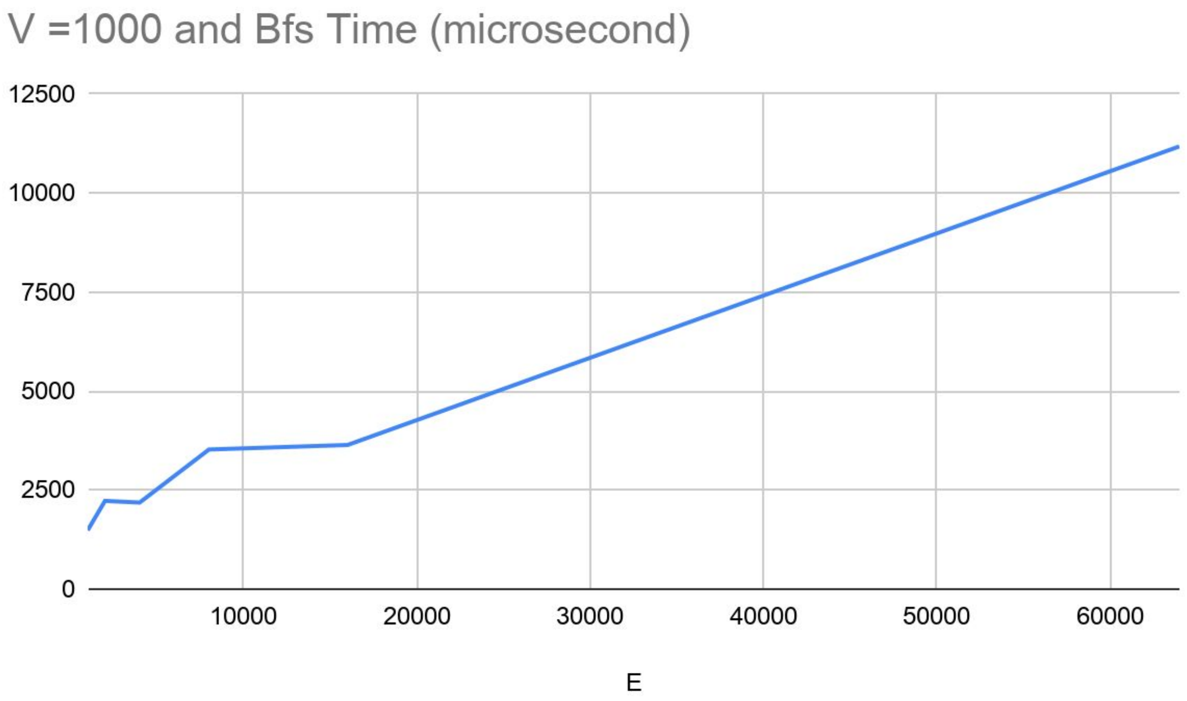
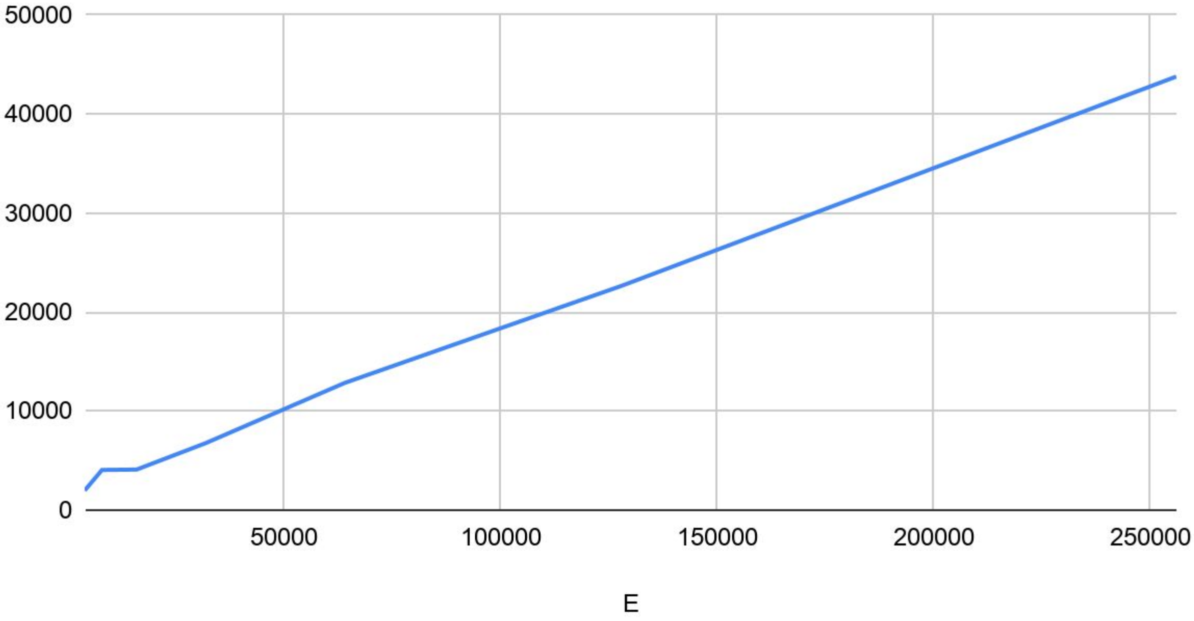
Adjacency List :

