CS301 - Algorithms

2022-2023 Summer

Project Report

Group 10

Zeynep Özgür Gün

1. Problem Description

In the Clique Problem, it is tried to find a subset W of k vertices, in which each vertex shares a common edge with the rest of the vertices in a graph G (which has a number of vertices larger than or equal to k) essentially. This subset is called a "clique". On the other hand, the optimized version of Clique Problem searches for the largest subset that forms a clique. For a real-life application example, it can represent a situation of finding a group of people that all know each other. Formally, the question that the problem handles can be describes as below:

General Version: *Does G contain a k-clique, i.e. a subset W of the nodes V such that W has size k and for each distinct pair of nodes u, v in W, {u,v} is an edge of G?*
Optimization Version: *Given an n-node undirected graph G(V,E) with node set V and edge set E; what is the largest cardinality subset W of the nodes V such that for each distinct pair of nodes u, v in W, {u,v} is an edge of G? In other words, what is the largest clique in G?*

The formal definition of the problem can be described as:
$G = (V, E)$ is an arbitrary undirected graph, where $V = \{1, 2, \ldots, n\}$ is the vertex set of G and $E \subseteq V \times V$ is the edge set of G. A graph $G = (V, E)$ is complete if all its vertices are pairwise adjacent, i.e. $\forall i, j \in V$ with $i \neq j$, we have $(i, j) \in E$. A clique C is a subset of V such that G(C) is complete. The clique number of G, denoted by $\omega(G)$ is the size of the maximum clique. The maximum clique problem asks for cliques of maximum cardinality (the cardinality of set S, i.e., the number of its elements which will be denoted by $|S|$) [1]:

$$\omega(G) = \max \{|S|: S \text{ is a clique in } G\} \text{ (Akbari, 2013)}$$

This project will handle the optimized version of the problem. In terms of hardness, it is an NP-hard problem. In order to show a problem is in NP-hard, another NP-hard or NP-complete (since all NP-complete problems are in NP-hard) problem can be reduced to it as a proof to show that the former is as hard as the reduced problem, making it NP-hard

as well. The SAT problem can be used for that; the SAT problem asks whether there is a way to arrange a Boolean array such that it would evaluate to "True" as a result.

Theorem: SAT problem (which is NP-complete) can be reduced to Clique problem to show that Clique Problem is in NP-hard.
Karp (1972), demonstrates the reduction in his paper, which can be shown as a proof of Clique problem being in NP-hard.

Reference:

- Richard M. Karp, *Reducibility among Combinatorial Problems*, In *Proceedings of a symposium on the Complexity of Computer Computations*, 1972, pp.85–103. doi: https://cgi.di.uoa.gr/~sgk/teaching/grad/handouts/karp.pdf

2. Algorithm Description

a. Brute Force Algorithm

Source: https://www.geeksforgeeks.org/maximal-clique-problem-recursive-solution/

The algorithm below uses a brute force approach to find the maximum clique, by iterating over each vertex one by one in the outer loop. For each vertex, it explores the possibility of a larger clique by adding other vertices in the graph and checking whether it still forms a clique with is_clique(). Meanwhile, it keeps track of the maximum clique size with the variable "max_". While exploring other vertices, the algorithm uses backtracking method with recursion, and explores other possibilities. The vertices are stored in the "store" array while being checked for their clique validity.

```
// n = number of vertices
store[n+1]
graph[n+1][n+1]
d[n+1]
```

is_clique(b):

```
    for i = 1 to b - 1:
        for j = i + 1 to b - 1:
            if graph[store[i]][store[j]] == 0:
                return false
    return true
```

maxCliques(i, l):

```
    max_ = 0
    for j = i + 1 to n:
        store[l] = j
        if is_clique(l + 1):
            max_ = max(max_, l)
            max_ = max(max_, maxCliques(j, l + 1))
    return max_
```

main():

```
        edges[][2] = {{1, 2}, {2,3}, {1,3}}
        size = length(edges)
        n = 3
        for i=0 to size:
            graph[edges[i][0]][edges[i][1]] = 1;
            graph[edges[i][1]][edges[i][0]] = 1;
            d[edges[i][0]]++;
            d[edges[i][1]]++;
         print maxCliques(0,1)
```

b. Heuristic Algorithm

Source: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6607889

The algorithm below is a heuristic implementation for the maximum clique problem. The basic idea is that it prunes the graph by removing the vertices of lowest degree on each step.

At the beginning, the algorithm checks whether the input graph is a clique or not by checking if all vertices' degrees are "size of the graph – 1". If it is a clique, for each vertex, the algorithm checks whether there is a larger clique by checking an array that stores the maximum clique for each vertex so far. If the input graph does not form a clique, then the algorithm prunes it until it reaches the maximum clique of it. It finds the vertex with the lowest degree, and creates the maximum subgraph in which the said vertex is a part of (a subgraph that includes the lowest degree vertex and all the nodes adjacent to it) and calls recursion with this subgraph. Then, it considers this subgraph without the said vertex, therefore pruning the lowest degree vertex. Then it makes a recursive call to the pruned subgraph as well. In the end, since the maximum clique for each vertex will be found, the maximum clique for the whole graph will be obtained. This algorithm is a divide and conquer algorithm, since it recursively divides the graphs into subgraphs and checks for cliques in each of them repeatedly. It is also a greedy algorithm, since it tries to make the locally optimal choice at each step by getting rid of the node with the lowest degree but it might not be the best option as that node may be a part of the maximal clique. However, the algorithm trades accuracy with time this way.

MaxClique(G):

```
1: if (G is a clique)
2:      for each vertex of G: v
3:           if (|V|-1 > maxC[v])
4:                maxC[v] := |V|;
5: else
6:      find the vertex of lowest degree: α
7:      find the largest subgraph of G in which α exists: G´ (V´,E´)
8:       MaxClique(G´ );
9:      if (V - α not empty)
10:          MaxClique(G – α);
```

3. Algorithm Analysis

a. Brute Force Algorithm

The correctness if maxCliques(i,l) can be proved with mathematical induction. The claim is that maxCliques(i,l) correctly finds the maximum clique starting from vertex i with l vertices.

Base case: When there is only one vertex left (n = 1), the function trivially returns 1, indicating a clique of size 1.

Inductive step: Assume that the function correctly identifies the maximum clique starting from vertex i with l vertices for all cases with fewer than n vertices.

Now, consider the case with n vertices. For each vertex j from i+1 to n, the function checks if the vertex can be added to the current clique (store) using the is_clique() function. If it can, the function compares the current clique size with the maximum found so far. It also calls itself recursively to explore other potential cliques.

By the inductive hypothesis, the recursive call correctly identifies the maximum cliques for smaller subgraphs. The function chooses the largest clique among the possibilities and returns its size.

Time Complexity:

The brute force algorithm works by checking all existing subsets each vertex is a part of by iterating over each vertex and checking all scenarios with recursion. The worst case would result in the maximum number of subsets for each vertex, which indicates a graph that has all the vertices connected to each other, with n-1 edges.

is_clique(): This function checks all edges with two nested loops, the outer one checking the vertex we are on, and the inner one checking the edges that are connected to it until it finds a missing one. In worst case, it will check n-1 existing edges for all n vertices one by one, which amounts to n^2 - n comparisons, thus having an **O(n^2)** complexity.

maxCliques(): This recursive function checks all subsets a vertex is a part of. In worst case, it will find 2^n - 1 subsets (total number of subsets in an n-element set) since there will be a clique of each cardinality up to n, because all vertices are connected. The function will calculate each possibility, making its complexity O(2^n).

Considering the complexities of the functions above, the overall complexity is **O(n^2 * 2^n)**. Since it is a brute-force algorithm, the complexity is not polynomial.

b. Heuristic Algorithm

Akbari (2013), demonstrates the algorithm's correction using mathematical induction. P(n) is denoted as the statement that the algorithm successfully identifies and yields the maximum clique for graphs of size n. The objective is to prove the validity of ∀n P(n).

Base case: When the graph consists of only one vertex (|V| = 1), it's a complete graph of size 1. Consequently, the algorithm can detect the maximum clique during its initial run, leading to termination. This establishes the truth of P(1).

Inductive step: It can be assumed that the algorithm correctly identifies the maximum clique when the graph contains at most k vertices. Now, the scenario gets examined with (k + 1) vertices in the graph. If the graph itself is complete, the maximum clique can be instantly identified during the algorithm's first execution.

For an incomplete graph, the vertex with the lowest degree is identified. Upon its removal, a subgraph of size at most k is left. By the inductive assumption, we know the algorithm accurately identifies the maximum clique for graphs of size 1 to k through recursive calls. Hence, the algorithm can determine the maximum clique for both subgraphs, including the one with size k. Since the desired answer must be present within these subgraphs, we can infer, utilizing the principle of mathematical induction, that ∀n P(n) holds true. Consequently, the algorithm's correctness is established.

Time Complexity:

Since the algorithm is a recursive divide and conquer algorithm, the time complexity can be found by calculating the time complexity of each execution of maxClique(G) * the number of recursive calls.

The time complexity of each execution of maxClique() is O(n). Because checking for each vertex for their degrees means iterating over the vertices once. Thus, the time complexity of each execution is at most O(n^2).

