Mobile & Pervasive Computing Project

Apostolopoulou Ioanna

February, 2024

1 Introduction

A connected dominating set (CDS) is a subset of nodes in a graph such that every node in the graph is either in the CDS or adjacent to a node in the CDS. In wireless networks, a CDS is used as a virtual backbone to support various network functions such as multi-hop communication and area monitoring. By using a CDS, energy consumption and interference can be reduced. The presented algorithm improves on previous solutions for finding an m-fold CDS in a general graph by achieving a better approximation ratio.

2 Connected Dominating Set

The "Five Queens" problem can be said to be the origin of the study of the dominating sets in graphs. The problem of determining the minimum number of queens that can be placed on a chess board, so that all the squares are either attacked by a queen or are occupied by a queen is called the five queen problem or the dominating queen problem. It was shown in 1850's, that five is the minimum number of queens that can dominate all of the squares of a chess board. The dominating queen problem can be stated in general as the domination of vertices of a graph.

A set $D \subseteq V$ of vertices in a graph G = (V, E) is called a dominating set if every vertex $v \in V$ is either in the set D, or is adjacent to a vertex in D. The minimum of the cardinalities of the dominating sets is the domination number of the graph G, denoted by $\gamma(G)$.

The problem of finding dominating sets in a graph G is applied in a variety of situations. The concept of domination is mainly used in network problems like, computer communication networks, in which a computer network is modeled by a graph G = (V, E), for which the vertices represent the computers and the edges represent direct communication links between pairs of computers. Each processor passes information to other processors connected to it. Thus the information is collected from all the processors. This is done by passing the information from each processor to one of the small set of collecting processors. The collecting processors form the dominating set, and the problem is to find a small set of processors which are connected to all other processors.

Another application worth mentioning is the ad hoc networks. Ad hoc networks are communication systems with no fixed infrastructure. These networks are used in applications such as mobile commerce, search and rescue, and military battlefields. In these networks, the information is passed between hosts in the network. The information is collected at selected hosts in the network called "virtual backbone" of the network. The problem of finding a minimum size backbone in ad hoc networks can be reduced to the problem of finding a minimum connected dominating set in a connected graph G.

3 Approximation Algorithm

The potential function composed of three parts (p, q, m) and the categorization of nodes into four different colors based on their state in the subgraph are key components of the greedy algorithm presented in the paper for computing an m-fold CDS in a general graph. The potential function is defined as f(C) = p(C) + q(C) + m(C), where p(C) is the number of components of the subgraph induced by C, q(C) is the number of components of the subgraph induced by the edges with at least one endpoint in C, and m(C) is a function that assigns a value to each node in V based on its distance from C. The categorization of nodes into four different colors (black, gray, red, and white) is based on their state in the subgraph induced by C. The algorithm uses these concepts to iteratively select nodes to add to C until every node in the graph is covered by at least m nodes in C. The use of the potential function and node categorization allows the algorithm to achieve an improved approximation ratio compared to previous solutions for finding an m-fold CDS in a general graph. The greedy Algorithm used is shown below:

Algorithm 1 Greedy Algorithm for finding an m-fold CDS in a general graph.

Require: A connected graph $G = (V, E)$ and an integer m.
Ensure: A $(1, m)$-CDS of G.
  $C \leftarrow \emptyset$
  while $\exists$ a node $x \in V \setminus C$ such that $-\Delta_x f(C) > 0$ **do**
    Find x that minimizes $-\Delta_x f(C)$.
    $C \leftarrow C \cup \{x\}$
  end while
Output $C_G \leftarrow C = 0$

