CS 165

Project 3 Report

Wenzhuo Wang

14444481

1) graph algorithm

Erdos-Renyi random graphs:

This algorithm requires input of number of nodes (n) and the probability (p). The output are all edges formed after algorithm. The pseudo-code is:

```
Edges = [];              //set an empty edges collection
V=1; w=-1;               //the initial vertex
While v <= n:            //from every vertex starts from 1 to n
    Draw R from [0,1), uniformly at random      //generate radius
    w = w+1+(log(1-r)/log(1-p));        //wait from the 0s interval
    while((w>=v) && (v<=n))
        w=w-v;
        v=v+1;
   if v<=n then Edges add {v,w}         //make sure that v in range(1,n), add this edge to Edges
```

Barabasi-Albert random graphs:

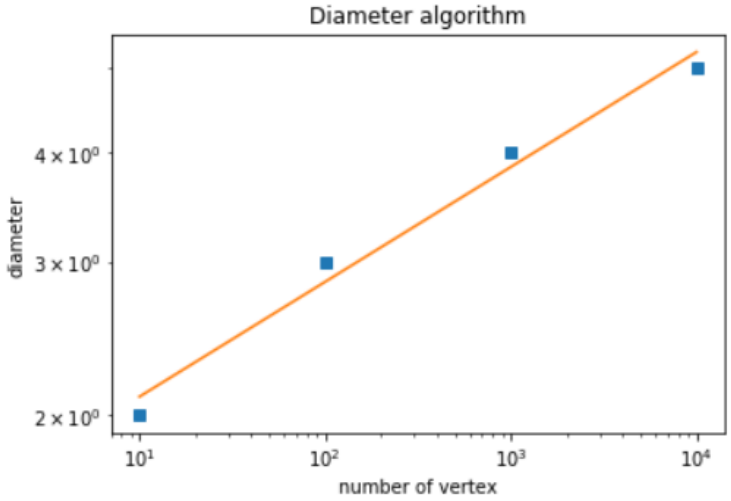
This algorithm requires number of vertices (n) and number of connection of every vertex (d). The The pseudo-code is:

```
Edges = [];              //set an empty edges collection
int M[2*n*d+1];          //the number of total edges is fixed to n*d
for every vertex:
    for i = 0 to d-1:
        M[2*(v*d+i)]=v;          //each vertex has d edges go out.
                //i.e. the first vertex will occupy M[0],M[2],M[4],M[6],M[8] as start of edges
        Draw R from [0, 2*(v*d+i)], uniformly at random
        M[2*(v*d+i)+1]=M[r];  //for even number of index of M, connect it with a random vertex r
For i = 0 to n * d:
    Edges add { M[2*i], M[2*i+1] }    //for every pair in M, add an edge to Edges
```

2) Diameter algorithm

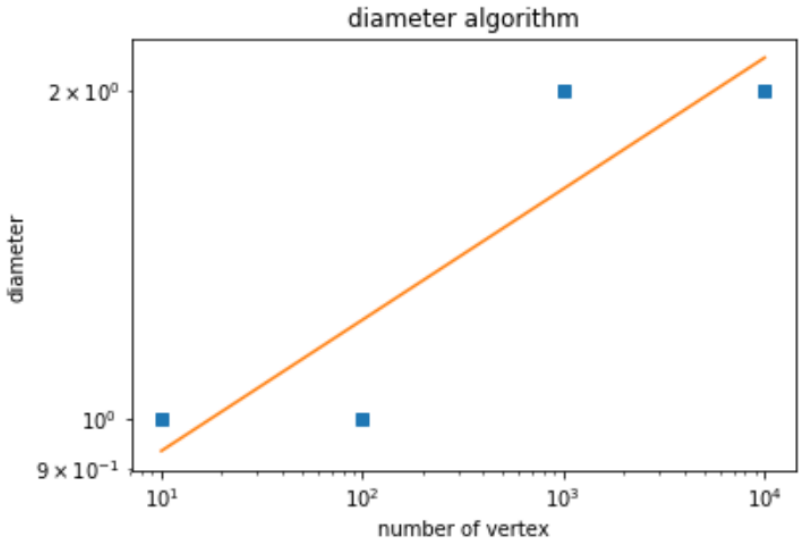
This algorithm finds the longest path in the graph. Because generating longest path from every vertex will take a long time if the number of vertices is large, I chose to use the second heuristic idea. The algorithm will randomly select one vertex and perform BFS from vertex r. Then it will store the longest distance it could achieve and the final vertex it achieved. Supposedly, this final vertex is at the outer link in the graph. Then the algorithm will perform another BFS from this final vertex and see if the longest path changed. If the new path is larger than the old one, the old one will be replaced with the new one.

