Hamiltonian Cycles

Introduction

- Hamiltonian cycle (HC): is a cycle which passes once and exactly once through every vertex of G (G can be digraph).
- Hamiltonian path: is a path which passes once and exactly once through every vertex of G (G can be digraph).
- A graph is Hamiltonian iff a Hamiltonian cycle (HC) exists.

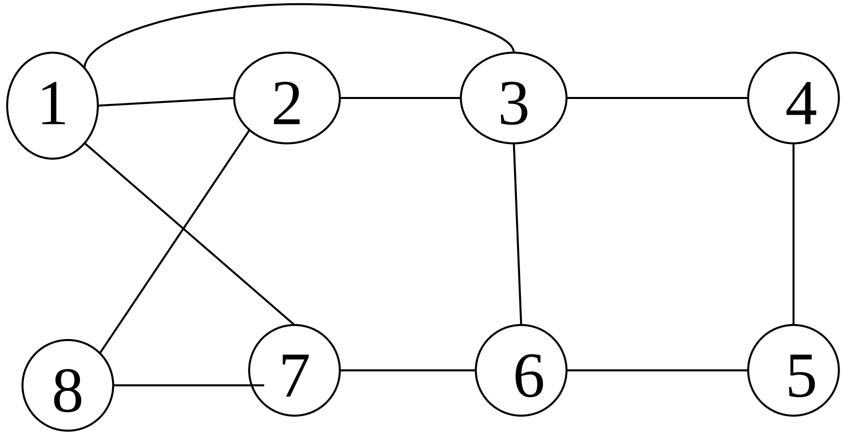
History

- Invented by Sir William Rowan Hamilton in 1859 as a game
- Since 1936, some progress have been made
- Such as sufficient and necessary conditions be given

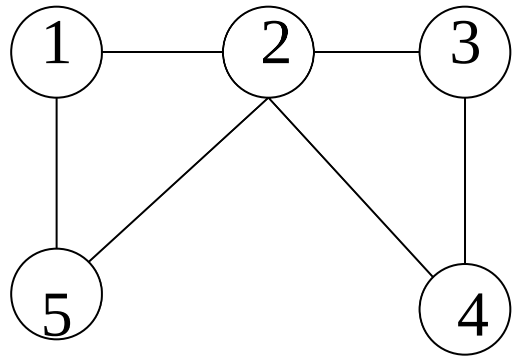
Application

- Hamiltonian cycles in fault random geometric network
- In a network, if Hamiltonian cycles exist, the fault tolerance is better.

Which of following graph contain Hamiltonion?



G1



G2

- Graph G1 contain hamiltonian cycle and path are 1,2,8,7,6,5,3,1
- Graph G2contain no hamiltonian cycle.
- Here solution vector (x1,x2,…,xn) is defined so that xi represent the I visited vertex of proposed cycle.
- The algorithm is started by initializing adjacency matrix G[1:n,1:n,then setting x[2:n] to zero & x[1] to 1, then executing Hamiltonian(2).

```
Algorithm Hamiltonian(k)
{
repeat
{
    Nextvalue(k)
    if(x[k]=0) then write( x[1:n]);
    if(k=n) then Hamiltonian(k+1);
}until(false)
}
```

```
Algorithm Nextvalue(k)
{
repeat
{
x[k]:=x[k]+1 mod(n+1)     //Next vertex
if(x[k]=0) then break
if(G(x[k-1],x[k])≠0) then
{//is there an edge?
for j:=1 to k-1 do
 if(x[j]=x[k]) then break;
//check for distinctness
if(j=k) then
     if(k<n) or(k=n) and G[x[n],x[1]]≠0)
               then return;
}}}
```

- Hamiltonian will find a minimum cost tour if a tour exists.
- If the edge cost is c, the cost of a tour is cn since there are n edges in a hamiltonian cycle

Scope of research

- To develop such an algorithm for Hamiltonian cycle which is more efficient in term of time complexity.

