Bachelor of Science in Computer Science & Engineering

Developing an Android Application for Visualizing Different Artificial Intelligence Algorithms

by

S. M. Abdus Salam

ID: 1504030

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

May, 2021

Developing an Android Application for Visualizing Different Artificial Intelligence Algorithms

Submitted in partial fulfilment of the requirements for
Degree of Bachelor of Science
in Computer Science & Engineering

by

S. M. Abdus Salam
ID: 1504030

Supervised by

Dr. Mohammad Shamsul Arefin
Professor
Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)
Chattogram-4349, Bangladesh.

The thesis titled '**Developing an Android Application for Visualizing Different Artificial Intelligence Algorithms'** submitted by ID: 1504030, Session 2019-2020 has been accepted as satisfactory in fulfilment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

Declaration of Originality

This is to guarantee that I am the sole author of this thesis and that no part of it, nor the whole thesis, has been submitted to another institution for a degree.

To the best of my understanding, my research does not contravene on anyone's copyright or breach any proprietary rights, and that any inventions, methods, samples, or other content from other people's work used in my thesis, whether published or not, is completely accepted in compliance with normal referencing practices. I'm also aware that if any copyright infringement is discovered, whether deliberate or not, I will face legal and disciplinary action from Dept. of CSE,CUET.

I hereby assign every rights in the copyright of this thesis work to Dept. of CSE, CUET, who shall be the owner of the copyright of this work and any reproduction or use in any form or by any means whatsoever is prohibited without the consent of Dept. of CSE, CUET.

Signature of the candidate

Date:

Acknowledgements

The progress and result of this thesis involved a great deal of support and assistance from many people, and I consider myself incredibly fortunate to have received it all the way through my thesis. Professionally and socially, it has been a rewarding experience. Much of what I've accomplished has been possible only because of such oversight and assistance. First of all, I am very thankful to almighty Allah for allowing me to complete this thesis successfully. Thereafter, I'd like to express my heartfelt appreciation to my respected supervisor Dr.Mohammad Shamsul Arefin, Professor, Department of Computer Science & Engineering, Chittagong University of Engineering & Technology (CUET) for his guidance, encouragement, and continuous support during my thesis work. I acknowledge his many critical questions, which forced me to consider things from various angles, as well as his unwavering support during the process..

Finally, I want to express my gratitude to my father and mother for their unwavering love, guidance, motivation, and contributions in every aspect of my life and academic career over the years..

Abstract

In this project, we have considered a framework that covers almost all important and well known classes of artificial intelligence algorithms. Artificial intelligence is the theory of changing our world and powering the 4th industrial revolution. It's creating dominance in every spheres of our day-to-day life. We rely on artificial intelligence much more because it enhances our speed, precision and efforts of work. So we need to know the basic principles of algorithms that runs the whole system of artificial intelligence. Data visualization, animations, simulation techniques are really helpful to get out the best knowledge of algorithms. So we developed an application where we can get the basic working principles of algorithms and the application will provide a new dimension to get an effective way of learning, which will be more constructive than the traditional educational system.

Keywords— Artificial Intelligence; well known; dominance; Data visualization; dimension; traditional

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

Artificial Intelligence is a process that provides human brain mechanism into machines. We are using A.I to serve a greater purpose of our daily life. So it is very important to be familiar with artificial intelligence and should bear a good knowledge how it works. So to know the working principle , we need to know it's algorithms that we are using more frequently. Visualization technology plays a vital role to be familiar with algorithms because it graphically illustrates how algorithms work. A well established visualization technique can easily helps us to get a better result. Despite its intuitive appeal, the technology has yet to gain traction in mainstream computer science education. [1]. For computer science educators, algorithm visualization is an important pedagogical method for students to better understand algorithms [2]. Algorithm visualization (AV) software has advanced quicker than our understanding of how such technology encourages student learning [3].

1.2 Framework Overview

Considering these facts, we developed a framework that provides the animations of visualizing different artificial intelligence algorithms. In our proposed system we have tried to visualize important classes which are using regularly in our work. Mainly graphical presentation has Java's influence and HTML5 and JavaScript content are increasingly being used online, putting visualizations, interactive exercises and due to technical constraints, combining information and material on a wide scale [4]. So we consider scenes for every classes of algorithms. The scenes will visualize the algorithms according the working principle of that particular class. We have used unity for front end development and visual studio for back end development. An algorithm is represented as a series of graphical snapshots of data structures that are worked on at "interesting events" during the algorithm's execution in this way [5].

1.3 Difficulties

The major difficulties for developing this framework are given below:

- If we set parameters according to our choice then the codes runs smoothly and output visualization is very clear. But making the system dynamic, it creates a lot of errors.
- It is very important to make the framework interesting with well animated system, which is quite challenging.
- There are hundreds of problem scenarios for every class. All the manual or graphical problems are hard to implement on our system.

1.4 Applications

Present education system needs a lot of changes to make sure that learning platforms must be well organized and attractive to the learners. [6]. So there is no alternate of visualization. Learning something with an animated view always makes the system more interesting. Some of the important and effective applications are mentioned below:

- Visualization of artificial intelligence algorithms makes the system more easier to understand than theoretical knowledge.
- Visualization creates a platform where one can learn the algorithms and implement them in related works.
- Pedagogical algorithm visualization system produce graphical representations that aim to assist learners in understanding the dynamic behavior of A.I algorithms

- This framework will encourage researchers or developers to develop a framework with more exclusive animated items.

1.5 Motivation

A.I serves a greater purpose for the development of mankind. The basic function of this system is it's algorithms. Algorithms are the skeleton of A.I. Many works are producing new dimensions for learning these algorithms everyday. We also tried to contribute on this platform by visualizing them. Our system provides a good knowledge for this. But during this development period we have to face some challenges:

- to built up large animation scenes for different classes.
- to run them smoothly and errors must be strictly prohibited.
- compilation time must be keep short to make the system fluent.

