IT203: Discrete Math Project
Cops and Robbers

Under guidance of Dr. Bhawana Rudra

Advaith Prasad Curpod
201IT204
Information Technology
NITK Surathkal
Surathkal, India, 575025
advaithprasadcurpod.201it204@nitk.edu.in

Durga Supriya H L
201IT121
Information Technology
NITK Surathkal
Surathkal, India, 575025
durgasupriyahl.201it121@nitk.edu.in

Dhruvil Lakhtaria
201IT119
Information Technology
NITK Surathkal
Surathkal, India, 575025
dhruvillakhtaria.201it119@nitk.edu.in

***Abstract*—A pursuit-evasion problem which involves the analysis of the cop-robber situation on different graphs and developing an algorithm that would prove if a given graph is cop-win or robber-win. A graph is said to be a cop-win graph if the cop can nab the robber in a finite number of moves, regardless of the strategy applied by the robber to escape; and vice-versa for robber-win graph. In this report, we have analysed a simple, bounded & connected graph G(V, E) where one cop chases one robber. We have also assumed that both the cop and robber move one step at a time.**

The fundamental concept used to identify the nature of a graph is "pitfalls". In the report we have explained in detail how a graph can be reduced to an equivalent smaller graph (a subgraph) after removing the pitfalls, without affecting the nature of the graph. If after removing all pitfalls, the graph could be reduced to a single vertex graph, then it is said to be a cop-win graph, else a robber-win. Note that this is a searching problem and not a sweeping one, i.e., at any given point of time, the pursuers and evaders must occupy a node and cannot halt anywhere on an edge.

I. Introduction

We often come across situations where we are expected to search for something in a given environment, be it a treasure, a thief who has escaped from the cops or a lost person. In such situations, it becomes extremely crucial to devise a look-up or search strategy to find what we are looking for, with the least possible manpower and time. Through this project, we wish to solve a pursuit-evasion problem where the pursuers (cops) look for an evader (robber), both of whose movement is restricted to a boundary.

II. Problem Statement

Given a simple, planar, connected, undirected and bounded graph, identify if the graph is a cop-win or robber win.

We shall assume that there is a finite connected graph in which the cop and the robber must move continuously. They make alternative moves.

If it is found that it is a Robber Win Graph, then we find the minimum number of cops required to make it a Cop Win graph.

III. Literature Survey

Before we move ahead with the methodology and implementation, we need to get our fundamental theoretical concepts cleared. Below we have listed certain theorems and lemmas of graph theory which we have used for graph analysis.

Summary of the references

1. Jordon S. Daugherty (2020), The Game of Cops and Robbers on Planar Graphs, Missouri State University.

We basically focused on 3 important and useful theorems from this paper
(1) Pitfall removal lemma - Let H(V, E) G(V, E) be the induced subgraph created by removing a pitfall from G, then c(H) = c(G)
(2) Condition for cop-win - A graph G is cop-win if and only if it is decomposable.
(3) Cop-Number - If G is a planar graph, then c(G) 3.
Limitation in Base Paper: The base paper theorems is limited to planar graphs only. To solve this, we have implemented certain special cases which also apply for non-planar graphs like trees, complete graph etc.

2. John R. Britnell And Mark Wildon (2012), Finding a Princess in a Palace: A Pursuit–Evasion problem, The Electronic Journal of Combinatorics.

This paper talks about Finding a Princess in a Palace problem. Basically even this is a type of pursuit evasion problem.Unlike the related and much studied 'Cops and Robbers Game', the prince has no knowledge of the position of the princess. So, we related the number of days it takes to find the princess in a place to Cop number in our problem.

3. Anthony Bonato, A and Richard. J. Nowakowski.

The Game of Cops of Robbers on Graphs.

This paper majorly discusses the theorems related to Cop and Robber problem and pursuit evasion problems in general. The features of Cop winning graphs are discussed. As mentioned in the methodology, some of the special graphs can be directly declared as Cop win graphs. It also considers the case where a single graph can have multiple number of robbers and the way to find the number of cops to make it Cop win is discussed.

