Int. J. Sci. R. Tech., 2026 3(8)
A Multidisciplinary peer-reviewed Journal
www.ijsrtjournal.com [ISSN: 2394-7063]

Original Article

An Algorithmic Exploration Of Dominating Set And Independent Dominating Set

A. Venkatesan*

PG and Research Department of Mathematics, St. Joseph's College of Arts & Science (Autonomous), Cuddalore-01

ABSTRACT

Objectives: This paper presents an algorithmic investigation into the identification and analysis of dominating sets and independent dominating sets in simple graphs. **Methods:** A dominating set and an independent dominating set are found in the following definitions (i) A set S ⊆ V is a dominating set of a graph G if every vertex v ∈ V is either an element of S or is adjacent to an element of S. (ii) An independent dominating set of G is a set that satisfies both dominating and independent in G. **Findings:** The possibilities of a dominating set and an independent dominating set are found in Python programming language. **Novelty:** The Python programming language to implement algorithms that precisely identify dominating sets and independent dominating sets in simple graphs.

Keywords: Domination – Independent Domination- python program code.

INTRODUCTION

The development of algorithms provides systematic and efficient solutions to a wide range of problems across different domains. Algorithms are a fundamental concept in computer science and critical for advancing technology and solving complex challenges. Through the paper, the graph G = (V, E) means a simple, finite, undirected, and connected graph. A set S ⊆ V is a dominating set of a graph G if every vertex $v \in S$ either an element of S or is adjacent to an element of S [4]. In 1958[1], Berge defined the concept of the domination number of a graph, calling this as "coefficient of External Stability". In 1962[3], Ore used the name "dominating set" and "domination number" for the same concept. In 1977[2] Cockayne and Hedetniemi made an interesting and extensive survey of the results known at that time about dominating sets in graphs. They have used the notation γ (G) for the domination number of a graph. The domination number of G, denoted by γ (G), is the minimum size of a dominating set. A set is independent (or stable) if no two vertices are adjacent. An independent dominating set of G is a set that is both dominating and independent in G. The independent domination number and the notation i(G) were introduced by Cockayne and Hedetniemi in [5,6] is the minimum size of an independent dominating set. The independence number of G, denoted α(G), is the maximum size of an independent set in G. It follows immediately that γ (G) ≤ i(G) ≤ α(G). A dominating set of G of size γ (G)is called a γ -set, while an independent dominating set of G of size i(G)is called an i-set [7].

2. RESULT AND DISCUSSION

2.1. An algorithm on Dominating set of any simple graph using Python language:

```
class MyGraph:
 def __init__(self, num_vertices):
    self.V = num_vertices
    self.adj = []
    for i in range(num_vertices):
       self. adj.append([])
 def add_my_edge(self, v1, v2):
    self.adj[v1].append(v2)
    self.adj[v2].append(v1)
```

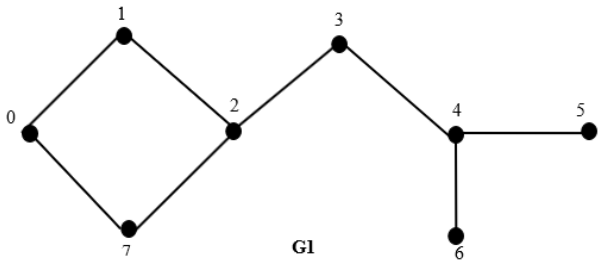
Relevant conflicts of interest/financial disclosures: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

```
    def is_my_dominating(self, dom_set):
        for i in range(self.V):
            if i not in dom_set:
                flag = False
                for j in self.adj[i]:
                    if j in dom_set:
                        flag = True
                        break
                if not flag:
                    return False
        for i in dom_set:
            for j in dom_set:
                if i != j and j in self.adj[i]:
                    return False
        return True
    def find_my_dominating_sets(self):
        all_sets = []
        for i in range(1 << self.V):
            dom_set = []
            for j in range(self.V):
                if i & (1 << j):
                    dom_set.append(j)
            if self.is_my_dominating(dom_set):
                all_sets.append(dom_set)
        return all_sets
g = MyGraph(8)
g.add_my_edge(0, 1)
g.add_my_edge(0, 7)
g.add_my_edge(1, 2)
g.add_my_edge(2, 7)
g.add_my_edge(2, 3)
g.add_my_edge(3, 4)
g.add_my_edge(4, 5)
g.add_my_edge(4, 6)
dom_sets = g.find_my_dominating_sets()
print(dom_sets)
```

