph ***G***(***V*** + ***E***): O(|***V***| + |***E***|), linear growth

Note: The reason why the first graph (Fixed Edges Probability) does not appear to be perfectly linear is that increasing the number of nodes with fixed edges probability also increases number of edges. Let ***N*** be the number of nodes, ***P*** be the probability of edges. Then the number of edges would be ((***N*** -1)****N***/2)****P*** in theory, assuming there are no cycles. So the increase in edges does not necessarily have a linear relationship with the increase in nodes. Thus the result O(|***V***| + |***E***|) does not appear perfectly linear in the first graph.

ALGORITHMS ARE FUN!