Knapsack Problem

- Here instead of considering no. of items, we consider that we have n types of items & that proper no. of items of each type is available. This 0/1 knapsack problem

```
algorithm BackKnapsack(T,W)
{
b=0
for i= 1 to n do
if(wi≤W) then
{
   b=max(b, pi  + BackKnapsack(I,W-wi))
}
return b
}
```

T=<T1,T2,T3>

w=<2,3,4>

p=<3,4,5>

W=5

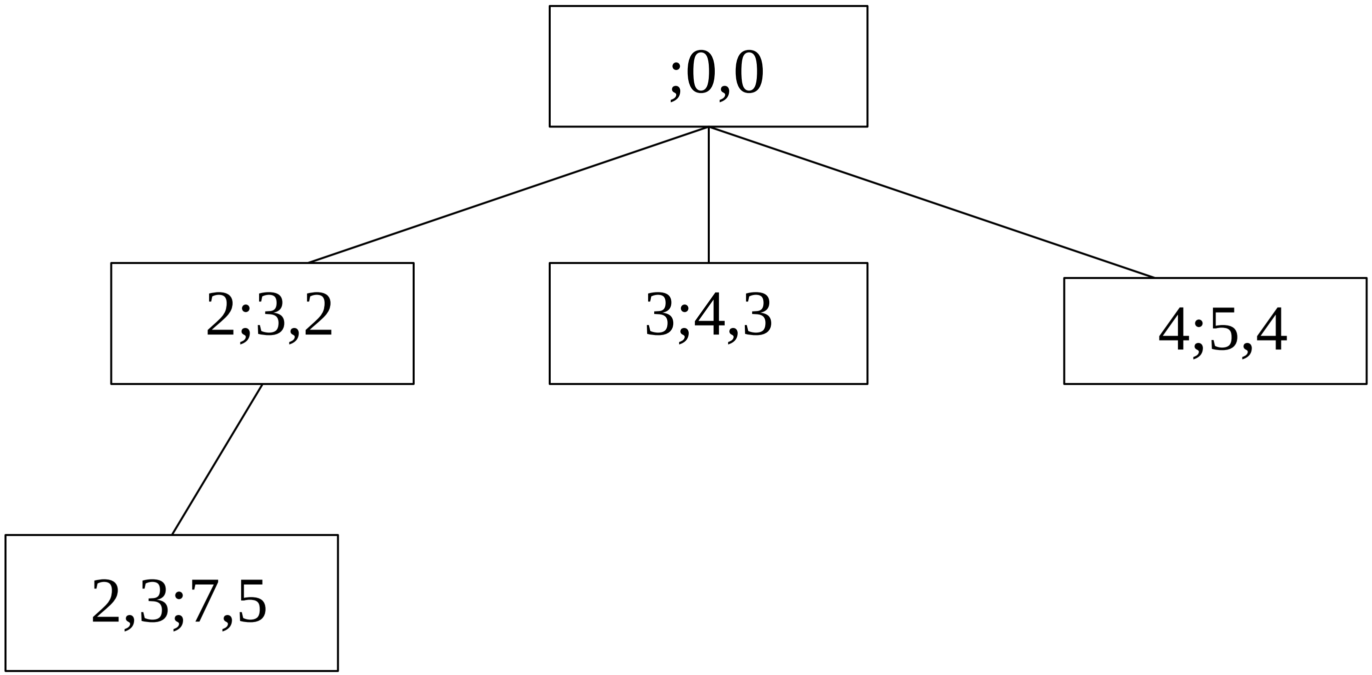


Fig1 :The implicit tree of Knapsack problem.

We have the following node structure

;0,0

where left hand side of semi-colon corresponds to the weight chosen & the first entry to the write hand side correspond to total profit & second entry correspond to the weight till now included in Knapsack. Such a node structure correspond to a partial solution. At every move down to the children we select type of item to be added in the knapsack. Here we select the item according order of weight(it can be increasing order of weight or you can select profit) so that search for the tree is decreased.

For example,, if we once visit node(2,3; 7,5) then next time we do not visit node

(3,2;7,5). The first node visited is (2;3,20 the next is (2,3;7,5). It can be seen that as each new node is visited the partial solution is also extended. After visiting theses two nodes the dead end comes as node(2,3;7,5)has no unvisited successor, since adding more items to the partial solution violates the knapsack capacity constraint. Also this partial solution violates knapsack capacity constraint. Also this partial solution produces the optimal solution so far, thus we memorize it. Next we backtrack one step and find that new addition(2,4;8,6) will also violate the capacity constraint.

In the same manner we backtrack one step & proceed the search and it will continue until we get optimal solution. Exploring whole tree in this way find node(2,3;7,5) to be optimal solution. This is optimal solution for our problem with maximum capacity 7 and T1 & T2 are include into the knapsack.

Assignment

- Q.1)Define hamiltonian cycle.
- Q.2)Write an algorithm for hamiltonian cycle.
- Q.3)Explain Knapsack problem using Backtracking method.