Sequential

1000 Nodes

```
Node 991 BC = 2612.422363
Node 992 BC = 7769.836426
Node 993 BC = 2853.305420
Node 994 BC = 3206.027588
Node 995 BC = 2526.590088
Node 996 BC = 6368.935059
Node 997 BC = 174.078094
Node 998 BC = 2785.130615
Node 999 BC = 501.523682
Node 1000 BC = 5148.713379
BC of graph: 13682.895508
Time taken by program is : 76.612306 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/seq$
```

10,000 Nodes

```
Node 9988 BC = 11826.147461
Node 9989 BC = 33284.664062
Node 9990 BC = 33844.531250
Node 9992 BC = 16429.302734
Node 9993 BC = 16541.593750
Node 9994 BC = 37307.871094
Node 9996 BC = 21152.587891
Node 9997 BC = 13749.150391
Node 9998 BC = 13781.196289
Node 9999 BC = 3701.997070
Node 10000 BC = 64646.992188
BC of graph: 255227.093750
Time taken by program is : 130.722783 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/seq$
```

Parallel

1000 Nodes

```
Node 715 BC = 3061.962158
Node 755 BC = 1946.609497
Node 795 BC = 7899.212402
Node 835 BC = 3710.549072
Node 875 BC = 1475.739258
Node 915 BC = 2468.165039
Node 955 BC = 664.320496
Node 995 BC = 2494.316406
BC of graph: 13429.852539
Time taken by program is : 0.053204 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/parallel$
```

10,000 Nodes

```
Node 9795 BC = 9878.930664
Node 9835 BC = 57948.007812
Node 9875 BC = 28230.451172
Node 9915 BC = 23155.443359
Node 9955 BC = 12078.264648
Node 9886 BC = 34953.761719
Node 9926 BC = 9509.125000
Node 9966 BC = 23660.789062
BC of graph: 255227.078125
Time taken by program is : 1.976344 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/parallel$
```

Parallel with varying logical Threads

1000 Nodes

```
Node 988 BC = 2321.466797
Node 989 BC = 457.508972
Node 990 BC = 921.835205
Node 991 BC = 2612.423340
Node 992 BC = 7768.070312
Node 993 BC = 2852.107178
Node 994 BC = 3206.025635
Node 995 BC = 2526.088623
Node 996 BC = 6368.437012
Node 997 BC = 173.578049
Node 998 BC = 2785.129639
Node 999 BC = 501.523712
Node 1000 BC = 5148.195312
BC of graph: 13667.792969
Time taken by program is : 0.141867 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/Parallel_With_Varrying_Threads$
```

10,000 Nodes

```
Node 6989 BC = 11890.654297
Node 6990 BC = 63935.953125
Node 6991 BC = 35618.812500
Node 6992 BC = 42750.527344
Node 6993 BC = 67390.929688
Node 6994 BC = 36646.730469
Node 6995 BC = 5259.403320
Node 6996 BC = 33870.761719
Node 6997 BC = 53197.234375
Node 6999 BC = 24555.515625
Node 7000 BC = 37096.304688
BC of graph: 255179.656250
Time taken by program is : 2.001445 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/Parallel_With_Varrying_Threads$
```

1,00,000 Nodes

```
Node 92495 BC = 427879.156250
Node 92496 BC = 461146.406250
Node 92497 BC = 793396.250000
Node 92498 BC = 718131.625000
Node 92499 BC = 89027.960938
Node 92500 BC = 627358.125000
Node 92501 BC = 352951.781250
BC of graph: 3900481.000000
Time taken by program is : 559.458491 sec
shavak@shavak-X11DAi-N:~/Documents/Ahad P0518 Project/Betweenness-Centrality/Parallel_With_Varrying_Threads$
```