4 Experimental Results

4.1 Fixed Graphs

4.1.1 Experiment 1

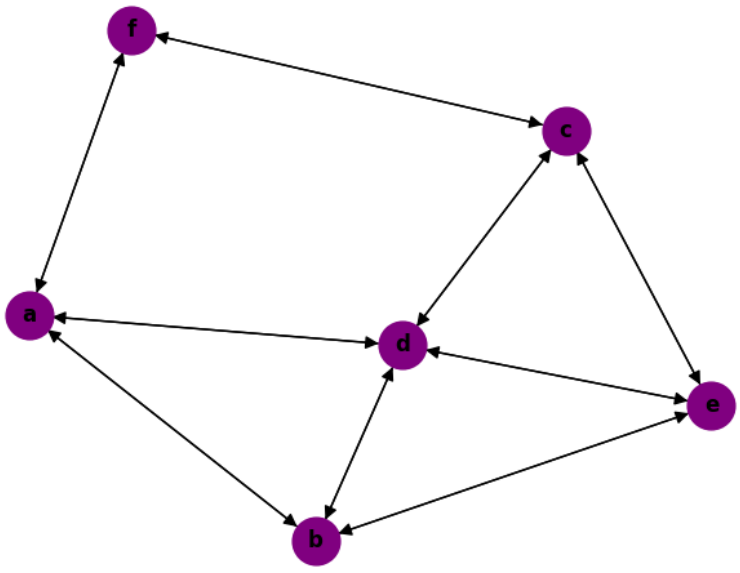

Figure 1: Fixed Graph.

For m = 2:

```
Initial set C: ['b', 'e']
Result of find_N: {'a': ['b'], 'b': [], 'f': [], 'd': ['b', 'e'], 'e': [], 'c': ['e']}
black_nodes: ['b', 'e']
gray_nodes: ['d']
red_nodes: ['a', 'c']
white_nodes: ['f']
```

4.1.2 Experiment 2

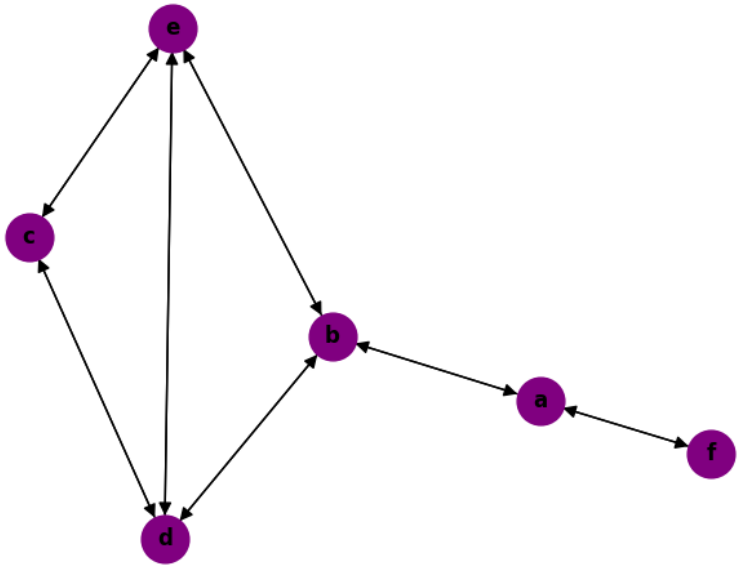

Figure 2: Fixed Graph.

For m = 1:

```
Initial set C: ['a', 'b', 'e']
Result of find_N: {'a': [], 'b': [], 'f': ['a'], 'd': ['b', 'e', 'a'], 'e': [], 'c': ['e']}
black_nodes: ['a', 'b', 'e']
gray_nodes: ['f', 'd', 'c']
red_nodes: []
white_nodes: []
```

For m = 2:

```
Initial set C: ['a', 'b', 'c', 'e', 'f']
Result of find_N: {'a': [], 'b': [], 'f': [], 'd': ['b', 'c', 'e', 'a'], 'e': [], 'c': []}
black_nodes: ['a', 'b', 'f', 'e', 'c']
gray_nodes: ['d']
red_nodes: []
white_nodes: []
```

For m = 2:

```
Initial set C: ['b', 'e']
Result of find_N: {'a': ['b'], 'b': [], 'f': [], 'd': ['b', 'e'], 'e': [], 'c': ['e']}
```

```
black_nodes: ['b', 'e']
gray_nodes: ['d']
red_nodes: ['a', 'c']
white_nodes: ['f']
```