Erdos-Renyi random graphs:



The diameter grows as the number of vertices increasing. The grow rate is proportional to log n.

Extra Barabasi-Albert random graphs:



The diameter grows as the number of vertices increasing. The grow rate is proportional to log n.

3) Clustering-coefficient algorithm

The idea of clustering-coefficient is that if A is connected to B and B is connected to C, then there is a high probability that A is connected to C.

The clustering coefficient C is:

C = 3*number of triangles divided by the number of 2-edge paths

Computing the number of 2-edge paths:

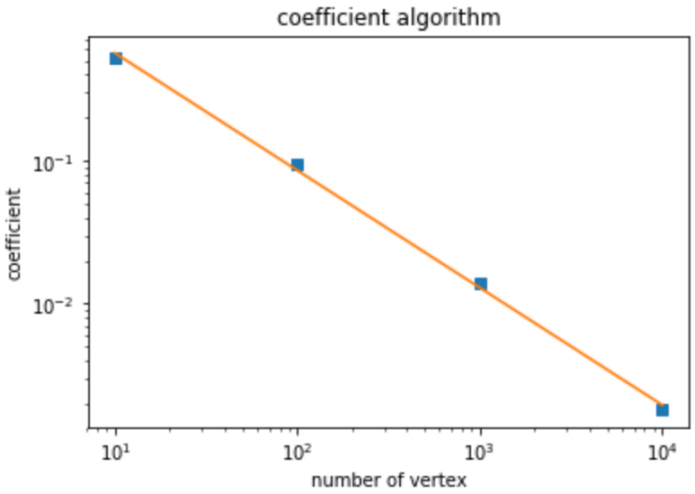
For each vertex, find its degree (its neighbor). Then the number of 2-edge paths for this vertex is degree * (degree-1) /2

Computing the number of triangles:

First compute a degeneracy order of the vertices.

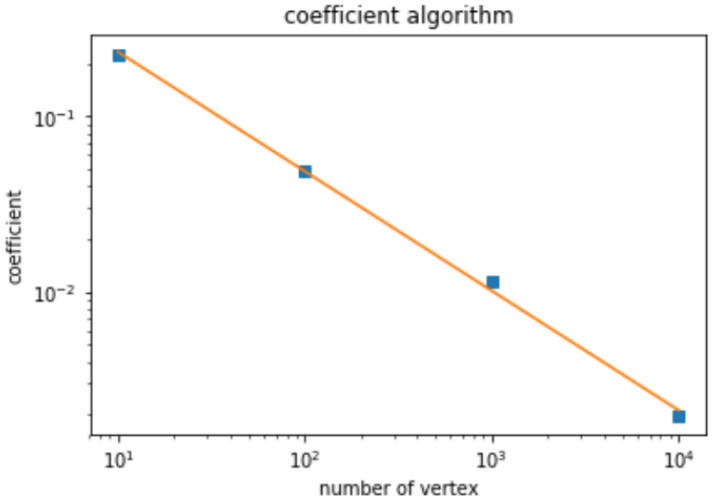
```
L = []              //the output degeneracy order
For each vertex V, compute dv which is the degree of V.
Initialize D, for each vertex V, D[dv] add V
Initialize N, for each vertex V, N[V] is a list contains the neighbors of v that come before v in L. Initially, it is empty for every vertex.
Repeat n times://n is the number of vertices
    int i=0;
    while (D[i].size()==0)
        i++;        //find smallest i such that D[i] is not empty.
    Node v = D[i].back();
    D[i].pop_back();            //Pop a vertex from D[i]
    l.push_front(v);            //push this vertex to L and mark this vertex as in L
    for each node W which is neighbor of V:
        if W is not in L:
            d[W]-=1;
            D[d[W]+1].remove(w);
            D[d[W]].push_back(w);        //move W to D which has one neighbor less.
            N[V].push_back(w);           //W comes before V in L
```

Erdos-Renyi random graphs:



The clustering-coefficient decreases as the number of vertices increasing. The coefficient approaches p as n grows.

