Given a graph find whether K(3,3) or K(5) is present as subgraph or not.

Jatin Goel [1], Rohit Raj[2], Mohd. Abdullah [3]

[1]IT Department,Indian Institute of Information Technology,Allahabad
Deoghat,Jhalwa-211015 India
icm2014503@iiita.ac.in
ism2014003@iiita.ac.in
ism2014004@iiita.ac.in

Abstract - Graph planarity is an important concept in fields of graph theory which has several applications in different fields of engineering, especially electrical engineering. Kuratowski mentioned about the graph planarity that a graph is certainly non planar if there exists a subgraph in the graph which is isomorphic to K(3,3) or K(5). In this report we are providing algorithm to check existence of K(3,3) or K(5) in a given graph.

1. Introduction

We are given a simple graph G.We have to find whether two special type of graphs K(5) or K(3,3) present in the graph G as a subgraph.

K(5) - This is complete graph of 5 vertices.

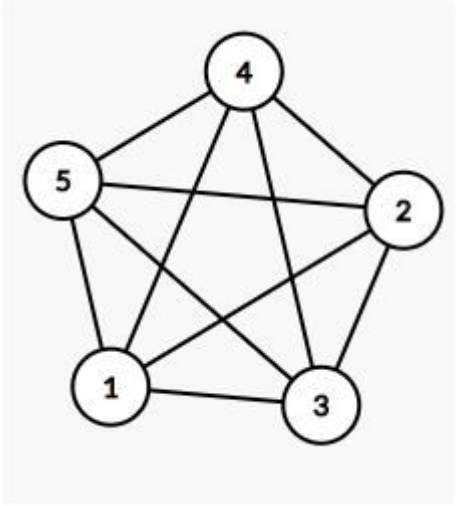

Figure - 1 [Kuratowski's first non-planar graph - K(5)]

K(3,3) - In this type of graph We have two set of 3 vertices each and there is no edge between vertices of same set.
There are edges between every vertex of one set to all the vertices of another set.

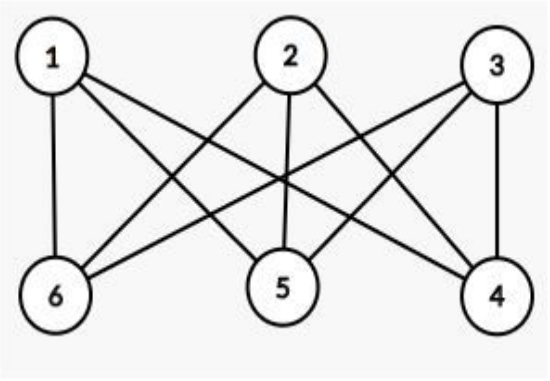

Figure - 2 [Kuratowski's second non-planar graph -K(3,3)]

Example Graphs Containing K(5) or K(3,3)

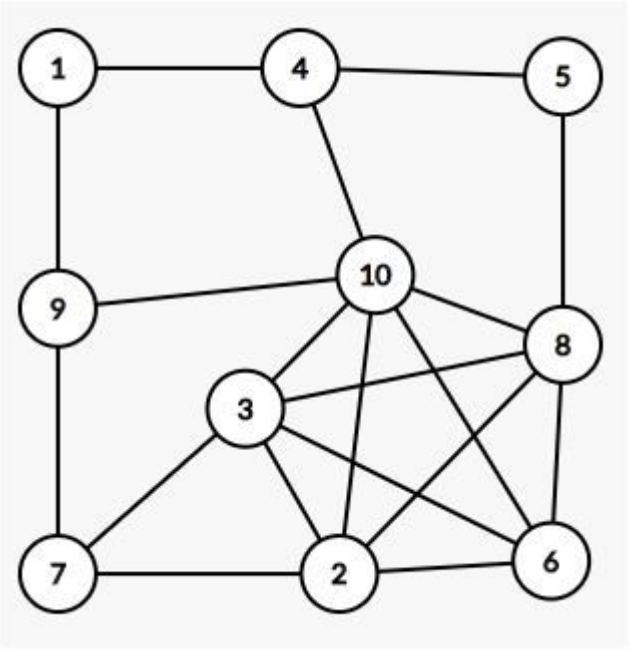

Figure - 3 [Graph of 10 vertices containing K(5)]

