

**Project report: Hamming distance and Maximal clique**

**Course Name: CMP3005Analysis of Algorithms**

Bahcesehir University

2022/2023 Fall Semester

1st Jan 2023

Project done by:

**Omnia Elmenshawy 2000007**

**Rayyan Al-haj 2017741**

**Qatrlnada Almrzoq 1900678**

**Introduction**

* The **hamming graph** is the special type of graph with 2n vertices.
* The **hamming distance** is the number of positions at which corresponding symbols are different of 2 equal length strings. Hamming Distance is used when a comparison method is needed to examine two binary data strings. If the compared strings length are the same, then the hamming distance here is the number of bit place where the difference between the two bits occur.
* The hamming distance is generally used to detect errors and correct them when data transactions happen in computer networks.
* A clique is a subset of vertices in a graph such that every two distinct vertices in the subset are adjacent, that is, they are connected by an edge of the graph. A maximal clique is a clique that cannot be extended by adding an adjacent vertex, that is, it is not a subset of any other clique in the graph. The problem of finding the maximal cliques in a graph is known as the maximal clique problem.

In this report, we present an algorithm for finding the maximal clique of a set of binary numbers. The input to the algorithm is a distance threshold and the number of digits in the binary numbers. The output is the maximal clique of binary numbers that are within the specified distance from each other. We analyze the time and space complexity of the algorithm and discuss its various components.

**LIBRARIES THAT HAVE BEEN USED IN THIS PROJECT :**

<bits/stdc++. h> 🡪 it is a header file that includes all standard libraries

is an implementation file for a precompiled header

#include<iostream> 🡪 iostream is a header file that contains functions for

input/output operations ( cin and cout ).

#include 🡪 a preprocessor directive that tells the preprocessor to

include header

using namespace std; 🡪 make all the things under the std namespace available

without having to prefix std:: before each of them

**Algorithm**

The maximal clique algorithm consists of two main functions: ***hamming\_distance*** and ***maximal\_clique.***

1. **The hamming\_distance Function:**

The ***hamming\_distance*** function calculates the hamming distance between two binary numbers. The hamming distance between two binary numbers is the number of positions in which the two numbers differ.

The function takes two strings a and b as input and returns an integer representing the hamming distance between the two numbers. It does this by iterating over the characters in the two strings and incrementing a counter variable distance whenever the characters at the same position differ.

The time complexity of the ***hamming\_distance*** function is O(n), where n is the length of the strings a and b. This is because the function performs a constant number of operations (incrementing the counter variable) for each character in the strings.

The space complexity of the ***hamming\_distance*** function is O(1), since it only uses a constant amount of memory (the counter variable distance) regardless of the size of the input.

1. **The *maximal\_clique* function:**

It calculates the maximal clique victor and the size of it based on the required problem.

**Analysis**

1. **Time complexity**

The maximal clique algorithm has a time complexity of O(n^2) and a space complexity of O(n), where n is the number of possible binary numbers with the specified number of digits. This is due to the nested loop in the ***maximal\_clique*** function, which generates all possible binary numbers and then checks the distance of each number from all other numbers in the clique.

The time complexity of the algorithm can be improved by using a more efficient method for calculating the hamming distance between two binary numbers. For example, instead of iterating over the characters in the two strings and incrementing a counter variable, we could use the built-in popcount function in C++ to calculate the number of set bits in the bitwise XOR of the two numbers. This would reduce the time complexity of the ***hamming\_distance*** function to O(1), which would in turn reduce the time complexity of the ***maximal\_clique*** function to O(n).

1. **Space complexity**

The space complexity of the algorithm can also be improved by using a more efficient data structure to store the maximal clique. For example, instead of using a vector, we could use a set or a hash table, which would reduce the space complexity to O(m), where m is the size of the maximal clique.

**Sample Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Conclusion**

In this report, we presented an algorithm for finding the maximal clique of a set of binary numbers. We analyzed the time and space complexity of the algorithm and discussed its various components. We also identified potential improvements to the algorithm to reduce its time and space complexity.

**Resources:**

* Gross, J. L., & Yellen, J. (2006). Handbook of Graph Theory (2nd ed.). Boca Raton, FL: Chapman & Hall/CRC.
* Skiena, S. S. (1998). The Algorithm Design Manual (1st ed.). New York, NY: Springer-Verlag.
* Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). Cambridge, MA: MIT Press.
* Knuth, D. E. (1998). The Art of Computer Programming, Volume 3: Sorting and Searching (2nd ed.). Reading, MA: Addison-Wesley.
* Sedgewick, R. (2011). Algorithms (4th ed.). Boston, MA: Addison-Wesley.