1.6 Contribution of the thesis

The major contributions of this article are listed as follows:

- We develop an application which helps a learner to understand basic principles of well known A.I algorithms.
- We developed a system which is more effective than the traditional text book knowledge.
- We developed a framework which is user friendly.
- We can also implement this framework in iOS and p.c
- Under a same platform we organized to visualize and describe all well known classes of A.I algorithms which is rare.
- We performed subjective and objective evaluations of our framework.

1.7 Thesis Organization

The rest of the paper is organized as follows.

- In Chapter 2 a brief discussion on the previous work related to "visualizing different artificial intelligence algorithms" is provided.
- In Chapter 3 we provide our visualizing classes of different artificial algorithms, methodology- in which we actually describe our working principles step by step.
- In Chapter 4 the experimental results and performance evaluations are presented.
- Finally, in Chapter 5 we conclude the paper and outline future research directions.

1.8 Conclusion

The summary of our work has been addressed in this chapter. Our work's challenges, applications, and inspiration have all been addressed in this chapter. We have listed our work's contributions. This chapter summarizes our efforts.

Chapter 2

Literature Review

2.1 Introduction

This section describes the types of works that have been done in the past that are close to ours. We looked at some articles that were very close to our own. Despite the fact that they have a high success rate on such projects, they do have limitations. In this part, we'll talk about these issues and try to come up with the best solutions.

2.2 Related Literature Review

Algorithms are at the heart of all non-trivial programming applications. In order to work more effectively and precisely, there is no alternative of learning basics of algorithms. The main goal of algorithm visualization is to refine and improve our skills of how algorithms work. There has been a lot of research and development in the area of algorithm visualization over the last few decades. There are several different types of solutions available today, each of which serves a different function, but it is also difficult to modify or extend a given solution to meet particular needs.

In [1] it has been considered that the algorithm visualization technique as an effective sector of learning. Independent variables are used to relate each study to a particular guiding theory of learning to see which one has the best forecasting success, and also used dependent variables to see the performance indicators are more stable and considerable of learning the benefits of algorithm visualizing development.

Abu-Naser [7] described a framework which was developed to visualize searching artificial algorithms. Three methods used here- single step mode, all steps mode and back step mode. This provides a good basic terminology for our work. They also have some limitations like- we cannot go back if we are forwarding the graph. In our proposed work, we have tried to overcome these obstacles.

A framework is produced [8] where researchers examine how software and algorithm visualizations have been used successfully in the classroom. Here we can see at first, they looked at 18 visualization structures and rated them in 33 different ways. They found that only the reliability of half of the systems had been tested, and that these tests were only superficial. Remaining learners were executed based on their educational effectiveness. viewing, responding, and changing engagement levels appears to be best suited for script-based systems, while constructing and presenting engagement levels appears to be best suited for compiler-based systems. We both envision animations for educational purposes, so the work is very close to ours.

A new tool [9] is proposed called Visual LinProg. It is a web-based educational platform that uses animation and visualization techniques to solve linear programming problems. Visual LinProg was created to complement the teaching of linear programming courses. This assessment concludes that Visual LinProg aids in the learning of the revised simplex algorithm, and then goes on to present additional findings on factors affecting students'/users' comprehension of this algorithm. They have some sort of same methodology but only web based system we think it is not up to the mark as mobile phones are cheaper than computers. So we developed to do this for both pc and android platform.

Wu, Yang, Shu [10] have performed a survey on the basis of data visualization by artificial intelligence approach. In this framework, ten different factors are discussed on how,why and when to apply artificial intelligence for visualizing data. The main theme is to discuss about the importance of data visualization by artificial intelligence in the field technology and information. Actually A.I plays a vital role in every spheres of sectors.

A framework [11] which is produced to analyse big datas and visualize them by using artificial intelligence. They have tried to provide a common ground for exchanging information as there wasn't any reference model for larger initiatives or partnership environments that made it impossible for researchers and large corporations to invent the wheel. So they use the A.I technology to produce a good model that can overcome from this obstacle. Actually by the increasing demand of A.I we tried to develop our framework to deliver a good knowledge and platform for learning these A.I algorithms.

2.3 Conclusion

In this chapter, a detailed literature review is discussed. We have given a brief description of previous work related to visualize algorithms on different techniques and developing a good number of tools which is very beneficial for our educational system. We believe learners should learn with joy and pleasure.

Chapter 3

Methodology

3.1 Introduction

The proposed system architecture consists of four modules. They are the starting module, input module, working module and output module. First we have to enter the app and where the selection module will produce the classes of algorithm. Entering the selected class if inputs are needed then it will enter the input module. Most important part is the working module where the scenes are ready to be triggered according to the needs of users. After running the simulation we will get our result in output module. The program was written in visual studio and the animations are created in unity.

3.2 System Architecture

We first make the scenes for different algorithms visualization in lucid chart. After completing the scenes we did our coding works in visual studio, which works collaboratively with unity-hub.

In the input module, when we selects the classes for visualization, the selected individual scenes are triggered and collaborated with unity-hub. Then the project simulates the animations and we get the results in output module.

