Handed out: April 16. Due: April 23.

1 2-Coloring the 3-Colorable

As mentioned in class, assume we're given a 3-colorable graph; that is, a graph for which each node can be assigned one of three colors, where each edge has different colors at the endpoints.

The problem is to give a polynomial time (randomized) algorithm that assigns only two colors to the nodes of the graph, in a way that no triangle in the graph is monochromatic (i.e. has the same color on all three of its nodes). The hint we gave in class was to essentially perform a random walk, starting with an arbitrary 2-coloring.

[You might ask: why do we care? One motivation is that 3-coloring a graph is hard, but "approximately" coloring it with two in the way described is easy. In general, coloring a graph has many applications, such as register allocation.]

2 Maximum-Cutting a graph

We saw a randomized algorithm for finding a minimum cut in a graph. What about finding a *maximum* cut? It turns out that in general, this problem is NP-hard, so this is probably not as easy as finding a minimum cut.

But we can find a cut that is "approximately" large, rather easily. Give a polynomial time randomized algorithm that, given a graph, always returns a cut such that at least $m/2$ edges cross the cut, where m is the total number of edges. (Note that m is an upper bound on the number of edges in a maximum cut, so this is an approximation to the maximum cut.)

3 Solving 2-SAT using randomness

Recall the 2-SAT algorithm given in class, where we begin with a random assignment to the variables. If the assignment makes some clause false, then we randomly choose one of the two variables in that clause and flip the assigned value for that variable.

Prove that, if a given 2-SAT formula is satisfiable, the expected number of flips necessary in order to satisfy it is $O(n^2)$.

Hint: This should be fairly short and involve very little derivation. Feel free to use any results stated in class.