CSC 448- Artificial Intelligence

Professor: Yunhua Zhao

Erlin Paredes

Empl ID: 23750371

11/18/2023

Abstract:

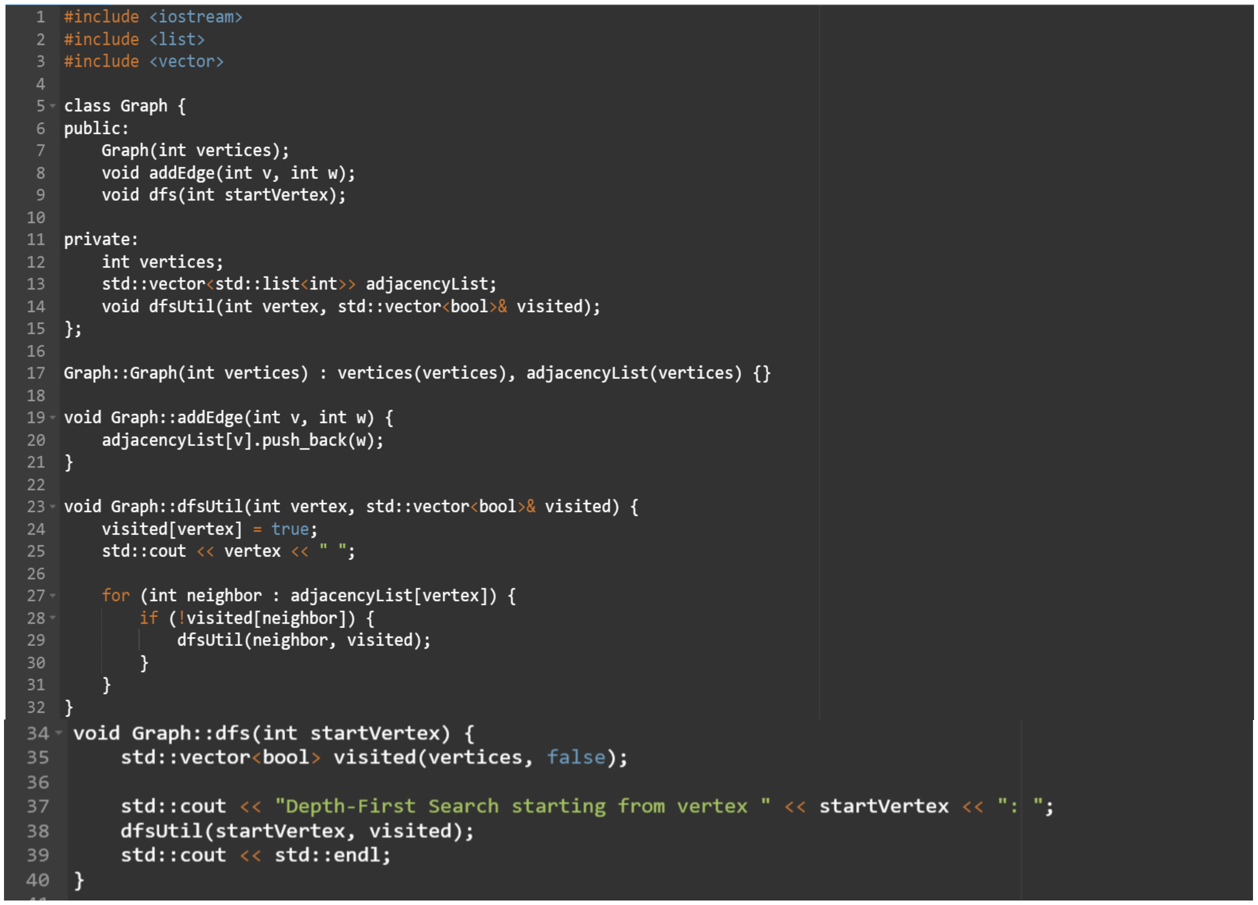
In this research paper, I will be describing the different concepts behind the development of artificial intelligence. Among these are Depth first search, breadth first search, blind search, heuristic algorithm, neural network and the n-queens problem. Each one of them has a key foundation in the formation of artificial intelligence which has substantially improved our society throughout time. Most of our society consists of devices that are mainly controlled by an AI software. Many people still do not know how artificial intelligence began and why it started so it is very important in this research paper to explain in detail. Throughout the years, different awards have been given to different scientists for these discoveries and one prestigious award is the Loebner Prize. Throughout this document I will be describing what the award consists about and how important it is to have one. I will also mention the different people that have won the award and why they got to win such accomplishment.

