Graph

Fredrik Berzins

Fall 2023

Introduction

A graph is any type of linked nodes, that can be a linked list, a double linked list or a tree. A graph is commonly used to map out road/tracks between intersections/station. Another use case for graphs is when manufacturing a complex assembly with multiple sub assembly's needing different parts in this use case it can give a weight to a relation to show the production time of that component and the sum of the relations for the longest path to the finished assembly shows the total time to build from scratch.

Implementation /W Explanation

Graph explanation

The graph in this case is a representation of cites(nodes) and the respective rail network(edges) between them. There are 52 cites and 75 tracks/ connections between them. Each track has a time(weight) this means if all times for each track a route passes are summed it gives a total route time.

Linear Implementation

Linear searches every possible solution up to a max route time, this is done so it can go on forever in a single loop since nothing is hindering it from thing it otherwise. The max value is also a major downside to both linear and Swap Max, since it means the route will only be found if the time is sufficient other wise it will just waste computing time.

Swap Max Implementation

Swap max works as linear but if when it finds a route it swaps the max route time to the current route time, since if it's longer then that it's not going to be the shortest route. If it's shorter it's an even better solution and then swaps the max value again.

No Duplicates Implementation

No duplicates holds an array with all previously visited cities and when moving it will check so it doesn't make a route that goes in a loop since that can never be the shortest route.

No Duplicates /w Swap Max Implementation

No duplicates /w swap max dose exactly what it says it combines the two. It uses a max value to eliminate possible routes earlier just like swap max but it also avoids loops by keeping track of previously visited cities.

Benchmarks

The beanchmarks was done as one single search for a routes, the routes tested was the following:

- Malmö to Göteborg (M/G)
- Göteborg to Stockholm (G/St)
- Malmö to Stockholm (M/St)
- Stockholm to Sundsvall (St/Su)
- Stockholm to Umeå (St/U)
- Göteborg to Sundsvall (G/Su)
- Sundsvall to Umeå (Su/U)
- Umeå to Göteborg (U/G)
- Göteborg to Umeå (G/U)

For the linear and swap max benchmarks the starting max values was set to the closest hole hour.

Result

The results show that the more restriction on the search the faster is going to be this is quite obvious, since their is fewer options to search its faster to go thru those options. The biggest difference in search time was from lowering the max route time. While eliminating possible loops gave a fewer options it did not have a max time this means it's still trying to effectively brute force but with a max number of cities in this case 52.

Route	**Linear**	**Swap Max**	**No Dup**	**No Dup /w Swap Max**
M/G	1.08	0.237	129	0.266
G/St	0.350	0.150	49.2	0.052
M/St	3.53	1.31	99.9	0.022
St/Su	5.98	9.52	75.8	1.61
St/U	4020	3680	106	6.94
G/Su	6350	3430	90.3	2.90
Su/U	0.008	0.008	270	171
U/G	0.241	0.151	92.0	0.034
G/U	6464000	4204000	132	4.48

Table 1: Time to in ms to find shortest route with different methods

Route	**Linear**	**Swap Max**	**No Dup**	**No Dup /w Swap Max**
M/G	2:33	2:33	2:33	2:33
G/St	3:31	3:31	3:31	3:31
M/St	4:33	4:33	4:33	4:33
St/Su	5:27	5:27	5:27	5:27
St/U	8:37	8:37	8:37	8:37
G/Su	8:35	8:35	8:35	8:35
Su/U	3:10	3:10	3:10	3:10
U/G	11:45	11:45	11:45	11:45
G/U	11:45	11:45	11:45	11:45

Table 2: Shortest route found for each respective search. (Same order as table 2.)

Conclusion

In conclusion no duplicates /w swap max was the fastest while swap max still might be worth the trade of in complexity for small graphs and especially since both no duplicate methods add extra memory complexity to store the previous cites. Linear is never going to be worth using since adding a changing max value is one row of code, this is also accurate when comparing no duplicates and no duplicates /w swap max