Theorem 1

Let H(V0, E0) G(V, E) be the induced subgraph created by removing a pitfall from G, then c(H) = c(G).[1]

Proof:

Assume c(G) = n and there exists a pitfall on G, then there exists a strategy for n cops to catch the robber, but not for n 1 cops. If the robber does not move to the dominating vertex or the pitfall, then removing the pitfall will not change the cop number. Thus assume the robber moves to the dominating vertex. Since all the vertices adjacent to the pitfall are also adjacent to the dominating vertex, the robber will not need to move to the pitfall. If the robber does move to the pitfall, then the cops will have the opportunity to decrease the distance between them and the robber by moving toward the dominating vertex as if the robber was on it instead. Hence moving to the pitfall will only hinder the robber so they will not move there. Therefore the robber will only move on vertices in H, which implies the strategy to catch the robber on G is also the strategy for H.

Since pitfalls can be removed without changing the cop number, a graph can be decomposed into a smaller graph by removing all the pitfalls.

Theorem 2

A graph G is cop-win if and only if it is decomposable.[1]

If a graph can be decomposed by successively removing pitfalls until there is only a single vertex remaining, then the graph is said to be decomposable.

Proof:

I will first prove the sufficiency condition, if a graph is decomposable then it is a cop-win. Since the graph G is decomposable, pitfalls can be successively removed until only one vertex remains. By Theorem1, c(G) is the same as the cop number of a single vertex, which is trivially one. Thus G is cop-win. Now assume that G is cop-win and I will prove that it is necessary that G is decomposable. Because G is cop-win, after a finite number of moves the cop will catch the robber. Now consider the robber's turn before the cop wins, I claim the robber is in a pitfall. Suppose, for contradiction, that the robber is not in a pitfall. Then there is no vertex that is adjacent to all the neighbors of the robber's vertex. Hence the robber has a safe move away from the cop no matter where the cop is. But this contradicts the fact that on the cop's turn the cop will catch the robber, therefore the robber must be in a pitfall. Let H1 be the induced subgraph of G defined by removing the previous pitfall. By Theorem1, c(H1) = c(G) = 1 and so the cop catches the robber after a finite number of moves. Once again the robber must be in a pitfall on their last turn, and so we can define H2 to be the induced sub graph of H1 with the pitfall removed. Notice that this pattern continues and so, by induction, pitfalls can be successively removed from G until some Hn which is only a single vertex. Thus G is decomposable and the theorem holds.

Theorem 3

Let G and H be graphs. Then G + H is a cop-win graph if and only if G or H is a cop-win graph.[4]

Proof :

Suppose G is a cop-win graph. Then successive removal of pitfalls reduces G to a single vertex, say v. Let G1, G2, ··· , Gk be the sequence of subgraphs of G obtained by successively removing pitfalls. Then, for each i = 1, 2, ··· , k, Gi is the subgraph obtained from Gi1 by removing from it its pitfalls. Here, G0 = G and Gk = v. Note that for each stage i, every pitfall p of Gi1 is a pitfall of Gi1 + H. Thus removal of these pitfalls (with relative to G + H) reduces G + H into the subgraph K = v + H. Clearly, removal of these n vertices of H reduces K to the subgraph v. Therefore, by Theorem 2, G + H is a cop-win graph. Suppose now that G + H is a cop-win graph. Suppose further that G and H are both robber-win graphs. Let the cop choose the vertex v V (G + H). Without loss of generality, assume that v V (G). Then the robber can choose a vertex in G where the cop could not catch him. Since G is a robber-win graph, the cop will not succeed in putting himself on top of the robber if he stays in G. Now, if the cop traverses through an edge to the graph H, then the robber can also move to graph H and stay on a vertex of H where the cop could not catch him. Again, since H is a robber-win graph, the cop will not succeed in putting himself on top of the robber if he stays in H. By staying on the same graph the cop stays, the robber always has a winning strategy on G+H. This implies that G + H is a robber-win graph, contrary to our assumption. Therefore, G or H must be a cop-win graph. The corona G ○ H of two graphs G and H is the graph obtained by taking one copy of G of order n and n copies of H, and then joining the i th vertex of G to every vertex in the i th copy of H. For every v V (G), denote by Hv the copy of H whose vertices are attached one by one to the vertex v. Subsequently, denote by v + Hv the subgraph of the corona G ○ H corresponding to the join v + Hv, v V (G).