The number of recursive calls does not exceed O(n^4), which can be proved by mathematical induction. Akbari (2013, pp. 3-4) proves this by mathematical induction, stating that for any graph with n vertices, the algorithm's recursive calls are polynomial, supported by the base case of one vertex and inductive step considering cases of k and (k+1) vertices, ultimately resulting in a polynomial time complexity for the algorithm.

Consequently, the running time will be at most **O(n^5)**, which is polynomial for the heuristic algorithm.

4. Sample Generation (Random Instance Generator)

The sample generator creates a sample n-vertex graph with the given number of nodes. It first pushes all the nodes' numbers in an empty vector called "all". Then for all the nodes, it connects them to one of the already visited nodes which is chosen randomly by giving a range between 1 and the number of visited nodes. After all iterations, a connected graph is ensured. However, to increase the chances of a large clique and to increase randomness, it iterates over the nodes once again and for each node, it decides whether or not to connect it to all the other nodes other than itself by a binary randomizer. During connection, the function adjusts the adjacency matrix accordingly.

randomInt(min, max):

return random number between min and max (inclusive)

generateRandomGraph(numNodes):

```
all = empty list
    for i from 1 to numNodes:
        append i to all

    for v from 2 to numNodes:
        w = randomInt(1, v - 1)
        while w == v:
            w = randomInt(1, numNodes)
        graph[v][w] = 1
        graph[w][v] = 1

    for k from 0 to length of all - 1:
        for l from 0 to length of all - 1:
            if k + 1 != all[l]:
                rand = randomInt(0, 1)
                if rand == 1:
```

```
graph[k + 1][all[l]] = 1
graph[all[l]][k + 1] = 1
```

5. Algorithm Implementations

a. Brute Force & Heuristic Algorithm

Sample 1: N = 4:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}, }

Max Clique Size for Brute-Force Algorithm: 4

Maximum Clique Vertices:

v1 v2 v3 v4

Time taken for Brute Force Algorithm: 25 microseconds

Max Clique Size for Heuristic Algorithm: 4

Maximum Clique Vertices:

v1 v2 v3 v4

Time taken for Heuristic Algorithm: 20 microseconds

Sample 2: N = 7:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 6}, {1, 7}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {3, 4}, {3, 6}, {3, 7}, {4, 5}, {4, 6}, {4, 7}, {5, 6}, {5, 7}, }

Max Clique Size for Brute-Force Algorithm: 5

Maximum Clique Vertices:

v1 v2 v3 v4 v6

Time taken for Brute Force Algorithm: 163 microseconds

Max Clique Size for Heuristic Algorithm: 5

Maximum Clique Vertices:

v1 v2 v3 v4 v6

Time taken for Heuristic Algorithm: 369 microseconds

Sample 3: N = 10:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 10}, {3, 4}, {3, 5}, {3, 7}, {3, 8}, {3, 9}, {3, 10}, {4, 5}, {4, 6}, {4, 7}, {4, 8}, {4, 9}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {7, 8}, {7, 9}, {7, 10}, {8, 9}, {9, 10}, }

Max Clique Size for Brute-Force Algorithm: 7
Maximum Clique Vertices:
v1 v2 v4 v6 v7 v8 v9
Time taken for Brute Force Algorithm: 1377 microseconds

Max Clique Size for Heuristic Algorithm: 7
Maximum Clique Vertices:
v1 v2 v4 v6 v7 v8 v9
Time taken for Heuristic Algorithm: 810 microseconds

Sample 4: N = 15:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 10}, {1, 11}, {1, 12}, {1, 14}, {1, 15}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 10}, {2, 12}, {2, 14}, {2, 15}, {3, 6}, {3, 8}, {3, 9}, {3, 10}, {3, 13}, {3, 14}, {4, 5}, {4, 6}, {4, 9}, {4, 10}, {4, 11}, {4, 12}, {4, 13}, {4, 14}, {4, 15}, {5, 6}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {5, 11}, {5, 12}, {5, 14}, {5, 15}, {6, 7}, {6, 9}, {6, 11}, {6, 12}, {6, 13}, {6, 14}, {6, 15}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 13}, {7, 14}, {7, 15}, {8, 9}, {8, 11}, {8, 14}, {8, 15}, {9, 10}, {9, 11}, {9, 12}, {9, 13}, {9, 14}, {10, 11}, {10, 12}, {10, 14}, {10, 15}, {11, 13}, {11, 14}, {11, 15}, {12, 13}, {12, 14}, {13, 14}, {14, 15}, }

Max Clique Size for Brute-Force Algorithm: 7
Maximum Clique Vertices:
v1 v2 v4 v5 v6 v12 v14
Time taken for Brute Force Algorithm: 5137 microseconds

Max Clique Size for Heuristic Algorithm: 7
Maximum Clique Vertices:
v1 v2 v4 v5 v6 v12 v14

Time taken for Heuristic Algorithm: 3686 microseconds

Sample 5: N = 20:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 9}, {1, 11}, {1, 12}, {1, 13}, {1, 14}, {1, 16}, {1, 18}, {1, 19}, {2, 3}, {2, 4}, {2, 5}, {2, 7}, {2, 9}, {2, 10}, {2, 13}, {2, 14}, {2, 16}, {2, 17}, {2, 18}, {2, 19}, {2, 20}, {3, 5}, {3, 6}, {3, 7}, {3, 8}, {3, 9}, {3, 10}, {3, 11}, {3, 12}, {3, 15}, {3, 16}, {3, 18}, {3, 20}, {4, 5}, {4, 7}, {4, 8}, {4, 9}, {4, 10}, {4, 12}, {4, 13}, {4, 14}, {4, 15}, {4, 16}, {4, 17}, {4, 18}, {4, 19}, {4, 20}, {5, 6}, {5, 8}, {5, 9}, {5, 10}, {5, 11}, {5, 13}, {5, 14}, {5, 15}, {5, 16}, {5, 17}, {5, 18}, {5, 19}, {5, 20}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {6, 11}, {6, 12}, {6, 13}, {6, 14}, {6, 15}, {6, 16}, {6, 17}, {7, 8}, {7, 10}, {7, 11}, {7, 12}, {7, 13}, {7, 14}, {7, 15}, {7, 16}, {7, 17}, {7, 19}, {7, 20}, {8, 9}, {8, 10}, {8, 12}, {8, 13}, {8, 14}, {8, 15}, {8, 16}, {8, 17}, {8, 18}, {8, 19}, {8, 20}, {9, 10}, {9, 11}, {9, 12}, {9, 13}, {9, 14}, {9, 15}, {9, 16}, {9, 17}, {9, 18}, {9, 19}, {9, 20}, {10, 11}, {10, 12}, {10, 14}, {10, 15}, {10, 16}, {10, 17}, {10, 18}, {10, 19}, {11, 12}, {11, 13}, {11, 15}, {11, 16}, {11, 20}, {12, 13}, {12, 14}, {12, 15}, {12, 16}, {12, 18}, {12, 20}, {13, 15}, {13, 16}, {13, 19}, {13, 20}, {14, 16}, {14, 17}, {14, 18}, {14, 19}, {14, 20}, {15, 16}, {15, 18}, {15, 19}, {15, 20}, {16, 18}, {16, 20}, {17, 18}, {17, 19}, {17, 20}, {18, 19}, {19, 20}, }

Max Clique Size for Brute-Force Algorithm: 9
Maximum Clique Vertices:
v2 v4 v5 v9 v10 v14 v17 v18 v19
Time taken for Brute Force Algorithm: 51945 microseconds

Max Clique Size for Heuristic Algorithm: 8
Maximum Clique Vertices:
v2 v4 v5 v9 v10 v14 v17 v18
Time taken for Heuristic Algorithm: 10092 microseconds

Sample 6: N = 25:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {1, 12}, {1, 13}, {1, 15}, {1, 16}, {1, 17}, {1, 18}, {1, 19}, {1, 22}, {1, 24}, {1, 25}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 11}, {2, 12}, {2, 13}, {2,

