Yi Li

HW3

CSE 520p

1)

a) The approximate algorithm takes advantage of the uniform nature of the distribution of random values. The first step, where each vertex is given a random number $s_v \in [0,1]$, is where the random value is chosen for each vertex.

Then, for each sub-section of the graph, the minimum value of the random probabilities gets propagated to all the nodes that can reach it. Thus, each partitioned graph will have the $minhash$ of all the probabilities that it points to.

Next, if given n exclusively connected elements in the graph, the $minhash$ of the this graph is expected to be $\frac{1}{n}$ (due to the uniform nature of the distribution of random values). Thus, after getting the $minhash$ of all the probabilities for a particular node, calculating $\frac{1}{h_v}$ will give an approximate value for $\widetilde{n_v}$.

b) The exact algorithm I am using is a depth first search on each vertex v in V. For each node v, I iterate through all the edges v is directly and indirectly connected to. For a single node, the complexity of this algorithm is m (total number of edges) $\cdot$ n (total number of nodes). Thus, the complexity of the algorithm is mn^2.

c) The two data sets I chose were from the Gnutella peer to peer network (https://snap.stanford.edu/data/p2p-Gnutella08.html) and the General Relativity and Quantum Cosmology collaboration network (https://snap.stanford.edu/data/ca-GrQc.html). These graph had a large number of nodes and a sizeable set of edges associated with each node, making it suitable for this experiment.

d) I've recorded the running time of the two algorithms below:

Experiment	Gnutella	Collab
5	71 ms	42 ms
15	230 ms	133 ms
25	338 ms	219 ms
Exact alg.	2767 ms	1561 ms

In addition, I added the exact time as well. As observed, the approximate algorithm scaled directly with the number of experiments that were run. Furthermore, the exact algorithm can be observed to take a much longer time than the approximate algorithms.

e) The errors for the approximate algorithms are specified below:

Iterations	Gnutella		Collab	
	Error	Max relative Error	Error	Max relative error
5	0.024144061	43.11585075	0.028242945	28.0866404
15	0.015280172	7.528070904	0.016579043	6.591704814
25	0.01194362	3.606884756	0.015123618	6.401773281

As observed, the errors for both datasets decrease as the number of iterations increase.

4)

For this algorithm, we will assume k is the size of the array $[m]$.

For a given element, the range of potential answers from the algorithm is 0 -> f_i. First, we will prove the upper limit to the any query is f_i . Next, we will show the lower bound of the frequency of element I is $f_i - \frac{T}{k}$.

We will examine the value of f_i and how frequencies are ejected. If the frequency of an element is greater than $floor\left(\frac{T}{k}\right)$, then it does not get ejected from the table. This is because after processing k elements, every frequency gets decremented by 1. If the frequency is less than $\frac{T}{k}$, that means that the count in the table has been decremented $\frac{T}{k}$ times. The algorithm states that if the $count == 0$, then it is removed from the table.

If the element in the list is not ejected from the table, then the count is equal to f_i minus the number of times the table has been updated with a $count[i]$—(which is equivalent to $floor\left(\frac{T}{k}\right)$). Thus, if the frequency is not ejected from the table (because it is greater than $floor\left(\frac{T}{k}\right)$, then it is in the range of f_i and $f_i - \frac{T}{k}$.

The next case is if the element is ejected from the table. If this is the case, that means $f_i <= \frac{T}{k}$. Thus, $\frac{f_i - T}{k} <= 0$. Since 0 is now within the range of f_i and $f_i - \frac{T}{k}$, the answer to the query falls within f_i and $f_i - \frac{T}{k}$.