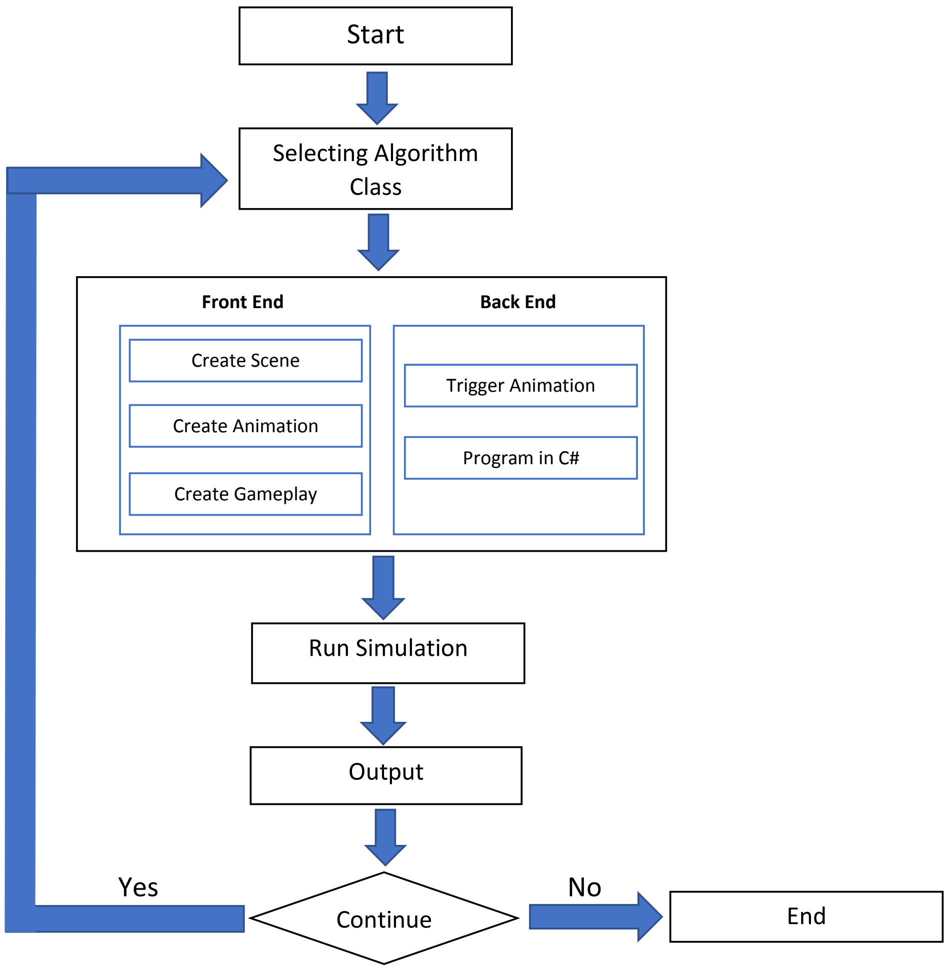


Figure 3.1: System architecture for visualizing A.I algorithms

3.3 Overview of Framework

As we mentioned in the introduction of our methodology, the framework works in four module. Each modules have their individual tasks. So if we summarize them, we can see the overview of the work, is given below:

- The scenes are created first in in lucid chart
- Then we have initialized the coding section in C#
- Then the scenes are triggered in unity, which works collaboratively with visual studio.

3.4 Detailed Explanation

3.4.1 Starting Module

The starting module means the starting steps of the application. That means whenever we enter into the app the first preview of the app is called starting module. In the starting module, there will be some tabs or buttons which indicates the next step (selecting algorithm classes).

3.4.2 Selecting Algorithm Classes

Artificial intelligence (AI) is intelligence demonstrated by computers as opposed to natural intelligence demonstrated by humans and animals, which includes consciousness and emotion. There are many classes of artificial intelligence algorithms. In this work we have suggested about 24 types of algorithms, the chart is given below-

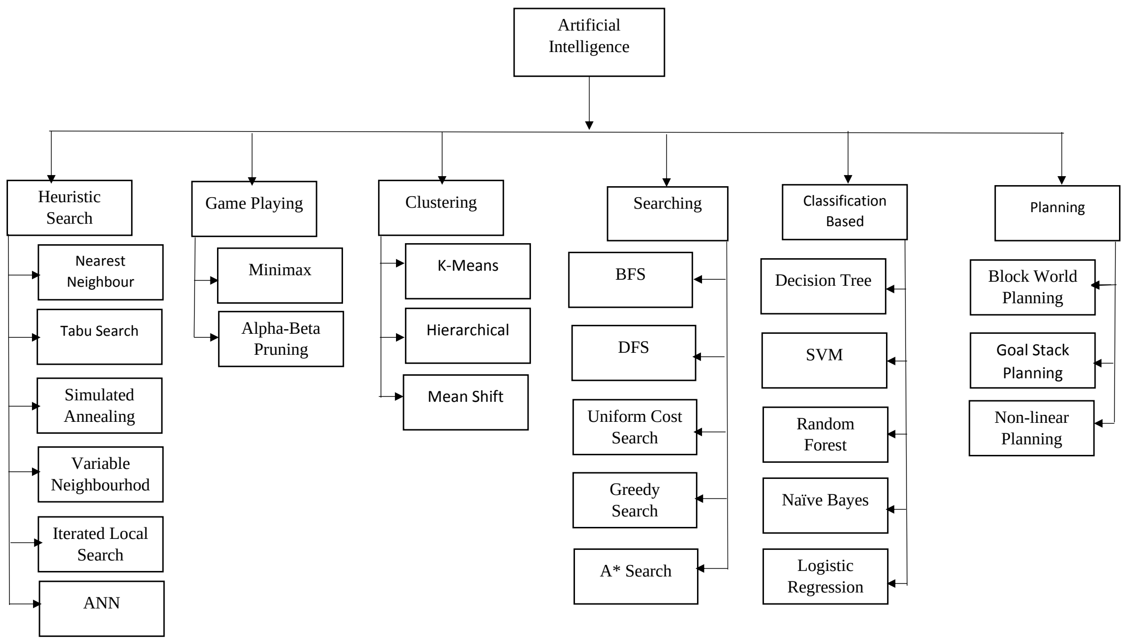


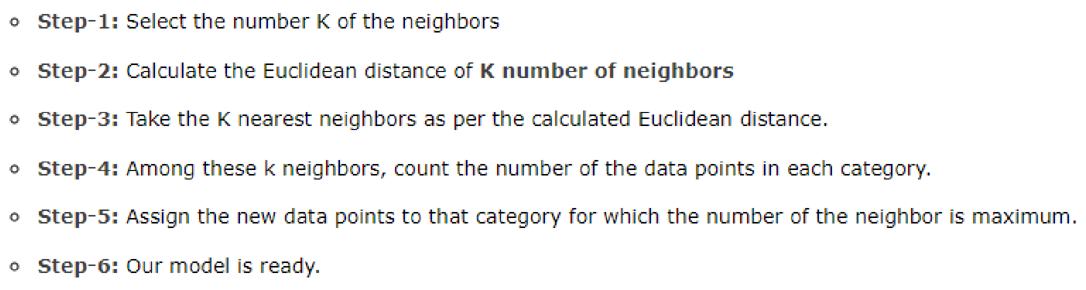
Figure 3.2: Classes of Artificial Intelligence Algorithm

