Report for HW3

-Tashi Tengyal (2020-15441)

Environment

Python 3.9.2

I/O

As per the directions given.

Time Measurement

Obtaining Strongly Connected Components using Kosaraju's algorithm mainly revolves around the Depth-First-Search technique. So, the generally accepted time complexity is O(|V|+|E|). However, this analysis attempts to dig deeper into the algorithm and find out the behaviors under different circumstances.

After building the three graph representations, various inputs were run to analyze and compare the difference in time taken by each graph representation. To acquire an even more in-depth analysis, each |V| were run two times i.e., first in the case of sparse graphs and second in the case of denser versions.

|V|	Type	AdjList (ms)	AdjMat (ms)	AdjArr (ms)
5	Sparse	0.04792213439941406	0.02980232238769531	0.0247955322265625
	Dense	0.04892321272141453	0.03180212668413283	0.02261395241413456
10	Sparse	0.12874603271484375	0.06008148193359375	0.05602836608886719
	Dense	0.13780593872070312	0.05698204040527344	0.05388259887695312
50	Sparse	2.0918846130371094	0.7512569427490234	0.5249977111816406
	Dense	3.865957260131836	0.8549690246582031	0.5629062652587891
100	Sparse	8.007287979125977	3.3540725708007812	1.9080638885498047
	Dense	26.585817337036133	3.0939579010009766	1.8057823181152344
1000	Sparse	953.596830368042	280.86400032043457	168.03503036499023
	Dense	1940.260887145996	251.58405303955078	137.13502883911133
5000	Sparse	18926.753759384155	7103.9721965789795	4270.1170444488525
	Dense	47178.03883552551	6760.72096824646	3384.039878845215

Discussion

Strictly judging by the results in the table above, it is quite intuitive to say that using Adjacency List takes a longer time compared to the other two. The reason is most probably because of the linked list implementation. To obtain the strongly connected components, Kosaraju's algorithm was used where depth first search is the primary tool. DFS requires traversal in a linked list which takes more time.

It turned out that AdjArr was the fastest among the three. AdjArr although, not as popular as AdjMat, is twice as fast as the latter. This shows that in situations where finding strongly connected components is the goal, AdjArr might be a good choice. Obviously, if given a dynamic graph, AdjArr would not be an ideal option.

It should be kept in mind that the data obtained in the table above might not be the most accurate because of the numerous ways a graph with |V| vertices can be used to have varying |E|. The table can be assumed as an average case.