Abstract – A problem of interest in Computer Science is determining a path that will visit each vertex exactly once, in an undirected graph. This is often referred to as the Traveling Salesman problem. In a typical Traveling Salesman problem each city or town is visited one time and then the salesman returns to his/her point of origin. This is also known as a Hamiltonian Cycle. In this project we explore the Hamiltonian Path which does not complete the ‘cycle’.

1. INTRODUCTION

The Hamiltonian Path problem is named after William Rowan Hamilton who in 1857 created the Icosian game. The object of this game is to find a Hamiltonian cycle around the edges of the dodecahedron such that each vertex is visited exactly once and the starting point is the same as the ending point. A Hamiltonian Path is essentially the same problem without the requirement that the path ends where it began.

In this project we examine the Hamiltonian Path problem. This gives a different result than the cycle as it is not important at which vertex we start or finish.

The figure above shows one possible solution for one problem. The edges are in blue and the Hamiltonian Path is in black. An infinite number of these problems exist depending on the application. This is an NP-Complete problem ideal for a computer algorithm.
2. THE PROBLEM

The problem used a grid of equal length and width with edges available between each vertex in the x and y direction as such.

```
X-X-X  
| | |  
X-X-X  
| | |  
X-X-X  
```

The application was written to find a path through the grid visiting each node exactly one time. A solution to the above problem may look like.

```
X X-X  
| | |  
X X X  
| | |  
X-X-X  
```

The larger the grid, the larger the number of Hamiltonian Paths available.

Each vertex shares two edges except for the start and end vertexes. The number of edges is:

\[\text{num\_edges} = \text{size} \times \text{size} - 1\]

3. DEVELOPMENT ENVIRONMENT

Visual Studio 2008 C# was used to implement the Hamiltonian Path algorithm.

4. ALGORITHM

A randomized algorithm was used:

Start at vertex(0,0).
While edges less than size \(*\) 2 -1
{
    if unvisted neighbor
    {
        // move
        Randomly move to unvisited neighbor. Add edge
    }
    else
    {
        // find a pivot
        move to a neighbor at random
        delete the most recent edge (other than the last).
        find the new end of the path
    }
}

This algorithm was modified from the Wikipedia randomized algorithm. The wiki algorithm deleted a random edge. However; deleting the wrong edge often created a cycle. After some experimentation it was determined that deleting the most recent edge (not the last move) solved this problem.
5. TEST DATA

For each size grid the test was run three times. The largest grid that was able to run, in a reasonable amount of time, was an 8x8 grid (64 nodes).

<table>
<thead>
<tr>
<th>size</th>
<th>Run 1</th>
<th></th>
<th></th>
<th>Run 2</th>
<th></th>
<th></th>
<th>Run 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time (sec)</td>
<td>time (msec)</td>
<td>time (sec)</td>
<td>time (msec)</td>
<td>time (sec)</td>
<td>time (msec)</td>
<td></td>
</tr>
<tr>
<td>3x3</td>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>4x4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>0</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>5x5</td>
<td>0</td>
<td>161</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>97</td>
<td></td>
</tr>
<tr>
<td>6x6</td>
<td>0</td>
<td>769</td>
<td>0</td>
<td>981</td>
<td>0</td>
<td>239</td>
<td></td>
</tr>
<tr>
<td>7x7</td>
<td>3</td>
<td>79</td>
<td>26</td>
<td>322</td>
<td>7</td>
<td>595</td>
<td></td>
</tr>
<tr>
<td>8x8</td>
<td>49</td>
<td>845</td>
<td>39</td>
<td>977</td>
<td>11</td>
<td>346</td>
<td></td>
</tr>
</tbody>
</table>

Hamiltonian Path - Data

6. RESULTS

Each time a test was run the time required varied. This is likely due to other tasks being run by windows. Therefore; each test was run three times to get an average.

Figure 4 shows the time required for each test run. Even though each test run varies, we can see how the time increases for each graph.

Averaging the results of the runs, in Figure 5, shows more clearly how the run time increases exponentially as more nodes are added.
7. CONCLUSION

The Hamiltonian path algorithm demonstrates how a computer can perform many operations and decisions much quicker than a human. However, as the data shows this is an NP Complete problem and even a computer has its limitations.

8. REFERENCES


**Samuel Barlow** received his B.S. in Computer Science from Florida Atlantic University in 2001 and is currently pursuing a MSCS from the Florida Institute of Technology. He joined Northrop Grumman Laser Systems in 2003 and currently works in the Embedded Software department.