P1 dataset 10,000 Nodes of dataset

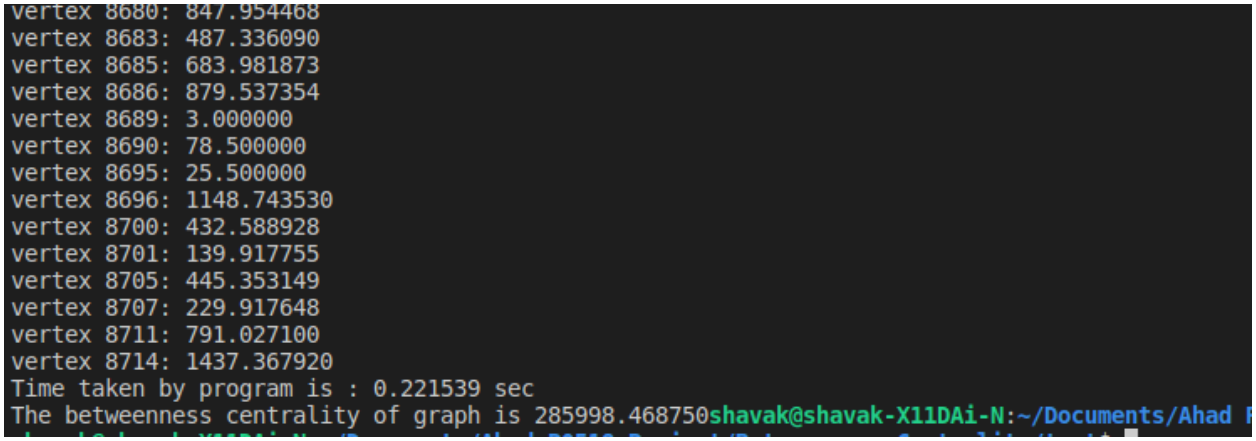
16 threads

```
vertex 8667: 850.843848
vertex 8669: 940.511892
vertex 8670: 402.752799
vertex 8671: 508.144673
vertex 8672: 360.196492
vertex 8674: 22.323676
vertex 8677: 1091.526021
vertex 8678: 1387.751688
vertex 8680: 852.980499
vertex 8683: 499.208700
vertex 8685: 694.457832
vertex 8686: 887.412263
vertex 8689: 3.000000
vertex 8690: 78.500000
vertex 8695: 25.500000
vertex 8696: 1155.159880
vertex 8700: 435.935865
vertex 8701: 138.056668
vertex 8705: 450.948737
vertex 8707: 232.615930
vertex 8711: 800.890380
vertex 8714: 1455.235239
Time taken by program is : 0.362151 sec
The betweenness centrality of graph is 294004.593750shavak@shavak-X11DAi
```

32 threads

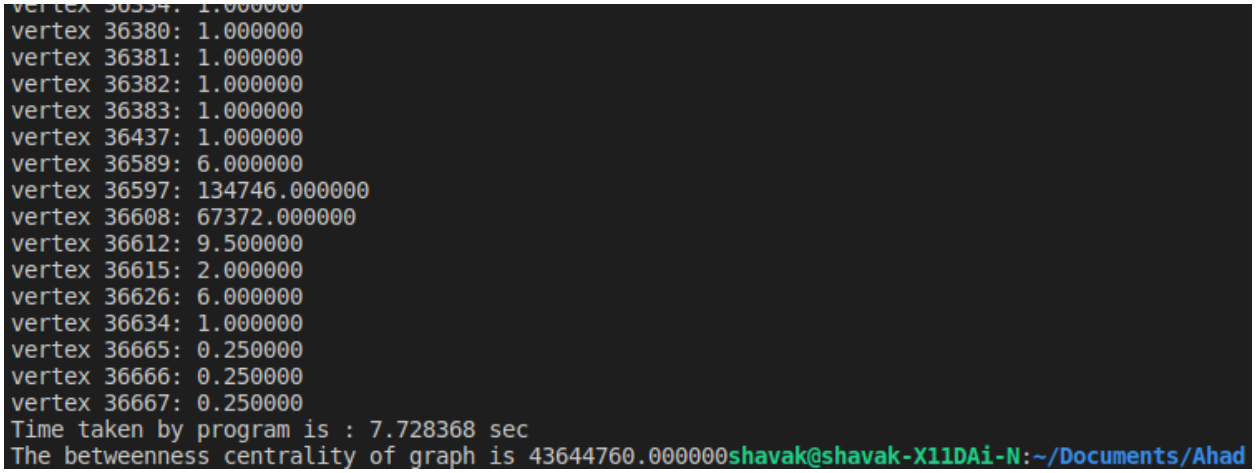
```
vertex 8685: 689.437317
vertex 8686: 878.456726
vertex 8689: 3.000000
vertex 8690: 78.500000
vertex 8695: 25.500000
vertex 8696: 1142.156494
vertex 8700: 431.910980
vertex 8701: 139.764984
vertex 8705: 445.353729
vertex 8707: 230.282730
vertex 8711: 786.771484
vertex 8714: 1447.455444
Time taken by program is : 0.230210 sec
The betweenness centrality of graph is 289995.437500shavak@shavak-X11DAi-N:~/Documents/Ahad P
```

