COMP20007 Design of Algorithms
Semester 1, 2016

Assignment 1 – Topological Sort Analysis

Geoffrey Law (Student ID: 759218)

Abstract

This report investigates the efficiency of two topological sorting algorithms, which are Depth-first search algorithm and Kahn's algorithm. A brief complexity analysis for each of the two algorithms is described in the form of big-O notation. The empirical model is represented by a graph of running times that data is collected from tests with arbitrary input of increasing size. The discussion is to analyse the running time of the two algorithms from the experiment relating to theoretical analysis.

Introduction

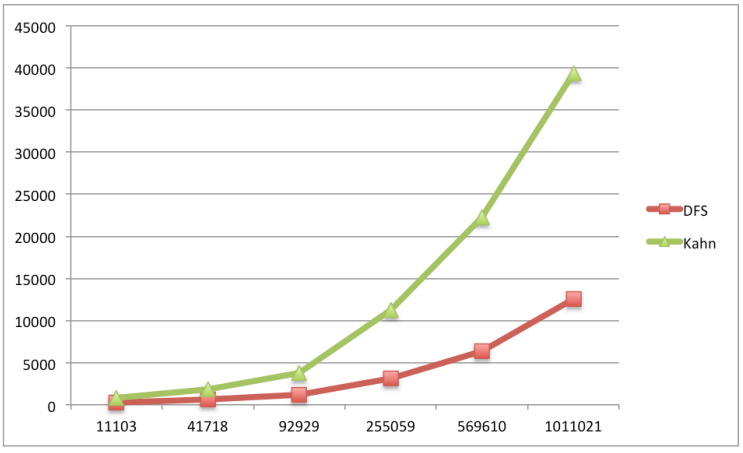
The depth-first search (DFS) algorithm is implemented to iterate all vertices in a graph and for each unvisited vertex to call 'visit' function recursively to explore all outgoing edges from that vertex to unvisited vertices. The iteration of all vertices takes $O(|V|)$ and the explore step takes $O(|E|)$, then the total running time is linear $O(|V| + |E|)$. Whilst Kahn's algorithm has the same big-O running time, however it has a different implementation of the algorithm. The algorithm has similar process as DFS which to step through all vertices and edges consistently but with additional loops and data structure manipulation steps during the procedure. Experimentally, in hypothesis, Kahn's algorithm should take more time in its exact total work done.

Method

1. To generate 6 input files by program 'daggen' with increasing size n, where n is number of vertices and edges
2. To test each input file to the main program in two different modes of topological sorting which DFS and Kahn's
3. To collect 10 clock data for each input file of each sorting mode
4. To calculate the average clock as resulted data to ensure the accuracy
5. To arrange the resulted data into a table then sketch a graph

Experiment

n	DFS algorithm	Kahn's algorithm
11103	238.5	773.2
41718	632.0	1820.2
92929	1194.9	3766.1
255059	3126.6	11307.7
569610	6410.1	22230.6
1011021	12537.2	39403.8



Since the running time for both topological sorting algorithms is linear $O(|V| + |E|)$, we can now consider the constant c of the big-O notation (by definition $O(g) = c \cdot g$) for each input with size n, where $n = |V| + |E|$.

Thus, we evaluate constant c from $O(n) = c \cdot n$.

n	$O_{DFS}(n)$	$O_{Khan}(n)$	c_{DFS}	c_{Khan}
11103	238.5	773.2	0.0215	0.0696
41718	632.0	1820.2	0.0151	0.0436
92929	1194.9	3766.1	0.0129	0.0405
255059	3126.6	11307.7	0.0123	0.0443
569610	6410.1	22230.6	0.0113	0.0390
1011021	12537.2	39403.8	0.0124	0.0390
Average constant c for each algorithm:			**DFS: 0.0143**	**Khan: 0.0460**

Discussion

DFS's algorithm has smaller constant c, which means the exact total work done of DFS's algorithm is less than Kahn's algorithm. Theoretically, they both have the same running time in linear $O(|V| + |E|)$, however the big-O representation only consider the worst case. The exact work done is depending on the constant c asymptotically. There are many factors and uncertainties to affect the constant or the exact running time, such as hardware performance of a computer, additional functions used by implementation etc. Now we investigate some factors would affect the running time that involved in the actual C code.

In the C code implementation of DFS's algorithm, we notice that there is only one list manipulation function, which is 'prepend', throughout the algorithm. However, Kahn's algorithm used four list manipulation functions in total that might affect the running time, includes one 'insert' function.

Since 'insert' function is inserting a node into the tail of the list, it is needed to access that from head to tail throughout the entire list, besides that 'insert' function is indented into a loop. The 'insert' function takes $O(|list|)$ to implement whilst 'prepend' takes only $O(1)$. The data type used in Kahn's algorithm is queue, so it is impossible to replace 'insert' with other more efficient list manipulation functions from the provided API. Moreover, another function that affecting the running time is the 'del' function, which used to delete edges from the graph. It would take $O(|list|)$ to step through the list in order to find the node that equal to the aiming node.

Last but not least, in the function 'kahn_sort', the 'reverse' function that takes $O(|list|)$ used before the list is returned. That is because, by considering efficiency, each vertex was consistently added into head of the list by using 'prepend' instead of 'insert', therefore the list needed to be reversed in order to return a valid topological ordering list.

Conclusion

The result from empirical model shows Kahn's algorithm is taking more running time than depth-first search algorithm by testing increasing size input. The experiment was done in C code, the running time might be different if we used other programming language such as C++ or Java that has more efficient functionalities for particular algorithm's implementation. In conclusion, practically in C language implementation, the depth-first search is more efficient for topological sorting. On the other hand, the depth-first search algorithm has less complexity in this particular circumstance.