Problem 1. Prove that any undirected simple bipartite graph with an odd number of vertices has no Hamiltonian cycle.

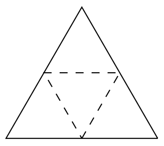
Solution: A bipartite graph cannot contain an odd cycle - the Hamiltonian cycle would be an odd cycle.

Problem 2. Someone gives you all the numbers from 1 to 99, and asks you to write them in a row so that the sum of any two consecutive number is divisible by 5, 7 or 13. For example, 4, 9, 1 works, but 1, 6, 2 does not. Express this problem as a graph theory problem (you do not need to solve it).

Solution: We can model this problem using an undirected graph $G = (V, E)$. Each vertex in the graph represents a number between 1 and 99. If the sum of two numbers i and j can be divided by 5,7 or 13, we connect the two vertices i and j using an undirected edge. Then original problem is equivalent to find a Hamiltonian path in this graph, as we can sequentially write the numbers represented by the vertices along the Hamiltonian path as a solution to the original problem.

Problem 3. There are five points inside an equilateral triangle of side length 2. Show that at least two of the points are within 1 unit distance from each other.

Solution: We can divide the equilateral triangle into four small equilateral triangles of side length 1, as shown in the following figure. We note that two points in the same equilateral triangles of side length 1 have a distance within 1. We consider the four small triangles as holes and the five points as pigeons. Then according to the pigeonhole principle, we have at least two points within 1 unit distance from each other.



Problem 4. Consider the complete bipartite graph with $n = 2k$ vertices on each side. How many edge-disjoint spanning trees can we find?

Solution: The number of edge-disjoint spanning trees we can find is k. First, we show that we cannot achieve more than k such trees. Suppose we have more than k edge-disjoint spanning trees. Since a spanning tree containing $4k$ vertices have $4k - 1$ edges, we need at least $(k + 1)(4k - 1) = 4k^2 + 3k - 1$ edges, which is greater than $4k^2$, the number of edges in a complete bipartite graph with $n = 2k$ vertices on each side.

Next, we need to show that we can find such k edge-disjoint spanning trees. Clearly, to be able to do so, each vertex needs to have degree at most 2 in each tree. We are thus going to try to find k edge-disjoint Hamiltonian paths. We denote the vertices on one side by $a_1, a_2, \ldots, a_{2k}$ and the vertices on the other side by $b_1, b_2, \ldots, b_{2k}$. Then we can find k paths (which are also trees) in many different ways. One construction is as follows.

$$a_1, b_1, a_2, b_2, a_3, b_3, \dots, b_{2k}$$

$$a_1, b_4, a_2, b_5, a_3, \dots, b_1, a_{2k}, b_2$$

$$a_1, b_6, a_2, b_7, a_3, \dots, a_{2k}, b_4$$

Basically rotate the right side by two steps, so that we always use edges that were not employed before.

Problem 5. In the following, we will give you some problem descriptions, and we will ask you to match them to graph theory problems. In each case, you need to specify what are the vertices and the edges of the graph, what problem does it map to (and mapping details). For example, if it maps to an edge coloring problem, you also need to say, what do the edge colors correspond to in the original problem.

1. In computer networks, a link can be attacked. In order to detect the attacks on time, we deploy anti virus tools on some of the computers. If a link is attacked, the computer directly connecting to it can discover the attack. We want to make sure that each link is under detection. How can we choose a minimum number of computers to deploy the tools?

   Solution:

   - vertices: we model the computers as the vertices.
   - edges: we model the links between computers as the edges.
   - problem it maps to: we map the problem into a vertex cover problem.
   - mapping details: we want to find a minimum size vertex cover to cover all edges. The minimum size vertex cover corresponds to the computers we choose to deploy the tools.

2. In vehicle to vehicle communications, we would like to build a network such that any two of the n vehicles can communicate with each other. There is a cost $c_{ij} = c_{ji}$ associated with any two vehicles i and j if there is a communication link between them. The total cost of the network is the summation of the cost for each built link. Find a network using the least cost.

   Solution:

   - vertices: we model the vehicles as the vertices.
   - edges: we add an edge between two vehicles i and j with a weight c_{ij} representing the cost if there is a link connecting them.
   - problem it maps to: we map the problem into a minimum (weight) spanning tree problem.
   - mapping details: we want to find a minimum spanning tree in such a graph. The edges in the minimum spanning tree are the links we need to keep in the vehicle to vehicle communication network.

3. Suppose you want to write all 2^n binary strings of length n in a column, so that, from row to row, only one bit changes. For example, for $n = 2$, a valid such column is:

01
00
10
11

That is, we can write, for example, after 00 the string 01, but not the string 11, as 00 and 11 differ in two bits. The question is whether we can always do this for the arbitrary integer $n \geq 2$.

Solution:

- vertices: we model the 2^n binary strings as vertices.
- edges: we add an edge between two binary strings if and only if they differ in one bit.
- problem it maps to: the problem can be mapped to whether we can find a Hamiltonian path in the graph.
- mapping details: if a Hamiltonian path exists, we can sequentially write the strings (vertices) along the Hamiltonian path in a column and any two strings differ in one bit. Otherwise, we cannot always do this. In fact, we learned from class in such a graph, a Hamiltonian path always exists.

4. Consider a set of binary vectors $S = \{s_1, \ldots, s_n\}$. Each vector s_i $(i = 1, 2, \ldots, n)$ in S has length k, and they are all different. We want to find the shortest length binary vector x that contains inside it all the vectors in S. For example, given $S = \{001, 010, 100\}$, we can find a string 10010 that contains each of the three elements in S as a substring, and we can always concatenate all vectors in S together and create a vector of length kn that contains all vectors in S.
Hint: we can slightly modify a canonical graph theory problem to solve this.

Solution:

- vertices: we model the n binary vectors as the vertices.
- edges: we add a directed edge from a vertex i to every other vertex j with weight $w_{i,j}$ being the maximum number of overlaps when putting j behind i. Here the maximum number of overlaps when putting a vector s_j behind a vector s_i is the maximum number l (which can be 0) such that the last l elements of s_i are the same as the first l elements of vector s_j. For example, the weight is 2 from vertex (100) to (001) and the weight is 0 from vertex (001) to (010).
- problem it maps to: we want to find the Hamiltonian path with the maximum weight.
- mapping details: for this complete directed graph, if we find a Hamiltonian path and concatenate the vectors along the path (by removing the overlaps) to get a vector x. Then the weight of this Hamiltonian path is $kn - \mathit{length}(x)$. So if we can find the maximum weight Hamiltonian path, we can find a minimum length vector x that contains all n vectors.