1 Nonhamiltonian Odd Bipartite Graphs (20 points)

Question: Prove that if G is an undirected bipartite graph with an odd number of vertices, then G is nonhamiltonian.

Solution:

From intuition that we know this bipartite graph has two parts that contain different number of vertices, and the sum is odd, then there must be one side that contains larger number of vertices, let's say the one side has m vertices, and the other side has n vertices($m < n, n - m \geq 1$).

Then no matter from which side we try to start the cycle, we will return to the larger side since it has more vertices, after that, there is either one vertice left in this side, or other odd numbers of vertices left in this side that is not visited. If we want to form a hamiltonian cycle, we should return to the start-point, so it means we need to add at least an edge from the same side so that it will contain all the points that are not visited. But it contradicts with the condition that it's bipartite, which mean that it should not have an edge for the same side, otherwise it cannot be divided into the same side. So this proof can demonstrate that it is nonhamiltonian.

2 Enumerating Vertices (20 points)

Question: Show that if HAM-CYCLE $\in P$, then the problem of listing the vertices of a hamiltonian cycle, in order, is polynomial-time solvable.

Solution:

From the description we know that it assumes HAM-CYCLE could be solved in polynomial time, so that it belongs to P-problem. From this condition, we could get a basic understanding that solving this problem seems "harder" than just listing the vertices of a hamiltonian cycle, as it seems that the second problem could be solved if the first is solved.

Above this assumption, we could add some extra steps in the solving process of HAM-CYCLE, so first when we try to solve the HAM-CYCLE, each step that moves the vertice from one to another, we record the start-point and end-point, as well as the edge between them, by this way, we will record all the possibility of the selection. And if there is some backward steps, then we just need to delete these edges and points, so that we could always maintain a point-sets and edge-sets, which record the right PATH that forms a solution for the hamiltonian cycle.

At last, we know that the original solution could take a polynomial time, so our method that records the vertices and edges could also take the **same** complexity, which means that if HAM-CYCLE $\in P$, then the problem of listing the vertices of hamiltonian cycle in order, is also polynomial-time solvable.

3 Hamiltonian Path (10 points)

Questioin: A **hamiltonian path** in a graph is a simple path that visits every vertex exactly once. Show that the language HAM-PATH=$\{< G, u, v >$: there is a hamiltonian path from u to v in graph $G\}$ belongs to NP.

Solution:

From a general view, if we want to prove that a problem is NP-problem, then we just need to prove that it can be verified in polynomial time. So that's say we just need to prove the given a solution of HAM-PATH, we could verify it in a polynomial time. Now let's give a formal proof.

If we have got a solution for HAM-PATH, which is to say that given two vertices (u, v), we could find a hamiltonian path from u to v. Then in order to verify it, we just need to do the following verification:

1) check all the vertices in this path that are **ALL** listed in the original graph, without any missing or abundant(which fits the definition of hamiltonian path).

2) check that the vertices are visited exactly only once, which is to say that no repeated vertices is visited.

3) check each edge in this path is contained in the original graph, no extra edges that doesn't belong to graph are added.

After all these verification, we could know whether this path is a solution or not for this problem. And let's compute the complexity of this procedure, it is $\boldsymbol{O(|V| + |E|_{path})}$. Since for a given graph its vertex and edges could be determined in polynomial time, it's easy to verify in polynomial time. So this problem could be verified in polynomial time, and it is a NP-problem.

4 Hamiltonian Path in DAG (10 points)

Question: Show that the hamiltonian-path problem from Exercise 34.2-6 can be solved in polynomial time on directed acyclic graphs. Give an efficient algorithm for the problem.

Solution:

For this particular problem, we could find a polynomial-time solution for it and then it's proven it could be solved in polynomial-time.

In this following step, I will give some steps that will demonstrate it will resolve the problem:

1) find a path in the original graph G that in-degree is zero, because for a DAG it must have a source node that in-degree is zero. If there're more than one nodes, then there is no solution, since every node must be visited, but the zero degree should start first and if there're more than one point, then there must be some nodes that left cannot be visited at first.

2) remove the source node which in-degree is zero and its neighboring edges so that we could find the next node to start, let's assume that the left graph is G'.

3) if the left graph G' has more than two nodes or vertices that have the in-degree which is zero, then we could judge that it's unsolvable, because as I said in 1) there is no way to visit those nodes since there in-degree is zero and there should only be one to visit after its predecessor.

4) keep doing the 3) until we have reached the end-point, which there is no other nodes to visit. If there is some nodes that are not visited but we cannot find an edge to reach it, then it's unsolvable, otherwise we should get the solution.

From the above steps, we have found the solution for the hamiltonian-path, and its complexity if $O(|E| + |V|)$, while here $|V|$ is the number of vertices in the graph and $|E|$ is the number of edges in the graph, because we have to trace the vertices and its neighborhood edges from v_1 to v_n. We know that it is done in polynomial time.