40 threads

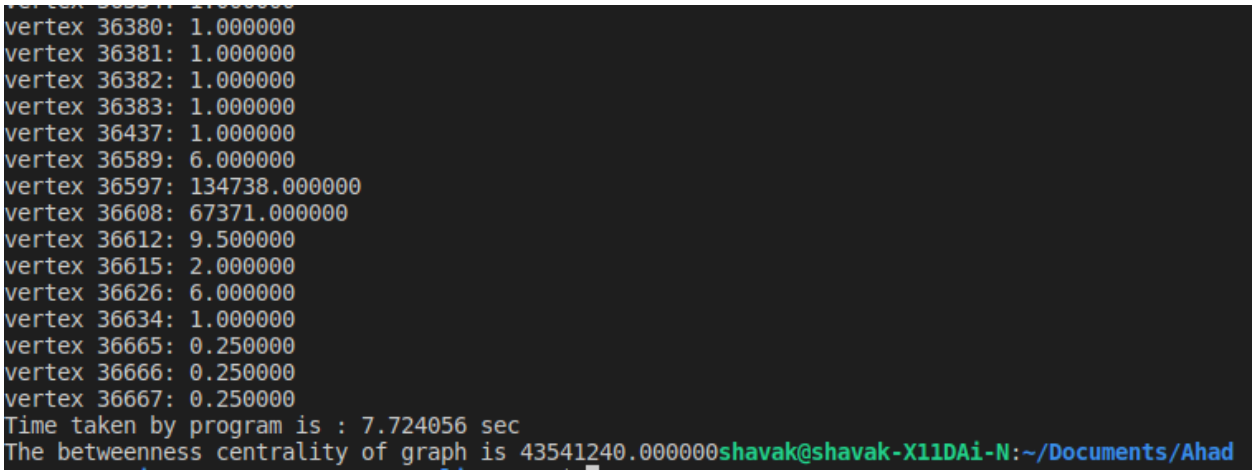


Email-Enron dataset

16 Threads



32 threads



40 threads

```
vertex 36380: 1.000000
vertex 36381: 1.000000
vertex 36382: 1.000000
vertex 36383: 1.000000
vertex 36437: 1.000000
vertex 36589: 6.000000
vertex 36597: 134738.000000
vertex 36608: 67378.000000
vertex 36612: 9.500000
vertex 36615: 2.000000
vertex 36626: 6.000000
vertex 36634: 1.000000
vertex 36665: 0.250000
vertex 36666: 0.250000
vertex 36667: 0.250000
Time taken by program is : 7.772647 sec
The betweenness centrality of graph is 43645580.000000shavak@shavak-X11DAi-N:~/Documents/Ahad
```