Data:

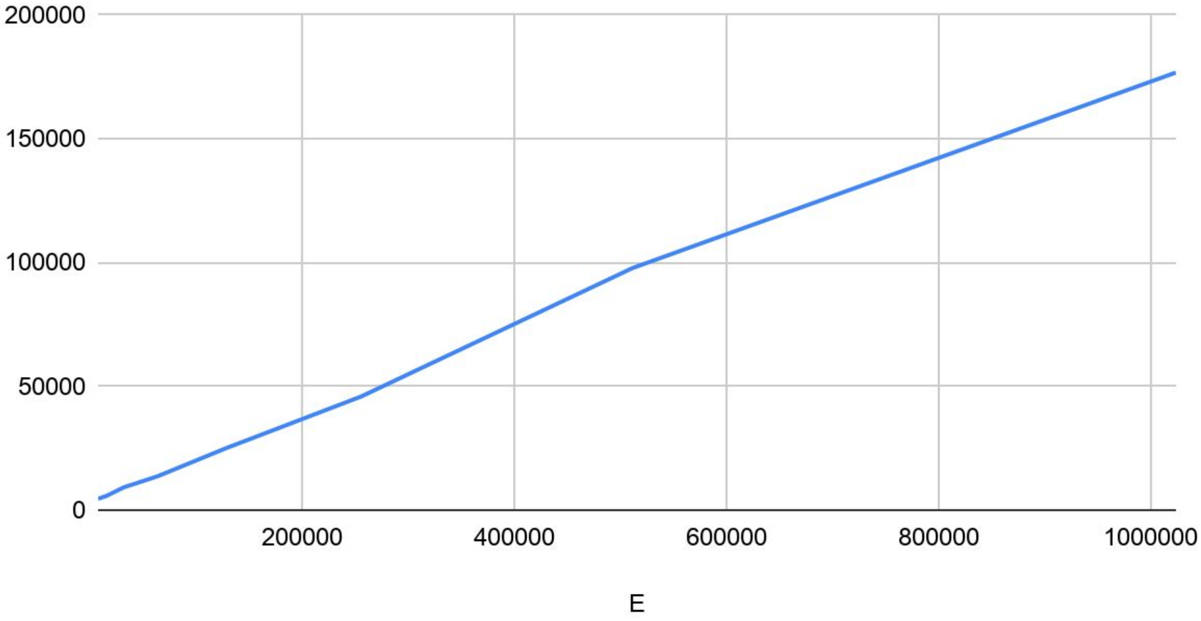
```
1000 1000 78.34
1000 2000 132.14
1000 4000 176.24
1000 8000 239.21
1000 16000 434.9
1000 32000 693.23
1000 64000 1276.8
mahdi@mahdi-HP-Pavilion-15-Notebook-PC:~/Desktop/New folder/2-2-Nodes/DSA 2/Offline 1$
cd "/home/mahdi/Desktop/New folder/2-2-Nodes/DSA 2/Offline 1"
mahdi@mahdi-HP-Pavilion-15-Notebook-PC:~/Desktop/New folder/2-2-Nodes/DSA 2/Offline 1$
./"solution"
1000 1000 82.22
1000 2000 132.59
1000 4000 166.92
1000 8000 274.16
1000 16000 394.94
1000 32000 723.88
1000 64000 1323.25
2000 2000 199.33
2000 4000 235.71
2000 8000 318.7
2000 16000 425.31
2000 32000 705.01
2000 64000 1202.52
2000 128000 2311.74
2000 256000 4493.37
4000 4000 264.36
4000 8000 471.26
4000 16000 619.72
4000 32000 957.22
4000 64000 1426.57
4000 128000 2724.56
4000 256000 5167.77
4000 512000 9810.81
4000 1024000 17277.6
8000 8000 514.57
8000 16000 919.1
8000 32000 1266.22
8000 64000 1805.28
8000 128000 2815.02
8000 256000 5285.69
8000 512000 9677.82
```

```
8000 1024000 19615.1
8000 2048000 37384.9
8000 4096000 76670.4
16000 16000 1285.38
16000 32000 1969.24
16000 64000 2935.89
16000 128000 4192.28
16000 256000 7565.89
16000 512000 11679.4
16000 1024000 20814.7
16000 2048000 39220.5
16000 4096000 72273.6
16000 8192000 143811
16000 16384000 284228
```

