

1. Recall the Bin-Packing problem:
INPUT: A set of positive integers $A = \{a_1 \dots a_n\}$, a positive bin size B , and a positive integer k .
QUESTION: Can we partition A into k disjoint sets such that the sum of the integers in any set is at most B .
 - (a) Use the Subset-Sum problem to prove that Bin-Packing is NP-Complete.
 - (b) Does your proof from part (a) show that Bin-Packing is strongly NP-Complete? Explain why or why not.
 - (c) Define an appropriate optimization version of the Bin-Packing problem, and prove that if there is a $(\frac{3}{2} - \epsilon)$ -approximation to this problem, for any $\epsilon > 0$, then P=NP.
2. [CLRS] 35-5: **Parallel machine scheduling**.
3. In class, we saw how to express the Max-Flow problem as a Linear Programming problem. In this question, you are asked to extend this construction to a more general flow problem, called the Multi-commodity Flow problem.

This is defined as follows. We are given a directed graph G with positive capacities $C(e)$ on its edges, as in the usual MaxFlow problem. Also, for each ordered pair of vertices $(x, y) \in V \times V$, we are given a non-negative *demand* $D(x, y)$. There is a separate *commodity* for each such pair (x, y) . A *flow* is a way of routing $F \times D(x, y)$ units of the (x, y) -commodity from x to y , for all pairs (x, y) simultaneously, such that the total quantity of all commodities flowing along any edge e does not exceed its capacity $C(e)$. There must be conservation of flow for each commodity separately. The *value* of the flow is F . The problem is to find a flow of maximum value.

 - (a) Show that the standard MaxFlow problem is a special case of this Multi-commodity Flow problem.
 - (b) Show how to express the Multi-commodity Flow problem as an instance of Linear Programming. Your constraints do not need to be in matrix form: it is sufficient for you to write down the objective function, and the list of constraints.
4. The Integer Programming problem is like Linear Programming, except that the variables are not able to take on arbitrary real values. Instead, they are required to be integers. Define a decision version of the Integer Programming problem, and show that this problem is NP-Complete. Hint: use 3SAT.