Theorem 4

Let G and H be connected graphs. Then G[H] is a cop-win graph if and only if G and H are cop-win graphs.[4]

Proof : Suppose G and H are cop-win graphs. By Theorem 1 , successive removal of pitfalls reduces H to a single vertex, say u. Let H1, H2, ··· , Hk be the sequence of subgraphs of H obtained by successively removing pitfalls, where H0 = H and Hk = u. By repeatedly applying Lemma 2.4, G[H]

can be successively reduced to graphs G[H1], G[H2], ··· , G[Hk]. Since Hk = u, it follows that G[Hk] = G. Since G is a cop-win graph, successive removal of pitfalls reduces it to a graph v. This implies that successive removal of pitfalls reduces G[H] to the graph (v, u). Therefore G[H] is a cop-win graph. Suppose G[H] is a cop-win graph. Suppose further that G or H, say G, is a robber-win graph. Then the robber can stay on a copy of G throughout the game without being caught by the cop. It follows that G[H] is a robber-win graph, contrary to the assumption. The Cartesian product GH of two graphs G and H is the graph with V (GH) = V (G) × V (H) and (u, u)(v, v) E(GH) if and only if either uv E(G) and u = v or u = v and u v E(H).

Theorem 5

Let G and H be connected graphs of orders at least two. Then GH is a robber-win graph.[4]

Proof : It suffices to show that GH has no pitfall. To this end, let (a, b) V (GH) and let (x, y) NG

Then by E(H). Pick c V (G) such that ac E(G). Then (a, b)(c, b) E(GH). Since b y, (x, y)(c, b) / E(GH). It follows that NG [(a, b)] is not contained in NG

Case2. Suppose y = b. Then ax E(G). Pick z V (H) such that yz E(H). Then (a, b)(a, z) E(GH). Since a = x, (x, y)(a, z) / E(GH). It follows that NG [(a, b)] is not contained in NGH[(x, y)].

Since (x, y) is an arbitrary neighbor of (a, b), it follows that (a, b) is not a pitfall of GH. Also, since (a, b) was arbitrarily chosen, GH has no pitfall. Accordingly, GH is a robber-win graph.

Theorem 6

If G is a planar graph, then c(G) 3. [3]

The idea of the proof of Theorem 8.1 is to increase the cop territory; that is, vertices that if the robber moved to he would be caught. Hence, the number of vertices the robber can move to without being caught is eventually is reduced to the empty set, and so the robber is captured. For a fixed graph H, Andreae [9] generalized this result by proving that the cop number of a K5-minor-free graph (or K3,3-minor-free graph) is at most 3 (recall that planar graphs are exactly those which are K5-minor-free and K3,3-minor-free). Andreae [10] also proved that for any graph H the cop number of the class of H-minor-free graphs is bounded above by a constant. Less is known about the cop number of graphs with positive genus. As such, the survey of such graphs is brief. The main conjecture in this area is due to Schroeder. In [129], Schroeder conjectured that if G is a graph of genus g, then c(G) g + 3. Quilliot [123] had shown the following.

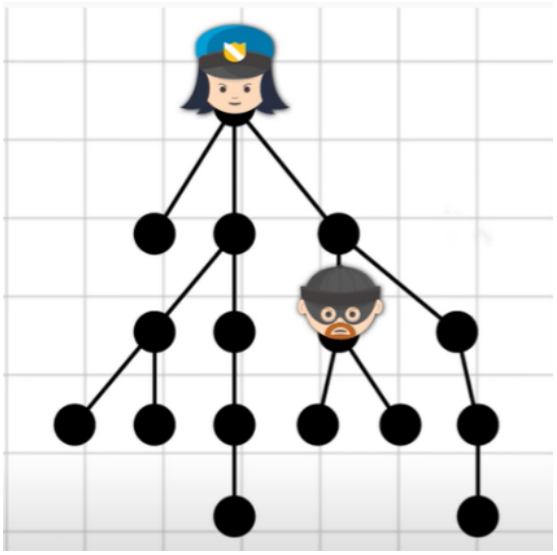
IV. Methodology

We are developing an algorithm to identify the given graph as cop-win or robber-win. We began the implementation for some specific cases/special graphs.We implemented the algorithm for Trees, Linked Lists,Complete Graph & Cyclic graph.

