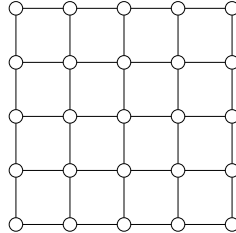


1. Consider the special case of the independent set problem where the graph G is an $n \times n$ grid graph, as in this figure:



Furthermore, assume that each node i has a weight w_i , which is a non-negative integer, and we want to find an independent set of maximum total weight.

- Consider the greedy algorithm, where we process the nodes from largest weight to smallest weight, and include each node in the solution if it does not have an edge to any node that has already been included. Show that this algorithm is a 4-approximation. (You should demonstrate that it is no worse than a 4-approximation, but also that it is no better than a 4-approximation.)
 - Describe a polynomial time algorithm that achieves no worse than a 2-approximation for this problem.
2. In the FPTAS for the knapsack problem that we saw in lecture, we round the values of the items down to the nearest multiple of 2^k , solve the resulting problem exactly, and then use the set of items from the resulting solution. Consider instead the algorithm where we round the values of the items **up** to the nearest multiple of 2^k , solve the resulting problem exactly, and then use the set of items from the resulting solution. Prove that this modified algorithm is also an FPTAS for the knapsack problem.
3. Consider the following problem: we are given a finite set $U = \{e_1, e_2, \dots, e_n\}$, as well as a collection $C = \{S_1, S_2, \dots, S_m\}$ of subsets of U . In addition, every element $e_i \in U$ has a weight $w(e_i)$, and we want to find a minimum weight subset $V \subseteq U$, such that every set S_i contains at least one element of V . In other words, $\forall i, V \cap S_i \neq \emptyset$. Note that this problem is different from the set cover problem, as we are looking for a subset of the elements in U , as opposed to a subcollection of the sets in C . Give the best upper and lower bounds you can on the approximation ratio achievable in polynomial time for this problem. For the lower bound, you should assume that $P \neq NP$.
4. In the integer programming problem, the input is a linear program, and the output is the optimal solution to the linear program where all of the variables have integer values. The integer programming problem is NP-Complete (you don't need to prove this).

On the other hand, we have seen that maximum flow can be expressed as a linear program, and yet, if the capacities of the flow problem are integers, then we can always find an optimal flow where all of the individual flows are integer. As a result, there always exists an optimal solution to the linear program for maximum flow that has integer values. In fact, it turns out that for this linear program, every basic feasible solution will be integer. Thus, for some linear programming problems, the integrality condition does not make the problem NP-Complete. In this problem, we will see a useful sufficient condition for this nice property to hold.

A square integer matrix A is called *unimodular* if its determinant $\det(B) = \pm 1$. An integer matrix A is called *totally unimodular* if every square, non-singular submatrix of A is unimodular. Here, a $k \times k$ submatrix refers to the entries of the matrix that appear in any k rows and any k columns. A square matrix is singular if its determinant is 0.

- (a) Let $\mathcal{F} = \{\mathbf{x} \text{ s.t. } A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\}$ be the feasible region of a linear program in standard form, where A is an $m \times n$ integer matrix and b is an integer m -vector. Show that if A is totally unimodular, then every basic feasible solution of \mathcal{F} is an integer point in \mathbb{R}^n .
- (b) Suppose that the integer matrix A has entries in $\{0, \pm 1\}$, and that no more than two non-zero entries appear in any column. Suppose further that the rows of A can be partitioned into two sets I_1, I_2 , so that two non-zero entries in the same column have the same sign if and only if their rows are in different sets. Show that A is totally unimodular.
- (c) Show that the maximum flow problem can be described as a linear programming problem in standard form, where the constraint matrix A is totally unimodular, and hence that all basic solutions are integer.
- (d) Consider the following problem: we are given m sources of flow, where source i has a supply of a_i units, and n sinks of flow, where sink j has a demand of b_j units, and we must send flow from the sources to the sinks in such a way that the demands are satisfied. Any source can send any integer amount of flow (up to its supply) to any sink, but there is a cost c_{ij} to send one unit of flow from source i to sink j . Our task is to satisfy all the demands at minimum total cost. Show how to find the minimum total cost solution to this problem in polynomial time using linear programming. You can assume that $\sum_i a_i = \sum_j b_j$, and that all the a_i s and b_j s are integers.