Extra Barabasi-Albert random graphs:



The clustering-coefficient decreases as the number of vertices increasing. The decreasing rate is 1/n.

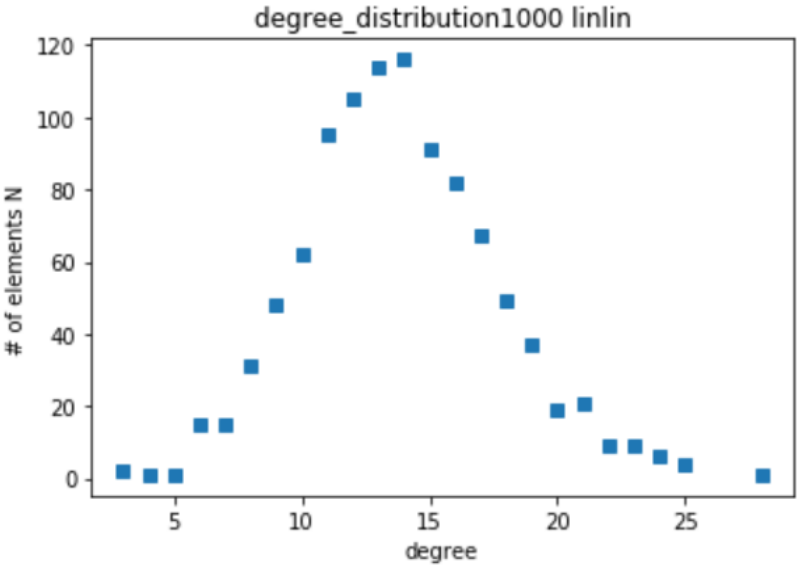
4) Degree-distribution algorithm

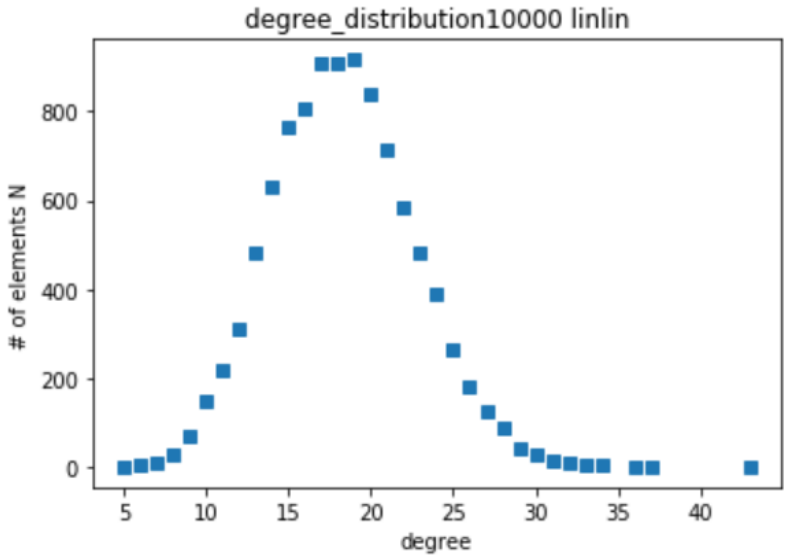
For each vertex v:

