1. (Greedy Matching) Given an undirected, unweighted graph, consider the following algorithm for finding a large matching. Begin with the empty matching. Repeat the following step until it no longer applies: find an edge both of whose ends are unmatched, and add it to the matching.

   (a) Describe an implementation of this algorithm that runs in linear time (in the number of vertices and edges of the graph).

   (b) Prove that this algorithm finds a matching that is within a factor of two of maximum size. Give an example to show that this constant factor of two is tight (an example on which the algorithm finds a matching only half the size of maximum).

2. (Scheduling with profits and deadlines: see CLRS Problem 34-4) Suppose you have one machine and n tasks a₁,a₂,…,aₙ. Each task aᵢ has a processing time tᵢ, a profit pᵢ and a deadline dᵢ. The machine can process only one task at a time, and the task must run without interruption for tᵢ consecutive time units. If you complete task aᵢ by its deadline dᵢ, you receive a profit pᵢ, but if you complete it after its deadline you receive nothing. As an optimization problem, you wish to find a schedule that completes all the tasks and returns the greatest profit.

   Formulate a “yes-no” version of the problem and show that it is NP-complete.

3. (Bin-packing: see CLRS Problem 35-1) Given a set of n objects, where the size sᵢ of the ith object satisfies 0<sᵢ<1, we wish to pack all the objects into the minimum number of bins. Each bin can hold any subset of the objects whose total size does not exceed 1. The “first-fit” algorithm puts the objects into bins sequentially according to the following rule: add the next object to the first bin into which it fits.

   (a) Give an efficient implementation of the first-fit algorithm, and analyze its running time.

   (b) Prove that the first-fit algorithm comes within a factor of 2 of minimizing the number of bins used.