CS 577: Introduction to Algorithms — Homework 5b

Out: 11/28/06 — **Due: 12/05/06**

Ground Rules

- Grading. You will be graded on the correctness as well as clarity of your solutions. You are required to prove any claims that you make. In particular, when you are asked to design an algorithm, you must argue its correctness and running time.
- Collaboration. You are allowed to discuss questions with other people in the class. However, **you must solve and write your answers yourself without any help**. You must also give explicit citations to any sources besides the textbook and class notes, including discussions with classmates. Solutions taken from external sources such as the WWW, even if cited, will receive no credit unless there is significant "value added". In cases of doubt, you may be asked to explain your answer to the instructor and this will determine your grade.
- Lateness. Please see the class webpage for details on the lateness policy.
- This homework is due in **one week**. Start working early. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later. Good luck!

Problems

1. **(Not-All-Equal-SAT.)** Consider the following variant of 3-SAT: Given a 3-CNF formula, decide whether there exists an assignment to the variables of the formula such that every clause contains at least one literal that is true and at least one literal that is false. Show that this problem is NP-complete.

2. **(Max-Cut.)** In the (unweighted) Min-Cut problem, our goal is to find a partition of the graph into two pieces with the fewest edges going across the partition. This problem can be solved in polynomial time using max-flow. In the Max-Cut problem, our goal is the opposite – we want to find a partition with the maximum number of edges going across. Surprisingly (as you will show in this problem), the complexity of finding a Max-Cut is very different from that of finding a Min-Cut!

   Formally, the problem is stated as follows:

   Given a graph G with edge set E and vertex set V, and a number k, does there exist a partition of the vertices into two sets V_1 and V_2 with $V_1, V_2 \neq \emptyset$ and $V_1 \cup V_2 = V$, such that $|(V_1 \times V_2) \cap E| \geq k$.

   Prove that the Max-Cut problem is NP-complete.

3. **(1/2-Cut.)** Given that Max-Cut is NP-complete, we attack a slightly easier problem instead – we consider the special case of $k = |E|/2$. In the $1/2$-Cut problem, our goal is to find a partition of the vertex set into two pieces V_1 and V_2 with $V_1, V_2 \neq \emptyset$ and $V_1 \cup V_2 = V$, such that the number of edges going across is at least half the total number of edges: $|(V_1 \times V_2) \cap E| \geq |E|/2$.

   Before we worry about finding a $1/2$-Cut, it is not even clear that such a partition always exists! In this question, you will not only prove the existence of a $1/2$-Cut but also learn a simple way of finding one. Consider the following randomized procedure: for every vertex $v \in V$, we independently toss a coin of bias $1/2$; if the coin comes up heads we put v in V_1 and otherwise we put it in V_2. (If either one of V_1 or V_2 turn out to be empty, we just repeat the procedure; the probability of this is so small that you can essentially ignore it.)

   (a) For a given edge $(u, v) \in E$, what is probability that (u, v) crosses the cut?

   (b) What is the expected number of edges in the cut? Prove your answer.

   (c) Use your answers to the previous parts to show that a $1/2$-Cut always exists.

   Extra credit: Give a deterministic algorithm for finding a $1/2$-Cut. Analyse its running time.

4. (**Integer Programming.)** Integer programming is one of the most widely used tools for real world optimization problems such as planning and scheduling problems. There are several commercial programs such as CPLEX that are used to solve integer programs arising in practice. Unfortunately, no polynomial-time algorithm is known for solving this problem in general, and commercial packages do not always guarantee optimal solutions in a short amount of time.

   Formally, an integer program is a sequence of linear inequalities on variables. The goal is to determine whether there is an assignment of integer values to all the variables such that all the inequalities are satisfied. The size of an integer program is defined to be the number of variables plus the number of constraints. The following is an example of an integer program of size 7 that is satisfied by the solution $x = 2$ and $y = 1$.

$$\begin{aligned} x + y &\geq 3 \\ x + 2y &\leq 4 \\ 2x + y &\leq 5 \\ x, y &\geq 0 \end{aligned}$$

   Prove that integer programming is NP-complete.