Analysis Of Algorithms II
Homework 2

Talha Çomak, 150160726

1- Intro

This is an individual assignment report includes graph algorithm. The aim of the assingment is simulating a Pokemon battle where a Pikachu is fighting against a Blastoise. And attributes of this battle should be stored in the graph.

There are 3 part in this assignment. For part 1 pikachu and blastoise HP values are 273 and 361 respectively. For part 2 and 3 both of that values are 200. The other attributes of the pokemons are determined according to given txt files and that values are hardcoded in the program.

2- Development and Operating Environments

The project was developed in a Visual Studio 2019 environment with C++ language. The program tested in Linux with g++ compiler. The command to compile the program is:

g++ -std=c++11 –Wall –Werror project1.cpp -o project1

To run the program some command line arguments are required.

For part1: ./project1 part1 <max_level>

For part2: ./project1 part2 <max_level> <dfs/bfs>

Where max_level is an integer between 0-11. In part2 one of dfs and bfs must be selected

For part3: ./project1 part3 <pikachu/blastoise>

In part3 pikachu or blastoise must be selected.

3- Data Structures and Libraries

For this assignment, i created two classes: Pokemon and Graph. To store the battle variables; fight, Stats, pikachu_node and blastoise_node structures are created. Also iostream, vector, cmath, ctime, cstdlib libraries are used in the program.

4- Program Flow

The flow is determined by which part is selected. For part 1 the graph is created according to max_level parameter. In the end of the program P_HP, P_PP, B_HP, B_PP and PROB variables of the nodes in the deepest level (equal to max_level) are printed out. Example run is given is the assignment pdf.

Also in the part 2 the graph is created according to max_level parameter. The output is determined by the third parameter: dfs or bfs. In both case the graph is traversing by these algorithms. Example runs are given below.

```
./project1 part2 9 dfs

(DFS) Node count: 1354731 Running time: 0.093

---------------------------------------------------------------
./project1 part2 9 bfs

(BFS) Node count: 1354731 Running time: 0.156

---------------------------------------------------------------
./project1 part2 10 dfs

(DFS) Node count: 5274927 Running time: 0.375

---------------------------------------------------------------
./project1 part2 10 bfs

(BFS) Node count: 5274927 Running time: 0. 641
```

In part 3 the graph is created according to end of the battle. The aim is find the easiest path for win condition. If pikachu wins, when the HP of the blastoise is zero the easiest path is found. To calculate the probabilty, the other nodes are created in this level. Then the easiest path is printed out. Example run is given below.

```
./project1 part3 pikachu

Pikachu used Thundershock. It's efficient.
Blastoise used Tackle. It's efficient.
Pikachu used Thundershock. It's efficient.
Blastoise used Tackle. It's efficient.
Pikachu used Slam. It's efficient.
Blastoise used Tackle. It's efficient.
Pikachu used Slam. It's efficient.

Level count: 7
Probability: 0.0000694444
```

5- Analyse the Results and Conclusion

If we want to keep all possibilities of an event in a graph, that process use very big memory for large values of max_level. However traversing on paths is very short process. Until the max_level 8, both dfs and bfs functions takes almost 0 second. For max_level = 10; (DFS) Node count: 5274927 Running time: 0.375. So traversing 5274927 path is takes less than half second with DFS algorithm. For level 10: (BFS) Node count: 5274927 Running time: 0.641. So BFS algorithm is also short. The Time complexity of both BFS and DFS should be O (V + E). However is a little time difference between that functions.