A. Trees

To identify a given graph as cop win/robber win,we don't actually require any algorithm. Every tree is a cop-win graph and a cop can always come up with a strategy to trap the robber. Hence we identify the given input as a tree or not. We need to check for –

- No of edges = No. of vertices - 1
- Any loops/cycles



Listing 1. Java language

```
private static boolean isCyclicUtil
(int v, boolean visited[], int parent,
LinkedList<Integer>adj[])
{
    // Mark the current
    // node as visited
    visited[v] = true;
}

 Integer i;

    // Recur for all the vertices
    // adjacent to this vertex
    Iterator<Integer> it =
        adj[v].iterator();
    while (it.hasNext())
    {
        i = it.next();

        if (!visited[i])
        {
            if (isCyclicUtil(
            i, visited, v,adj))
                return true;
        }
        else if (i != parent)
        return true;
    }
    return false;
}
```

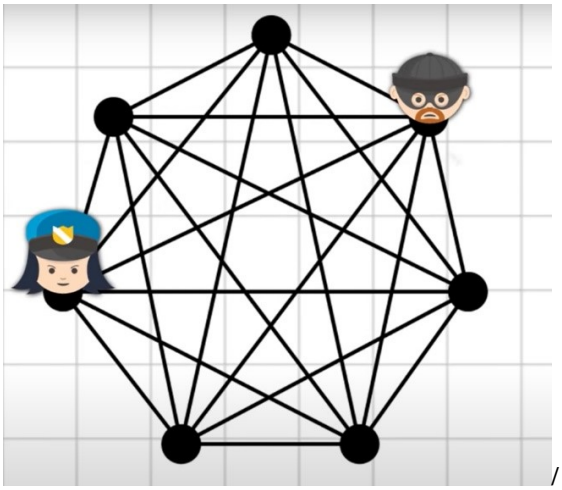
B. Complete Graph

A complete graph is a special graph where a vertex is connected to every other vertex. This case is also similar to the above one and is a cop-win graph. Hence we just have to verify if every node is connected to every other node in the graph.
We have stored the edges from vertex i to vertex j in 2 arrays. We merge the arrays and use a hashmap to count the occurrences of each vertex. If the count of each vertex is = No. of edges - 1, then the graph is a complete graph.

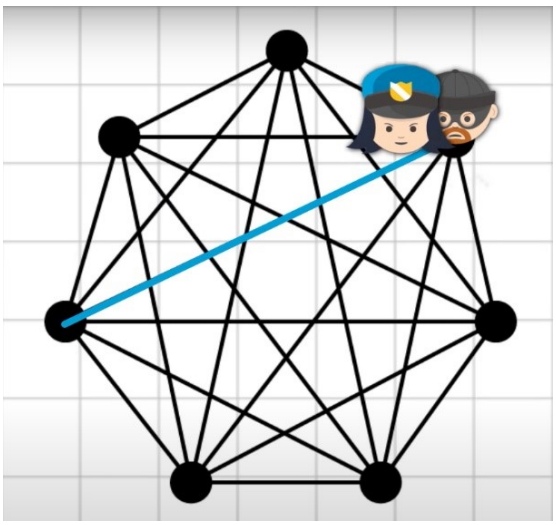
C. Cyclic Project

A graph is said to be cyclic if every node on the graph is connected to exactly 2 other nodes on the graph on either side. Any cyclic graph with more than 3 nodes will always be a robber-win graph as the robber can always find a way to avoid the cop if he stays 2 steps away. We have stored the vertices in 2 arrays, and an edge is represented with the line joining the vertices corresponding to the same index in both the arrays. We then check if every node is connected to exactly 2 different nodes on the graph. If yes, it is a cyclic graph, else not.