14}, {2, 15}, {2, 16}, {2, 17}, {2, 18}, {2, 19}, {2, 20}, {2, 21}, {2, 22}, {2, 23}, {2, 24}, {2, 25}, {3, 4}, {3, 5}, {3, 6}, {3, 7}, {3, 9}, {3, 10}, {3, 11}, {3, 12}, {3, 13}, {3, 14}, {3, 15}, {3, 16}, {3, 17}, {3, 18}, {3, 19}, {3, 20}, {3, 21}, {3, 22}, {3, 23}, {3, 24}, {4, 5}, {4, 6}, {4, 7}, {4, 8}, {4, 12}, {4, 13}, {4, 14}, {4, 16}, {4, 17}, {4, 18}, {4, 19}, {4, 20}, {4, 21}, {4, 22}, {4, 23}, {4, 25}, {5, 6}, {5, 7}, {5, 9}, {5, 10}, {5, 11}, {5, 12}, {5, 13}, {5, 14}, {5, 15}, {5, 17}, {5, 19}, {5, 20}, {5, 21}, {5, 22}, {5, 23}, {5, 24}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {6, 11}, {6, 12}, {6, 13}, {6, 14}, {6, 16}, {6, 17}, {6, 18}, {6, 20}, {6, 21}, {6, 22}, {6, 23}, {6, 24}, {6, 25}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 12}, {7, 13}, {7, 14}, {7, 15}, {7, 16}, {7, 17}, {7, 18}, {7, 22}, {7, 23}, {7, 24}, {7, 25}, {8, 10}, {8, 11}, {8, 12}, {8, 14}, {8, 15}, {8, 16}, {8, 17}, {8, 20}, {8, 21}, {8, 22}, {8, 23}, {8, 24}, {9, 10}, {9, 11}, {9, 12}, {9, 13}, {9, 14}, {9, 17}, {9, 18}, {9, 19}, {9, 20}, {9, 21}, {9, 22}, {9, 23}, {9, 24}, {10, 11}, {10, 12}, {10, 13}, {10, 15}, {10, 16}, {10, 17}, {10, 18}, {10, 19}, {10, 20}, {10, 22}, {10, 23}, {10, 24}, {11, 12}, {11, 13}, {11, 14}, {11, 16}, {11, 18}, {11, 19}, {11, 20}, {11, 21}, {11, 22}, {11, 23}, {11, 24}, {11, 25}, {12, 15}, {12, 17}, {12, 19}, {12, 21}, {12, 22}, {12, 23}, {12, 24}, {12, 25}, {13, 14}, {13, 15}, {13, 16}, {13, 17}, {13, 19}, {13, 20}, {13, 21}, {13, 22}, {13, 23}, {13, 24}, {13, 25}, {14, 15}, {14, 16}, {14, 18}, {14, 20}, {14, 21}, {14, 22}, {14, 23}, {14, 25}, {15, 16}, {15, 17}, {15, 18}, {15, 19}, {15, 20}, {15, 21}, {15, 22}, {15, 23}, {15, 25}, {16, 17}, {16, 19}, {16, 20}, {16, 21}, {16, 22}, {16, 23}, {16, 24}, {16, 25}, {17, 18}, {17, 19}, {17, 21}, {17, 22}, {17, 24}, {17, 25}, {18, 19}, {18, 21}, {18, 22}, {18, 23}, {18, 24}, {19, 20}, {19, 21}, {19, 22}, {19, 23}, {19, 24}, {19, 25}, {20, 21}, {20, 22}, {20, 23}, {20, 24}, {20, 25}, {21, 22}, {21, 23}, {21, 24}, {21, 25}, {22, 24}, {22, 25}, {23, 24}, {24, 25}, }

Max Clique Size for Brute-Force Algorithm: 11

Maximum Clique Vertices:

v1 v2 v3 v5 v6 v7 v9 v12 v17 v22 v24

Time taken for Brute Force Algorithm: 406148 microseconds

Max Clique Size for Heuristic Algorithm: 11

Maximum Clique Vertices:

v1 v2 v3 v5 v6 v7 v9 v12 v17 v22 v24

Time taken for Heuristic Algorithm: 21268 microseconds

Sample 7: N = 30:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {1, 11}, {1, 13}, {1, 14}, {1, 15}, {1, 16}, {1, 17}, {1, 18}, {1, 19}, {1, 20}, {1, 21}, {1, 23}, {1, 24}, {1, 25}, {1, 26}, {1, 27}, {2, 3}, {2, 5}, {2, 6}, {2, 8}, {2, 9}, {2, 10}, {2, 11}, {2, 13}, {2, 14}, {2, 15}, {2, 16}, {2, 17}, {2, 18}, {2, 19}, {2, 20}, {2, 21}, {2, 23}, {2, 25}, {2, 26}, {2, 27}, {2, 28}, {2, 29}, {3, 4}, {3, 9}, {3, 10}, {3, 11}, {3, 14}, {3, 16}, {3, 17}, {3, 19}, {3, 20}, {3, 22}, {3, 24}, {3, 26}, {3, 30}, {4, 5}, {4, 6}, {4, 8}, {4, 9}, {4, 10}, {4, 11}, {4, 14}, {4, 16}, {4, 17}, {4, 22}, {4, 23}, {4, 24}, {4, 25}, {4, 27}, {4, 28}, {4, 29}, {4, 30}, {5, 6}, {5, 7}, {5, 10}, {5, 12}, {5, 13}, {5, 14}, {5, 15}, {5, 17}, {5, 18}, {5, 19}, {5, 20}, {5, 21}, {5, 22}, {5, 23}, {5, 24}, {5, 25}, {5, 26}, {5, 27}, {5, 28}, {6, 8}, {6, 9}, {6, 10}, {6, 11}, {6, 13}, {6, 15}, {6, 17}, {6, 18}, {6, 19}, {6, 20}, {6, 21}, {6, 24}, {6, 25}, {6, 27}, {6, 28}, {6, 29}, {6, 30}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 15}, {7, 16}, {7, 17}, {7, 18}, {7, 19}, {7, 20}, {7, 21}, {7, 22}, {7, 23}, {7, 24}, {7, 25}, {7, 26}, {7, 29}, {7, 30}, {8, 9}, {8, 10}, {8, 11}, {8, 12}, {8, 14}, {8, 15}, {8, 16}, {8, 17}, {8, 18}, {8, 20}, {8, 21}, {8, 22}, {8, 23}, {8, 24}, {8, 25}, {8, 27}, {8, 28}, {8, 30}, {9, 10}, {9, 11}, {9, 12}, {9, 13}, {9, 14}, {9, 15}, {9, 16}, {9, 17}, {9, 18}, {9, 19}, {9, 20}, {9, 21}, {9, 22}, {9, 24}, {9, 25}, {9, 26}, {9, 27}, {9, 28}, {9, 29}, {9, 30}, {10, 11}, {10, 12}, {10, 13}, {10, 15}, {10, 16}, {10, 18}, {10, 21}, {10, 22}, {10, 23}, {10, 24}, {10, 25}, {10, 26}, {10, 27}, {10, 28}, {10, 29}, {11, 13}, {11, 14}, {11, 15}, {11, 16}, {11, 17}, {11, 19}, {11, 20}, {11, 21}, {11, 22}, {11, 23}, {11, 25}, {11, 26}, {11, 27}, {11, 28}, {11, 30}, {12, 14}, {12, 16}, {12, 17}, {12, 18}, {12, 19}, {12, 20}, {12, 22}, {12, 23}, {12, 24}, {12, 25}, {12, 27}, {12, 28}, {13, 15}, {13, 16}, {13, 17}, {13, 18}, {13, 19}, {13, 20}, {13, 21}, {13, 23}, {13, 24}, {13, 25}, {13, 27}, {13, 28}, {13, 30}, {14, 16}, {14, 17}, {14, 18}, {14, 20}, {14, 22}, {14, 24}, {14, 25}, {14, 26}, {14, 27}, {14, 28}, {14, 29}, {14, 30}, {15, 16}, {15, 17}, {15, 18}, {15, 19}, {15, 20}, {15, 21}, {15, 22}, {15, 23}, {15, 24}, {15, 26}, {15, 28}, {15, 29}, {15, 30}, {16, 17}, {16, 19}, {16, 20}, {16, 21}, {16, 22}, {16, 23}, {16, 26}, {16, 27}, {16, 28}, {16, 30}, {17, 18}, {17, 19}, {17, 20}, {17, 21}, {17, 23}, {17, 24}, {17, 26}, {17, 27}, {17, 28}, {17, 29}, {18, 20}, {18, 21}, {18, 22}, {18, 23}, {18, 24}, {18, 25}, {18, 26}, {18, 28}, {18, 29}, {18, 30}, {19, 20}, {19, 21}, {19, 23}, {19, 24}, {19, 25}, {19, 26}, {19, 28}, {19, 29}, {19, 30}, {20, 21}, {20, 22}, {20, 23}, {20, 24}, {20, 25}, {20, 26},

{20, 27}, {20, 28}, {20, 30}, {21, 22}, {21, 23}, {21, 25}, {21, 26}, {21, 27}, {21, 29}, {22, 24}, {22, 25}, {22, 26}, {22, 27}, {22, 28}, {22, 29}, {23, 24}, {23, 25}, {23, 26}, {23, 27}, {23, 28}, {23, 29}, {23, 30}, {24, 26}, {24, 27}, {24, 28}, {24, 29}, {24, 30}, {25, 27}, {25, 28}, {25, 30}, {26, 28}, {26, 29}, {27, 28}, {27, 29}, {27, 30}, {28, 29}, {28, 30}, {29, 30}, }

Max Clique Size for Brute-Force Algorithm: 11
Maximum Clique Vertices:
v1 v2 v6 v9 v11 v13 v15 v17 v19 v20 v21
Time taken for Brute Force Algorithm: 765004 microseconds

Max Clique Size for Heuristic Algorithm: 10
Maximum Clique Vertices:
v1 v2 v6 v9 v11 v13 v15 v17 v19 v20
Time taken for Heuristic Algorithm: 38832 microseconds

Sample 8: N = 36:

Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 6}, {1, 8}, {1, 9}, {1, 10}, {1, 12}, {1, 13}, {1, 14}, {1, 15}, {1, 16}, {1, 17}, {1, 18}, {1, 19}, {1, 20}, {1, 21}, {1, 24}, {1, 25}, {1, 27}, {1, 30}, {1, 31}, {1, 33}, {1, 34}, {1, 36}, {2, 3}, {2, 5}, {2, 6}, {2, 7}, {2, 10}, {2, 12}, {2, 13}, {2, 16}, {2, 17}, {2, 18}, {2, 20}, {2, 23}, {2, 24}, {2, 26}, {2, 27}, {2, 28}, {2, 29}, {2, 32}, {2, 33}, {2, 35}, {2, 36}, {3, 4}, {3, 5}, {3, 6}, {3, 7}, {3, 8}, {3, 10}, {3, 12}, {3, 13}, {3, 15}, {3, 16}, {3, 18}, {3, 19}, {3, 21}, {3, 22}, {3, 23}, {3, 26}, {3, 27}, {3, 28}, {3, 29}, {3, 30}, {3, 31}, {3, 32}, {3, 33}, {3, 34}, {3, 35}, {3, 36}, {4, 5}, {4, 7}, {4, 8}, {4, 9}, {4, 11}, {4, 12}, {4, 14}, {4, 15}, {4, 16}, {4, 17}, {4, 19}, {4, 20}, {4, 21}, {4, 24}, {4, 25}, {4, 26}, {4, 27}, {4, 30}, {4, 31}, {4, 32}, {4, 33}, {4, 34}, {4, 35}, {5, 6}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {5, 13}, {5, 14}, {5, 15}, {5, 16}, {5, 17}, {5, 18}, {5, 19}, {5, 20}, {5, 21}, {5, 22}, {5, 23}, {5, 24}, {5, 25}, {5, 26}, {5, 27}, {5, 28}, {5, 29}, {5, 30}, {5, 31}, {5, 34}, {5, 35}, {5, 36}, {6, 8}, {6, 9}, {6, 10}, {6, 12}, {6, 13}, {6, 15}, {6, 16}, {6, 17}, {6, 18}, {6, 19}, {6, 20}, {6, 21}, {6, 22}, {6, 23}, {6, 24}, {6, 26}, {6, 27}, {6, 28}, {6, 29}, {6, 30}, {6, 31}, {6, 34}, {6, 35}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 15}, {7, 16}, {7, 17}, {7, 18}, {7, 19}, {7, 20}, {7, 22}, {7, 23}, {7, 24}, {7, 25}, {7, 26}, {7, 27}, {7, 28}, {7, 29}, {7, 30}, {7, 31}, {7, 32}, {7,

34}, {7, 35}, {8, 10}, {8, 11}, {8, 12}, {8, 13}, {8, 15}, {8, 16}, {8, 17}, {8, 19}, {8, 20}, {8, 22}, {8, 23}, {8, 25}, {8, 26}, {8, 28}, {8, 29}, {8, 30}, {8, 31}, {8, 32}, {8, 33}, {8, 34}, {8, 35}, {8, 36}, {9, 10}, {9, 11}, {9, 13}, {9, 14}, {9, 15}, {9, 16}, {9, 18}, {9, 19}, {9, 20}, {9, 21}, {9, 22}, {9, 23}, {9, 24}, {9, 25}, {9, 26}, {9, 27}, {9, 28}, {9, 29}, {9, 30}, {9, 32}, {9, 33}, {9, 34}, {9, 36}, {10, 11}, {10, 12}, {10, 14}, {10, 15}, {10, 16}, {10, 17}, {10, 18}, {10, 19}, {10, 20}, {10, 22}, {10, 23}, {10, 24}, {10, 25}, {10, 28}, {10, 30}, {10, 32}, {10, 33}, {10, 34}, {10, 35}, {11, 12}, {11, 13}, {11, 14}, {11, 16}, {11, 17}, {11, 20}, {11, 23}, {11, 25}, {11, 26}, {11, 27}, {11, 28}, {11, 29}, {11, 30}, {11, 31}, {11, 33}, {11, 34}, {11, 35}, {11, 36}, {12, 13}, {12, 14}, {12, 16}, {12, 17}, {12, 18}, {12, 20}, {12, 21}, {12, 24}, {12, 25}, {12, 28}, {12, 29}, {12, 30}, {12, 31}, {12, 32}, {12, 33}, {12, 34}, {13, 14}, {13, 15}, {13, 16}, {13, 17}, {13, 18}, {13, 19}, {13, 20}, {13, 22}, {13, 23}, {13, 25}, {13, 26}, {13, 27}, {13, 30}, {13, 31}, {13, 32}, {13, 33}, {13, 34}, {13, 35}, {13, 36}, {14, 15}, {14, 16}, {14, 17}, {14, 19}, {14, 20}, {14, 21}, {14, 23}, {14, 25}, {14, 27}, {14, 28}, {14, 29}, {14, 30}, {14, 32}, {14, 33}, {14, 34}, {14, 35}, {14, 36}, {15, 16}, {15, 17}, {15, 18}, {15, 19}, {15, 20}, {15, 21}, {15, 22}, {15, 23}, {15, 24}, {15, 25}, {15, 26}, {15, 27}, {15, 28}, {15, 30}, {15, 31}, {15, 33}, {15, 34}, {15, 35}, {16, 17}, {16, 18}, {16, 20}, {16, 21}, {16, 22}, {16, 23}, {16, 25}, {16, 26}, {16, 27}, {16, 28}, {16, 29}, {16, 30}, {16, 31}, {16, 32}, {16, 33}, {16, 34}, {16, 35}, {16, 36}, {17, 19}, {17, 20}, {17, 22}, {17, 23}, {17, 24}, {17, 25}, {17, 27}, {17, 28}, {17, 29}, {17, 30}, {17, 34}, {17, 35}, {18, 19}, {18, 20}, {18, 22}, {18, 23}, {18, 25}, {18, 28}, {18, 30}, {18, 32}, {18, 33}, {18, 34}, {18, 35}, {18, 36}, {19, 21}, {19, 22}, {19, 23}, {19, 24}, {19, 25}, {19, 26}, {19, 29}, {19, 31}, {19, 35}, {19, 36}, {20, 21}, {20, 22}, {20, 23}, {20, 24}, {20, 25}, {20, 26}, {20, 27}, {20, 29}, {20, 31}, {20, 33}, {20, 34}, {20, 35}, {20, 36}, {21, 26}, {21, 28}, {21, 30}, {21, 31}, {21, 34}, {21, 35}, {21, 36}, {22, 24}, {22, 25}, {22, 26}, {22, 27}, {22, 29}, {22, 31}, {22, 32}, {22, 34}, {22, 36}, {23, 24}, {23, 26}, {23, 27}, {23, 28}, {23, 29}, {23, 30}, {23, 31}, {23, 33}, {23, 34}, {23, 35}, {23, 36}, {24, 26}, {24, 27}, {24, 28}, {24, 29}, {24, 30}, {24, 31}, {24, 32}, {24, 33}, {24, 34}, {24, 35}, {24, 36}, {25, 26}, {25, 27}, {25, 30}, {25, 31}, {25, 33}, {25, 34}, {26, 27}, {26, 28}, {26, 29}, {26, 30}, {26, 31}, {26, 34}, {26, 35}, {27, 28}, {27, 29}, {27, 30}, {27, 31}, {27, 32}, {27, 33}, {27, 34}, {27, 35}, {28, 30}, {28, 31}, {28, 32}, {28, 34}, {28, 35}, {28, 36}, {29, 30}, {29, 31}, {29, 32}, {29, 33}, {29, 34}, {29, 35}, {29, 36}, {30, 31}, {30, 32}, {30, 33}, {30,

34}, {30, 35}, {30, 36}, {31, 32}, {31, 33}, {31, 35}, {31, 36}, {32, 33}, {32, 34}, {32, 36}, {33, 35}, {33, 36}, {34, 35}, {35, 36}, }

Max Clique Size for Brute-Force Algorithm: 12
Maximum Clique Vertices:
v3 v5 v6 v8 v10 v15 v16 v23 v28 v30 v34 v35
Time taken for Brute Force Algorithm: 2397308 microseconds

Max Clique Size for Heuristic Algorithm: 11
Maximum Clique Vertices:
v3 v5 v6 v8 v10 v15 v16 v23 v28 v30 v34

Sample 9: N = 40:
Max Clique Size for Brute-Force Algorithm: 13
Maximum Clique Vertices:
v4 v8 v9 v10 v11 v14 v17 v19 v23 v26 v36 v39 v40
Time taken for Brute Force Algorithm: 4991007 microseconds

Max Clique Size for Heuristic Algorithm: 11
Maximum Clique Vertices:
v4 v8 v9 v10 v11 v14 v17 v19 v23 v26 v36
Time taken for Heuristic Algorithm: 101680 microseconds

Sample 10: N = 45:
Max Clique Size for Brute-Force Algorithm: 13
Maximum Clique Vertices:
v2 v3 v5 v6 v9 v10 v19 v23 v27 v35 v36 v40 v44
Time taken for Brute Force Algorithm: 14440087 microseconds

Max Clique Size for Heuristic Algorithm: 13
Maximum Clique Vertices:
v2 v3 v5 v6 v9 v10 v19 v23 v27 v35 v36 v40 v44
Time taken for Heuristic Algorithm: 145386 microseconds

