Student name: Quan Hong Doan

Student ID: 986956

Assignment: Lab 12

Week: 12

Due: Jan 6, 2020

**Lab 12**

**Problem 1**:

Hamiltonian Graphs

**Answer**:

From the given graph, the Hamilton cycle is below.



**Problem 2**:

Vertex Covers. Create an algorithm for computing the smallest size of a vertex cover for a graph. The input of your algorithm is a set V of vertices along with a set E of edges.

**Answer**:

Algorithm: findMinimumVertexCover()

Input: set V of vertices and set E of edges.

Output: number of smallest size of a vertex cover

Arrays of size of vertex covers arrSize[]

Set W <- all subsets of V

For each subset w of set W

vertexCoverFlag <- true

For each edge *e* of set E

Vertex *v1, v2* <- computeEndPoints(*e*)

if belongsTo(*v1, w*) or belongsTo(*v2, w*) <- FALSE

vertexCoverFlag <- FALSE

break;

if vertexCoverFlag = TRUE

arrSize[++] <- size of subset *w*

Sort arrSize[] in ascending order

Return arrSize[0]

**Problem 3**:

The decision problem formulation of the Vertex Cover problem is this: Given a positive integer k, and a graph G, is there a vertex cover for G having size ≤ k? Show that this decision problem belongs to NP.

**Answer**:

The solution of the vertex problem is a subset V’ of V, which contains the vertices in the vertex cover. We need to decide whether the set V’ is a vertex cover of size k for graph G(V, E). Let assume k is the smallest size of the solution. The following is the simplest way to decide if the solution is right.

Set counter as integer, and initially equals 0.

For each vertex *v* in V’

Remove all edges adjacent to *v* from set E

Increment the counter by 1

If counter = k and E is empty then

The given solution is correct

Else

The given solution is wrong.

Obviously, this can be done in polynomial time, and we can see there’s no other minimum vertex cover size less than *k*. Therefore the vertex cover problem is NP class.

**Problem 4**:

The goal of this exercise is to devise a feasible algorithm that decides whether an input integer is prime.

What could be done to improve the degree of correctness of FirstTry but still preserve a reasonably good running time? Explain.

**Answer**:

In fact if the input is composite, the f(a,n) would return 1 with a probability less then ½. That means if we try after certain times, we can determine the input number is prime or composite. This is similar to the Fermat’s Little Theorem <https://en.wikipedia.org/wiki/Proofs_of_Fermat's_little_theorem>.

* If n is prime, then always return true.
* If n is composite then return false with high probability
* Higher value of k times of trying increases the probability of the correct result.

Below is a sample of Fermat implementation.

**static** **boolean** isPrimeFermat(**int** n) {

**int** k = 3;

**if** (n <= 1 || n == 4)

**return** **false**;

**if** (n <= 3)

**return** **true**;

// Try k times

**while** (k > 0) {

// Pick a random number in [2..n-2]

// Above corner cases make sure that n > 4

**int** a = 2 + (**int**) (Math.*random*() % (n - 4));

// Fermat's little theorem

**if** (powerFermat(a, n - 1, n) != 1)

**return** **false**;

k--;

}

**return** **true**;

}

**Problem 5**:

Show that if a graph G has |V| -1 edges and has no cycle, then G is connected.

**Answer**:

G is a graph with *v* vertices and *e* edges.

Assuming G is a tree, then every two vertices are joined by a unique path, and the G is acyclic (no cycle) as given. So if *v = e + 1* then G is connected since any two vertices are connected by a path. V = e + 1 (in other words, e = v - 1). We need to prove it.

Assume that it’s true for less than *v* vertices. Removing any edge from G will break G into two components, since each path is unique. Suppose the size are v1 and v2,

Which v1 + v2 = v

Then v1 = e1 + 1

And v2 = e2 + 1.

But v = v1 + v2 = e1 + 1 + e2 + 1 = e1 + e2 + 2 = e – 1 + 2 = e + 1

In other words, if G has |V| - 1 edges, no cycle, then G is connected.