Listing 2. Java language



/



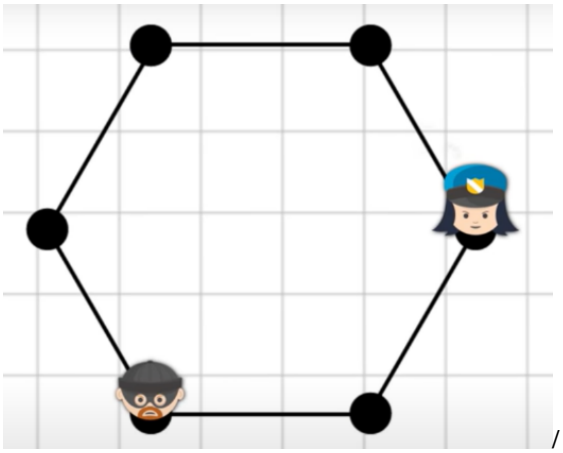
```
public class CyclicGraph {
public static boolean checkCyclicGraph
(int N, int E, int[] U, int[] V) {
    /**for a cyclic graph,
    number of vertices =
    number of edges*/

    if(N != E)
        return false;

    /**if N < 4, then it is cop-win*/
    if(N < 4)
        return false;

    int[] arr = new int[N+E];
    System.arraycopy(U, 0, arr, 0, N);
    System.arraycopy(V, 0, arr, N, E);

    /**store unique elements of
    the array in a HashSet*/
    Set<Integer> set = new HashSet<>();
```



/

```
        for(int i : arr)
            set.add(i);

        /**check if every number
        occurs exactly twice*/
        for(int i : set){
            int count = 0;
            for(int j : arr){
                if(i == j)
                    count++;
            }
            if(count != 2)
                return false;
        }

        /**every vertex must have an
        edge pointing to another
        different vertex; it shouldn't
        be a self loop*/
        for(int i = 0; i < N; i++){
            if(U[i] == V[i])
                return false;
        }
        return true;
    }
}
```

D. Generalized Graph

After checking for these special cases, we go with the general case. In a general graph A dismantling order can be found by a simple greedy algorithm that repeatedly finds and removes any pitfalls. The process succeeds, by reducing the graph to a single vertex, if and only if the graph is cop-win. Therefore, as well as providing an algorithm for finding dismantling orders, this method provides an algorithm for testing whether a given graph is cop-win. One way for this algorithm to find the dominated vertices that it removes is to perform the following steps:

Find all triangles in the graph, and count the number of triangles that each edge participates in. Repeatedly find a vertex v that is an endpoint of an edge participating in a number of triangles equal to the degree of v minus one, delete v, and decrement the triangles per edge of each remaining edge that formed a triangle with v.

Listing 3. Java language

```
public boolean checkCycle() {
    boolean visited[] = new boolean[this.N];
    for(int i = 0;i<this.N;i++)
    visited[i] = false;
    this.length = 0;

    for(int i = 0;i<this.N;i++)
    {
        if(visited[i] == false)
        {
            if(this.dfs(i,visited,-1)
            == true)
            {
                if(this.length >= 4)
                {
                    this.rob = true;
                }
                this.length = 0;
            }
        }
    }
    return this.rob;
}

public boolean dfs
(int i,boolean visited[],int parent) {
    visited[i] = true;
    for(int j : adjList[i])
    {
        if(this.end) this.length = 0;
        if(visited[j] == false)
        {
            this.length+=1;
            if(this.dfs(j,visited,i));
            return true;
        }
        else if(parent != j)
        return true;
    }
    this.end = true;
    return false;
}
```

E. Cop Number

Cop-Number – The upper limit for cop-number of any planar graph can be found out using the chromatic number of a graph. The 4 color Theorem – "The chromatic number of a planar graph is no greater than 4." For example consider the following graph. Let red be assigned to robber and all other colours to cops. So, the number of colours except red will be 2. Here

it is ensured that the robber is surrounded by cops and hence can be caught. Thus, the cop number would be 2.

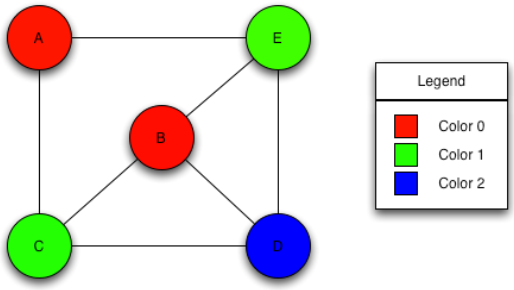


Fig. 1. Cop Number

1) Assign the first color to the first node of the graph.
2) Do the following for the remaining N - 1 node.
3) Consider the currently picked node and assign a color to it with the lowest numbered color. Provided that the color has not been applied on any of the previously colored nodes adjacent to it. If all colors that are previously used appear on the nodes that are adjacent to, assign a new color to it.
4) So, once we find the colours of each vertex findChromaticNo() finds the number of different colours used.
5) Finally, the cop number will be 1 less than chromatic number as mentioned earlier.