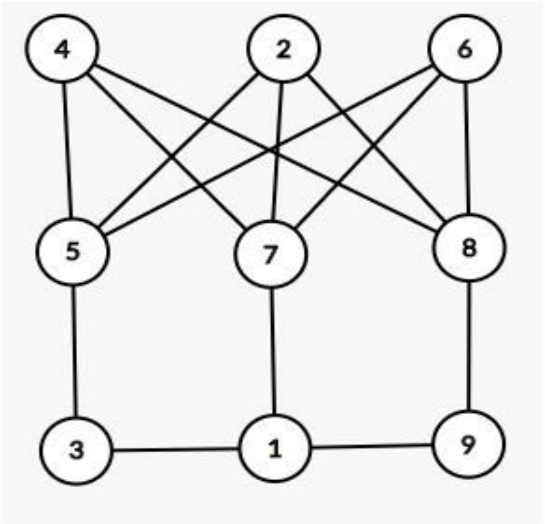

Figure - 4 [Graph of 9 vertices containing K(3,3)

K(5) - Kuratowski's Non Planar Graph of Minimum Number of Vertices
#Vertices = 5
#Edges = 10

K(3,3) - Kuratowski's Non Planar Graph of Minimum Number of Edges
#Vertices = 6
#Edges = 9

Properties of Kuratowski's Non Planar Graphs

1. Both K(5) and K(3,3) are Regular Graphs. [Regular Graphs are the graphs in which all the vertices have same degree]
2. Both are Non Planar.
3. Removal of one edge or a vertex makes each a planar graph.
4. Both are simplest nonplanar graphs of minimum number of vertices and edges.
5. K(5) is the non planar graph with the smallest number of vertices.
6. K(3,3) is the non planar graph with the smallest number of edges.

2. Approach

To Check for K(5)

Idea :

The main idea behind finding the subgraph K5 is that we can pick any set of 5 vertices and check whether the subgraph formed by them is K5 or not. We repeat this procedure until either we have not found the K5 subgraph or we have checked all the set of 5 vertices. In former case we can conclude that the graph has K5 subgraph present in it and hence it is not planar, while in latter case we can clearly say that the graph doesn't contains K5 subgraph.
Although it seems like a complete brute force, but we have adopted few optimizations to make the solution faster.

Algorithm :
Here we are using a nested loop of size 5 to find the K5 subgraph. Loops represents 5 nodes i,j,k,l and m. We construct the subgraph incrementally.
Since the subgraph should be a complete graph so at any step of its construction the completeness property of subgraph should be satisfied.
The final subgraph is represented by nodes i,j,k,l and m. So at any step lets say we have formed subgraph of nodes i,j and k , then these should also satisfy the completeness property , and if they satisfy the property then we pick another node l and form a subgraph of size 4 and if i,j,k and l form a complete subgraph then we proceed for node m and if i ,j,k,l and m forms a complete subgraph then we say that K5 is found.

```
Found = false
For i : 1 to N
  For j : i+1 to N
    If (!edge(i,j)) pick next j
    For k : j+1 to N
      If (!edge(i,k)) pick next k
      If (!edge(j,k)) pick next k
      For l : k+1 to N
        If (!edge(i,l)) pick next l
        If (!edge(j,l)) pick next l
        If (!edge(k,l)) pick next l
        For m : l+1 to N
          If (!edge(i,m)) pick next m
          If (!edge(j,m)) pick next m
          If (!edge(k,m)) pick next m
          If (!edge(l,m)) pick next m
          Found = true
          //K5 is composed of
          //vertices {i,j,k,l,m}
          Terminate()
```

Complexity :
Since nested loop of size 5 are involved so an upper bound on the complexity can be O(N^5) where N is number of nodes.
But in practice the algorithm is much faster than O(N^5) because of the fact that the subgraphs are built incrementally and if at any stage the subgraph built is not complete then we don't proceed further, instead we pick another vertex.