Depth first search:

The Depth First Search (DFS) algorithm is compared to exploring a maze by a lot of people. A maze usually has a path that you need to follow in order to solve it. If you follow depth first search ideology you would be exploring all possible paths at once, you choose one path and keep going until you cannot go any further. If in case you cannot find an exit then you backtrack and try another path. Depth First Search works in a similar way when solving problems in computer science. It starts at a specific point and explores as far as possible along each branch before backtracking. This type of algorithm is similar to a detective following leads one by one until the solution is found or all possibilities are exhausted. This practical and a simple algorithm is used in various applications, such as finding a path in a maze or searching for specific information in a database.

Depth-First Search (DFS) time complexity is mainly based on terms of the number of vertices (nodes) and edges visited in the graph. In the worst-case scenario, where every edge and vertex must be visited, DFS has a temporal complexity of O (V + E), where V is the number of vertices in the graph and E is the number of edges. This is due to the fact that, in the worst-case situation, DFS may need to search all vertices and edges in order to locate the needed information or reach a certain node. It's crucial to note, however, that the actual time complexity might vary based on things like the data structure used to represent the graph and the specifics of the problem being solved.

Code Example:



```cpp
#include <iostream>
#include <list>
#include <vector>

class Graph {
public:
    Graph(int vertices);
    void addEdge(int v, int w);
    void dfs(int startVertex);

private:
    int vertices;
    std::vector<std::list<int>> adjacencyList;
    void dfsUtil(int vertex, std::vector<bool>& visited);
};

Graph::Graph(int vertices) : vertices(vertices), adjacencyList(vertices) {}

void Graph::addEdge(int v, int w) {
    adjacencyList[v].push_back(w);
}

void Graph::dfsUtil(int vertex, std::vector<bool>& visited) {
    visited[vertex] = true;
    std::cout << vertex << " ";

    for (int neighbor : adjacencyList[vertex]) {
        if (!visited[neighbor]) {
            dfsUtil(neighbor, visited);
        }
    }
}
void Graph::dfs(int startVertex) {
    std::vector<bool> visited(vertices, false);

    std::cout << "Depth-First Search starting from vertex " << startVertex << ": ";
    dfsUtil(startVertex, visited);
    std::cout << std::endl;
}
```

Figure 1: Depth First search

In this code, the Graph Class represents an undirected graph. Constructor (`Graph (int vertices)) basically Initializes the graph with a given number of vertices. Meanwhile, AddEdge(int v, int w) adds an edge between vertices v and w. Dfs(int startVertex) Initiates DFS traversal starting from the specified vertex.

Private Members:

int vertices: Stores the number of vertices in the graph.