Wiki-vote data set

16 threads

```
vertex 7862: 4427.322754
vertex 7874: 36.425983
vertex 7882: 7269.521484
vertex 7910: 2536.212158
vertex 7924: 58.701458
vertex 7928: 1337.742920
vertex 8021: 1157.871704
vertex 8037: 114.185532
vertex 8042: 1201.307983
vertex 8050: 57.760818
vertex 8051: 6141.307129
vertex 8134: 12.472527
vertex 8204: 2.016459
vertex 8263: 7.486150
vertex 8271: 1.698715
Time taken by program is : 0.371150 sec
The betweenness centrality of graph is 444838.218750shavak@shav
```

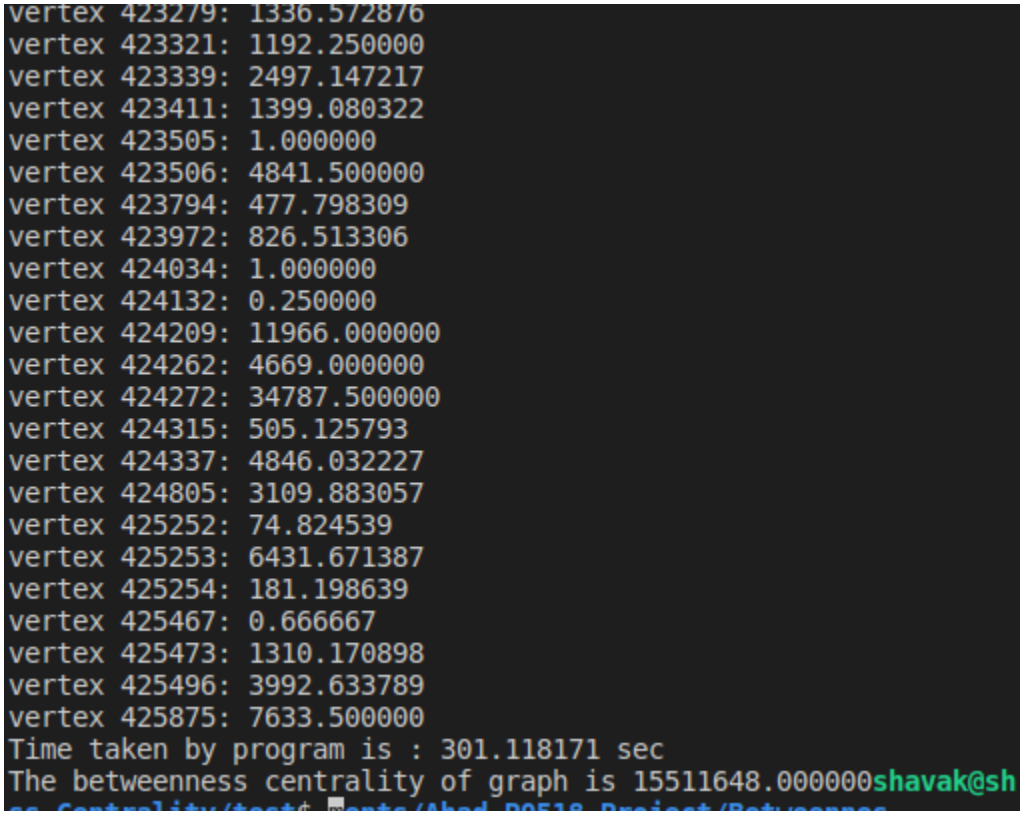
32 threads

```
vertex 7860: 3537.561523
vertex 7862: 4418.170898
vertex 7874: 36.130730
vertex 7882: 7265.913574
vertex 7910: 2535.841797
vertex 7924: 58.466805
vertex 7928: 1321.925903
vertex 8021: 1157.871704
vertex 8037: 113.539597
vertex 8042: 1201.259766
vertex 8050: 57.350941
vertex 8051: 6140.718750
vertex 8134: 12.972528
vertex 8204: 2.016459
vertex 8263: 7.468472
vertex 8271: 1.698715
Time taken by program is : 0.256721 sec
The betweenness centrality of graph is 442168.093750shavak@shav
```