3.4.2.1 Heuristic Search Algorithms

A heuristic is a technique for solving a problem faster than traditional methods or for estimating a solution when traditional methods fail. We often swap one of optimal, completeness, accuracy, or precision for speed. In artificial intelligence there are some kinds of heuristic searches that we included in this framework are-

- Nearest Neighbour
- Tabu Search
- Simulated Annealing
- Variable Neighbourhood
- Iterated Local Search
- ANN

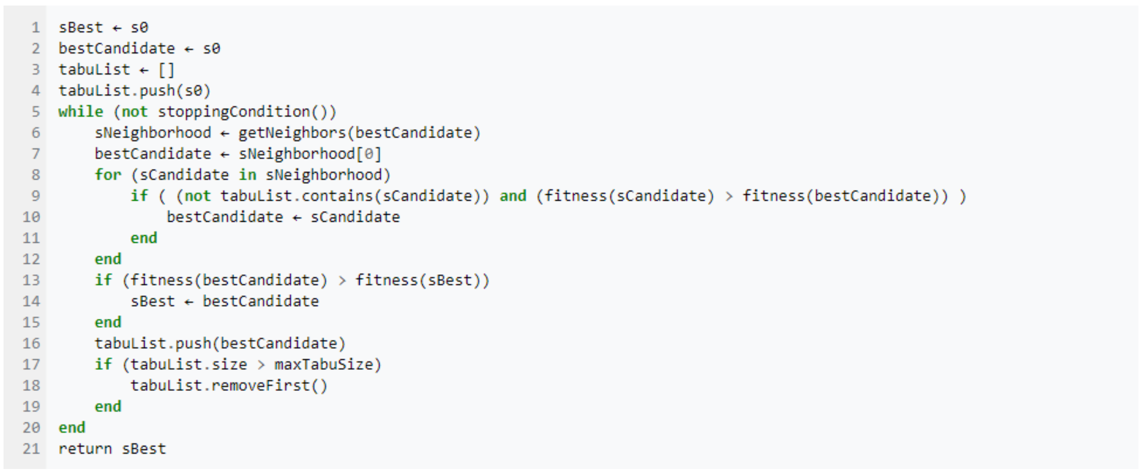
The optimization problem of finding the point in a given set that is closest to a given point is known as nearest neighbour search (NNS). Dissimilarity functions are often used to express closeness: the greater the function values, the less related the artifacts are.



- **Step-1:** Select the number K of the neighbors
- **Step-2:** Calculate the Euclidean distance of **K number of neighbors**
- **Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.
- **Step-4:** Among these k neighbors, count the number of the data points in each category.
- **Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.
- **Step-6:** Our model is ready.

Figure 3.3: Nearest Neighbour Search

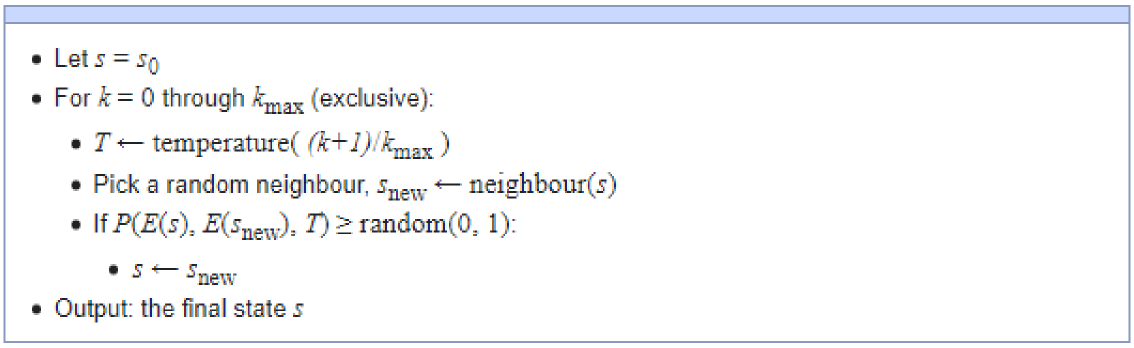
Tabu search is a meta-heuristic search system that use is used to optimize mathematical problems. Local (neighborhood) searches compare a possible solution to its immediate neighbors in the hopes of discovering a better one.



```
sBest ← s0
bestCandidate ← s0
tabuList ← []
tabuList.push(s0)
while (not stoppingCondition())
    sNeighborhood ← getNeighbors(bestCandidate)
    bestCandidate ← sNeighborhood[0]
    for (sCandidate in sNeighborhood)
        if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) > fitness(bestCandidate)) )
            bestCandidate ← sCandidate
        end
    end
    if (fitness(bestCandidate) > fitness(sBest))
        sBest ← bestCandidate
    end
    tabuList.push(bestCandidate)
    if (tabuList.size > maxTabuSize)
        tabuList.removeFirst()
    end
end
return sBest
```