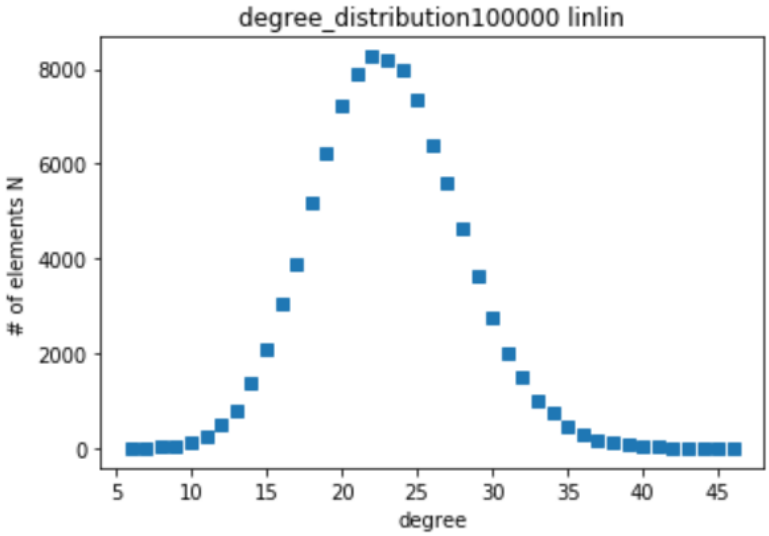
H[v] = number of neighbor of v;

Erdos-Renyi random graphs:

