COS 423
Spring 2003

Problem Set No. 6

Due: Tuesday May 13

1. Prove that the following problem is NP-complete: given an undirected graph G and a vertex constraint $d(v)$ at each vertex v, does G have a spanning tree that satisfies the degree constraint at every vertex? (The tree satisfies the constraint at a vertex v if the degree of v in the tree is at most $d(v)$. That is, the degree constraints are upper bounds on the allowed degrees.)

2. Bonnie and Clyde
Bonnie and Clyde have just robbed a bank. They have a bag of money and want to divide it up. For each of the following scenarios, either give a polynomial-time algorithm, or prove that the yes-no version of the problem is NP-complete. The input in each case is a list of the n items in the bag, along with the value of each, expressed in ordinary decimal notation.

   a. There are n coins, but only 2 different denominations: some coins are worth x dollars, and some are worth y dollars. They wish to divide the money exactly evenly.

   b. There are n coins, with an arbitrary number of different denominations, but each denomination is a nonnegative integer power of 2, i.e., the possible denominations are 1 dollar, 2 dollars, 4 dollars, etc. They wish to divide the money exactly evenly.

   c. There are n checks, which are, in an amazing coincidence, made out to "Bonnie or Clyde." They wish to divide the checks so that they each get the exact same amount of money.

   d. There are n checks as in part (c), but this time they are willing to accept a split in which the difference is no larger than 100 dollars.

3. In the MAX-CUT problem, we are given an unweighted undirected graph $G = (V, E)$. We define a cut $(S, V - S)$ as in Chapter 23 and the ***weight*** of a cut as the number of edges crossing the cut. The goal is to find a cut of maximum weight. Suppose that for each vertex v, we randomly and independently place v in S with probability $1/2$ and in $V - S$ with probability $1/2$. Show that this algorithm is a randomized 2-approximation algorithm. Does this algorithm give a 2-approximation if the edges have arbitrary non-negative weights?

4. ***Parallel machine scheduling***

In the ***parallel-machine-scheduling problem***, we are given n jobs, $J_1, J_2, \ldots, J_n$, where each job J_k has an associated nonnegative processing time of p_k. We are also given m identical machines, $M_1, M_2, \ldots, M_m$. A ***schedule*** specifies, for each job J_k, the machine on which it runs and the time period during which it runs. Each job J_k must run on some machine M_i for p_k consecutive time units, and during that time period no other job may run on M_i. Let C_k denote the ***completion time*** of job J_k, that is, the time at which job J_k completes processing. Given a schedule, we define $C_{\max} = \max_{1 \le k \le n} C_k$ to be the ***makespan*** of the schedule. The goal is to find a schedule whose makespan is minimum.

For example, suppose that we have two machines M_1 and M_2 and that we have four jobs J_1, J_2, J_3, J_4, with $p_1 = 2$, $p_2 = 12$, $p_3 = 4$, and $p_4 = 5$. Then one possible schedule runs, on machine M_1, job J_1 followed by job J_2, and on machine M_2, it runs job J_4 followed by job J_3. For this schedule, $C_1 = 2$, $C_2 = 14$, $C_3 = 9$, $C_4 = 5$, and $C_{\max} = 14$. An optimal schedule runs J_2 on machine M_1, and it runs jobs J_1, J_3, and J_4 on machine M_2. For this schedule, $C_1 = 2$, $C_2 = 12$, $C_3 = 6$, $C_4 = 11$, and $C_{\max} = 12$.

Given a parallel-machine-scheduling problem, we let $C^*_{\max}$ denote the makespan of an optimal schedule.

a. Show that the optimal makespan is at least as large as the greatest processing time, that is,

$$C^*_{\max} \ge \max_{1 \le k \le n} p_k \ .$$

b. Show that the optimal makespan is at least as large as the average machine load, that is,

$$C^*_{\max} \ge \frac{1}{m} \sum_{1 \le k \le n} p_k \ .$$

Suppose that we use the following greedy algorithm for parallel machine scheduling: whenever a machine is idle, schedule any job that has not yet been scheduled.

c. Write pseudocode to implement this greedy algorithm. What is the running time of your algorithm?

d. For the schedule returned by the greedy algorithm, show that

$$C_{\max} \le \frac{1}{m} \sum_{1 \le k \le n} p_k + \max_{1 \le k \le n} p_k \ .$$

Conclude that this algorithm is a polynomial-time 2-approximation algorithm.