Figure 3.4: Pseudo-code of Tabu Search

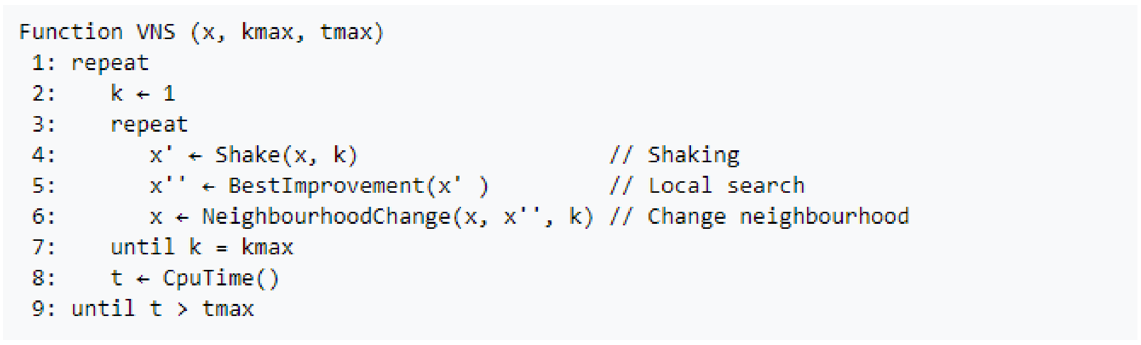
Simulated annealing (SA) is a probabilistic method for approximating a function's global optimum. It approximates global optimization for an optimization problem in a broad search space. When the search space is discrete, it is often used.



```
• Let $s = s_0$
• For $k = 0$ through $k_{max}$ (exclusive):
    • $T \leftarrow$ temperature( $(k+1)/k_{max}$ )
    • Pick a random neighbour, $s_{new} \leftarrow$ neighbour($s$)
    • If $P(E(s), E(s_{new}), T) \geq$ random(0, 1):
        • $s \leftarrow s_{new}$
• Output: the final state $s$
```

Figure 3.5: Pseudo-code of Simulated Annealing

Variable neighbourhood search (VNS) is a kind meta-heuristic way for solving different kind of optimization problems. It looks through the far reaches of the current incumbent solution before matches from the past and moves to the next.

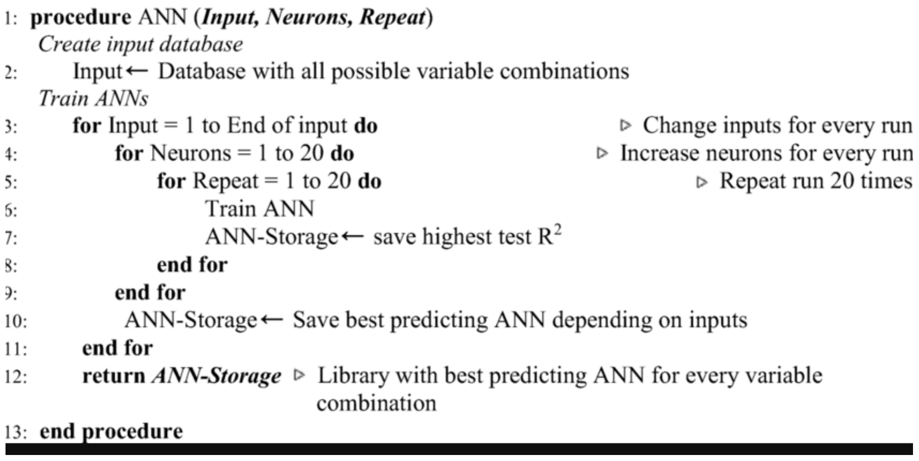


```
Function VNS (x, kmax, tmax)
repeat
    k ← 1
    repeat
        x' ← Shake(x, k)                  // Shaking
        x'' ← BestImprovement(x' )        // Local search
        x ← NeighbourhoodChange(x, x'', k) // Change neighbourhood
    until k = kmax
    t ← CpuTime()
until t > tmax
```

Figure 3.6: Pseudo-code of Variable Neighbourhood Search

Iterated local search (ILS) is a concept used in applied mathematics and computer science for solving discrete optimization problems that is a modification of local search or hill climbing. Local search methods may become trapped in a local minimum, where there are no better neighbors.

An artificial neural network (ANN) is a component of a computational system that mimics how the human brain analyzes and processes data. artificial intelligence (AI) is built on this basis, and it solves problems that would be impossible or difficult to solve by human or statistical standards.



```
procedure ANN (Input, Neurons, Repeat)
  Create input database
    Input ← Database with all possible variable combinations
  Train ANNs
    for Input = 1 to End of input do                      ▷ Change inputs for every run
        for Neurons = 1 to 20 do                          ▷ Increase neurons for every run
            for Repeat = 1 to 20 do                       ▷ Repeat run 20 times
                Train ANN
                ANN-Storage ← save highest test $R^2$
            end for
        end for
        ANN-Storage ← Save best predicting ANN depending on inputs
    end for
    return ANN-Storage  ▷ Library with best predicting ANN for every variable
                          combination
end procedure
```

