Understanding Algorithms: Visualize Them In Action

Madhav (12028)
Siddharth Gupta (12056)

Mentor: Prof. Shobha Bagai
Cluster Innovation Centre
University of Delhi

Abstract

In this report, an e-Learning tool for A* Pathfinder, Bubble Sort,Insertion Sort, N Queens, Knight's tour,Prim's visualization is described.For example, In sorting the animation tool would represent information as a bar and once choosing a data-ordering and algorithms, the user will run an automatic animation or step through it at their own pace. In path finding making the starting and the end node be able to move around or the user to choose wherever he wants it to start or end.

Introduction

Algorithm Visualizer is a dynamic graphic tool that allows the visualization of computation for a given algorithmic program. This study aims to design and implement a system of algorithmic visualization that shows the abstract algorithmic idea behind a certain computing method. The goal is to help students understand algorithms effectively by visualizing the run time for each implemented algorithm.

Data Structures and Algorithms

Data structures refer to the way that data is organized and stored within a computer program. A data structure defines a way of representing and manipulating data in a way that makes it easy to access and process. Examples of common data structures include arrays, linked lists, trees, graphs, and hash tables.
Algorithms, on the other hand, are sets of instructions that are designed to perform specific tasks or solve specific problems. Algorithms are typically developed using one or more data structures as the underlying framework for organizing and processing data. Examples of common algorithms include sorting algorithms, search algorithms, and graph traversal algorithms.

Backtracking Algorithm

Backtracking algorithm is a systematic way of searching for a solution to a problem that incrementally builds a partial solution and backtracks to previous steps when it determines that the current solution cannot be completed successfully. It is commonly used in solving combinatorial problems such as generating permutations, subsets, and graphs. Backtracking algorithms are powerful tools in optimization problems, constraint satisfaction, and artificial intelligence. They are particularly useful in solving problems where there is no known efficient algorithm. However, backtracking algorithms can be computationally expensive, especially when dealing with large search spaces.

A* Pathfinding

A* Pathfinding Algorithm is a search algorithm used to find the shortest path between two nodes in a graph or a grid. It is a heuristic-based algorithm that uses a combination of the actual cost to reach a node from the starting node and an estimate of the cost to reach the destination node. This estimate is calculated using a heuristic function, such as the Euclidean distance or Manhattan distance. The algorithm explores the graph or grid by selecting the node with the lowest f-value, which is the sum of the actual cost and the estimated cost. A* algorithm is optimal and complete, meaning it always finds the shortest path if one exists and terminates if no path exists. It is widely used in robotics, video games, and map routing applications. A* algorithm has a time complexity of O(b^d), where b is the branching factor and d is the depth of the shallowest solution. However, the use of a heuristic function can significantly reduce the search space and improve the algorithm's efficiency.

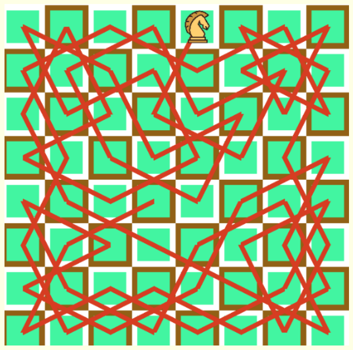
Sorting Algorithms

Sorting algorithms are algorithms used to put a collection of data elements in a specific order. There are numerous types of sorting algorithms, each with its unique advantages and disadvantages. Some popular sorting algorithms are bubble sort, insertion sort, selection sort, merge sort, quicksort, heapsort, and radix sort. The efficiency of a sorting algorithm is measured by its time complexity, which is usually denoted as O(n log n) or O(n^2), where n is the number of elements to be sorted. Sorting algorithms have numerous applications in computer science, such as in databases, search algorithms, data compression, and data analysis. The choice of sorting algorithm depends on the specific use case, the size of the data set, and the desired time complexity.

Prim's Algorithm

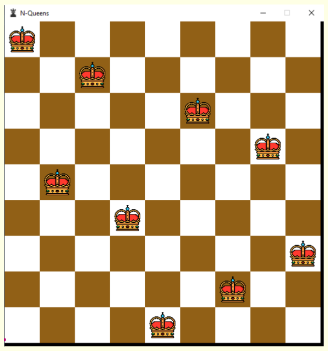
Prim's algorithm is a greedy algorithm used to find the minimum spanning tree of a weighted undirected graph. It starts with an arbitrary vertex and adds the minimum weight edge to connect it to the rest of the graph. It then adds the minimum weight edge from the existing tree to a new vertex until all vertices are connected. The algorithm guarantees that the tree is always connected and has the minimum weight possible. Prim's algorithm has a time complexity of O(E log V), where E is the number of edges and V is the number of vertices in the graph. It is widely used in network design, cluster analysis, and computer networks.

Algorithm Visualisation



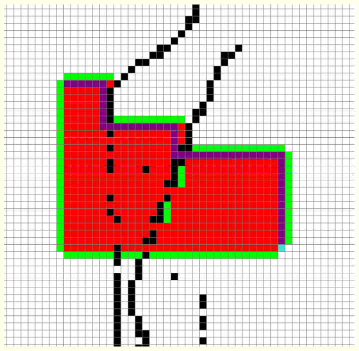
Knight's Tower

A knight's tour is a sequence of moves of a knight on a chessboard such that the knight visits every square exactly once



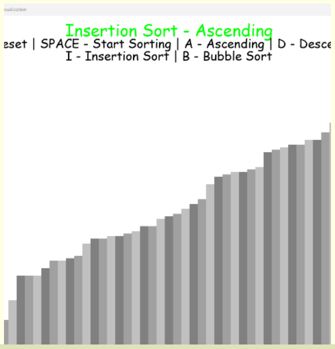
N Queens

The visualizer allows users to specify the size of the chessboard and the speed of the execution. The solution is shown on a graphical chessboard, where each queen is represented by an image.



A* Pathfinding

It allows the user to create a grid of nodes, designate start and end points, and customise the grid by creating barriers. It allows you to find shortest distance between start and end points.

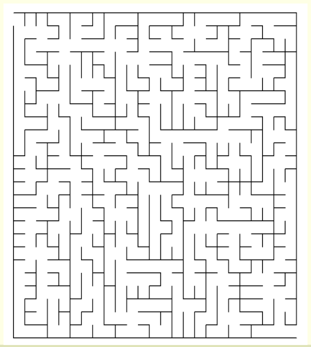

Insertion Sort - Ascending
eset | SPACE - Start Sorting | A - Ascending | D - Desce
I - Insertion Sort | B - Bubble Sort

Sorting

The program generates a random list of numbers, displays it on the screen, and animates the sorting process using two popular sorting algorithms. Bubble Sort and Insertion Sort

Prim's Algorithm

The aim of this program is to find a subset of edges in a connected, weighted graph that connects all the vertices without any cycles and with the minimum possible total edge weight.



Conclusion

An algorithm visualization project can be a powerful tool for teaching and understanding algorithms. By creating visual representations of algorithms, it becomes easier to see how they work and understand their behaviour. It's helps us students to better understand the logic behind algorithms and how they can be used to solve problems. To create an effective algorithm visualization project, it is important to have a clear understanding of the algorithm you are trying to visualize. The visualization should be clear, concise, and easy to understand. Finally, it is important to test your algorithm visualization project with users to ensure that it is effective in achieving its goals. User feedback can be used to refine the visualization and make it more effective for future users.