40 threads

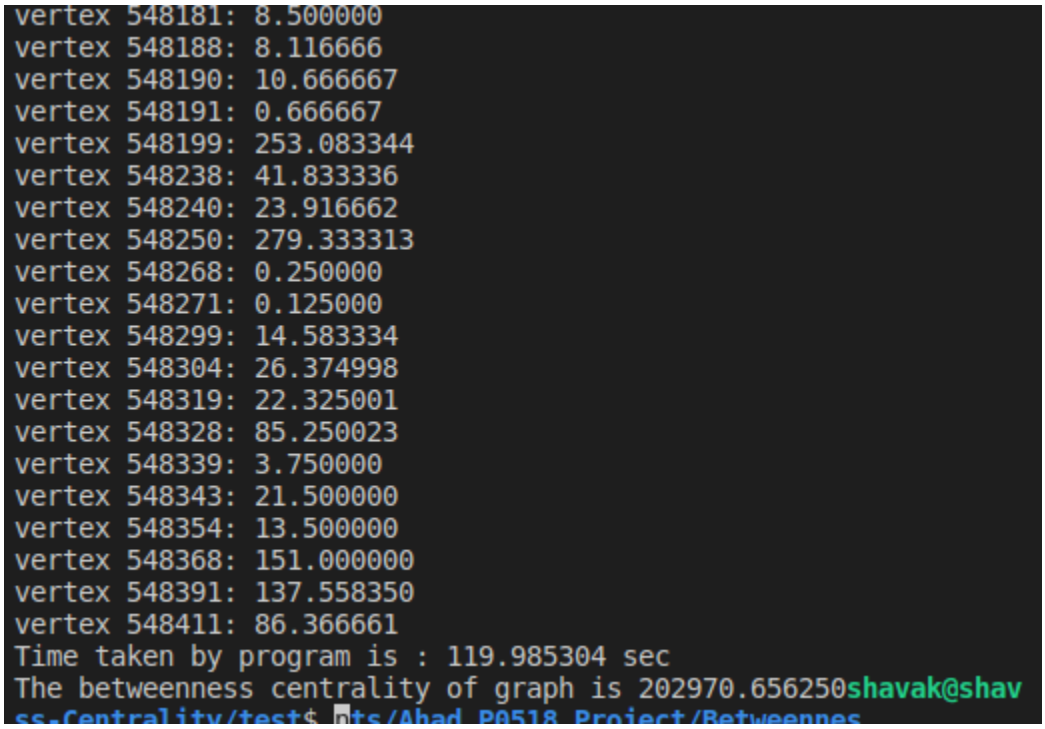
```
vertex 7862: 4416.470215
vertex 7874: 36.386360
vertex 7882: 7265.417969
vertex 7910: 2536.366455
vertex 7924: 58.023907
vertex 7928: 1337.659912
vertex 8021: 1157.871704
vertex 8037: 113.798042
vertex 8042: 1201.302124
vertex 8050: 57.650249
vertex 8051: 6142.503418
vertex 8134: 12.801097
vertex 8204: 2.016459
vertex 8263: 7.421719
vertex 8271: 1.698715
Time taken by program is : 0.216035 sec
The betweenness centrality of graph is 441462.437500shavak@shav
```