4.1.3 Experiment 3

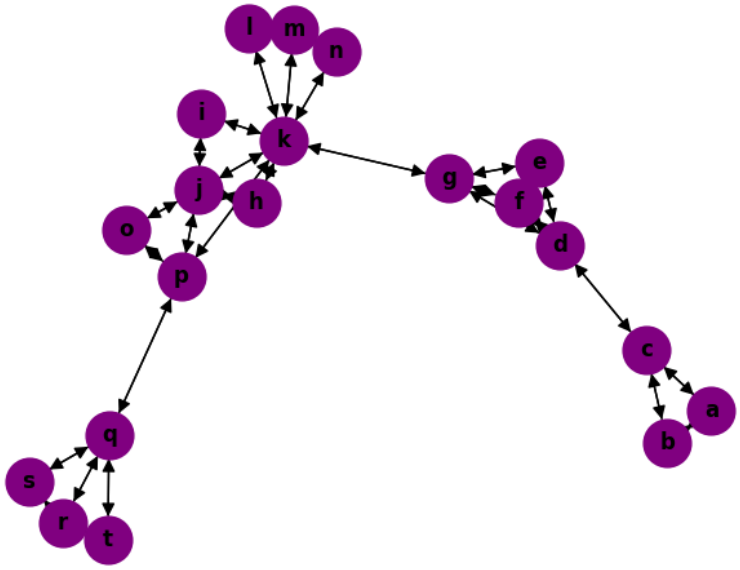

Figure 3: Fixed Graph.

For m = 1:

```
Initial set C: ['c', 'd', 'g', 'k', 'q', 'p']
Result of find_N: {'a': ['c'], 'b': ['c'], 'c': [], 'd': [],
'e': ['d', 'g'], 'f': ['d', 'g'], 'g': [], 'k': [], 'l': ['k'], 'm': ['k'],
'n': ['k'], 'h': ['k'], 'i': ['k'], 'j': ['k', 'p'], 'p': [], 'o': ['p'],
'q': [], 's': ['q'], 'r': ['q'], 't': ['q']}
black_nodes: ['c', 'd', 'g', 'k', 'p', 'q']
gray_nodes: ['a', 'b', 'e', 'f', 'l', 'm', 'n', 'h', 'i', 'j', 'o', 's', 'r', 't']
red_nodes: []
white_nodes: []
```

For m = 2:

```
Initial set C: ['a', 'c', 'd', 'g', 'k', 'j', 'm', 'q', 'p', 'r']
Result of find_N: {'a': [], 'b': ['a', 'c'], 'c': [], 'd': [],
'e': ['d', 'g'], 'f': ['d', 'g'], 'g': [], 'k': [], 'l': ['k', 'm'],
'm': [], 'n': ['k', 'm'], 'h': ['k', 'j'], 'i': ['k', 'j'], 'j': [],
'p': [], 'o': ['j', 'p'], 'q': [], 's': ['q', 'r'], 'r': [],
't': ['r', 'q']}
black_nodes: ['a', 'c', 'd', 'g', 'k', 'm', 'j', 'p', 'q', 'r']
gray_nodes: ['b', 'e', 'f', 'l', 'n', 'h', 'i', 'o', 's', 't']
red_nodes: []
white_nodes: []
```

For m = 3:

```
Initial set C: ['k', 'd', 'j', 'q', 'a', 'e', 'm', 'r', 'c', 'g', 'o', 'b',
'i', 'h', 'l', 'n', 'p', 's', 't']
Result of find_N: {'a': [], 'b': [], 'c': [], 'd': [], 'e': [], 'f': ['e',
'd', 'g'], 'g': [], 'k': [], 'l': [], 'm': [], 'n': [], 'h': [], 'i': [],
'j': [], 'p': [], 'o': [], 'q': [], 's': [], 'r': [], 't': []}
black_nodes: ['a', 'b', 'c', 'd', 'e', 'g', 'k', 'l', 'm', 'n', 'h', 'i',
'j', 'p', 'o', 'q', 's', 'r', 't']
gray_nodes: ['f']
red_nodes: []
white_nodes: []
```

4.2 Randomized Garaphs

By using the doctoral thesis of Dimetrios Papakostas, and the Networkx Python3 Library,in order to visualize the data collected from the application, I created the following Randomized Fully Connected Graph.