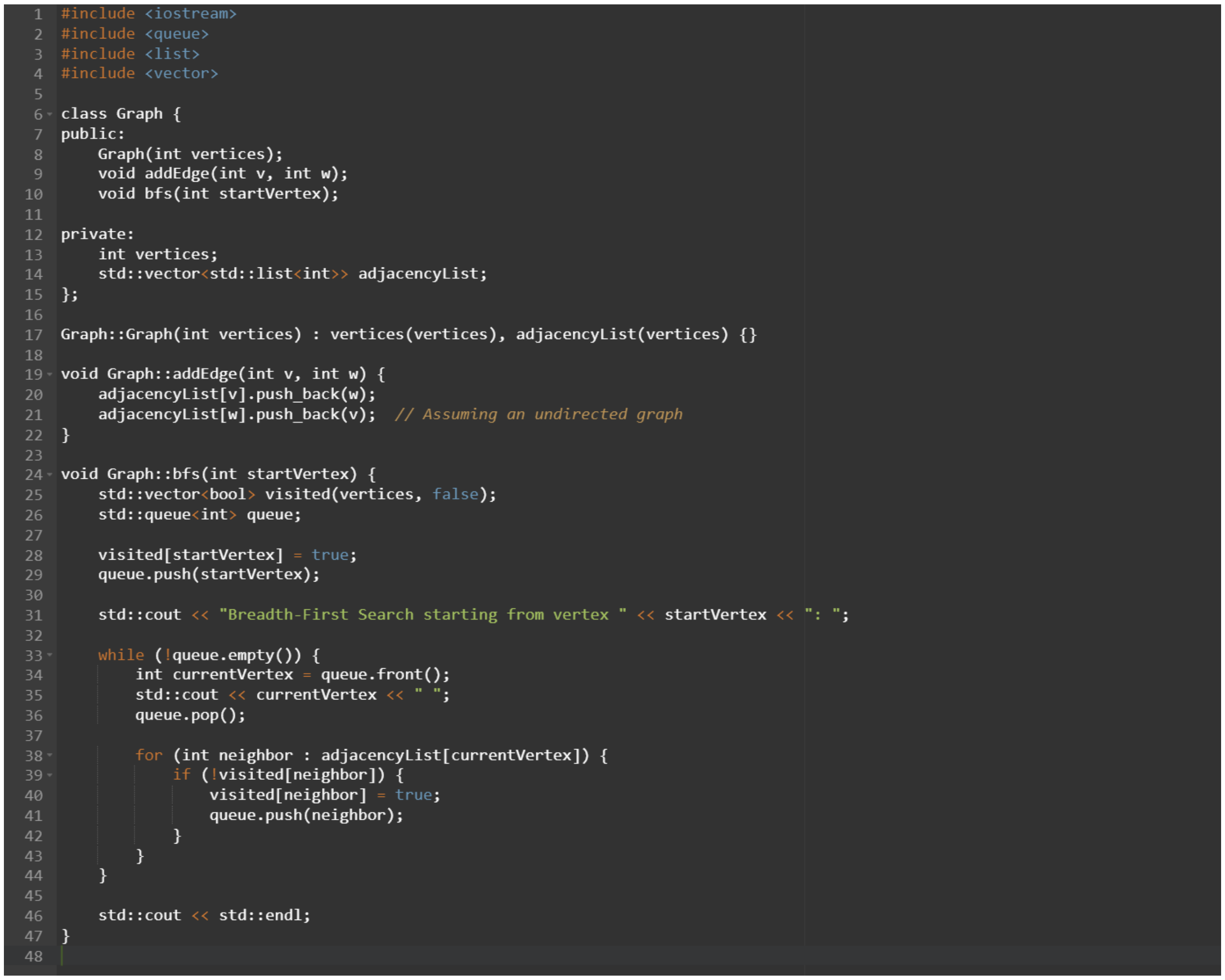
std: vector<std: list<int>> adjacencyList: Represents the graph using an adjacency list. Each element of the vector corresponds to a vertex, and the associated list contains the neighbors of that vertex.

dfsUtil(int vertex, std::vector<bool>& visited) is an utility function used by the DFS traversal.

Breadth-First Search:

Breadth-First Search basically consists of traversing a graph level by level, beginning at a selected source vertex. In order words, it visits every particular node in a methodical manner before going on to their neighbors. It can be compared to waves going outward and layer by layer investigating nodes. The sequence in which nodes are visited is maintained by BFS using a queue data structure, which makes sure that nodes that are nearby are examined before moving on to those that are farther away. This algorithm is helpful for tasks like network analysis, social network algorithms and puzzle solving since it can be used to discover the shortest path between two nodes in an unweighted graph. Usually, BFS has an O (V + E) temporal complexity, where V is the number of vertices and E is the number of nodes. A cool fact about BFS is that a German civil engineer and computer pioneer Konrad Zuse first presented the Breadth-First