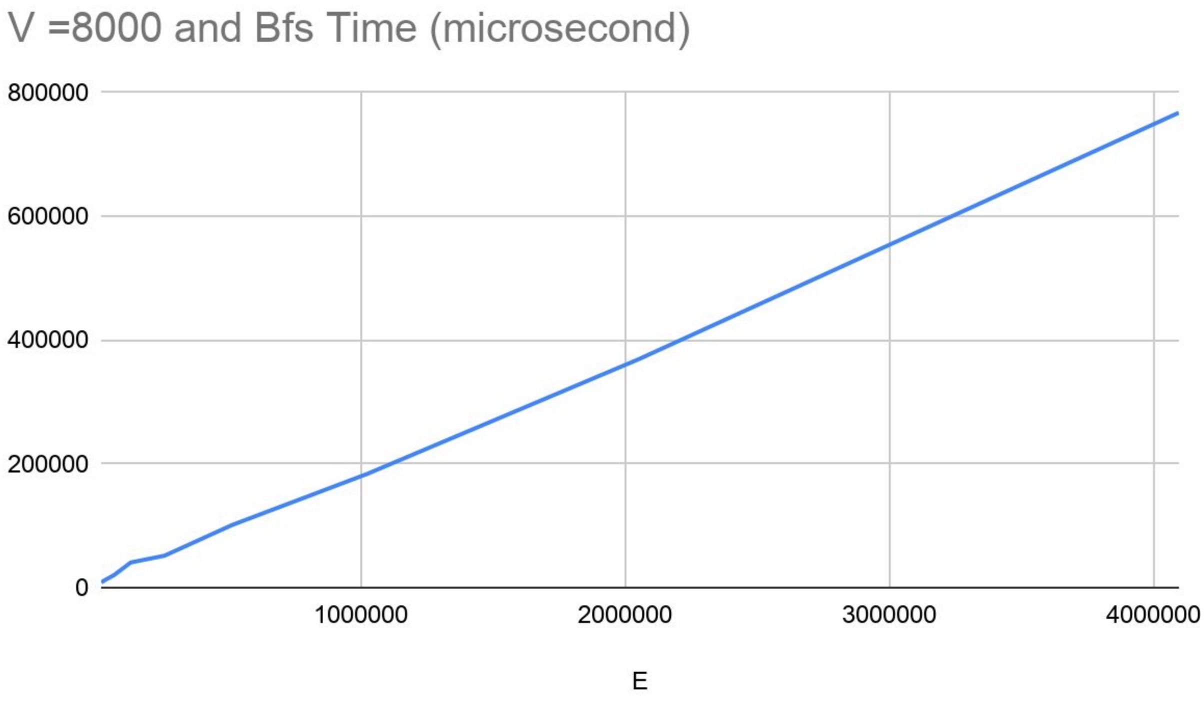


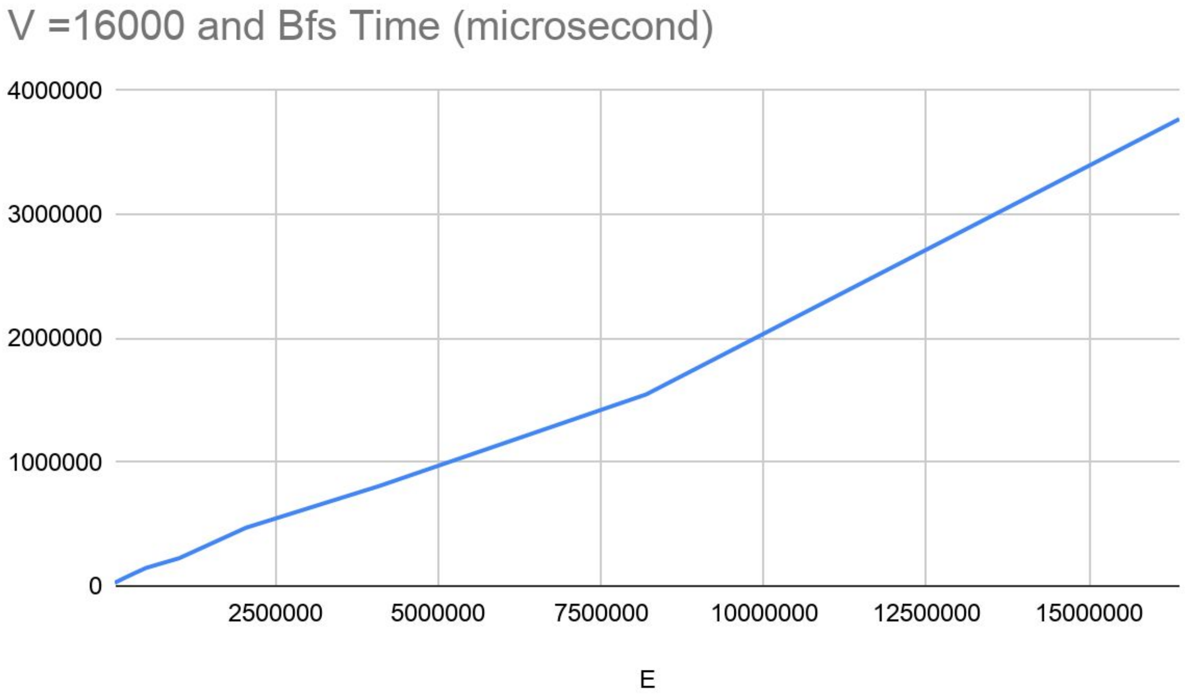


V =2000 and Bfs Time (microsecond)



V =4000 and Bfs Time (microsecond)



