Design & Analysis of Algorithms
(Prof. Parminder Kaur Walthwa)

CAMPUS 2 Date _____ Page _____

Hamiltonian Cycles

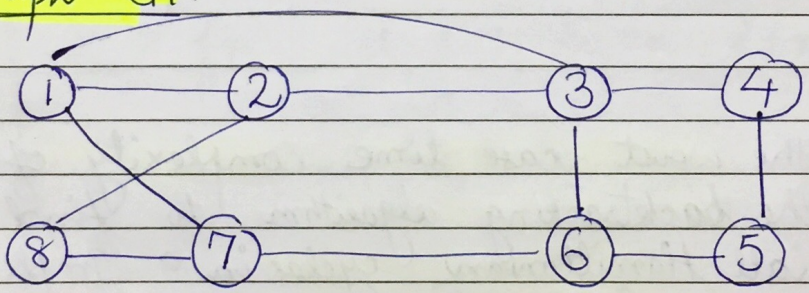
(using backtracking)

Let $G = (V, E)$ be a connected graph with n vertices. A Hamiltonian cycle is a round-trip path along n edges of G that visits every vertex once and returns to its starting position.

(It is same as was the tour in a Travelling Salesperson Problem, but it just finds the cycle and have no concern with costs of edges.)

For example:-

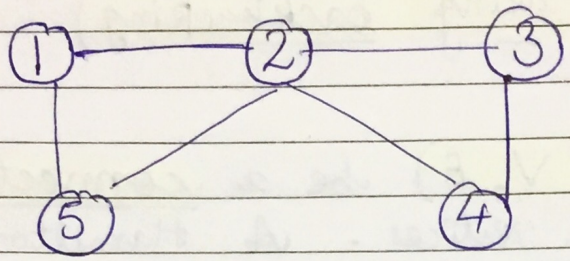
Graph G1:



This graph contains the following Hamiltonian Cycle:-

1, 2, 8, 7, 6, 5, 4, 3, 1

Graph G2 :-



The above graph contains no Hamiltonian cycle.

The backtracking algorithm finds all the Hamiltonian cycles in a graph. The graph may be directed or undirected. Only distinct cycles are output.

The worst case time complexity of the backtracking algorithm to find all Hamiltonian cycles in a graph is

$\Rightarrow O(n!)$

A backtracking algorithm to find all the Hamiltonian cycles in a given graph G.

```
Void Hamiltonian (int k)
// The graph is stored as an adjacency
// matrix G[1:n][1:n]. All cycles begin
//at node 1.
{
    do
    {
        NextValue(k); // Assign a legal next value to x[k]
        if (!x[k]) return; // no vertex is possible
        if (k==n) // All vertices are visited
        {
            for (int i=1; i<=n; i++)
                cout << x[i] << " "; cout<<"1";
                cout << endl;
        }
        else Hamiltonian (k+1);
    } while (1);
}
```

true

false

end the cycle at vertex 1

recursive call

The detail of NextValue function for Hamiltonian Cycles problem

```
Void NextValue(int k)
{
  do
  {
    x[k] = (x[k]+1) % (n+1);        // Next vertex
    if (!x[k]) return;              // all vertices are selected and no vertex is left
    if (G[x[k-1]][x[k]])            // (k-1) edge? Is there any edge?
    {                               // True
      for(int j=1; j<=k-1; j++)
        { if x[j] == x[k]) break; } // It has already been visited in the past; reject the vertex
      if (j==k)                     // j will be k only if the for loop is successfully completed means the vertex has not appeared in the past.
      {
```

all vertices are not visited,

all vertices are visited

```
if ((k < n) || ((k == n) &&
        G[x[n]][x[1]]))
    return;
} while (1);
```

and there is an edge from the last chosen vertex to vertex 1 to complete the cycle

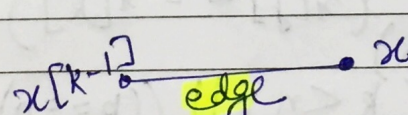
check any another vertex to be included in the cycle

In the above mentioned function, $x[k]$ is assigned to the next highest numbered vertex which :-

(i) does not already appear in $x[1], x[2], \ldots, x[k-1]$

and

(ii) and is connected by an edge to $x[k-1]$.



Summaizing:- Backtracking Algorithm to find all Hamiltonian Cycles in a graph:-

```
void Hamiltonian (int k)
{
  do
  {
    NextValue (k);
    if (!x[k]) return;
    if (k==n)          // all vertices are visited
    { for (int i=1; i<=n; i++)
        cout << x[i] << " ";
      cout << "1"; cout << "\n";
    }
    else Hamiltonian (k+1);
  } while(1);
```

```
void NextValue (int k)
do {
    x[k] = (x[k]+1) % (n+1);
    if (!x[k]) return;
    if (G[x[k-1]][x[k]])   // if edge is there
    { for (int j=1; j<=k-1; j++)
        if (x[j]==x[k]) break;
      if (j==k)
      { if ((k<n) || ((k==n) && G[x[n]][x[1]]))
          return;
      }
    }
} while (1);   // find any other vertex
```

true

false