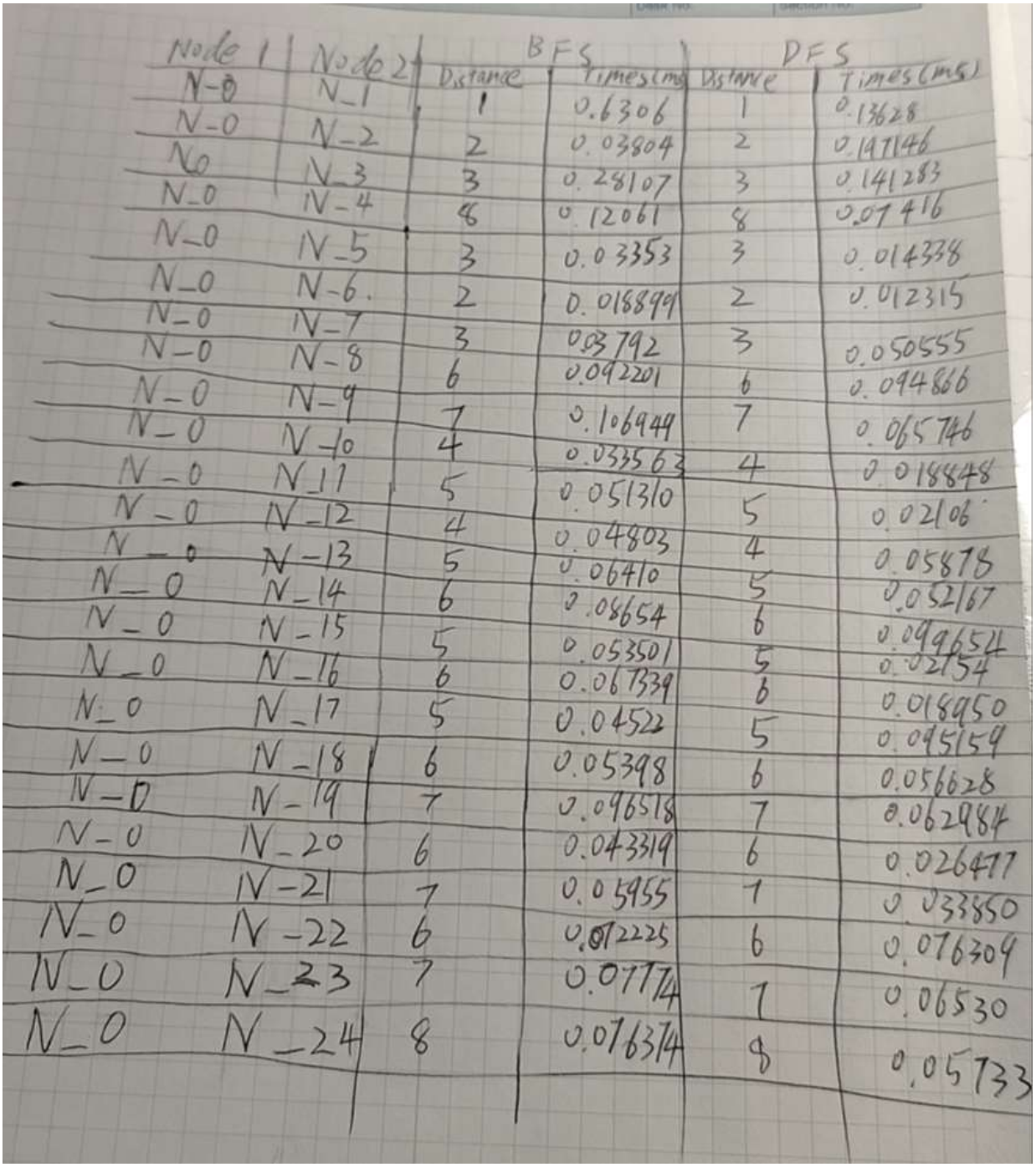
First, I made a HashMap<String, List<String>> graph to map the hash table. HashSet is used to check whether a vertex has been visited. Next is the bfs implementation part, I used Queue to implement it. I used Stack to implement the dfs part. The Node part is used to find distance, and the Result part is used to store distance and times. Then import the data through the Main part to get the output of distance and times. Since times is too small, I first converted ms to ns for output. Then convert ns to ms to create a table.



Node 1	Node 2	BFS Distance	BFS Times(ms)	DFS Distance	DFS Times(ms)
N-0	N_1	1	0.6306	1	0.13628
N-0	N_2	2	0.03804	2	0.147146
No	N_3	3	0.28107	3	0.141283
N_0	N-4	4	0.12061	8	0.07416
N_0	N_5	3	0.03353	3	0.014338
N_0	N-6.	2	0.018899	2	0.012315
N_0	N_7	3	0.03792	3	0.050555
N-0	N-8	6	0.092201	6	0.094866
N_0	N_9	7	0.106949	7	0.065746
N_0	N_10	4	0.033563	4	0.018848
N_0	N_11	5	0.051310	5	0.02106
N_0	N_12	4	0.04803	4	0.05878
N_0	N-13	5	0.06410	5	0.052167
N_0	N_14	6	0.08654	6	0.094654
N_0	N_15	5	0.053501	5	0.02154
N_0	N_16	6	0.067339	6	0.018950
N_0	N_17	5	0.04522	5	0.095159
N_0	N_18	6	0.05398	6	0.056628
N_0	N_19	7	0.096518	7	0.062984
N_0	N_20	6	0.043319	6	0.026477
N_0	N_21	7	0.05455	1	0.033850
N_0	N_22	6	0.012225	6	0.076304
N_0	N_23	7	0.0774	7	0.06530
N_0	N_24	8	0.076374	8	0.05733

Table1

1. Suppose you want to find a path between nodes at a shallow depth to your start node. Would you use BFS or DFS?

Answer1. According to Table 1, N_0 to N_2 and N_0 to N_3, their distances are 2 and 3 respectively. They belong to the paths between nodes with shallower depth. Comparing the time they take, it is obvious that BFS takes less time, so I will use BFS.

2. Suppose that the end node is at a very large depth from the start node. Would you use BFS or DFS?

Answer2. According to Table 1, N_0 to N_23 and N_0 to N_24, their distances are 7 and 8 respectively. They belong to the paths between nodes with deeper depth. Comparing the time they take, it is obvious that DFS takes less time, so I will use DFS.