V. RESULTS

```
Cops & Robbers
Given a graph, this program will determine if the graph is cop win or robber win

Enter number of vertices: 9
Enter number of edges: 11

First vertex number: 1
Connected vertex number: 2

First vertex number: 1
Connected vertex number: 3

First vertex number: 3
Connected vertex number: 4

First vertex number: 3
Connected vertex number: 2

First vertex number: 4
Connected vertex number: 5

First vertex number: 4
Connected vertex number: 6

First vertex number: 5
Connected vertex number: 6

First vertex number: 7
Connected vertex number: 8

First vertex number: 7
Connected vertex number: 9

First vertex number: 7
Connected vertex number: 6

First vertex number: 8
Connected vertex number: 9

COP-WIN graph
Cop-number = 1
```

Fig. 2. Output 1

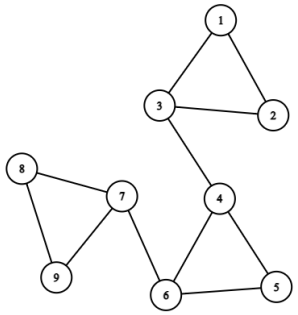


Fig. 3. Result 1

```
Cops & Robbers
Given a graph, this program will determine if the graph is cop win or robber win

Enter number of vertices: 9
Enter number of edges: 10

First vertex number: 1
Connected vertex number: 2

First vertex number: 2
Connected vertex number: 3

First vertex number: 4
Connected vertex number: 7

First vertex number: 3
Connected vertex number: 5

First vertex number: 5
Connected vertex number: 6

First vertex number: 6
Connected vertex number: 1

First vertex number: 4
Connected vertex number: 8

First vertex number: 8
Connected vertex number: 7

First vertex number: 1
Connected vertex number: 9

First vertex number: 4
Connected vertex number: 9

ROBBER-WIN graph
The cop number of the graph is at-most: 2
```

Fig. 4. Output 2

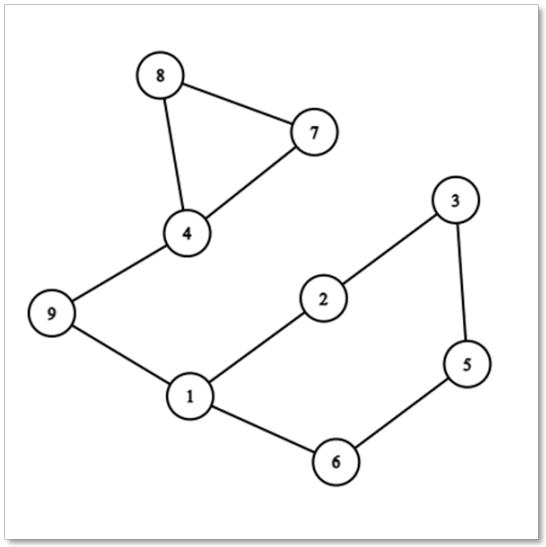


Fig. 5. Result 2

The GitHub link of the project: **GitHub**

REFERENCES

[1] Jordon S. Daugherty (2020), *The Game of Cops and Robbers on Planar Graphs* , Missouri State University.

[2] John R. Britnell AND Mark Wildon (2012), *Finding a Princess in a Palace: A Pursuit–Evasion problem*, The Electronic Journal of Combinatorics.

[3] Anthony Bonato, A and Richard. J. Nowakowski. *The Game of Cops of Robbers on Graphs.*

[4] Sergio R. Canoy Jr. and Mhelmar A. Labendia, *Cop-Win Graphs and Robber-Win* Graphs in Mindanao State University-Iligan Institute Technology.

[5] M. Aigner and M. Fromme (1982), *A Game of Cops and Robbers*, Math Institute, Freie Universitat Berlin.

[6] Anthony Bonato (2012), *What is Cop Number?*, American Mathematical Society.

[7] PBS Infinite Series: *"The Cops and Robbers Theorem"*

[8] PBS Infinite Series: *"How Many Cops to Catch a Robber?"*