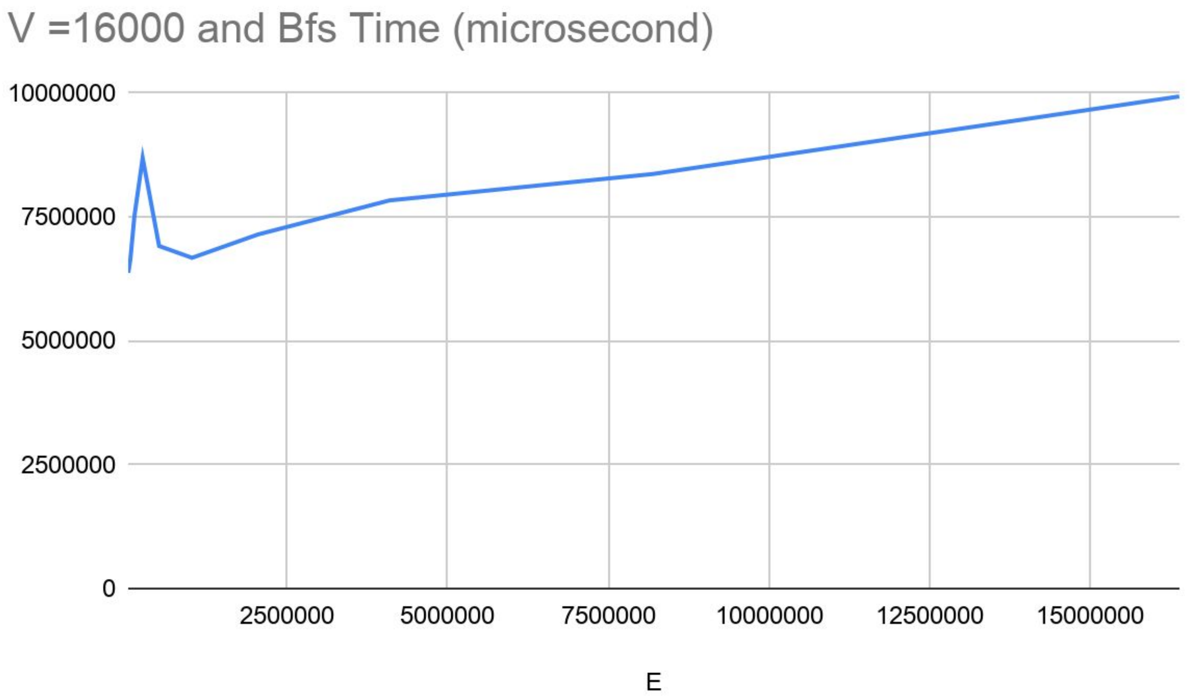


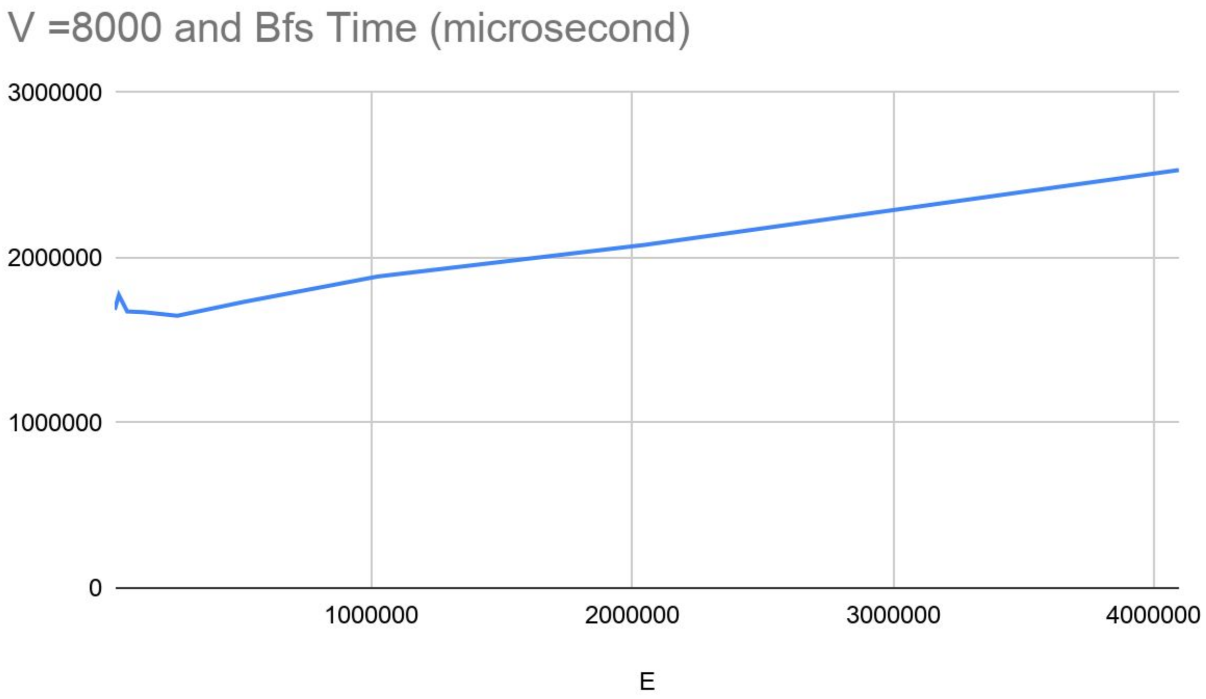
Charts for graph with adjacency matrix implementation :