Sample 11: N = 50:

Max Clique Size for Brute-Force Algorithm: 13

Maximum Clique Vertices:

v1 v2 v6 v14 v17 v20 v26 v27 v34 v35 v40 v42 v48

Time taken for Brute Force Algorithm: 23536946 microseconds

Max Clique Size for Heuristic Algorithm: 11

Maximum Clique Vertices:

v1 v2 v6 v14 v17 v20 v26 v27 v34 v35 v40

Time taken for Heuristic Algorithm: 210449 microseconds

Sample 12: N = 55:

Max Clique Size for Brute-Force Algorithm: 14

Maximum Clique Vertices:

v2 v3 v6 v12 v13 v14 v17 v19 v22 v27 v31 v37 v45 v48

Time taken for Brute Force Algorithm: 51432025 microseconds

Max Clique Size for Heuristic Algorithm: 13

Maximum Clique Vertices:

v2 v3 v6 v12 v13 v14 v17 v19 v22 v27 v31 v37 v45

Time taken for Heuristic Algorithm: 288362 microseconds

Sample 13: N = 60:

Max Clique Size for Brute-Force Algorithm: 16

Maximum Clique Vertices:

v2 v3 v4 v6 v9 v11 v14 v17 v19 v36 v37 v44 v47 v49 v51 v57

Time taken for Brute Force Algorithm: 355056670 microseconds

Max Clique Size for Heuristic Algorithm: 15

Maximum Clique Vertices:

v2 v3 v4 v6 v9 v11 v14 v17 v19 v36 v37 v44 v47 v49 v51

Time taken for Heuristic Algorithm: 419875 microseconds

Sample 14: N = 65:

Max Clique Size for Brute-Force Algorithm: 16
Maximum Clique Vertices:
v1 v2 v4 v14 v19 v21 v23 v26 v27 v47 v51 v59 v63 v64 v69 v76
Time taken for Brute Force Algorithm: 1937939990 microseconds

Max Clique Size for Heuristic Algorithm: 15
Maximum Clique Vertices:
v1 v2 v4 v14 v19 v21 v23 v26 v27 v47 v51 v59 v63 v64 v69
Time taken for Heuristic Algorithm: 1195409 microseconds

Sample 15: N = 70:
Max Clique Size for Brute-Force Algorithm: 15
Maximum Clique Vertices:
v1 v2 v5 v7 v9 v10 v12 v20 v22 v25 v28 v34 v50 v51 v55
Time taken for Brute Force Algorithm: 892621871 microseconds

Max Clique Size for Heuristic Algorithm: 13
Maximum Clique Vertices:
v1 v2 v5 v7 v9 v10 v12 v20 v22 v25 v28 v34 v50
Time taken for Heuristic Algorithm: 746241 microseconds

6. Experimental Analysis of The Performance (Performance Testing)

For each input size, both algorithms have been run 20 times. Running times are in microseconds. The constant "t" used for performance testing is 2.093 for %95 confidence. The interval for each input size is calculated in its respective python code and the interval is between (m - 2.093*sm) and (m + 2.093*sm), but the actual mean value taken is the upper bound M1 (m + 2.093*sm), for a more consistent graph with single values rather than the whole interval.

2 to 20 node graphs are tried for the brute force algorithm and 2 to 60
The line's slope in the brute force graph is 349.06, and the fitted line is linear.
The line's slope in the heuristic graph is 3829.36, and the fitted line is linear.

The growth being linear is expected for the graphs as the input size is relatively small, so it might not line up with the theoretical calculations. However, when the M values are observed, it can be seen that when the growth happens after some certain input size, it is quite larger than the previous input, almost exponential.

Running times are calculated as below:

Brute Force Algorithm:

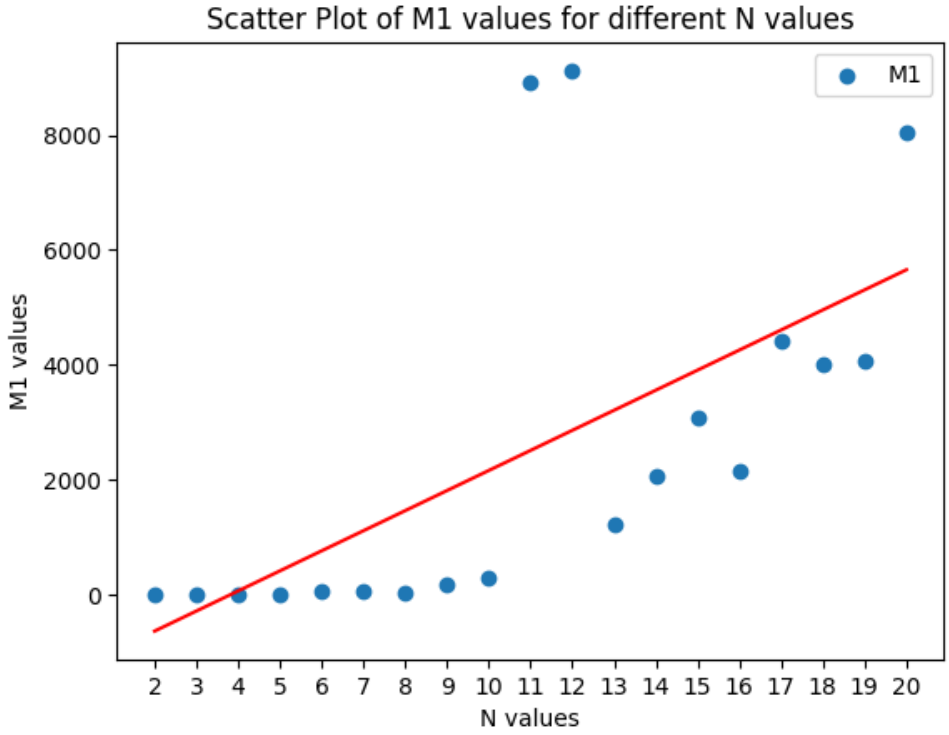
N=2: [1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
N=3: [4, 2, 3, 3, 2, 2, 3, 3, 3, 4, 2, 2, 4, 2, 2, 4, 2, 3, 3, 3]
N=4: [6, 8, 7, 8, 6, 7, 7, 6, 7, 7, 10, 4, 7, 6, 5, 7, 8, 5, 7, 7]
N=5: [21, 14, 13, 11, 17, 12, 15, 20, 9, 12, 12, 22, 11, 11, 12, 15, 12, 18, 12, 12]
N=6: [25, 25, 23, 26, 36, 129, 31, 51, 111, 52, 35, 31, 24, 23, 24, 23, 23, 24, 24]
N=7: [50, 49, 47, 39, 48, 42, 50, 46, 56, 53, 40, 41, 78, 48, 38, 38, 39, 39, 39]
N=8: [43, 43, 43, 42, 42, 45, 42, 41, 42, 44, 46, 46, 45, 45, 31, 30, 31, 44, 40]
N=9: [168, 220, 185, 174, 164, 165, 170, 163, 161, 174, 163, 163, 182, 165, 164, 184, 164, 164, 169]
N=10: [368, 286, 288, 311, 287, 280, 271, 292, 282, 282, 313, 278, 276, 274, 291, 280, 284, 283, 274]
N=11: [521, 502, 42334, 862, 722, 720, 3774, 720, 760, 689, 683, 683, 710, 754, 738, 725, 661, 714, 704]
N=12: [1510, 1267, 1234, 1331, 1221, 1307, 1397, 40884, 1860, 1879, 5930, 1826, 1816, 1586, 1266, 1272, 1255, 1610, 1825]
N=13: [1308, 1158, 1171, 1240, 1145, 1171, 1160, 1165, 1125, 1141, 1178, 1279, 1170, 1165, 1211, 1202, 1149, 1181, 1245]
N=14: [1345, 1200, 1162, 1541, 1238, 1184, 1195, 1214, 1177, 1175, 1172, 1180, 1171, 1195, 1162, 5585, 1752, 1757, 1709]
N=15: [2062, 1988, 1974, 1953, 1985, 1996, 2456, 7037, 3053, 2956, 2663, 2011, 2005, 2745, 2035, 2007, 1978, 1985, 1990]
N=16: [2167, 2198, 2141, 2066, 2053, 2114, 2176, 2047, 2158, 2200, 2251, 2143, 2185, 2085, 2063, 2205, 2025, 1994, 2160]

N=17: [3876, 3930, 3966, 3956, 4062, 4243, 3950, 4076, 4691, 4050, 4152, 4172, 3979, 4258, 3991, 4002, 4144, 3976, 5973]

N=18: [5902, 3597, 3587, 3662, 3761, 3589, 3812, 3739, 3704, 3657, 3570, 3592, 3656, 3553, 3590, 3583, 3560, 3613, 3628]

N=19: [3957, 3868, 5264, 3842, 4021, 3796, 3777, 3953, 3871, 4042, 3774, 3830, 3827, 3828, 3892, 3802, 3604, 3792, 3831]

N=20: [8125, 7870, 7929, 8437, 7926, 7934, 7601, 7728, 7527, 7683, 7462, 7583, 7362, 8983, 7898, 7207, 7444, 7424, 8677]



Heuristic Algorithm:

