

Name:

Student ID:

Conditions: No notes, books, consultation, or any kind of assistance allowed. The time limit will be announced. Be sure to show your work where indicated.

Problem 1:

- In the vertex cover problem, we are given a graph G and a value k . The question is, are there k vertices of G that cover all edges of G ?
- The baseball card collector problem is as follows. Given packets P_1, \dots, P_m , each of which contains a subset of that year's baseball cards, is it possible to collect all the year's cards by buying $\leq k$ packets?

Prove that the baseball card collector problem is NP-complete using the fact that vertex cover problem is NP-complete.

See the chapter six review (a link is on the homepage)

Problem 2:

- In the Hamiltonian path problem, we are given a graph G and the question is, "Does there exist a path in G that visits every vertex exactly once?"
- The *low degree spanning tree problem* is as follows: Given a graph G and an integer k , does G contain a spanning tree such that all vertices in the tree have degree at most k ?
- The *high degree spanning tree problem* is as follows: Given a graph G and an integer k , does G contain a spanning tree such that the highest degree is at least k ?

(a) (3pt) : Prove that the low degree spanning tree problem is NP-complete using the fact that the Hamiltonian path problem is NP-complete.

See the chapter six review (a link is on the homepage)

(b) (3pt) : Give an efficient algorithm to solve the high degree spanning tree problem and an analysis of its time complexity.

The algorithm here turns out to be simple:

HDST(G, k)

1. Find the highest degree vertex
2. If the highest degree is less than k then return FALSE else return TRUE

This works since if the highest degree is less than k , clearly no spanning tree exists with a vertex of degree at least k . If the highest degree is at least k , then we can build a spanning tree that includes the highest degree vertex and all of its incident edges. One way to do this is to assign edge weights of 1 to the incident edges and 2 to all others and then apply Prim's or Kruskal's algorithm.

The time complexity of the algorithm is the time complexity of finding the highest degree vertex. This can be done using BFS in $O(n + m)$.