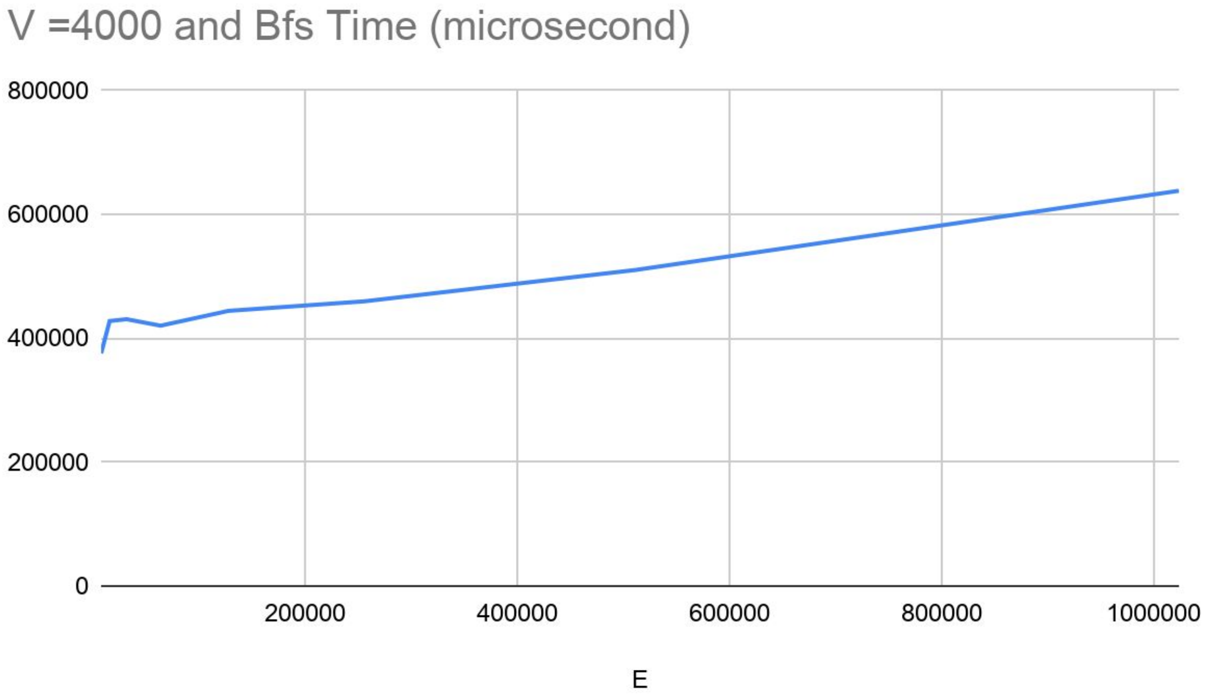
Data:

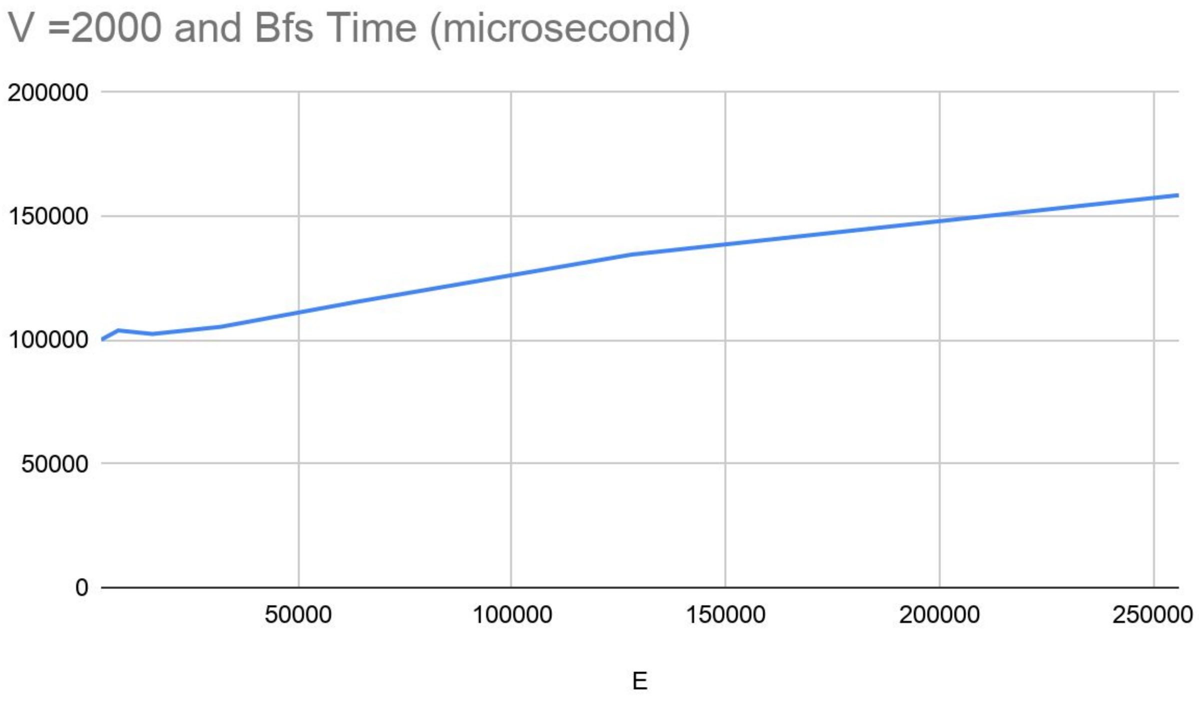
```
1000 1000 2648.6
1000 2000 3201.3
1000 4000 2979.8
1000 8000 3099.7
1000 16000 3430.8
1000 32000 4179.4
1000 64000 4823.6
2000 2000 7913.7
2000 4000 10356.3
2000 8000 17130.7
2000 16000 13374.7
2000 32000 10895.6
2000 64000 11364.4
2000 128000 12605.4
2000 256000 15844.2
4000 4000 22660.3
4000 8000 41319.6
4000 16000 43403.4
4000 32000 42709.6
4000 64000 43211.8
4000 128000 42923.7
4000 256000 46421.5
4000 512000 51245
4000 1024000 61967.7
8000 8000 91697.2
8000 16000 163065
8000 32000 165844
8000 64000 159744
8000 128000 168415
8000 256000 185123
8000 512000 194397
8000 1024000 233129
8000 2048000 235774
8000 4096000 265998
16000 16000 472327
16000 32000 725503
16000 64000 691829
16000 128000 657945
16000 256000 699442
16000 512000 668236
16000 1024000 757597
16000 2048000 739797
16000 4096000 778102
```