4.2.1 Experiment 1

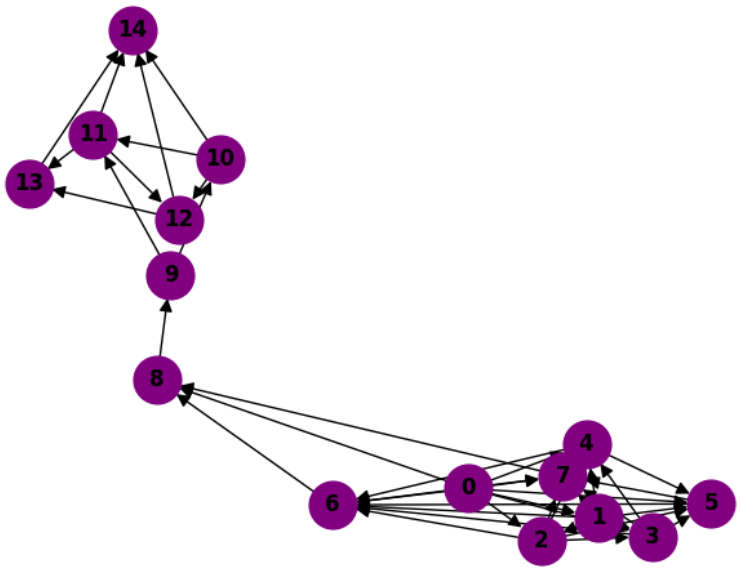

Figure 4: Randomized Graph.

For m = 1:

```
Initial set C: ['14', '0', '10', '4', '9']
Result of find_N: {'0': [], '1': ['4'], '2': ['4'], '3': ['4'],
'4': [], '5': [], '6': [], '7': [], '8': ['9'], '9': [], '10': [],
'11': ['14'], '12': ['14'], '14': [], '13': ['14']}
black_nodes: ['0', '4', '9', '10', '14']
gray_nodes: ['1', '2', '3', '8', '11', '12', '13']
red_nodes: []
white_nodes: ['5', '6', '7']
```

For m = 2:

```
Initial set C: ['14', '0', '10', '4', '9']
Result of find_N: {'0': [], '1': ['4'], '2': ['4'],
'3': ['4'], '4': [], '5': [], '6': [], '7': [],
'8': ['9'], '9': [], '10': [], '11': ['14'],
'12': ['14'], '14': [], '13': ['14']}
black_nodes: ['0', '4', '9', '10', '14']
gray_nodes: []
red_nodes: ['1', '2', '3', '8', '11', '12', '13']
white_nodes: ['5', '6', '7']
```

For m = 3:

```
Initial set C: ['14', '0', '10', '4', '9']
Result of find_N: {'0': [], '1': ['4'], '2': ['4'],
'3': ['4'], '4': [], '5': [], '6': [], '7': [],
'8': ['9'], '9': [], '10': [], '11': ['14'],
'12': ['14'], '14': [], '13': ['14']}
black_nodes: ['0', '4', '9', '10', '14']
gray_nodes: []
red_nodes: ['1', '2', '3', '8', '11', '12', '13']
white_nodes: ['5', '6', '7']
```

4.2.2 Experiment 2

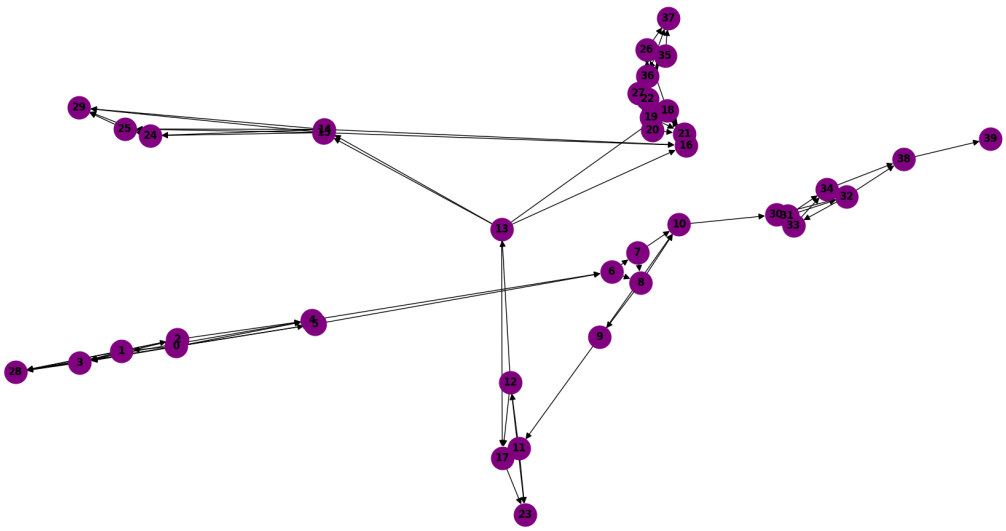

Figure 5: Randomized Graph.