Com_dblp.ungraph
40 threads

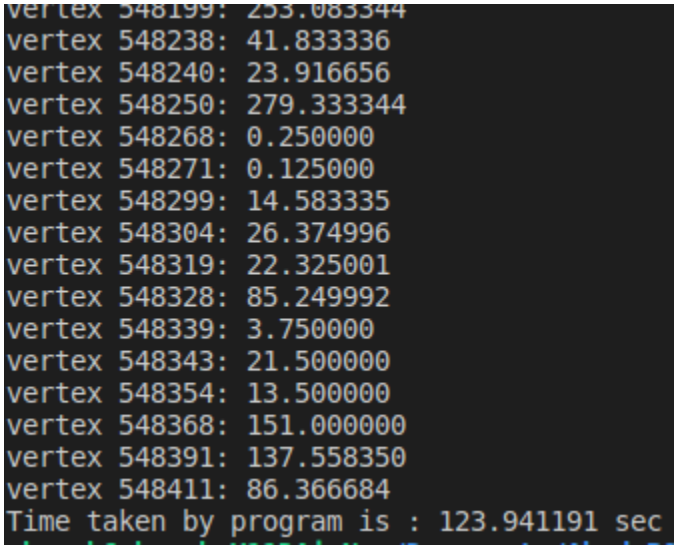


Com_amazon.ungraph

32 Threads



40 Threads

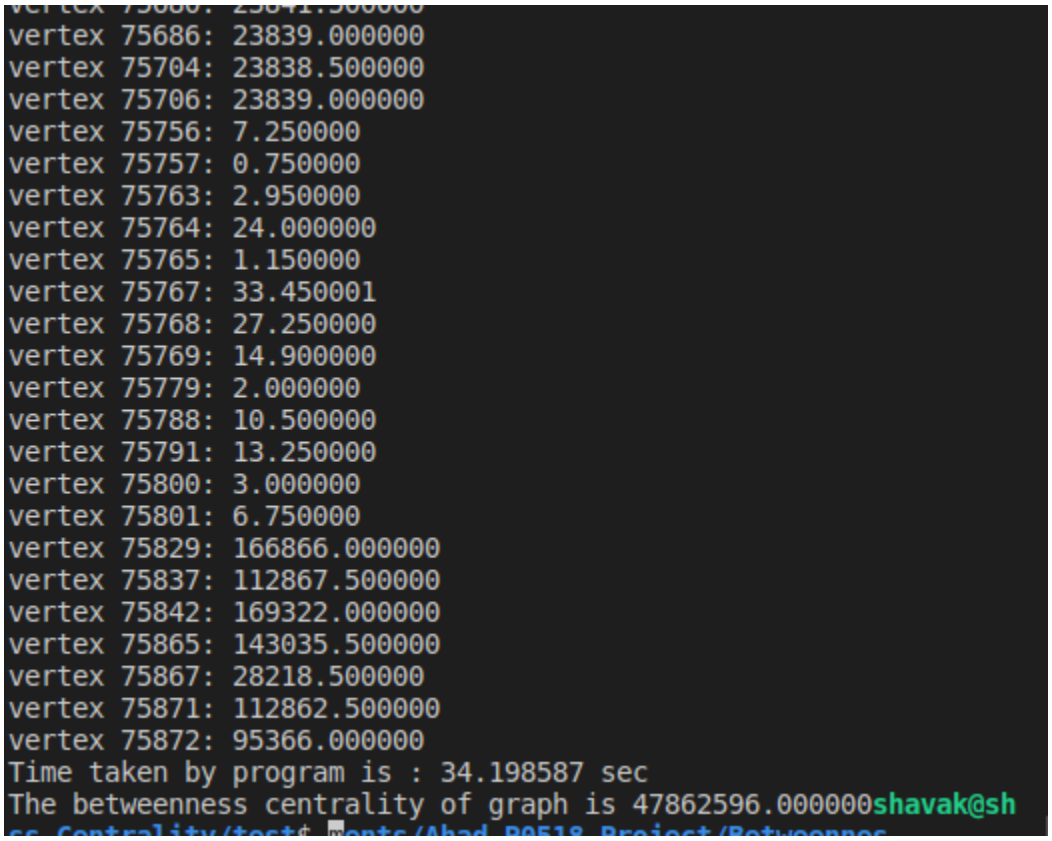


Com-youtube.ungraph

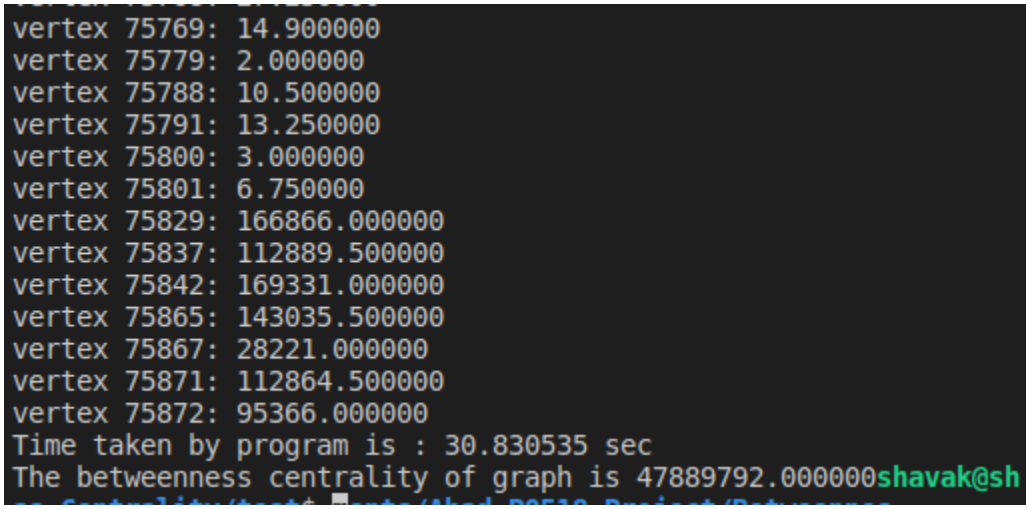
32 Threads
// could not run
40 Threads
// could not run

Soc-epinions dataset

32 Threads



40 Threads



Code:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>
#include <set>
#include <bits/stdc++.h>
#include <string>
#include <queue>
#include <stack>
#include <mutex>
#include <thread>
#include <chrono>
#include <time.h>
using namespace std;

// #define v_size 367663

vector<vector<int>> graph;
vector<int> vertices;
int numberOfVertices;

int main(int argc, char *argv[])
{
```

```
string inputFileName(argv[1]); //, outputFileName(argv[2]);
// int numberOfThreads = stoi(strThreads);
ifstream input(inputFileName);
// ofstream output(outputFileName);
int freeIndex = 0;
struct timespec start, end;

vector<pair<int, int>> edges;
map<int, int> myMap;
// start = clock();
printf("clock started\n");
clock_gettime(CLOCK_MONOTONIC, &start);
ios_base::sync_with_stdio(false);
printf("Reading file\n");
while (!input.eof())
{
    int u, v;
    input >> u >> v;
    edges.push_back(make_pair(u, v));
    myMap.insert(make_pair(u, 0));
    myMap.insert(make_pair(v, 0));
    if (input.eof())
        break;
}
input.close();
printf("Done reading file\n");
for (auto &m : myMap)
    m.second = freeIndex++;
for (const auto &m : myMap)
    vertices.push_back(m.first);
numberOfVertices = freeIndex;
graph.resize(freeIndex);
for (const auto &p : edges)
{
    int u = p.first, v = p.second;
    graph[myMap[u]].push_back(myMap[v]);
}


//vector<double> betweenness(numberOfVertices, 0.0);
float betweenness[numberOfVertices];
printf("graph created...\n\n");
// for (int i = 0; i < graph.size(); i++) {
//     for (int j = 0; j < graph[i].size(); j++)
```

```
//         cout << graph[i][j] << " ";
//     cout << endl;
// }

printf("Initializing BC\n");