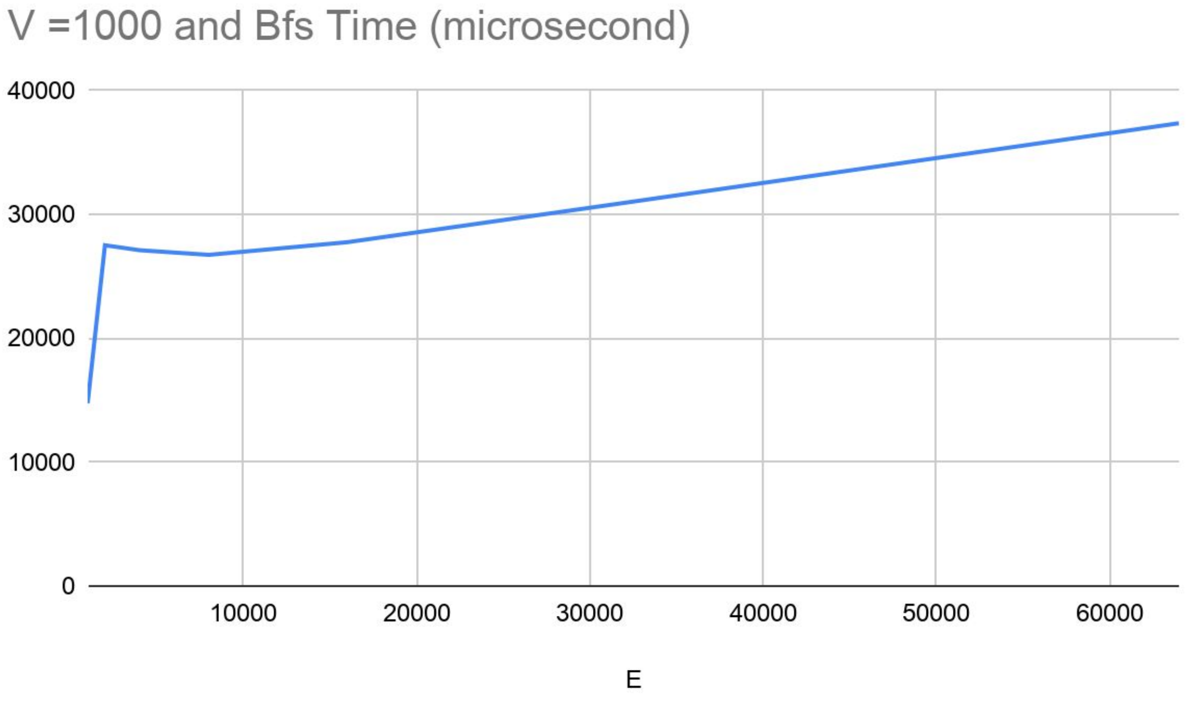
```
16000 8192000 852042
16000 16384000 1.03918e+06
```




V =8000 and Bfs Time (microsecond)
3000000
2000000
1000000
0
1000000
2000000
3000000
4000000
E


V =4000 and Bfs Time (microsecond)
800000
600000
400000
200000
0
200000
400000
600000
800000
1000000
E


V =2000 and Bfs Time (microsecond)


V =1000 and Bfs Time (microsecond)

1. What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency list? Why is it so?

Nodes , edges , bfs time
1000 1000 78.34
1000 2000 132.14
1000 4000 176.24
1000 8000 239.21
1000 16000 434.9
1000 32000 693.23
1000 64000 1276.8
From the data we can say runtime increase rapidly with the increase of edges.
Since time complexity of bfs in adjacency list implementation = O(n+m)

2. What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency list? Why is it so?
   Node , edge , bfs time
   1000 1000 83.58
   2000 2000 198.25
   4000 4000 281.62
   8000 8000 592.01
   16000 16000 1251.91
   From the data we can say runtime increase rapidly with the increase of node.
   Since time complexity of bfs in adjacency list implementation = O(n+m)

3. What is the impact on runtime if we keep |V| unchanged and double |E| for adjacency matrix? Why is it so?

   Node , edge , bfs time for matrix
   1000 1000 2011.57
   1000 2000 2773.58
   1000 4000 2916.38
   1000 8000 2770.23
   1000 16000 2887.57
   1000 32000 3126.6
   1000 64000 3754.11

From the Data we can say runtime does not increase so much with increase of edges

We also know that bfs complexity in Adjacency Matrix Implementation = O(n*n)

4. What is the impact on runtime if we keep |E| unchanged and double |V| for adjacency matrix? Why is it so?

   Node , edge , bfs time for matrix
   1000 1000 2018.18
   2000 2000 6697.65

4000 4000 24776.8
8000 8000 102175
16000 16000 457264

From the data we can say runtime increase rapidly with increase of edges

We also know that bfs time complexity in Adjacency matrix implementation = O(n*n)

5. For the same |E| and |V|, why are the runtimes for adjacency list and adjacency matrix representation different? Which one is higher and why?

   Time complexity of bfs depends on implementation of Graph .
   In case of adjacency list implementation complexity = O(n+m)
   In case of adjacency matrix implementation complexity = O(n*n)
   So for sparse graph Adjacency list implementation Graph perform better since complexity = O(n)
   For dense graph any of two implementation will result almost same running time.