N=2: [3, 2, 3, 2, 4, 2, 2, 2, 2, 2, 3, 3, 3, 3, 2, 2, 3, 2, 4, 2]

N=3: [3, 5, 4, 3, 11, 5, 6, 5, 3, 5, 3, 5, 3, 4, 4, 5, 4, 3, 3, 3]

N=4: [21, 13, 4, 22, 22, 13, 27, 16, 15, 4, 6, 28, 20, 13, 16, 12, 22, 9, 4, 4]

N=5: [6, 23, 24, 24, 4, 42, 34, 34, 46, 24, 24, 6, 42, 64, 36, 35, 45, 4, 25, 36]

N=6: [46, 39, 51, 38, 39, 38, 38, 38, 37, 37, 37, 38, 37, 37, 64, 39, 38, 37, 37]

N=7: [73, 67, 63, 64, 62, 64, 74, 74, 89, 67, 63, 86, 70, 62, 63, 64, 63, 63, 63]

N=8: [174, 122, 112, 113, 112, 119, 112, 110, 110, 135, 115, 115, 111, 129, 112, 111, 113, 134, 111]

N=9: [295, 292, 281, 281, 292, 275, 283, 275, 285, 279, 288, 289, 278, 285, 280, 288, 277, 279, 293]

N=10: [304, 249, 250, 220, 238, 276, 245, 224, 224, 251, 246, 222, 220, 226, 220, 239, 215, 227, 222]

N=11: [475, 411, 421, 1329, 274, 254, 287, 277, 259, 253, 278, 318, 259, 275, 257, 253, 253, 278, 256]

N=12: [521, 519, 356, 316, 300, 317, 426, 385, 298, 311, 298, 383, 413, 350, 390, 464, 455, 312, 322]

N=13: [579, 552, 514, 545, 523, 526, 525, 527, 544, 599, 528, 511, 623, 767, 561, 861, 663, 654, 574]

N=14: [797, 833, 726, 699, 726, 680, 693, 697, 692, 703, 679, 686, 782, 838, 703, 682, 683, 692, 691]

N=15: [1007, 986, 984, 978, 958, 954, 1365, 1293, 1053, 1002, 981, 974, 1600, 1005, 980, 1006, 976, 995, 959]

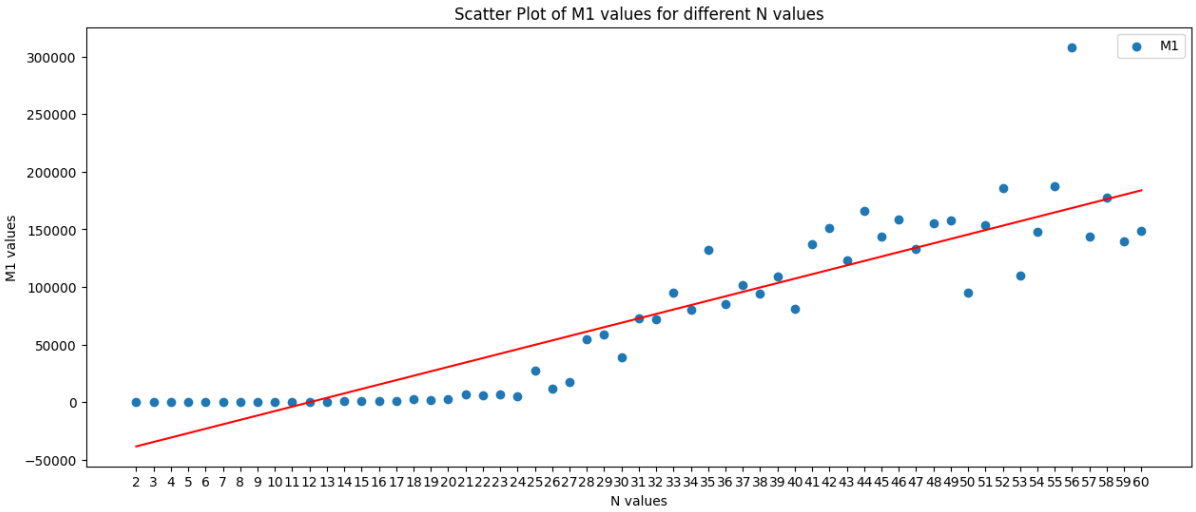
N=16: [1368, 1287, 1307, 1300, 1300, 1293, 1304, 1311, 1321, 1318, 1374, 1601, 1394, 1647, 1283, 1285, 1351, 1314, 1347]

N=17: [1494, 1324, 1717, 1371, 1401, 1662, 1662, 1456, 1314, 1294, 1462, 1311, 1295, 1290, 1309, 1251, 1281, 1300, 1459]

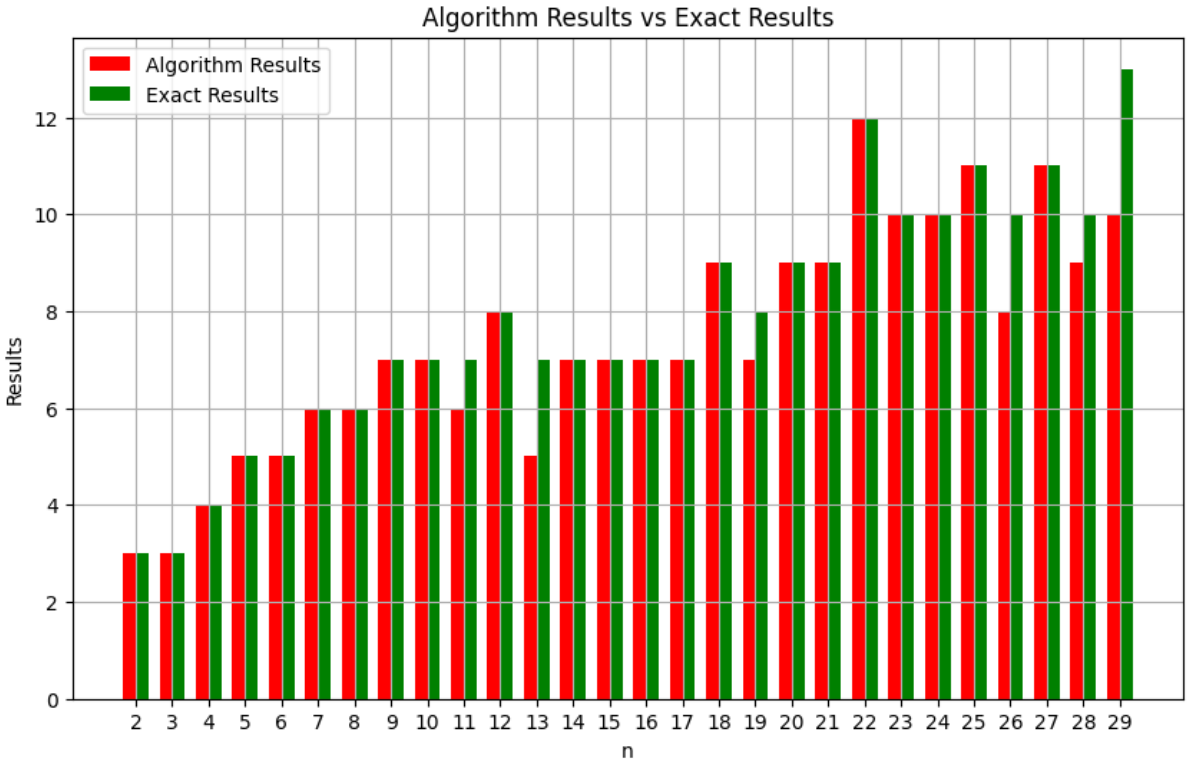
N=18: [2090, 1868, 2206, 2896, 2863, 2866, 2862, 2873, 2830, 2916, 2863, 2906, 2718, 1911, 1842, 1825, 1951, 1822, 1820]

N=19: [2191, 2262, 2172, 2152, 2104, 2109, 2110, 2314, 2159, 2164, 2119, 2162, 2262, 2249, 2207, 2157, 2121, 2137, 2121]

N=20: [2932, 2841, 2769, 2766, 2735, 2790, 3568, 3034, 2799, 2758, 2737, 2749, 2751, 2808, 2747, 2729, 3333, 3284, 2781]



7. Experimental Analysis of the Quality



Graphs of N=2 to N=30 are tried to see the accuracy of the results. As seen from the bar graph, the heuristic algorithm did not always find the maximal clique unlike the brute-force algorithm, due to its pruning and greedy nature. The algorithm sometimes got rid of the nodes that are a part of the actual maximum clique, hence found in smaller cliques than the brute force's results.

8. Experimental Analysis of the Correctness (Functional Testing)

Black Box Testing:

For black box testing, multiple graph combinations are used with a specified node and edge count with the help of generateSpecificGraph().

As it can be seen from the marked outputs, though the brute force algorithm worked correctly for the inputs, there occurred some mismatches between the maximal clique and the vertices in the heuristic algorithm. Also, the maximum size sometimes overflowed in the heuristic algorithm.