#pragma omp parallel for num_threads(40)
for (int v = 0; v < numberOfVertices; v++)
{
   betweenness[v] = 0.0;
}
printf("Starting BFS for each vertex...\n");

clock_gettime(CLOCK_MONOTONIC, &start);
ios_base::sync_with_stdio(false);

#pragma omp parallel for num_threads(40)
for (int s = 0; s < numberOfVertices-1; s++)
{
   stack<int> S;
   vector<vector<int>> P(numberOfVertices);
   vector<int> sigma(numberOfVertices, 0);
         vector<int> d(numberOfVertices, -1);
         vector<double> delta(numberOfVertices, 0.0);

   sigma[s] = 1;
   d[s] = 0;
   queue<int> Q;
   Q.push(s);
   while (!Q.empty())
   {
      int v = Q.front();
      Q.pop();
      S.push(v);
      for (const auto &w : graph[v])
      {
         if (d[w] < 0)
         {
            Q.push(w);
            d[w] = d[v] + 1;
         }
         if (d[w] == d[v] + 1)
         {
            sigma[w] += sigma[v];
```

```
                P[w].push_back(v);
            }
        }
    }
    // #pragma omp parallel for num_threads(40)
    // for (int i = 0; i < numberOfVertices; i++)
    // {
    //     delta[i] = 0.0;
    // }

    while (!S.empty())
    {
        int w = S.top();
        S.pop();
        for (const auto &v : P[w])
        {
            delta[v] += ((double)sigma[v] / (double)sigma[w]) * (1.0 + delta[w]);
        }
        if (w != s)
        {
            betweenness[w] += delta[w] / 2;
        }
    }
}
printf("Completed BFS..\n");
clock_gettime(CLOCK_MONOTONIC, &end);

float max = -1.0;
// #pragma omp parallel for num_threads(40)
for(int i=0; i<freeIndex; i++) {
        if(graph[i].size() > 0 && betweenness[i]>0){
                printf("\nvertex %d: %f", vertices[i], betweenness[i] );
        }
    if(betweenness[i] > max)
    {
        max = betweenness[i];
    }
}

printf("\nThe betweenness centrality of graph is %f", max);


double time_taken;
time_taken = (end.tv_sec - start.tv_sec) * 1e9;
```

```
    time_taken = (time_taken + (end.tv_nsec - start.tv_nsec)) * 1e-9;

    cout << "Time taken by program is : " << fixed
         << time_taken << setprecision(9);
    cout << " sec" << endl;
    return 0;
}
```

Algorithm:

Algorithm Parallel Betweenness Centrality

Input Graph $G = (V, E)$
DECLARE r_size := 10001
DECLARE c_size := 30100

Begin
Procedure parallelBC_With_BFS(Graph G, r_size, c_size)
For each node s in G **in Parallel do**
DEFINE S AS stack
DEFINE d AS array //Distance vector
DEFINE Q AS queue //Queue for BFS
DEFINE $sigma$ AS array
DEFINE P AS map of vector //Paths
For i:=0 To 10 **in Parallel do** //Initialize BFS variables
$D[i]$:= -1
$Sigma[i]$:= 0
End For
$Sigma[s]$:= 1
$d[s]$:= 0
ENQUEUE Q := s
While Q is NOT empty **do** //Perform BFS
v := FRONT of Q
DEQUEUE Q
PUSH v INTO S
DECLARE m AS integer
If v + 1 >= r_size **then**

```
        m := c_size
    Else
        m := r[v+1]
    End if
    For j:=r[v] To r[v+1] do                    //Traverse neighbors of v
        W := c[j]
        If d[w] < 0 then
            PUSH w into Q
            d[w] := d[v] + 1
        End if
        //Update sigma value of w if path through v is shortest
        If d[w] == d[v] + 1 then
            Sigma[w] := sigma[w] + sigma[v]
            PUSH_BACK v into P[w]
        End if
    End For
```