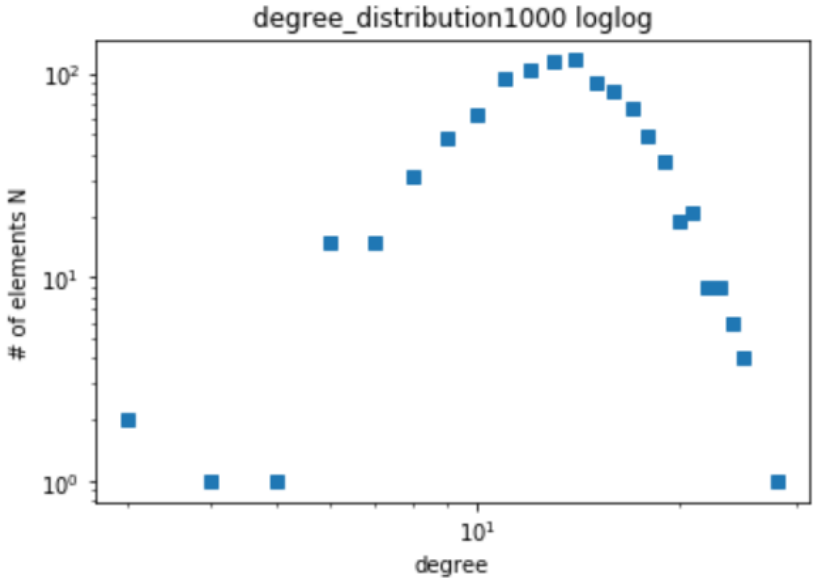
Regular (lin-lin) scale

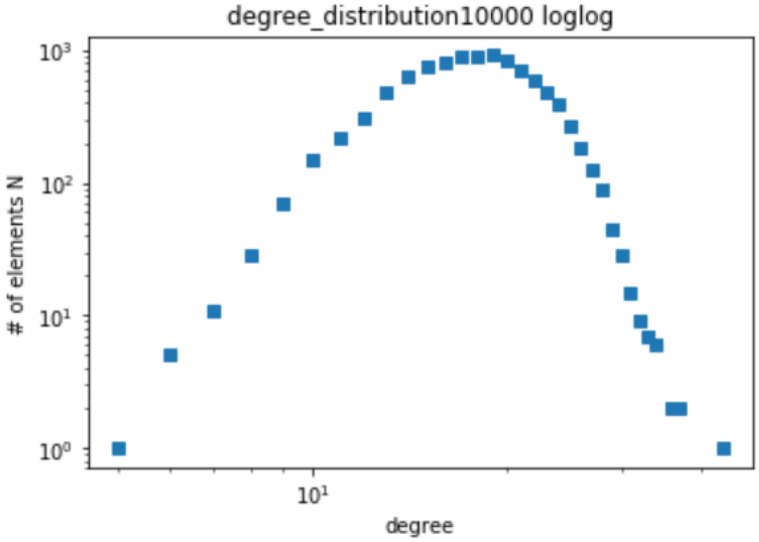


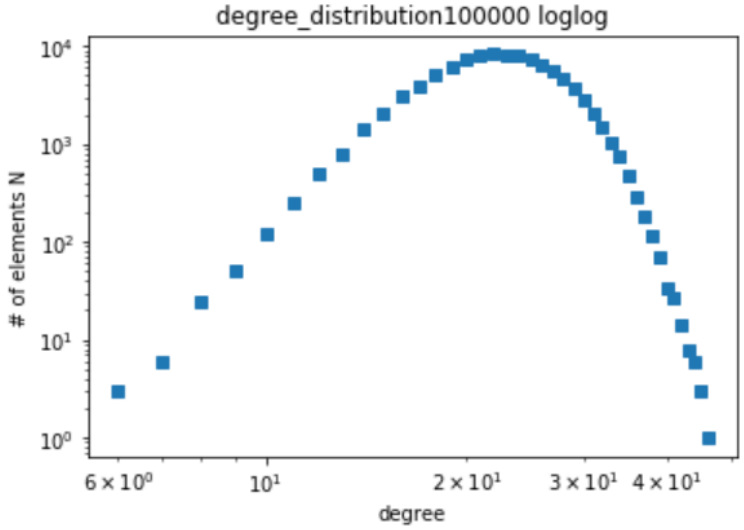




Log-log scale



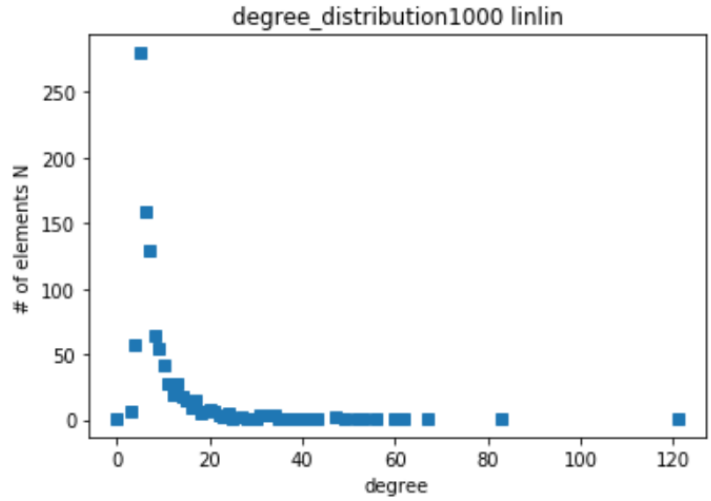


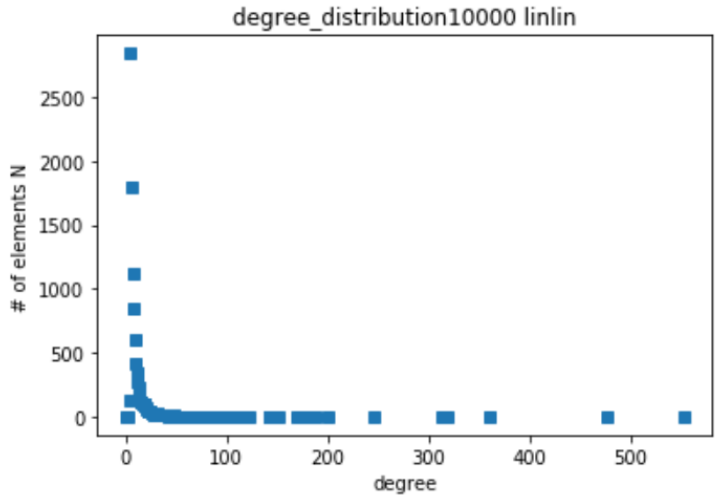


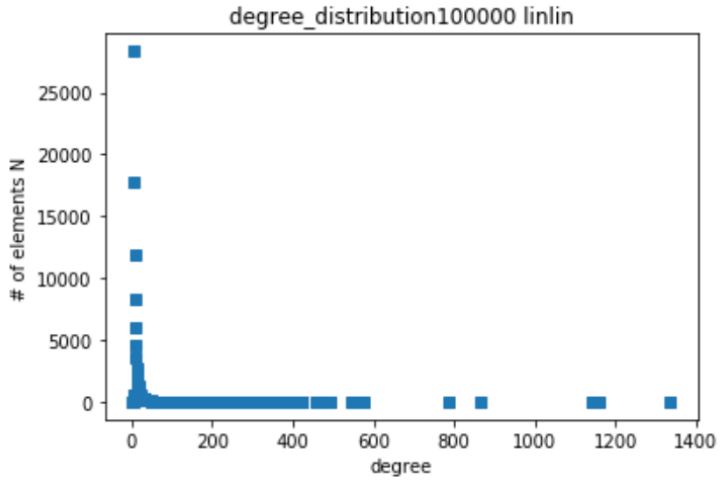
By observing the graph, it can be inferred that Erdos-Renyi random graph does not exhibit power laws.

