EECS 279: Approximation Algorithms — Release: 08-08-2014
Homework 0 — This homework won't be graded.

This graduate course requires decent background in discrete algorithms and complexity. The following problems are to test your background and were chosen from courses taught by several faculty members at UIUC, Chandra Chekuri, Jeff Erickson, and Sariel Har-Peled. If you cannot solve most of these problems, perhaps you are not well prepared to take this course.

1. *Induction.* The following induction-based proof shows that all balls always have the same color in any case, which is obviously wrong. Find the error.

   > We prove the claim by induction on the number n of balls. First consider the case when $n = 1$. In this case, we have only one ball, so the claim is obviously true. Now suppose that we have $n + 1$ balls where $n > 0$. Let $B_1, B_2, ..., B_{n+1}$ denote the balls. First consider the first n balls, $B_1, B_2, ..., B_n$. By induction hypothesis, all these balls have the same color. Now consider the last n balls, $B_2, B_3, ..., B_{n+1}$. Likewise, we know that all these balls have the same color by induction hypothesis. Then note that B_1 has the same color as the balls $B_2, B_3, ...B_n$. So does B_{n+1}. Hence we conclude that all balls, $B_1, B_2, ...B_{n+1}$ have the same color.

2. *Asymptotic Running Time.* Sort the following functions in increasing order of asymptotic magnitude. Indicate ties if there are any.

$$\begin{array}{cccc}
n^{2.5} - (n-1)^{2.5} & n & n^{1.5} & n^{1.4} \\
1 + \log\log n & (\log n)! & 2^{\log\log n} & 2^{\log n} \\
\sum_{i=1}^{n} i & \sum_{i=1}^{n} \sqrt{i} & \sum_{i=1}^{n} 1/i & n^{\log\log n} \\
n\log^4 n & \log^{20} n & (1.05)^{20n} & 10 + \lceil \log\log n \rceil
\end{array}$$

3. *Recurrence.* Solve the following recurrences, and state tight asymptotic bounds for each function in the form of $\Theta(f(n))$ for a simple function $f(n)$.

   (a) $A(n) = A(n/3 + 5 + \lfloor \log n \rfloor) + n \log\log n$.

   (b) $B(n) = 3B(\lceil n/2 \rceil - 5) + n/\log n$.

   (c) $C(n) = \frac{n}{n-2} C(n-1) + 1$.

   (d) $D(n) = D(\lfloor n/4 \rfloor) + 1/\sqrt{n}$.

   (e) $E(n) = E(\lfloor n^{1/7} \rfloor) + 2$.

4. Let G be an undirected graph with n nodes. Suppose that G contains two nodes s and t such that every path from s to t contains more than $n/2$ edges.

   (a) Prove that G must contain a vertex v that lies on *every* path from s to t.

   (b) Describe an algorithm that finds such a vertex v in $O(V + E)$ time.

5. Imagine that you have just been confirmed as the head of the super-secret spy agency SRU (Spies R Us). There are a number of countries that you wish to spy upon. These countries are of various sizes and speak a variety of languages. SRU has a number of spies each of whom speaks some languages, but not others. For each country you know how many spies are required to cover its entire territory and the language spoken. For each spy you know what languages he speaks. Obviously, you do not want to assign a spy to a country where he doesn't speak the language. Give a good algorithm to assign spies to countries.

6. NP-hardness.

   (a) In the Sub-graph problem, we are given two graphs G and H. The goal is to decide if H is a subgraph of G or not. Prove that the Sub-graph problem is NP-complete by a reduction from Clique. The input to Clique is an undirected graph G and a non-negative integer k, and the goal is to decide if G has a clique of size k (as a subgraph) or not. We say that a graph G' is clique if any pair of vertices are connected by an edge, and the size of a clique is simply the number of verticies in the clique.

   (b) In the Long-Path problem, we are given as input an directed graph G (all edges have the same length 1) together with a source-sink pair s and t, and a target length k. The goal is to decide if there is a simple path (using no edge or node twice) of length k from s to t. Show that the Long-Path is NP-complete. You can use the fact that deciding if a directed graph has a Hamiltonian path [1] is NP-complete.

7. The classical 0, 1 knapsack problem is the following. We are given a set of n items. Item i has two positive integers associated with it: a size s_i and a profit p_i. We are also given a knapsack of integer capacity B. The goal is to find a maximum profit subset of items that can fit into the knapsack. (A set of items fits into the knapsack if their total size is less than the capacity B.) Use dynamic programming to obtain an exact algorithm for this problem that runs in $O(nB)$ time. Also obtain an algorithm with running time $O(nP)$ where $P = \sum_{i=1}^{n} p_i$. Note that both these algorithms are not polynomial time algorithms. Do you see why?

8. Longest Common Subsequence. Let $X[1...m]$ and $Y[1...n]$ be two arbitrary arrays. A common subsequence of X and Y is another sequence that is a subsequence of both X and Y. Describe an efficient algorithm to compute the length of the longest common subsequence of X and Y. A subsequence is anything obtained from a sequence by extracting a subset of elements, but keeping them in the same order; the elements of the subsequence need not be contiguous in the original sequence. For example, the strings C, DAMN, and YAIOAI, and DYNAMICPROGRAMMING are all subsequences of the sequence DYNAMICPROGRAMMING

[1] We say that a path is Hamiltonian if it visits every node exactly once