Output:

[[0,2,4], [0,2,5,6], [0,3,5,6], [1,4,7], [1,3,5,6,7]]

Manual output:



The possible dominating sets of G1 are: {[0,2,4], [0,2,5,6], [0,3,5,6], [1,4,7], [1,3,5,6,7]}

2.2 An algorithm on Independent Dominating set using Python language

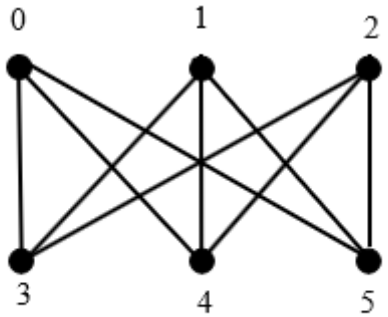
```
class MyGraph:
    def __init__(self, num_nodes):
        self.num_nodes = num_nodes
        self.adj_list = []
        for node in range(num_nodes):
            self.adj_list.append([])
    def add_my_edge(self, node1, node2):
        self.adj_list[node1].append(node2)
        self.adj_list[node2].append(node1)
    def is_my_independent_set(self, my_set):
        for node1 in my_set:
            for node2 in my_set:
```

```
            if node1 != node2 and node2 in self.adj_list[node1]:
                return False
        return True
    def is_my_dominating_set(self, my_set):
        covered = my_set.copy()
        for node1 in my_set:
            for node2 in self.adj_list[node1]:
                covered.append(node2)
        covered_set = set(covered)
        return len(covered_set) == self.num_nodes
    def find_my_independent_dominating_sets(self):
        all_sets = []
        for i in range(1 << self.num_nodes):
            my_set = []
            for j in range(self.num_nodes):
                if i & (1 << j):
                    my_set.append(j)
            if self.is_my_independent_set(my_set) and self.is_my_dominating_set(my_set):
                all_sets.append(my_set)
        return all_sets
g = MyGraph(6)
g.add_my_edge(0, 3)
g.add_my_edge(0, 4)
g.add_my_edge(0, 5)
g.add_my_edge(1, 3)
g.add_my_edge(1, 4)
g.add_my_edge(1, 5)
g.add_my_edge(2, 3)
g.add_my_edge(2, 4)
g.add_my_edge(2, 5)
ids = g.find_my_independent_dominating_sets()
print(ids)
```

Output:

[[0,1,2], [3,4,5]]

Manual output:



G2

The possible independent dominating sets of G2 are: {[0,1,2], [3,4,5]}

CONCLUSION

This paper presents an algorithm developed to determine the possibilities of dominating and independent dominating sets for simple graphs, implemented using Python programming.

REFERENCES

1. Berge, C. (1962). The theory of graphs and its applications. Methuen.
2. Cockayne, E. J., & Hedetniemi, S. T. (1977). Towards a theory of domination in graphs. Networks, 7, 247–261.
3. Ore, O. (1962). Theory of graphs. American Mathematical Society.
4. Haynes, T. W., Hedetniemi, S. T., & Henning, M. A. (Eds.). (2009). Topics in domination in graphs. Springer.
5. Haynes, T. W., Hedetniemi, S. T., & Slater, P. J. (1998). Domination in graphs: Advanced topics. Marcel Dekker.

6. Haynes, T. W., Hedetniemi, S. T., & Slater, P. J. (1998). Fundamentals of domination in graphs. Marcel Dekker.
7. Goddard, W., & Henning, M. A. Independent domination in graphs: A survey and recent results. Discrete Mathematics, 313(7) (2013), 839–854.

HOW TO CITE: A. Venkatesan*, An Algorithmic Exploration Of Dominating Set And Independent Dominating Set, Int. J. Sci. R. Tech., 2026, 3 (8), 690-693. https://doi.org/10.5281/zenodo.21990050