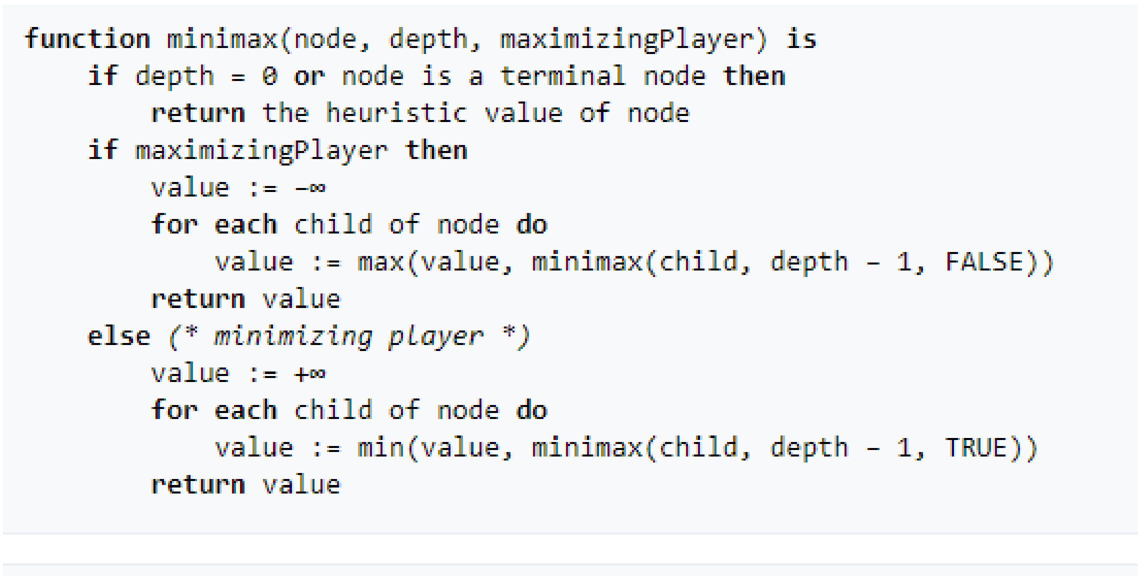
Figure 3.7: Pseudo-code of ANN

3.4.2.2 Game Playing Algorithms

The creation of artificial intelligence systems to be able to play more than one game successfully is known as general game playing. A chess-playing computer program, for example, cannot play checkers. Playing general games is regarded as a required step on the path to Artificial General Intelligence. We considered 2 types of game playing algorithms for this work-

- Minimax
- Alpha-Beta Pruning

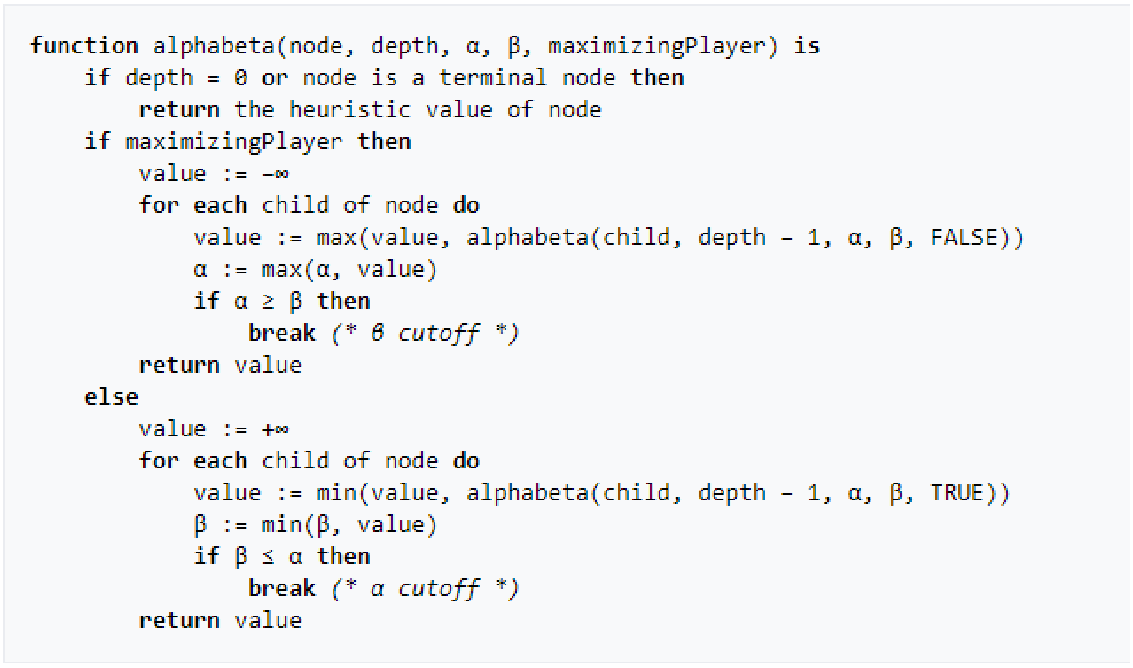
In decision-making and game theory, the mini-max algorithm is a recursive or backtracking algorithm. It suggests the best move for the player, assuming that the opponent is also playing well. This algorithm calculates the current state's minimax judgment.



```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

Figure 3.8: Pseudo-code of Minimax

A updated variant of the minimax algorithm is alpha-beta pruning. It's a strategy for improving the minimax algorithm. Alpha-Beta Algorithm is another name for it. Alpha-beta pruning can be done at any depth in a forest, and it can also prune not only the tree leaves but whole sub-trees as well.