5 DNF-SAT$\in P$(20 points)

Question: Show that the problem of determining the satisfiability of boolean formulas in disjunctive normal form is polynomial-time solvable.

Solution:

First, according to the definition of DNF, it has the form like $A_1 \cup A_2 \cup A_3 \cup ... \cup A_n$, and each of the element $A_i (i \leq n)$ should also have the form like $B_1 \cap B_2 \cap B_3 \cap ... \cap B_m$.

Unlike CNF that we need to check all the elements A_i at the same time in order to get the solution, here since it is the '$\cup$' relation, so in order to check the satisfiability, we just need to check anyone of the element A_i is satisfiable.

Image that we check one of them, for example A_i, then to determine the satisfiability of it we have to check whether it is possible that all its sub-elements B_j are true at the same time. Since it has finite elements in DNF, so it takes less than $O(n^2)$ to finish the determining(n is the number of the sub-elements). The specific algorithm is just to use two loops to determine that there is no conflicts of them.

So for the whole problem it is just to select anyone of them and it's also $O(n^2$, it is solvable in polynomial time. And the proof is reached.

6 Finding SAT Assignments (20 points)

Question: Suppose that someone gives you a polynomial-time algorithm to decide formula satisfiability. Describe how to use this algorithm to find satisfying assignments in polynomial time.

Solution:

The steps are as follows:

1) Find all the sub-elements and decide its satisfiability. Since the original algorithm could determine the satisfiability of the whole formula, then for SAT, each of its element should be checked to be taken into consideration about the final decision. Whether each of them are satisfiable could be determined. In this way, the problem is divided into some small problem sets $A_1, A_2, A3, ...An$, and any of their satisfiablility is determined.

2) For any A_i, since its satisfiability is determined, then it forms another SAT formula, and according to 1), we could also determine its sub-elements' satisfiability for $B_1, B_2, B_3, ...B_n$.

3) Apply this scheme iteratively, then we could divide all the sub-element's satisfiability. If it works then the value is ture, otherwise it's false. By iteration, we could finally get the formula that just contain one element like Z or $\neg Z$, and we also know its satifiable or not.

4) For the last step, when getting only one element in the formula, if it's satisfiable, then $Z's$ value is true, otherwise its value is false. In this way, **all** the elements' value is determined.

So according to the algorithm, by using the same complexity algorithm of determining the satisfiability of formula, we could also get the its solution in polynomial time.

7 Longest Simple Cycle (20 points)

Question: The **longest-simple-cycle problem** is the problem of determining a simple cycle (no repeated vertices) of maximum length in a graph. Formulate a related decision problem, and show that the decision problem is NP-complete.

Solution:

The decision problem could be described as follows:

1) First randomly pick one vertice of the graph.

2) Find the neighborhood vertices that is connected with the previous vertice, and for each of them, **decide** whether we should add it or not.

3) Keep the process as described in 2), for those we picked that are not visited before, decide the next vertice that connects to the previous node we have choosen, for those we don't pick, just ignore it(maybe it will be chosen in next decision round)

4) Until we have reached the a vertice which could connect to the first vertice, it's a circle and calculate the length of this circle and mark it down. If we have finished all the nodes but still don't reach the first node, then it is not a circle, just ignore these cases.

5) Try the above process for every node in the graph, and calculate the length of the circle.

6) For all the candidates in the above steps, by comparison get the longest circle and its selection vertice with path.

Now that I will show that it's a NP-complete problem:

1) First, we should show that it's a NP problem, which could be verified in polynomial time. So to verify it, we just need to verify that given a solution it can be verified in polynomial time, then first we verify that it is a circle, the last vertice connects back to the first vertice, then it has no repeated vertices. All of these could be checked in polynomial time, so it is a NP problem.

2) Second, we should do a polynomial reduction to show that this decision problem could be reduced from another NP-complete problem. If this is reached, then we could know that the decision problem is more complex than the previous NP-complete problem, and it is also a NP-complete problem.

3) From the given knowledge we know that Hamiltonian cycle and traveling salesman problem are both NP-complete problem, for traveling salesman problem, we know that it tries to find the smallest path that visited all the nodes exactly once. So our problem could be reduced from TSP problem.

4) Image that in the original graph G, if we pick just some of their vertices$v_1, v_2, ...v_k$, then for these points, if we want to find the smallest path, it is a TSP problem. Then mapping this problem to ours, add a negative function $f(x) = -g_{TSP}(x)$, it TSP reached the smallest path, then the value of $f(x)$ is maximum in this situation, we should know that it's at least as hard as TSP problem.

5) In another way, the picking of some vertices in 4) is also a subset of the whole vertices in graph G, so our decision problem has an upper lay that decide the number of vertices, which is $O(n)$, that's to say that our problem has another pre-decision process of the vertices, so it should be harder then TSP problem.

6) From the above proof, we know that it is harder and meets the defintion of NP-complete problem. The conclusion is reached.