Handout 1: graph coloring

March 14, 2011

http://www.wisdom.weizmann.ac.il/∼dinuri/courses/11-BoundaryPNP/

The chromatic number of a graph is the minimum number of colors that suffice for a vertex coloring (in which endpoints of an edge receive different colors). This problem is NP-hard. However, this does not preclude the existence of interesting exponential time algorithms, and some of these algorithms will be discussed in today's lecture.

Homework assignment.

Given a graph $G(V, E)$, a *maximal independent set* is an independent set $S \subset V$ that cannot be extended by additional vertices (every vertex $v \notin S$ has some neighbor in S). Throughout we use the convention that n denotes $|V|$, and that O^* notation suppresses multiplicative terms that are polynomial in n. For example, $n^3 2^n$ is expressed as $O^*(2^n)$.

1. Give a polynomial time algorithm that given as input a graph G, outputs a maximal independent set.

2. Let $i(G)$ denote the number of maximal independent sets in the graph G. Give an algorithm that outputs *all* maximal independent sets in G, whose running time is proportional to the size of the output, namely, $O^*(i(G))$.

3. Prove that for every graph, $i(G) \leq 3^{n/2}$. (Better bounds are known, but not required in this question.)

4. For every p, show a graph on $n = 3p$ vertices (need not be connected) for which $i(G) = 3^p$, and a connected graph on $n = 3p + 1$ vertices for which $i(G) > 3^p$.

5. It is known that $i(G) \leq 3^{n/3}$ [J. W. Moon, L. Moser: On cliques in graphs. Israel Journal of Mathematics 3: 23-28 (1965)], and in this question you may use this fact without proof. Give a 3-coloring algorithm that runs in time $O^*(3^{n/3}) \leq O^*(1.45^n)$.

State of the art and research questions. Consider running times in the form $O^*(b^n)$ for 3-coloring algorithms. What is the smallest value of b attainable? The homework assignment shows that $b < 1.45$. Other algorithms achieve better bounds [Richard Beigel, David Eppstein: 3-coloring in time $O(1.3289^n)$. J. Algorithms 54(2): 168–204 (2005)]. Can algorithms with a value of b arbitrarily close to 1 be designed? Alternatively, can one give a convincing argument why this cannot be done?