CS5234: Combinatorial and Graph Algorithms

Problem Set 5

Due: September 20th, 5:29pm

Exercises (and Review) (*Do not submit.*)

Exercise 1. Here we give an alternate method for deciding whether a graph is bipartite. As in class, our goal is to design a streaming algorithm. The stream will consist of a sequence of edges, and your goal is to decide whether or not the graph consisting of these edges is bipartite or not. Your algorithm should be deterministic, and it should always be correct.

a. Imagine you are given a graph $G = (V, E)$. (Forget about streaming, for the moment.) Construct a new graph $D = (V', E')$ as follows:

   - For every node $u \in V$, add two nodes u_1 and u_2 to $V;$.
   - For every edge $(x, y) \in E$, add two edges (x_1, y_2) and $(x_2, y_1$ to E'.

   (Draw a picture!) How does the number of connected components in D relate to whether or not G is bipartite? Give a criterion that you can check on D to determine whether G is bipartite. (Try a few examples, and count the connected components in D.)

b. Now give a streaming algorithm that determines whether G is bipartite by simulating D and checking the criterion previously specified.

Standard Problems (to be submitted)

Problem 1. Counting Triangles

In this problem, we develop an alternate approach to estimating the number of triangles in a stream based on random sampling. Here is the basic idea: choose a possible triangle and check whether or not it appears in your stream; if it appears, estimate that there are a lot of triangles; if not, estimate that there are very few triangles. More specifically, here is the definition of the basic triangle estimator:

Algorithm 1: EstimateTriangles(V)

Choose one edge $e = (u, v)$ uniformly at random from the stream.
Choose a node z uniformly at random from $V \setminus \{u, v\}$.
if edges (u, z) and (v, z) appear in the stream after e **then**
    return $m(n-2)$
else
    return 0

Problem 1.a. Describe in more detail how you would implement this estimation algorithm (e.g., how you would choose a random edge, etc.). How much space does the estimator require?

Problem 1.b. Let C be the value returned by the estimator. Let T_3 be the number of triangles in the graph. Show that $\mathrm{E}[C] = T_3$.

Problem 1.c. Show that the $\mathrm{E}\left[C^2\right] = T_3(m)(n-2)$, and thus that $\mathrm{Var}[C] \le T_3(m)(n-2)$. (Hint: you may want to define random variables that indicate whether or not a given triangle was found, and write C in terms of these random variables.)

Problem 1.d. Now consider the following algorithm: run s instances of the triangle estimator, and return the average value. What is the expected value and variance of the resulting output? (An upper bound on variance, as in the previous part, is expected.)

Problem 1.e. Use Chebychev's Inequality to determine a value for s as a function of n, m, and T_3 such that the algorithm returns a $(1 \pm \epsilon)$ approximation with probability at least $1 - \delta$. (Your answer may include a factor that looks like $\frac{m(n-2)}{T_3}$, even though T_3 is unknown in advance.)
Recall: Chebychev's Inequality says that for any random variable X, $\Pr[|X - \mathrm{E}[X]| \geq t] \leq \frac{\mathrm{Var}[X]}{t^2}$.

Problem 1.f. Given that T_3 is unknown in advance, how might you choose a reasonable sample size? For what sort of graphs does this yield a reasonable estimator? (For what sort of graphs does this yield an inefficient estimator?) How does the computational cost of processing the stream compare to the triangle estimator we saw in class?

Problem 1.g. (Optional.) You can reduce the dependence on δ by running many copies of the averaging-estimator and taking the median. Design an algorithm that runs s' instances of the averaging-estimator. In order to achieve a $(1 \pm \epsilon)$ estimator that is correct with probability at least $1 - \delta$, what are good values for s and s'?