```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1, α, β, FALSE))
            α := max(α, value)
            if α ≥ β then
                break (* β cutoff *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1, α, β, TRUE))
            β := min(β, value)
            if β ≤ α then
                break (* α cutoff *)
        return value
```

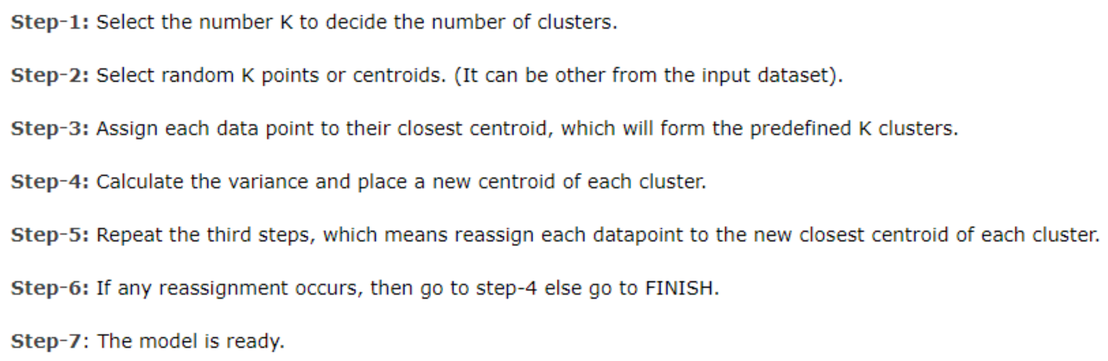
Figure 3.9: Pseudo-code of Alpha-Beta Pruning

3.4.2.3 Clustering Algorithms

Clustering is an artificial intelligence strategy that involves grouping data points. Data points in the same group should, in principle, have identical properties and/or features, while data points in separate groups should have extremely dissimilar properties and/or features. We considered 3 types of clustering algorithms here-

- K-means Clustering
- Hierarchical Clustering
- Mean-shift Clustering

A vector quantization approach that aims to partition n observations into k clusters, with each observation belonging to the cluster with the closest mean, which serves as the prototype of cluster's is called k-means clustering.



Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).

Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.

Step-4: Calculate the variance and place a new centroid of each cluster.

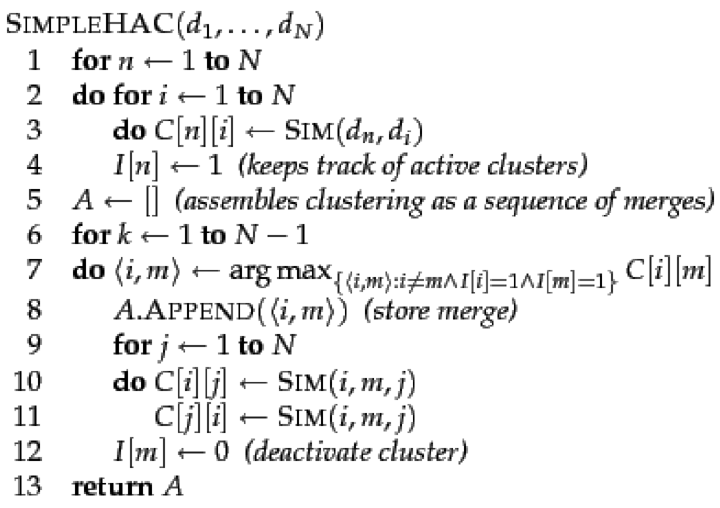
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.

Step-7: The model is ready.

Figure 3.10: K-Means Clustering

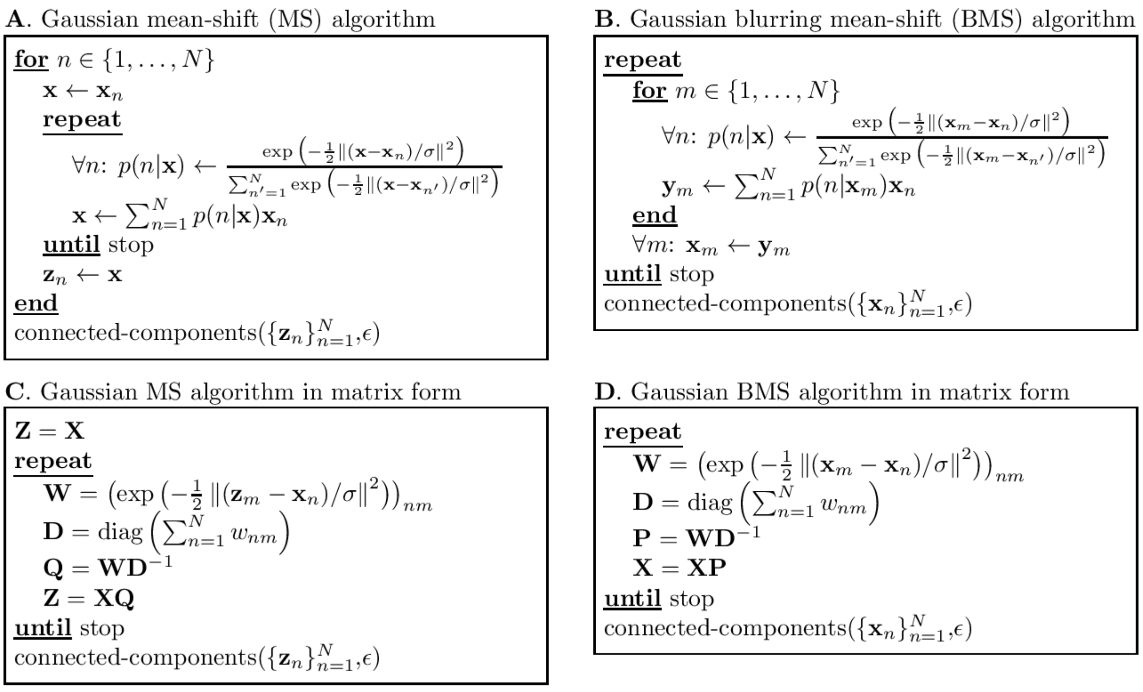
Separating data into groups based on some measure of similarity, finding a way to measure how they're alike and different, and narrowing down the data is what hierarchical clustering is all about.



```
SIMPLEHAC($d_1, \ldots, d_N$)
for $n \leftarrow 1$ to $N$
do for $i \leftarrow 1$ to $N$
    do $C[n][i] \leftarrow$ SIM($d_n, d_i$)
    $I[n] \leftarrow 1$ (keeps track of active clusters)
$A \leftarrow []$ (assembles clustering as a sequence of merges)
for $k \leftarrow 1$ to $N-1$
do $\langle i, m\rangle \leftarrow \arg\max_{\{\langle i,m\rangle : i \neq m \wedge I[i]=1 \wedge I[m]=1\}} C[i][m]$
    $A$.APPEND($\langle i, m\rangle$) (store merge)
    for $j \leftarrow 1$ to $N$
    do $C[i][j] \leftarrow$ SIM($i, m, j$)
        $C[j][i] \leftarrow$ SIM($i, m, j$)
    $I[m] \leftarrow 0$ (deactivate cluster)
return $A$
```

Figure 3.11: Pseudo-code of Hierarchical Clustering

Using a flat kernel for mean shift clustering. It's a centroid-based algorithm that works by changing partners for centroids to be the mean of the points within a field. If the seeds aren't set, they're determined by clustering.



A. Gaussian mean-shift (MS) algorithm