Search (BFS) method in 1945. The Z3, which Zuse finished in 1941, is frequently regarded with being the first programmable digital computer in history. BFS has changed throughout time and has been separately found and improved upon by several researchers. Because of the algorithm's ease of use and efficiency in exploring and analyzing graphs, it became widely known and adopted.



```cpp
#include <iostream>
#include <queue>
#include <list>
#include <vector>

class Graph {
public:
    Graph(int vertices);
    void addEdge(int v, int w);
    void bfs(int startVertex);

private:
    int vertices;
    std::vector<std::list<int>> adjacencyList;
};

Graph::Graph(int vertices) : vertices(vertices), adjacencyList(vertices) {}

void Graph::addEdge(int v, int w) {
    adjacencyList[v].push_back(w);
    adjacencyList[w].push_back(v);  // Assuming an undirected graph
}

void Graph::bfs(int startVertex) {
    std::vector<bool> visited(vertices, false);
    std::queue<int> queue;

    visited[startVertex] = true;
    queue.push(startVertex);

    std::cout << "Breadth-First Search starting from vertex " << startVertex << ": ";

    while (!queue.empty()) {
        int currentVertex = queue.front();
        std::cout << currentVertex << " ";
        queue.pop();

        for (int neighbor : adjacencyList[currentVertex]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                queue.push(neighbor);
            }
        }
    }

    std::cout << std::endl;
}
```

Figure 2: Breadth first search

This is the basic code for breadth first search in c++

Blind Search

The "blind search algorithm" basically describes a class of search algorithms that do not use any domain-related information to direct the search. These algorithms usually search without taking the nature of the problem to be addressed into account. Uninformed search algorithms are another name for blind search algorithms. The Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms are two of the most widely used blind search techniques. These algorithms do not use any information about the objective or the problem's structure; instead, they methodically scan a search space. When there is little to no information available regarding the issue and a thorough search is required, blind search algorithms are frequently used. Even though blind search algorithms are easy to use and may be used to a variety of situations, they might not always be the most effective in terms of time and space complexity, particularly for vast and intricate search environments. Compared to heuristic algorithms, blind search algorithms usually are very distinct.

Heuristic:

A heuristic algorithm is like a smart, quick-thinking strategy used to solve problems. Instead of trying every possible solution, a heuristic takes a shortcut based on practical rules or common sense to find a good or satisfactory answer. This type of algorithm is very different from the blind search algorithms and they are considered more effective in a way because they waste less time. The number of trials to look for a solution are less. Another way to see this algorithm is as a way of making educated guesses and decisions without having to check every single option. While it might not guarantee the absolute best solution, heuristics are handy because they save effort, especially when dealing with complex problems where exploring every possibility is not feasible. Heuristic algorithms have been used for a long time, and their history can be traced back to various fields.

Loebner Prize:

An annual competition in artificial intelligence (AI) called the Loebner Prize is centered around the Turing test, which was first proposed in 1950 by British mathematician and computer scientist Alan Turing. The Turing test quantifies a machine's capacity to behave intelligently in a way that cannot be distinguished from human behavior. To promote developments in conversational AI, American philanthropist and businessman Hugh Loebner founded the Loebner Prize in 1990. Under the format of a chatbot competition, computer programs, or chatbots, try to converse with human judges, who don't know if they are speaking with a machine or a person. The Turing test is passed by a chatbot if a sizable percentage of the judges believe it to be human. Over the years, the competition has had several format and regulation changes. It was limited to text-based interactions at first, but it eventually grew to include multimedia elements. The nature of artificial intelligence, language comprehension, and the difficulties in building robots that can converse naturally and human-like have all been topics of discussion and debate in the wake of the Loebner Prize. It is important to note that there are proponents and opponents of the Loebner Prize. While some believe it's a useful tool for advancing AI, others counter that the Turing test as it was used in the competition might not be the most accurate or rigorous indicator of actual machine intelligence.