For m = 1:

```
Initial set C: ['2', '17', '9', '4', '23']
Result of find_N: {'0': ['2', '4'], '1': ['2'], '2': [], '3': [],
'4': [], '5': [], '28': [], '6': [], '7': [], '8': ['9'],
'10': [], '9': [], '11': ['17', '23'], '30': [],
'12': ['17', '23'], '17': [], '23': [], '13': ['17'],
'14': [], '15': [], '16': [], '18': [], '24': [], '25': [],
'29': [], '19': [], '21': [], '26': [], '20': [], '22': [],
'27': [], '36': [], '37': [], '35': [], '31': [], '32': [],
'33': [], '34': [], '38': [], '39': []}
black_nodes: ['2', '4', '9', '17', '23']
gray_nodes: ['0', '1', '8', '11', '12', '13']
red_nodes: []
white_nodes: ['3', '5', '28', '6', '7', '10', '30', '14', '15',
'16', '18', '24', '25', '29', '19', '21', '26', '20', '22', '27',
'36', '37', '35', '31', '32', '33', '34', '38', '39']
```

For m = 2:

```
Initial set C: ['2', '17', '9', '4', '23']
Result of find_N: {'0': ['2', '4'], '1': ['2'], '2': [],
'3': [], '4': [], '5': [], '28': [], '6': [], '7': [],
'8': ['9'], '10': [], '9': [], '11': ['17', '23'], '30': [],
'12': ['17', '23'], '17': [], '23': [], '13': ['17'], '14': [],
'15': [], '16': [], '18': [], '24': [], '25': [], '29': [],
'19': [], '21': [], '26': [], '20': [], '22': [], '27': [],
'36': [], '37': [], '35': [], '31': [], '32': [], '33': [],
'34': [], '38': [], '39': []}
black_nodes: ['2', '4', '9', '17', '23']
gray_nodes: ['0', '11', '12']
red_nodes: ['1', '8', '13']
white_nodes: ['3', '5', '28', '6', '7', '10', '30', '14', '15',
'16', '18', '24', '25', '29', '19', '21', '26', '20', '22', '27',
'36', '37', '35', '31', '32', '33', '34', '38', '39']
```

For m = 3:

```
Initial set C: ['2', '17', '9', '4', '23']
Result of find_N: {'0': ['2', '4'], '1': ['2'], '2': [],
'3': [], '4': [], '5': [], '28': [], '6': [], '7': [],
'8': ['9'], '10': [], '9': [], '11': ['17', '23'], '30': [],
'12': ['17', '23'], '17': [], '23': [], '13': ['17'], '14': [],
'15': [], '16': [], '18': [], '24': [], '25': [], '29': [],
'19': [], '21': [], '26': [], '20': [], '22': [], '27': [],
'36': [], '37': [], '35': [], '31': [], '32': [], '33': [],
'34': [], '38': [], '39': []}
black_nodes: ['2', '4', '9', '17', '23']
gray_nodes: []
red_nodes: ['1', '8', '13', '0', '11', '12']
white_nodes: ['3', '5', '28', '6', '7', '10', '30', '14', '15',
'16', '18', '24', '25', '29', '19', '21', '26', '20', '22', '27',
'36', '37', '35', '31', '32', '33', '34', '38', '39']
```

4.2.3 Experiment 3

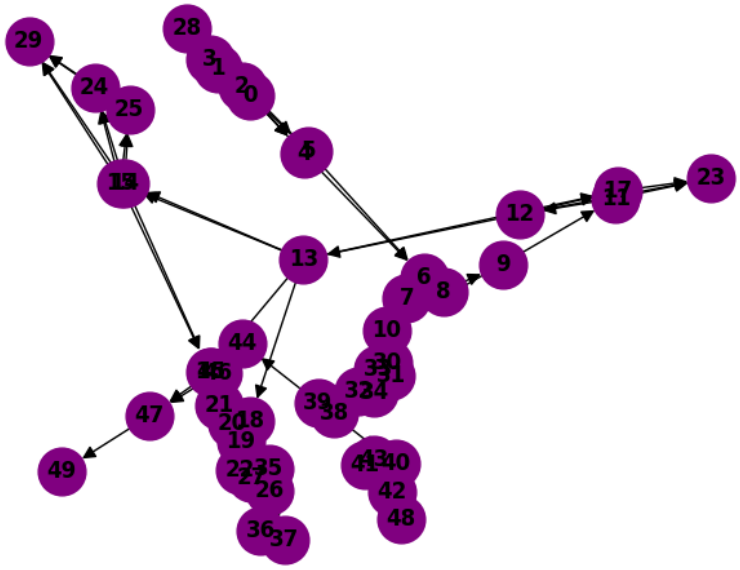

Figure 6: Randomized Graph.

