Homework Assignment #3 (50 points, weight 3%)
Note: this is a half assignment; Adjusted weights: A1 6%, A2 6%, A3 3% = total 15%)
Due: Friday April 8, by 3:30 p.m. (drop under my office door)

---

1. (25 points) Chapter 12 - Exercise 2 page 703.

2. Recall that for a problem in which the goal is to maximize some underlying quantity, gradient descent has a natural "upside-down" analogue, in which one repeatedly moves from the current solution to a solution of strictly greater value. Naturally, we could call this a *gradient ascent algorithm*. (Often in the literature you'll also see such methods referred to as *hill-climbing* algorithms.)

   By straight symmetry, the observations we've made in this chapter about gradient descent carry over to gradient ascent: For many problems you can easily end up with a local optimum that is not very good. But sometimes one encounters problems—as we saw, for example, with the Maximum-Cut and Labeling Problems—for which a local search algorithm comes with a very strong guarantee: Every local optimum is close in value to the global optimum. We now consider the Bipartite Matching Problem and find that the same phenomenon happens here as well.

   Thus, consider the following Gradient Ascent Algorithm for finding a matching in a bipartite graph.

   As long as there is an edge whose endpoints are unmatched, add it to the current matching. When there is no longer such an edge, terminate with a locally optimal matching.

   (a) Give an example of a bipartite graph G for which this gradient ascent algorithm does not return the maximum matching.

   (b) Let M and M' be matchings in a bipartite graph G. Suppose that $|M'| > 2|M|$. Show that there is an edge $e' \in M'$ such that $M \cup \{e'\}$ is a matching in G.

   (c) Use (b) to conclude that any locally optimal matching returned by the gradient ascent algorithm in a bipartite graph G is at least *half* as large as a maximum matching in G.

2. (25 points) Chapter 13 - Exercise 1 page 782.

1. *3-Coloring* is a yes/no question, but we can phrase it as an optimization problem as follows.

Suppose we are given a graph $G = (V, E)$, and we want to color each node with one of three colors, even if we aren't necessarily able to give different colors to every pair of adjacent nodes. Rather, we say that an edge (u, v) is *satisfied* if the colors assigned to u and v are different.

Consider a 3-coloring that maximizes the number of satisfied edges, and let c^* denote this number. Give a polynomial-time algorithm that produces a 3-coloring that satisfies at least $\frac{2}{3}c^*$ edges. If you want, your algorithm can be randomized; in this case, the *expected* number of edges it satisfies should be at least $\frac{2}{3}c^*$.