Samples:

N = 5, Edge number: 4:

Graph Edges: { {1, 3}, {1, 5}, {2, 4}, {3, 5}, }
Max Clique Size for Brute-Force Algorithm: 3
Maximum Clique Vertices:
v1 v3 v5
Time taken for Brute Force Algorithm: 7 microseconds

Max Clique Size for Heuristic Algorithm: 3
Time taken for Heuristic Algorithm: 14 microseconds
Maximum Clique Vertices:
v1 v3 v5

N = 5, Edge number: 5:

Graph Edges: { {1, 2}, {1, 5}, {2, 3}, {3, 4}, {3, 5}, }
Max Clique Size for Brute-Force Algorithm: 2
Maximum Clique Vertices:
v1 v2
Time taken for Brute Force Algorithm: 7 microseconds

Max Clique Size for Heuristic Algorithm: 2

Time taken for Heuristic Algorithm: 35 microseconds

Maximum Clique Vertices:

v1 v2

N = 5, Edge number: 8:

Graph Edges: { {1, 2}, {1, 4}, {1, 5}, {2, 3}, {2, 4}, {2, 5}, {3, 4}, {3, 5}, }

Max Clique Size for Brute-Force Algorithm: 3

Maximum Clique Vertices:

v1 v2 v4

Time taken for Brute Force Algorithm: 14 microseconds

Max Clique Size for Heuristic Algorithm: 2

Time taken for Heuristic Algorithm: 48 microseconds

Maximum Clique Vertices:

v1 v2

N = 8, Edge number: 10:

Graph Edges: { {1, 2}, {1, 5}, {1, 8}, {2, 4}, {2, 5}, {2, 6}, {3, 4}, {3, 6}, {5, 8}, {6, 8}, }

Max Clique Size for Brute-Force Algorithm: 3

Maximum Clique Vertices:

v1 v2 v5

Time taken for Brute Force Algorithm: 212 microseconds

Max Clique Size for Heuristic Algorithm: 2

Time taken for Heuristic Algorithm: 427 microseconds

Maximum Clique Vertices:

v1 v5 v8

N = 10, Edge number: 5:

Graph Edges: { {1, 6}, {2, 3}, {4, 7}, {6, 8}, {8, 10}, }

Max Clique Size for Brute-Force Algorithm: 2

Maximum Clique Vertices:

v1 v6

Time taken for Brute Force Algorithm: 15 microseconds

Max Clique Size for Heuristic Algorithm: 2
Time taken for Heuristic Algorithm: 66 microseconds
Maximum Clique Vertices:
v1 v6

N = 10, Edge number: 10:
Graph Edges: { {1, 8}, {2, 4}, {2, 9}, {3, 10}, {4, 7}, {5, 9}, {5, 10}, {6, 9}, {7, 9}, {8, 10}, }
Max Clique Size for Brute-Force Algorithm: 2
Maximum Clique Vertices:
v1 v8
Time taken for Brute Force Algorithm: 16 microseconds

Max Clique Size for Heuristic Algorithm: 2
Time taken for Heuristic Algorithm: 159 microseconds
Maximum Clique Vertices:
v1 v8

N = 10, Edge number: 35:
Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 10}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 10}, {3, 4}, {3, 5}, {3, 7}, {3, 8}, {3, 9}, {3, 10}, {4, 6}, {4, 8}, {4, 9}, {5, 6}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {8, 9}, {8, 10}, {9, 10}, }
Max Clique Size for Brute-Force Algorithm: 6
Maximum Clique Vertices:
v2 v3 v5 v8 v9 v10
Time taken for Brute Force Algorithm: 228 microseconds

Max Clique Size for Heuristic Algorithm: 6
Time taken for Heuristic Algorithm: 341 microseconds
Maximum Clique Vertices:
v2 v3 v5 v8 v9 v10

N = 15, Edge number: 27 :

Graph Edges: { {1, 4}, {1, 10}, {1, 12}, {2, 6}, {2, 10}, {3, 4}, {3, 7}, {3, 8}, {3, 11}, {4, 5}, {4, 6}, {4, 8}, {4, 10}, {4, 11}, {4, 12}, {5, 14}, {6, 13}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 14}, {8, 14}, {10, 14}, {11, 12}, {13, 15}, {14, 15}, }

Max Clique Size for Brute-Force Algorithm: 3

Maximum Clique Vertices:

v1 v4 v10

Time taken for Brute Force Algorithm: 577 microseconds

Max Clique Size for Heuristic Algorithm: 526070208

Time taken for Heuristic Algorithm: 1656 microseconds

Maximum Clique Vertices:

v3 v7 v11

N = 20, Edge number : 10:

Graph Edges: { {1, 2}, {1, 5}, {1, 19}, {2, 3}, {3, 10}, {4, 11}, {7, 10}, {7, 13}, {9, 14}, {13, 16}, }

Max Clique Size for Brute-Force Algorithm: 2

Maximum Clique Vertices:

v1 v2

Time taken for Brute Force Algorithm: 63 microseconds

Max Clique Size for Heuristic Algorithm: 2

Time taken for Heuristic Algorithm: 251 microseconds

Maximum Clique Vertices:

v4 v11

N = 20, Edge number: 30:

Graph Edges: { {1, 5}, {2, 8}, {3, 7}, {3, 11}, {3, 19}, {3, 20}, {4, 5}, {4, 8}, {5, 7}, {5, 9}, {5, 16}, {5, 18}, {6, 12}, {6, 17}, {7, 8}, {8, 14}, {9, 17}, {10, 14}, {10, 20}, {11, 14}, {12, 15}, {12, 17}, {12, 19}, {12, 20}, {13, 14}, {14, 19}, {15, 16}, {15, 17}, {16, 18}, {17, 19}, }

Max Clique Size for Brute-Force Algorithm: 3

Maximum Clique Vertices:

v5 v16 v18

Time taken for Brute Force Algorithm: 125 microseconds

Max Clique Size for Heuristic Algorithm: 2

Time taken for Heuristic Algorithm: 1245 microseconds

Maximum Clique Vertices:

v12 v17 v19

N = 20, Edge number: 100:

Graph Edges: { {1, 2}, {1, 3}, {1, 5}, {1, 10}, {1, 11}, {1, 12}, {1, 15}, {1, 16}, {1, 17}, {1, 18}, {1, 19}, {2, 4}, {2, 5}, {2, 7}, {2, 9}, {2, 10}, {2, 14}, {2, 15}, {2, 16}, {2, 17}, {2, 19}, {3, 5}, {3, 6}, {3, 7}, {3, 10}, {3, 12}, {3, 14}, {3, 15}, {3, 17}, {3, 20}, {4, 5}, {4, 7}, {4, 10}, {4, 11}, {4, 13}, {4, 14}, {4, 16}, {4, 20}, {5, 7}, {5, 8}, {5, 10}, {5, 12}, {5, 13}, {5, 15}, {5, 16}, {5, 20}, {6, 8}, {6, 9}, {6, 10}, {6, 16}, {6, 17}, {6, 20}, {7, 11}, {7, 12}, {7, 14}, {7, 17}, {7, 18}, {7, 19}, {8, 9}, {8, 10}, {8, 12}, {8, 14}, {8, 16}, {8, 17}, {8, 18}, {8, 19}, {9, 10}, {9, 13}, {9, 17}, {9, 19}, {9, 20}, {10, 13}, {10, 14}, {10, 15}, {10, 18}, {10, 19}, {10, 20}, {11, 14}, {11, 15}, {11, 16}, {11, 19}, {12, 16}, {13, 14}, {13, 15}, {13, 18}, {13, 20}, {14, 15}, {14, 16}, {14, 17}, {14, 18}, {14, 19}, {15, 16}, {15, 18}, {15, 19}, {15, 20}, {16, 18}, {16, 19}, {16, 20}, {17, 18}, {19, 20}, }

Max Clique Size for Brute-Force Algorithm: 5

Maximum Clique Vertices:

v1 v2 v5 v10 v15

Time taken for Brute Force Algorithm: 1208 microseconds

Max Clique Size for Heuristic Algorithm: 5

Time taken for Heuristic Algorithm: 2498 microseconds

Maximum Clique Vertices:

v2 v14 v15 v16 v19

N = 20, Edge number: 150:

Graph Edges: { {1, 2}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 11}, {1, 12}, {1, 13}, {1, 14}, {1, 15}, {1, 16}, {1, 17}, {1, 19}, {1, 20}, {2, 3}, {2, 6}, {2, 8}, {2, 9}, {2, 10}, {2, 12}, {2, 13}, {2, 14}, {2, 16}, {2, 17}, {2, 18}, {3, 4}, {3, 5}, {3, 6}, {3, 8}, {3, 9}, {3, 10},