Neural network:

A neural network is a type of computer model that is based on the biological neural networks seen in the human brain. Pattern recognition, classification, regression, and decision-making are among the activities it is used for in machine learning and artificial intelligence. Neural networks are made up of linked nodes arranged in layers; these nodes are also called neurons or units. Every neuron takes in information, uses an activation function to process it, and then outputs something. An input layer, one or more hidden layers, and an output layer make up the layers that make up neurons. Weights are a representation of the connections between neurons. The strength of a connection is determined by the weight assigned to it. The network is able to learn patterns that do not travel via the origin by adding a bias component to the weighted sum of inputs. A neuron's output is determined by its input, which is defined by its activation function. By adding non-linearity, it makes the network capable of learning intricate patterns. The sigmoid, rectified linear unit (ReLU), and hyperbolic tangent (tanh) are examples of common activation functions. The loss function measures the difference between the predicted output and the actual target values. At the end an optimization algorithm is used to minimize the loss function by adjusting the weights and biases. Neural networks can be either basic or complicated; the phrase "deep learning" refers to neural networks with numerous hidden layers. Neural networks are used in a wide range of applications today such as Speech Recognition, Natural Language Processing (NLP), Recommendation Systems, Healthcare and finance.

N-queens:

The N-Queens problem is a traditional chess puzzle that requires the player to arrange N queens on a N×N chessboard so that no two queens are in danger of one another. Since a queen in chess can attack in three different directions—horizontally, vertically, and diagonally—the objective is to position the queens so that no two of them occupy the same row, column, or diagonal. This is the best illustration of a constraint satisfaction problem which finds use in computer science, artificial intelligence, and optimization, among other fields. The N-Queens problem is a benchmark for evaluating combinatorial problem-solving algorithms; as N increases, it gets harder and harder to solve. It is often solved by applying backtracking methods. Backtracking is the process of methodically experimenting with various configurations and reversing decisions made when they result in an instance of constraints being broken. For big N values, it would be impracticable to explore every conceivable combination in order to discover a solution. Efficient algorithms, like the backtracking approach, can find solutions for the N-Queens problem, but it remains a computationally complex problem, especially for large values of N. Another important detail about the N-queens problem is that it was not initially explored by a single scientist and has no single origin. It's a well-known mathematical puzzle that many mathematicians and computer scientists have examined and debated over time. The issue has existed since at least the 19th century, and several researchers have contributed to its history. A lot of people were not interested on doing research for this type of problem because it is time consuming. the N-Queens problem gained more attention in the 20th century, especially with the advent of computer science and algorithmic approaches to problem-solving. It became a well-known problem in the field of combinatorics and constraint satisfaction.

Conclusion:

In conclusion, given how quickly technology is advancing, artificial intelligence (AI) is becoming increasingly important. AI has the revolutionary potential to completely change a number of industries, including healthcare, banking, education, and more. Its capacity to sort through massive volumes of data, spot trends, and come to wise judgments not only boosts productivity but also creates new avenues for creativity. Artificial intelligence (AI) has the power to solve difficult problems, enhance decision-making, and advance scientific discoveries. Furthermore, AI has the power to enhance human abilities, resulting in breakthroughs that could benefit society as a whole. Adopting and ethically growing AI will probably be crucial to creating a more intelligent, efficient, and networked environment as we navigate the future. As we can see the N queens among other algorithms.

References:

"N Queen Problem | Backtracking-3." GeeksforGeeks, 21 July 2011, www.geeksforgeeks.org/n-queen-problem-backtracking-3/.

IBM. "What Are Neural Networks? | IBM." Www.ibm.com, www.ibm.com/topics/neural-networks.

"Loebner Prize | Computer Science | University of Exeter." Loebner.exeter.ac.uk, loebner.exeter.ac.uk/.

"Heuristic Algorithms - Cornell University Computational Optimization Open Textbook Optimization Wiki." Optimization.cbe.cornell.edu, optimization.cbe.cornell.edu/index.php?title=Heuristic_algorithms.

Programiz. "DFS Algorithm for Graph (with Pseudocode, Example and Code in C, C++, Java, Python)." Programiz.com, 2019, www.programiz.com/dsa/graph-dfs.