Consider the following closet-point heuristic for building an approximate traveling-salesman tour whose cost function satisfies the triangle inequality. Begin with a trivial cycle consisting of a single arbitrarily chosen vertex. At each step, identify the vertex \( u \) that is not on the cycle but whose distance to any vertex on the cycle is minimum. Suppose that the vertex on the cycle that is nearest \( u \) is vertex \( v \). Extend the cycle to include \( u \) by inserting \( u \) just after \( v \). Repeat until all vertices are on the cycle. Prove that this heuristic return a tour whose total cost is not more than twice the cost of an optimal tour.

**Solution:**

Let’s denote the optimal tour at step \( i \) as \( H^*_i \), and the tour produced by the heuristic as \( H_i \). Suppose the vertex on the cycle that is nearest \( u \) is \( v \). Since the cost function satisfies the triangle inequality, it is easy to get \( c(H_i) \leq c(H_{i-1}) + 2c(u,v) \). So \( c(H_i) \leq 2\sum_i c(u,v) \).

Besides, we may notice that the way nodes and edges are added in closest-point heuristic is exactly the same as Prim’s algorithm. So the cost of MST produced by Prim’s algorithm is equal to \( \sum_i c(u,v) \).

In the test it is proved that: \( c(MST) \leq c(H^*) \), so we have \( c(H) \leq 2c(MST) \leq 2c(H^*) \). So it is proved.

**35.4-2**

The MAX-CNF satisfiability problem is like the MAX-3-CNF satisfiability problem, except that it does not restrict each clause to have exactly 3 literals. Give a randomized 2-approximation algorithm for the MAX-CNF satisfiability problem.

**Solution:**

Let each clause to have \( n \) literals. For a formula with \( k \) clauses, \( i = 1, 2, \ldots, k \), define

\[
y_i = \begin{cases} 0 & \text{if the } i\text{-th clause is not satisfied by the solution} \\ 1 & \text{if the } i\text{-th clause is satisfied by the solution} \end{cases}
\]

Then, number of satisfied clauses is \( \sum_{i=1}^{k} y_i \).

The expected cost of the solution of the randomized algorithm is \( \tau = E(\sum_{i=1}^{k} y_i) = \sum_{i=1}^{k} E(y_i) \).

The cost of optimal solution is \( c^* = k \).

Then, we have
\[
y_i = \begin{cases} 
0 & \text{with the probability } \frac{1}{2^n}, \\
1 & \text{with the probability } 1 - \frac{1}{2^n}.
\end{cases}
\]

So \( T = \left(\frac{2^n - 1}{2^n}\right)k \).

So the approximation ratio is \( \frac{2}{k} = \frac{2^n - 1}{2^n} = \frac{2^n}{2^n - 1} \).

35.4-3

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 is algorithm is a randomized 2-approximation algorithm.

Solution:

Suppose that for each vertex \( v \), we randomly and independently place \( v \) in \( S \) with probability \( 1/2 \) and in \( V-S \) with probability \( 1/2 \). For an edge \( e_i \), we define the indicator random variable \( Y_i = I\{e_i \text{ crossing a cut}\} \). For an edge \( e_i \) crossing a cut, its two vertices \( u, v \) have to be in \( S \) and \( V-S \) separately. The probability of such an event is

\[
P_i \{e_i \text{ crossing a cut}\} = P_i \{u \text{ in } S \text{ and } v \text{ in } V-S\} = P_i \{u \text{ in } V-S \text{ and } v \text{ in } S\} = 1/2 \times 1/2 + 1/2 \times 1/2 = 1/2.
\]

So \( E(Y) = 1/2 \).

Let \( Y \) be the number of edges crossing a cut, so that \( Y = Y_1 + Y_2 + \ldots + Y_n = |E| \).

We have

\[
E(Y) = \frac{1}{2}n.
\]

Let \( c^* \) be the weight for the max-cut. The upper bound of \( c^* \) is the total number of edges, i.e. \( c^* \leq n \). Then, we have

\[
c^* \leq n
= 2^\frac{n}{2}
= 2E(Y)
\]

So \( \frac{c^*}{E(Y)} \leq 2 \).
35.4-4

Show that the constraints in line (35.19) are redundant in the sense that if we remove them from the linear program in lines (35.17)-(35.20), any optimal solution to the resulting linear program must satisfy $x(v) \leq 1$ for each $v \in V$.

Solution:
Assuming that $\bar{x}$ is an optimal solution to the linear program in (35.17) and satisfies the constraints without the redundant (35.19). Round the solution to an integer algorithm as follow:

$$
\forall v \in V, \bar{x}(v) = \begin{cases} 
0 & \text{if } \bar{x}(v) < 1/2 \\
1 & \text{if } \bar{x}(v) \geq 1/2 
\end{cases}
$$

Return all the vertices in set $P = \{v \in V | \bar{x}(v) = 1\}$. The set $P$ is still a vertex cover, because for every edge $(u,v)$, $\bar{x}(v) + \bar{x}(u) \geq 1$ and $\max\{\bar{x}(v),\bar{x}(u)\} \geq 1/2$, therefore at least one of $u, v$ belongs to $P$.

Considering this LP in (35.17) is a minimization problem, $\max\{\bar{x}(v),\bar{x}(u)\}$ need to be as small as possible. Because $\forall v \in V, \bar{x}(v) \geq 0$ and $\bar{x}(v) + \bar{x}(u) \geq 1$, $\max\{\bar{x}(v),\bar{x}(u)\}$ will be at most 1. Therefore (35.19) is a redundant constraint.