CME 305: Discrete Mathematics and Algorithms

Arun Jambulapati

Final Review Session Problems - March 11th, 2016

This is a list of questions that I covered during my review session on March 11th. I attempted to choose questions which were indicative of the breadth of material and required techniques of the final, if not the difficulty. The questions vary in difficulty from slightly below final level to above final level but below homework level.

Although solutions to these problems are not provided here, I will provide hints and outlines of solutions and leave the details of the proofs to you. In addition, some of these questions were taken from other sources. In the interest of proper attribution, I will instead provide a link to the page the problem was taken from, and the problem number at that page.

1. Let G be a graph with minimum degree 2. Show that G must have a cycle.

   Hint: Try using the tree theorem from class.

2. A *regular* graph is a graph where every node has the same degree. Show that every regular graph with an odd number of nodes is Eulerian.

   Hint: What do we know about the degrees of the vertices in an Eulerian graph? Apply the handshake lemma.

3. Let T be a tree, and let $v \in T$ be a node of degree d. Show that T contains at least d leaf nodes.

   Hint: Root T at v, and see what happens to each of v's neighbors.

The following two questions were taken from Jeff Erickson's excellent course on algorithms,
`http://web.engr.illinois.edu/~jeffe/teaching/algorithms/all-algorithms.pdf`

4. Erickson 24.7

   Hint: Try formulating the checkerboard as a bipartite graph, with the white squares on one side, the black squares on the other, and an edge connecting two squares if and only if they are adjacent. Note that this is bipartite as no two squares of the same color are next to each other. What does a domino become in this graph?

5. Erickson 24.9

   Hint: Try a maximum flow argument, with a source connecting to each terminal and a sink connecting to all of the boundary points.

6. Let G be a graph with $nd/2$ edges ($d > 1$). Prove that G contains an independent set of at least $n/2d$ vertices.

   Hint: Try the following randomized procedure: For each vertex in G, delete it from G with probability $1 - 1/d$. In this (much smaller) graph, for each edge remaining, delete one of its endpoints. This clearly results in an independent set, as in the final step we remove one endpoint of each edge and thereby delete every remaining edge from G.

Show that this procedure will produce an independent set in G of size at least $n/2d$ in expectation, and therefore there must exist at least one independent set of that size in the graph.

7. We will now prove a stronger version of the statement above. Let G be a graph. Show G contains an independent set of size at least $\sum_{v \in V} 1/(\delta(v) + 1)$, where $\delta(v)$ is the degree of v.

   Hint: Try the following procedure to generate an independent set I: Begin by numbering the nodes in the graph in some random order. Iterating over the nodes in this order from lowest to highest, add v to I if doing so does not add an edge to I. This clearly forms an independent set. Show that I has the desired size in expectation, by first computing for each v the probability v is in I and then applying linearity of expectations.

   Double Hint: To bound the probability v is in I, what would happen if v was first among its neighbors in the random ordering we created at the beginning? What would happen if it wasn't?

8. The Erdos-Renyi random graph $G(n, p)$ is a graph on n nodes such that for every pair of nodes u, v, u and v are connected by an edge in G with probability p. Show that as n approaches infinity, $G(n, 3/4)$ is Hamiltonian with probability 1.

   Hint: By Dirac's theorem, we know that if G has minimum degree $n/2$, it is Hamiltonian. We also know that the degree of every node v in $G(n, 3/4)$ is a sum of Bernoulli trials (ie, coin flips) corresponding to the event that each possible neighbor is connected to v. Thus the degree of any vertex in $G(n, 3/4)$ is a Binomial random variable, since the events that two different edges exist in the graph are independent. With these facts in mind, try a judicious application of the union and Chernoff bounds.

9. The SETCOVER problem is as follows: Given a set E of elements and a collection $S_1, \ldots, S_n$ of subsets of E, is there a collection of at most k of these sets whose union equals E? Prove that SETCOVER is NP-Complete.

   Hint: Reduce VERTEX COVER to SET COVER.

10. Show that the hypercube graph in n dimensions (with 2^n nodes) has cover time $O(2^n n^2)$.

    Hint: Apply the cover time upper bounds from class, and show that the effective resistance of the hypercube graph is $O(1)$. This one is long, but a good way to get used to thinking about effective resistances.

11. Consider the following variant to the SET COVER problem discussed above: for every element $e \in E$, at most k sets from $S_1 ... S_n$ contain it, for some constant k. Find a k approximation to this question.

    Hint: try a linear programming relaxation, similar to VERTEX COVER.

12. Consider scheduling n jobs to m identical machines to minimize the time taken by the machine with the heaviest load (i.e. to minimize the makespan). One algorithm is to

order the jobs by decreasing processing times $t_1 \geq t_2 \geq \ldots \geq t_n$, then greedily assign jobs to the machine whose load is the smallest so far (starting with the heaviest job). Show that this algorithm achieves a 2-approximation.

Hint: We know that OPT must be greater than t_1 (some machine must do that job), and also greater than the average $\sum_{i=1}^{i=n} t_i/n$ (not every machine can do above average). Let machine k be the one with the highest weight at the end of the algorithm and let job l be the last one assigned to it. Its weight before getting that final job is at most the average job weight, and job l clearly has cost less than the highest cost. Use these facts to prove the ratio.

13. Show that the above makespan algorithm in fact also achieves a 3/2-approximation.

    Hint: I didn't do this one in the session, but it is an interesting problem to think about! Note that if the ratio of the most expensive to the least expensive job is less than 1/2, the same analysis as above would prove the ratio. One way to solve this problem is to prove that if the ratio was greater than 1/2, this algorithm actually performs optimally. Interestingly, this stronger analysis isn't tight either– this relatively straightforward greedy algorithm achieves a 4/3-approximation, by a fairly tricky argument.