Assignment 1 Report

BLG 336E–Analysis of Algorithms II

Name and surname: Jilan Alrehaili

Student number: 150160922

31.Mar.2020

Department of Computer Engineering

(a) Graph Implementation:

struct Node to keep my public values such as (PHP, PPP, BHP... etc) modified within every action that could take place later on. Also, created visited to check if this node is visited before or haven't been visited before. vector Node to keep the nodes values in it. Vector graph simulates the graph. StartAttack method has the nessecary nodes that will be used like thunder-shock, tackle, and skip...etc. And each node has the informations that were declared from attack names to damage level. Further, I inserts these nodes with their respective information into the graph. And at the end when the level is 3 and more we use the skip node on those levels. generateNodes method handles nodes by their level as it will be requested when starting the program. PrintGraph method handles the printing part of the simulated graph.

(b) BFS-DFS Implementation:

Breadth first search (BFS) : used graph to find the neighoring values of the vertex to assign two values for each then find the vertice that are not visited. Visit them then mark as visited. NodeCount keeps track of how many times this function is called.

Depth first search (DFS) : used to explore neighbor nodes first that are not visited, before moving to the next level. In more details, stack is used to push all the values to it and pop in a recursive call to avoid traversing the same node again and again infinite times, I made a boolean visited array to keep the values which was visited. NodeCount keeps track of how many times this function is called.

Analyze results: Based on the below information we gained from the table, I can say that BFS performance is better compared to DFS as the tree grows till level 10. While in this particular case it out performs DFS because the tree is really deep but when the tree gets very wide we may need to choose DFS; as BFS will take too much memory.

# of nodes	LEVEL	DFS (seconds)	BFS (seconds)
21	2	0.00247	0.000203
111	3	0.000869	0.000628
471	4	0.002248	0.001486
2631	5	0.015089	0.009935
11271	6	0.047324	0.035296
63111	7	0.240132	0.152813
270471	8	0.856983	0.468876
1514631	9	4.23069	2.31542
6491271	10	15.9739	7.12125

How to run the program:

g++ -o name hw.cpp
./name part2 2 <dfs/bfs>
or
./name part1 2