Analysis of Algorithms	**Problem Set 5**
CS 6820 Fall 2014	**Due 5pm, Wednesday, December 5, 2014**

This problem set has 5 problems with parts of varying difficulty. For full credit for a grade of A, solve four of the five problems. A full solution for each problem includes proving that your answer is correct. If your group cannot solve a problem, but can do some parts, or have partial results, write down how far you got, and why are you stuck. This problem set contains asks you to prove some problems NP-complete. To do this, you may use that any of the following problems are NP-complete: Vertex Cover, SAT, 0/1 integer programming, Circuit Satisfiability, 3D-Matching, and Hamiltonian Circuit (of these we didn't cover Hamiltonian Circuit in class, but the NP-completeness proof is in both of our recommended books).

Students may work on homework in groups of up to 2-3 people. Each group may turn in a single solution set that applies to all members of the group. However, students are asked to understand each of their group's solutions well enough to give an impromptu whiteboard presentation of the solution. You may use any fact we proved in class without proving the proof or reference, and may read the relevant chapters of the Kleinberg-Tardos or Kozen books, provided you state them clearly. However, you may **not use other published papers, or the Web to find your answer**.

Solutions can be submitted on CMS in pdf format (only). Please type your solution or write extremely neatly to make it easy to read. If your solution is complex, say more than about half a page, please include a 3-line summary to help us understand the argument.

Please ask any clarifying questions using Piazza, where we will post all answers.

(1) Prove that the directed disjoint path problem is NP-complete. The problem is as follows. Input is a directed graph G and k pairs of nodes s_i and t_i for $i = 1, \ldots, k$. The problem is to decide if there are a node-disjoint paths P_i so that path P_i goes from s_i to t_i. Hint: Consider using SAT in your proof. Start with an $s - t$ pair for each variable with two separate paths between each pair, thinking of one path as representing setting the variable "true", the other path as setting the variable "false". Now add further $s - t$ pairs to make sure that disjoint paths will be corresponding to a satisfying assignment of the SAT problem.

(2) The *Multi-Cut problem* is given by a undirected graph $G = (V, E)$ with nonnegative capacities $c_e \geq 0$ on the edges $e \in E$ and k pairs of nodes $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$, and a threshold value γ. The problem is to decide if there is a subset F of the edges E of total capacity at most γ, that is $\sum_{e \in F} c_e \leq \gamma$, such that each given pairs s_i and t_i is in separate components after deleting F. Note that the special case, when $k = 1$ is the traditional (s, t) min-cut problem.

Show that the Multi-Cut Problem is NP-complete even when G is a tree. Hint: try very simple kinds of trees.

(3) Construct the edge probabilities for a three state Markov chain where each pair of states is connected by an edge so that the stationary probability is $\frac{1}{2}, \frac{1}{3} \cdot \frac{1}{6}$. Recall that a Markov chain is given by matrix P of transition probabilities p_{ij} for $i, j \in \{1, 2, 3\}$, where p_{ij} is the probability that in state i the Markov chain goes to state j, so we must have $\sum_j p_{ij} = 1$. A probability distribution q on the states is stationary of $qP = q$. Can you make the matrix symmetric? if you cannot, explain why not.

(4) For an undirected graph G with n vertices and m edges, recall that the Laplacian matrix is an n by n matrix L where the diagonal entry l_{ii} is the degree of node i in G, and the entry l_{ij} is -1 if (i, j) is an edge of the graph and 0 otherwise. Also recall that this symmetric matrix is guaranteed to have all nonnegative eigenvalues with 0 as the smallest one (with the eigenvector of the all 1 vector). Let

$$0 = \lambda_1(G) \leq \lambda_2(G) \leq \ldots, \lambda_n(G)$$

be the set of eigenvalues of this matrix. Define a random induced subgraph of G to be a graph obtained by the following random sampling procedure: We include each vertex in the sample independently with probability 1/2, and then let H be the subgraph of G consisting of the selected vertices together with all the edges of G that have both their endpoints selected.

Prove or disprove that there exist constants $n_0 < \infty$ and $\delta > 0$ such that for every connected undirected graph G with at least n_0 vertices, if H is a random induced subgraph of G, then

$$Pr(\lambda_2(H) \geq \frac{1}{2}\lambda_2(G)) \geq \delta.$$

To make $\lambda_2(H)$ well-defined for all H, we adopt the convention that $\lambda_2(H) = 0$ when H has fewer than two vertices.

(5) In class we have seen a randomized algorithm to test if a number is prime. Suppose you run a version of the algorithm that makes a mistake with probability $\delta = 1/2^k$. So given a number n as input, if n is prime, the algorithm returns "probably prime" with probability at least $(1 - \delta)$, and if n is not prime, the algorithm returns "probably not prime" with probability at least $(1 - \delta)$.

For cryptographic applications we often need large random primes. The way to find a random n digit primes is to test random numbers with n digits for being prime. A useful fact here is that the probability that a random n digit number is prime is c/n for some constant c. So the algorithm is as follows.

```
While algorithm returns "probably not prime"
    Let $N$ be random $n$ digit number
    Run algorithm to test if $N$ is a prime
```

```
endwhile
return $N$
```

(a) Suppose we run a version of the prime testing algorithm from that make a mistake with probability at most $\delta = 0.1\% = 0.001$. What is the probability that the number N returned is prime? What δ do you have to use to make the provability that number returned is prime at least 99% (express your choice as δ as a function of n). If $O(T(n))$ is the running time of a single run of our prime testing algorithm (computing $a^{(n-1)/2} \bmod N$), what is the running time of the version of the above algorithm to find an n digit number that is prime with probability at least 99%?

Consider a generic randomized algorithm that makes a mistake with probability δ. Suppose the algorithm is deciding a yes/no question, and can return the wrong answer with probability 1/3.

(b) If we run this randomized algorithm k times (with independent random choices) we expect to get the correct answer in at least $2/3rd$ of the runs. Give a bound (using Chernoff bound) of the probability that the majority answer is wrong (i.e., that for an input N that is not prime, the majority of the runs will return "probably prime" or vica versa). How high a value k will guarantee that the probability of a wrong answer is at most δ for a small $\delta > 0$? (express your answer as a function of $\log \frac{1}{\delta}$).