Extra Barabasi-Albert random graphs:

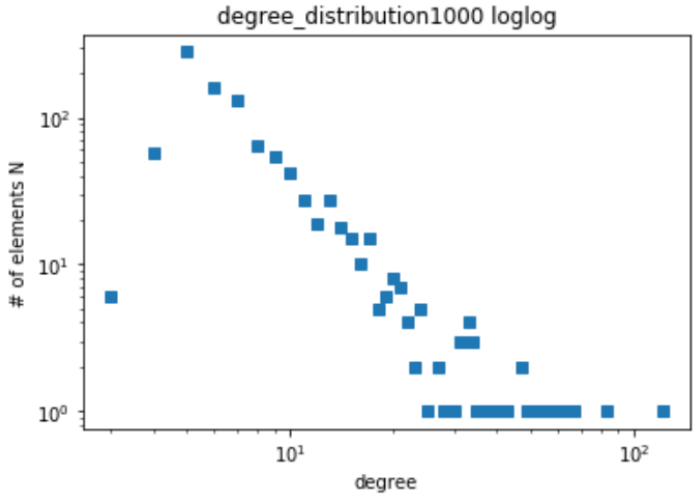
Regular (lin-lin) scale:

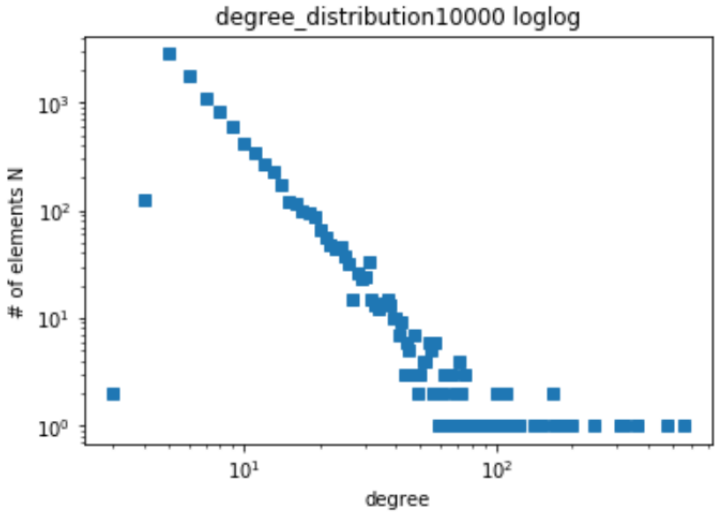


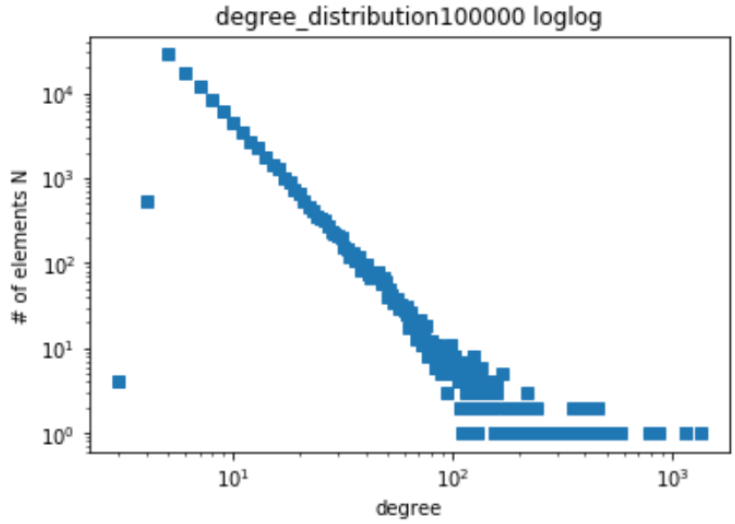




Log-log scale







It has a power law. The exponent of power law is -3.