worst case complexity:

no of vertex = V

no of edges = E

visiting each vertex = O(V)

visiting each edge = O(E)

checking adjacent edges of every vertex = O(V*2E)

∴ Time complexity = O(V * E)

Algorithm: greedy - colouring(V[], E[])

where, V[] is the set of vertex
E[] is the set of edges

step 1 : START

step 2 : Declare L[] as the set of colours from 1 to N

step 3 : for i = 1 to N do step 4

step 4 : L[i-1] = i

step 5 : for i = 1 to N repeat step 6 to 8

step 6 : set c_i colour to v_i

step 7 : for each j till $i < j \wedge (v_i - v_j) \in E(G)$ do step 8

Step 8 : set $L_j = L_i / c_i$

step 9 : END

worst case complexity:

If the no of vertex in graph = n

The program is designed to visit all the vertices possible by connecting all the vertices from a starting source vertex and all the vertices should be visited exactly one.

- Starting from one vertex the algorithm has to connect rest of the n-1 number of vertices.
- Number of possible routes included n-1 number of vertices = (n-1)!
- Time complexity = $O(n!) \simeq O(n^n)$

Algorithm : Hamiltonian(v,k)

where, v is the vertex

k is the position

step 1: START

step 2: check if k is valid, if not go to step 3

step 3: If no node edge bet^n node(k-1) to v

return false

step 4: If v taken

return false

step 5: Else, return true;

step 6: If there is an edge bet^n k and 0

return true;

else return false

step 7: If is valid (v,k) then

add v to path

step 8 If cycle bound(k+1) is true

return true

else remove v from path

step 9: END