Randomized Algorithms 2017A – Final (Take-Home Exam)

Robert Krauthgamer and Moni Naor

February 26, 2017
Due within 48 hours

General instructions. The exam has 2 parts.

Policy: You may consult textbooks and the class material (lecture notes and homework), but no other sources (like web search). You should work on these problems and write up the solutions by yourself with no help from other students.

Part I (25 points)

Answer 2 of the following 3 questions. Give short answers, sketching the proof or giving a convincing justification in 2-5 sentences (even for true/false questions). You may use without proof theorems stated in class, provided you state the appropriate theorem that you are using. As usual, assume n (or $|V|$) is large enough.

A. Let G be a graph drawn from the distribution $G(n,p)$ for $p = 1/100$.

   Is it true that $\Pr[G \text{ is bipartite}] \geq 1/2$?

B. A *triangle* in a graph is just a 3-cycle subgraph. Notice that two triangles can be edge-disjoint even if they share a vertex.

   Is it true that K_n, the complete graph on n vertices, contains $p = \Omega(n^2)$ triangles that are pairwise edge-disjoint (i.e., every edge of K_n belongs to at most one triangle)?

   Hint: Can you add a triangle at random?

C. Is it true that in every undirected graph G and for every two vertices u, v, adding edges to G can only reduce the commute time $C_{u,v}$?

Part II (75 points)

Answer 3 of the following 4 questions.

1. A tournament is a *directed* graph, where for every two nodes u, v, exactly one of the two directed edges (u, v) and (v, u), appears. A *dominating set* in a tournament is a set of nodes S, where for every node $v \notin S$ there is a node $u \in S$ such that the edge (u, v) exists.

   Suppose a tournament on n nodes is given as an adjacency matrix. Suggest a Las Vegas algorithm that finds a dominating set of size $O(\log n)$ in expected runtime $O(n \log n)$.

2. Recall that we used the method of compression to analyze Cuckoo Hashing. Use the method to show that when throwing n balls to n bins, the expectation of the maximum load is $O(\log n / \log \log n)$.

3. In the *Euclidean MST* problem, that input is n points $x_1, \ldots, x_n \in [m]^d$ for $m = d = n/10$, and the goal is to compute their minimum spanning tree, where costs are by ℓ_2-distances, i.e., the complete graph with weights $w_{ij} = \|x_i - x_j\|_2$.

   Describe a randomized $(1+\varepsilon)$-approximation algorithm that solves this problem faster than the naive computation that computes all pairwise distances and then runs Kruskal's algorithm, which in our case takes $\tilde{O}(n^3)$ time. (You can omit logarithmic factors.)

4. Analyze the construction below of a distance oracle with *additive stretch* $+2$ for *unweighted* graph $G = (V, E)$, which clearly implies a multiplicative stretch 3 (strictly speaking, it is not a distance oracle because its query time is not fast enough). You should analyze its accuracy (additive stretch) and its storage requirement, where $s = s(n)$ is a parameter that you should optimize.

   Preprocess(G):
   (1) store $L = \{v \in V : \deg(v) \leq s\}$ (the low-degree vertices) and the induced subgraph $G[L]$;
   (2) choose a random subset $W \subset V$ of $O(s^{-1} n \log n)$ vertices (with or without repetitions);
   (3) for each $w \in W$, compute a BFS tree T_w rooted at w.

   Query(u, v): return

   $$\min\{d_{G[L]}(u,v), \min_{w \in W} d_{T_w}(u,v)\},$$

   using the convention that $d_{G[L]}(u,v) = \infty$ whenever at least one of u,v is not in L.

   Notation: $\deg(\cdot)$ denotes degree in G, and $G[L]$ denotes the subgraph *induced* on L by G. As usual, $d_H(\cdot,\cdot)$ is the shortest-path distance in graph H.

Good Luck.

THE END.