For m = 1:

```
Initial set C: ['17', '38', '11', '6', '9']
Result of find_N: {'0': [], '1': [], '2': [], '3': [], '4': ['6'],
'5': ['6'], '28': [], '6': [], '7': [], '8': ['9'], '10': [],
'9': [], '11': [], '30': [], '12': ['17'], '17': [], '23': [],
'13': ['17'], '14': [], '15': [], '16': [], '18': [], '24': [],
'25': [], '29': [], '19': [], '21': [], '26': [], '20': [], '22': [],
'27': [], '36': [], '37': [], '35': [], '31': [], '32': ['38'],
'33': [], '34': ['38'], '38': [], '39': [], '40': [], '44': [],
'41': [], '42': [], '43': [], '48': [], '45': [], '46': [],
'47': [], '49': []}
black_nodes: ['6', '9', '11', '17', '38']
gray_nodes: ['4', '5', '8', '12', '13', '32', '34']
red_nodes: []
white_nodes: ['0', '1', '2', '3', '28', '7', '10', '30', '23', '14',
'15', '16', '18', '24', '25', '29', '19', '21', '26', '20', '22',
'27', '36', '37', '35', '31', '33', '39', '40', '44', '41', '42',
'43', '48', '45', '46', '47', '49']
```

For m = 2:

```
Initial set C: ['17', '38', '11', '6', '9']
Result of find_N: {'0': [], '1': [], '2': [], '3': [],
'4': ['6'], '5': ['6'], '28': [], '6': [], '7': [],
'8': ['9'], '10': [], '9': [], '11': [], '30': [],
'12': ['17'], '17': [], '23': [], '13': ['17'],
'14': [], '15': [], '16': [], '18': [], '24': [],
'25': [], '29': [], '19': [], '21': [], '26': [],
'20': [], '22': [], '27': [], '36': [], '37': [],
'35': [], '31': [], '32': ['38'], '33': [], '34': ['38'],
'38': [], '39': [], '40': [], '44': [], '41': [], '42': [],
'43': [], '48': [], '45': [], '46': [], '47': [], '49': []}
```

```
black_nodes: ['6', '9', '11', '17', '38']
gray_nodes: []
red_nodes: ['4', '5', '8', '12', '13', '32', '34']
white_nodes: ['0', '1', '2', '3', '28', '7', '10', '30', '23',
'14', '15', '16', '18', '24', '25', '29', '19', '21', '26', '20',
'22', '27', '36', '37', '35', '31', '33', '39', '40', '44', '41',
'42', '43', '48', '45', '46', '47', '49']
```

For m = 3:

```
Initial set C: ['17', '38', '11', '6', '9']
Result of find_N: {'0': [], '1': [], '2': [], '3': [], '4': ['6'],
'5': ['6'], '28': [], '6': [], '7': [], '8': ['9'], '10': [], '9': [],
'11': [], '30': [], '12': ['17'], '17': [], '23': [], '13': ['17'],
'14': [], '15': [], '16': [], '18': [], '24': [], '25': [], '29': [],
'19': [], '21': [], '26': [], '20': [], '22': [], '27': [], '36': [],
'37': [], '35': [], '31': [], '32': ['38'], '33': [], '34': ['38'],
'38': [], '39': [], '40': [], '44': [], '41': [], '42': [], '43': [],
'48': [], '45': [], '46': [], '47': [], '49': []}
black_nodes: ['6', '9', '11', '17', '38']
gray_nodes: []
red_nodes: ['4', '5', '8', '12', '13', '32', '34']
white_nodes: ['0', '1', '2', '3', '28', '7', '10', '30', '23', '14',
'15', '16', '18', '24', '25', '29', '19', '21', '26', '20', '22', '27',
'36', '37', '35', '31', '33', '39', '40', '44', '41', '42', '43', '48',
'45', '46', '47', '49']
```