To check for K(3,3)

Idea :
We follow the similar idea as what we did for K5. Here we pick any set of 6 vertices and check whether the subgraph formed by them is K3,3 or not. We repeat this procedure until either we have not found the K3,3 or the search space has been exhausted. In the former case we can conclude that the graph contains K3,3 and in the latter case we say that graph doesn't contains K3,3 because the search space is exhausted.

The vertices are incrementally picked and this results in speedup of the solution.

Algorithm :

Algorithm working is similar to that of K5.
Here we are using nested loop of size 6 , operated by variables i,j,k,l,m and n. vertex set {i,j,k} represent partition A and vertex set {l,m,n} represent partition B.
At every step we are incrementally constructing the subgraph , starting with vertex i, then we add vertex j such that i and j has no edge. Further we add k such that there is no edge between i and k, and j and k. Till this step we will get partition A. Similarly we get partition B incrementally and before adding any vertex in partition B, we check whether it has edges with i,j and k or not, If it doesn't have edge with any of i,j or k we pick another candidate for set B.
Proceeding in this manner we will have two partitions A and B in the end, satisfying the criteria that :
1 : there is no edge between any vertex of partition A
2 : there is no edge between any vertex of partition B
3 : every vertex of partition A is connected by every vertex of partition B
4 : size of both partition is 3

```
Found=false
For i:1 to N
 For j:i+1 to N
  if (!edge(i,j)),pick next j
  For k:j+1 to N
   if (!edge(i,k)),pick next k
   if (!edge(j,k)),pick next k
   For l:k+1 to N
     if(!edge(i,l)),pick next l
     if(!edge(j,l)),pick next l
     if(!edge(k,l)),pick next l
      For m:l+1 to N
        if(!edge(i,m)),pick next m
        if(!edge(j,m)),pick next m
        if(!edge(k,m)),pick next m
        if(edge(l,m)),pick next m
          For n:m+1 to N
           if(!edge(i,n)),pick next n
           if(!edge(j,n)),pick next n
           if(!edge(k,n)),pick next n
           if(edge(l,n)),pick next n
           if(edge(m,n)),pick next n
           Found=true
           //K(3,3) is composed of
            //vertices {{i,j,k},{l,m,n}}
            Terminate()
```

Complexity :
An upper bound on the time complexity can be justified as O(N^6) since 6 nested loops are involved, but in practice it is expected to run faster because subgraphs are built incrementally.

3. Result

We generated 950 graphs randomly consisting of nodes from 6 to 100. 20% to 30% edges were randomly dropped. For every N (N : number of nodes) 10 graphs were generated. So in total we have 10*(100-6+1) = 950 graphs.
Both the algorithms were tested on these 950 graphs. Although theoretically O(N^6) or O(N^5) seems to be of very high complexity and execution time for them is supposed to very high , but most of them executed in less than 1 seconds. Maximum execution time was 4 seconds.
It is difficult to generated a large(random) graph which would guarantee the worst case running time of our algorithm because the algorithm picks nodes (i,j,k,etc)

incrementally and it is less probably that execution control will reach in the innermost loop.

4. Conclusion

We successfully implemented the above mentioned algorithms for K(3,3) and K(5) on randomly generated graphs (20-30% edges dropped randomly) and analysed the execution time. For small graphs we manually checked the results and results very correct. We concluded that execution time is much less as compared to theoretical execution time (O(N^5) or O(N^6)). We also found that in case of randomly generated graphs K(5) has more occurrence than K(3,3).

5. References

1. "*Algorithms for Planar Graphs and Graphs in Metric Spaces*" by Christian Wulff-Nilsen Department of Computer Science, University of Copenhagen, Denmark
2. Graph Theory with Applications to Engineering and Computer Science by N Deo
3. "An Introduction to 1-planarity" by Stephen G. Kobourov1
4. Cs stackexchange Blog on Non Planar graphs K(3,3) & K(5)