{3, 11}, {3, 12}, {3, 14}, {3, 15}, {3, 16}, {3, 17}, {3, 18}, {3, 19}, {3, 20}, {4, 5}, {4, 6}, {4, 7}, {4, 8}, {4, 11}, {4, 12}, {4, 13}, {4, 15}, {4, 16}, {4, 17}, {4, 18}, {4, 19}, {4, 20}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {5, 12}, {5, 14}, {5, 15}, {5, 16}, {5, 17}, {5, 18}, {5, 19}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {6, 11}, {6, 12}, {6, 13}, {6, 14}, {6, 17}, {6, 18}, {6, 19}, {6, 20}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 12}, {7, 13}, {7, 14}, {7, 15}, {7, 17}, {7, 18}, {7, 19}, {7, 20}, {8, 9}, {8, 10}, {8, 11}, {8, 12}, {8, 13}, {8, 14}, {8, 15}, {8, 17}, {8, 18}, {8, 19}, {9, 10}, {9, 11}, {9, 12}, {9, 14}, {9, 18}, {9, 19}, {9, 20}, {10, 11}, {10, 12}, {10, 13}, {10, 14}, {10, 15}, {10, 16}, {10, 18}, {10, 19}, {10, 20}, {11, 12}, {11, 18}, {11, 19}, {11, 20}, {12, 13}, {12, 14}, {12, 15}, {12, 16}, {12, 17}, {12, 18}, {12, 19}, {12, 20}, {13, 14}, {13, 15}, {13, 17}, {13, 19}, {13, 20}, {14, 15}, {14, 16}, {14, 17}, {14, 18}, {14, 20}, {15, 16}, {15, 17}, {15, 20}, {16, 17}, {16, 18}, {16, 19}, {16, 20}, {17, 18}, {17, 19}, {17, 20}, {18, 19}, {18, 20}, {19, 20}, }
Max Clique Size for Brute-Force Algorithm: 9
Maximum Clique Vertices:
v1 v4 v6 v7 v8 v12 v13 v17 v19
Time taken for Brute Force Algorithm: 34137 microseconds

Max Clique Size for Heuristic Algorithm: 7
Time taken for Heuristic Algorithm: 5123 microseconds
Maximum Clique Vertices:
v3 v12 v16 v17 v18 v19 v20

White Box Testing:

White box testing tries to cover each line of the algorithms to see whether it handles each case accurately. Statement coverage will be used for white box testing, which will try go over each line with different inputs.

Brute Force:

maxCliques(i, l):

1. max_ = 0
2. for j = i + 1 to n:
3. store[l] = j

```
    if is_clique(l + 1):
        max_ = max(max_, l)
        max_ = max(max_, maxCliques(j, l + 1))
return max_
```

Heuristic:

MaxClique(G):

```
1: if (G is a clique)
2:      for each vertex of G: v
3:              if (|V|-1 > maxC[v])
4:                      maxC[v] := |V|;
5: else
6:      find the vertex of lowest degree: α
7:      find the largest subgraph of G in which α exists: G´ (V´,E´)
8:       MaxClique(G´ );
9:      if (V - α not empty)
10:             MaxClique(G – α);
```

Samples:

N = 1, covers [1,2, 3, 4, 7] for brute force, [1-11] for heuristic:

Graph Edges: { }

Max Clique Size for Brute-Force Algorithm: 1

Maximum Clique Vertices:

Time taken for Brute Force Algorithm: 3 microseconds

Max Clique Size for Heuristic Algorithm: 0

Time taken for Heuristic Algorithm: 18 microseconds

Maximum Clique Vertices:

N = 4, input graph that has no cliques [1,2, 3, 4, 7] for brute force and [1, 6-11] for heuristic:

Graph Edges: { }

Max Clique Size for Brute-Force Algorithm: 1

Maximum Clique Vertices:

Time taken for Brute Force Algorithm: 29 microseconds

Max Clique Size for Heuristic Algorithm: 0
Time taken for Heuristic Algorithm: 89 microseconds
Maximum Clique Vertices:

N = 4, edges: 6, input graph is already a clique, covers [1-7] for brute force and [1-5] for heuristic:
Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {2, 3}, {2, 4}, {3, 4}, }
Max Clique Size for Brute-Force Algorithm: 4
Maximum Clique Vertices:
v1 v2 v3 v4
Time taken for Brute Force Algorithm: 84 microseconds

Max Clique Size for Heuristic Algorithm: 4
Time taken for Heuristic Algorithm: 68 microseconds
Maximum Clique Vertices:
v1 v2 v3 v4

N = 15, edges: 105, input graph is already a clique, covers [1-7] for brute force and [1-5] for heuristic:
Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {1, 8}, {1, 9}, {1, 10}, {1, 11}, {1, 12}, {1, 13}, {1, 14}, {1, 15}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {2, 8}, {2, 9}, {2, 10}, {2, 11}, {2, 12}, {2, 13}, {2, 14}, {2, 15}, {3, 4}, {3, 5}, {3, 6}, {3, 7}, {3, 8}, {3, 9}, {3, 10}, {3, 11}, {3, 12}, {3, 13}, {3, 14}, {3, 15}, {4, 5}, {4, 6}, {4, 7}, {4, 8}, {4, 9}, {4, 10}, {4, 11}, {4, 12}, {4, 13}, {4, 14}, {4, 15}, {5, 6}, {5, 7}, {5, 8}, {5, 9}, {5, 10}, {5, 11}, {5, 12}, {5, 13}, {5, 14}, {5, 15}, {6, 7}, {6, 8}, {6, 9}, {6, 10}, {6, 11}, {6, 12}, {6, 13}, {6, 14}, {6, 15}, {7, 8}, {7, 9}, {7, 10}, {7, 11}, {7, 12}, {7, 13}, {7, 14}, {7, 15}, {8, 9}, {8, 10}, {8, 11}, {8, 12}, {8, 13}, {8, 14}, {8, 15}, {9, 10}, {9, 11}, {9, 12}, {9, 13}, {9, 14}, {9, 15}, {10, 11}, {10, 12}, {10, 13}, {10, 14}, {10, 15}, {11, 12}, {11, 13}, {11, 14}, {11, 15}, {12, 13}, {12, 14}, {12, 15}, {13, 14}, {13, 15}, {14, 15}, }
Max Clique Size for Brute-Force Algorithm: 15
Maximum Clique Vertices:
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15

Time taken for Brute Force Algorithm: 346495 microseconds

Max Clique Size for Heuristic Algorithm: 1998659584
Time taken for Heuristic Algorithm: 52 microseconds
Maximum Clique Vertices:
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15

N = 4, edges: 4, input graph that is not a clique at start, covers [1-7] for brute force and [1-11] for heuristic:
Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {2, 3}, }
Max Clique Size for Brute-Force Algorithm: 3
Maximum Clique Vertices:
v1 v2 v3
Time taken for Brute Force Algorithm: 60 microseconds

Max Clique Size for Heuristic Algorithm: 3
Time taken for Heuristic Algorithm: 95 microseconds
Maximum Clique Vertices:
v1 v2 v3

N = 15, edges: 50, input graph that is not a clique at start, covers [1-7] for brute force and [1-11] for heuristic:
Graph Edges: { {1, 2}, {1, 3}, {1, 4}, {1, 7}, {1, 8}, {1, 10}, {1, 11}, {2, 3}, {2, 5}, {2, 6}, {2, 12}, {2, 15}, {3, 8}, {3, 9}, {3, 10}, {3, 11}, {3, 12}, {3, 14}, {4, 6}, {4, 7}, {4, 8}, {4, 9}, {4, 10}, {4, 11}, {4, 13}, {4, 14}, {4, 15}, {5, 6}, {5, 7}, {5, 8}, {5, 13}, {5, 14}, {6, 7}, {6, 9}, {6, 13}, {7, 10}, {7, 11}, {7, 14}, {7, 15}, {8, 12}, {8, 14}, {8, 15}, {9, 12}, {9, 15}, {10, 11}, {11, 13}, {12, 14}, {12, 15}, {13, 14}, {14, 15}, }
Max Clique Size for Brute-Force Algorithm: 5
Maximum Clique Vertices:
v1 v4 v7 v10 v11
Time taken for Brute Force Algorithm: 3363 microseconds

Max Clique Size for Heuristic Algorithm: 5
Time taken for Heuristic Algorithm: 8218 microseconds

Maximum Clique Vertices:
v1 v4 v7 v10 v11

9. Discussion

In this report, two algorithm approaches for the clique problem, brute force and heuristic. The brute force algorithm works by calculating each possible instance for the given graph, therefore is more costly and slower. On the other hand, the heuristic algorithm is much faster but not guaranteed to give the correct answer every time since it makes greedy choices doesn't take each possibility into account unlike brute force.

This situation can be observed in the correctness analyses and the sample outputs as well. Their running times vary for the smaller inputs but as the input sizes grow, the heuristic algorithm gets noticeably faster. However, it is occasionally faulty at calculating the max clique, as it produces smaller cliques than the brute force's. Also, there are some coding errors in the heuristic one, based on the testing results. It gives a bigger clique when it shows the vertices but the max size is lower than the given clique. Also, the max size sometimes overflows.

The running time analysis showed a linear growth for both algorithms, even though the theoretical running times are much larger. However, this is probably due to the small size of the range of data. Though, from the slopes of the lines, it is possible to see that the brute force algorithm's growth is still much larger than the heuristic algorithm.

References:

- Richard M. Karp, *Reducibility among Combinatorial Problems*, In *Proceedings of a symposium on the Complexity of Computer Computations*, 1972, pp.85–103. doi: https://cgi.di.uoa.gr/~sgk/teaching/grad/handouts/karp.pdf
- https://www.geeksforgeeks.org/maximal-clique-problem-recursive-solution/
- Akbari, 2013, A Polynomial-Time Algorithm for the Maximum Clique Problem https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6607889