```
for $n \in \{1, \ldots, N\}$
    $\mathbf{x} \leftarrow \mathbf{x}_n$
    repeat
        $\forall n$: $p(n|\mathbf{x}) \leftarrow \frac{\exp\left(-\frac{1}{2}\|(\mathbf{x}-\mathbf{x}_n)/\sigma\|^2\right)}{\sum_{n'=1}^{N}\exp\left(-\frac{1}{2}\|(\mathbf{x}-\mathbf{x}_{n'})/\sigma\|^2\right)}$
        $\mathbf{x} \leftarrow \sum_{n=1}^{N} p(n|\mathbf{x})\mathbf{x}_n$
    until stop
    $\mathbf{z}_n \leftarrow \mathbf{x}$
end
connected-components($\{\mathbf{z}_n\}_{n=1}^{N}$,$\epsilon$)
```

B. Gaussian blurring mean-shift (BMS) algorithm

```
repeat
    for $m \in \{1, \ldots, N\}$
        $\forall n$: $p(n|\mathbf{x}) \leftarrow \frac{\exp\left(-\frac{1}{2}\|(\mathbf{x}_m-\mathbf{x}_n)/\sigma\|^2\right)}{\sum_{n'=1}^{N}\exp\left(-\frac{1}{2}\|(\mathbf{x}_m-\mathbf{x}_{n'})/\sigma\|^2\right)}$
        $\mathbf{y}_m \leftarrow \sum_{n=1}^{N} p(n|\mathbf{x}_m)\mathbf{x}_n$
    end
    $\forall m$: $\mathbf{x}_m \leftarrow \mathbf{y}_m$
until stop
connected-components($\{\mathbf{x}_n\}_{n=1}^{N}$,$\epsilon$)
```

C. Gaussian MS algorithm in matrix form

```
$\mathbf{Z} = \mathbf{X}$
repeat
    $\mathbf{W} = \left(\exp\left(-\frac{1}{2}\|(\mathbf{z}_m - \mathbf{x}_n)/\sigma\|^2\right)\right)_{nm}$
    $\mathbf{D} = \operatorname{diag}\left(\sum_{n=1}^{N} w_{nm}\right)$
    $\mathbf{Q} = \mathbf{W}\mathbf{D}^{-1}$
    $\mathbf{Z} = \mathbf{X}\mathbf{Q}$
until stop
connected-components($\{\mathbf{z}_n\}_{n=1}^{N}$,$\epsilon$)
```

D. Gaussian BMS algorithm in matrix form

```
repeat
    $\mathbf{W} = \left(\exp\left(-\frac{1}{2}\|(\mathbf{x}_m - \mathbf{x}_n)/\sigma\|^2\right)\right)_{nm}$
    $\mathbf{D} = \operatorname{diag}\left(\sum_{n=1}^{N} w_{nm}\right)$
    $\mathbf{P} = \mathbf{W}\mathbf{D}^{-1}$
    $\mathbf{X} = \mathbf{X}\mathbf{P}$
until stop
connected-components($\{\mathbf{x}_n\}_{n=1}^{N}$,$\epsilon$)
```

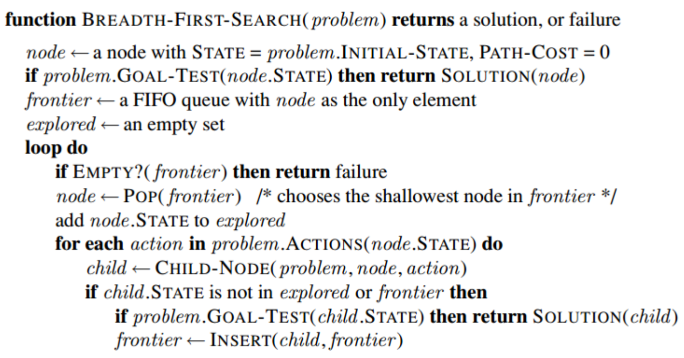
Figure 3.12: Pseudo-code of Mean Shift Clustering

3.4.2.4 Searching Algorithms

One of the most important fields of Artificial Intelligence is search algorithms.In A.I, search is the process of transitioning from a starting state to a target state through intermediate states. Every search technique is consist of three states-i)starting state, ii)intermediate state, iii)goal state. Here we considered the visualization of 5 searching techniques-

- Breadth-first Search
- Depth-first Search
- Uniform-cost Search
- Greedy Search
- A* Search

The Breadth First Search (BFS) algorithm is used to traverse or search data structures such as trees and graphs. Before moving on to the nodes at the next depth stage, it examines all of the nodes at the current depth.



```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)  /* chooses the shallowest node in frontier */
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
              frontier ← INSERT(child, frontier)
```

Figure 3.13: Pseudo-code of Breadth First Search

The depth-first search (DFS) algorithm is used to traverse or search data structures such as trees and graphs. The algorithm begins at the root node (in the case of a graph, any arbitrary node may be used as the root node) and explores each branch as far as possible before backtracking.

```
DFS(s)
    for each vertex u ∈V
        do color[u] ←White                 ; not visited
    time ←1                                ; time stamp
    for each vertex u ∈V
        do if color[u]=White
            then DFS-Visit(u,time)

DFS-Visit(u,time)
    color[u] ←Gray                         ; in progress nodes
    d[u] ←time                             ; d=discover time
    time ←time+1
    for each v ∈Adj[u] do
        if color[u]=White
            then DFS-Visit(v,time)
    color[u] ←Black
    f[u] ←time ←time+1                     ; f=finish time
```

Figure 3.14: Pseudo-code of Depth First Search

A searching algorithm for traversing a weighted tree or graph is uniform-cost search. The uniform-cost always searches the shortest path and obviously the path will cost lower than the others. The root node is expanded using uniform-cost search, which extends nodes based on their path costs.

```
function UNIFORM-COST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  frontier ← a priority queue ordered by PATH-COST, with node as the only element
  explored ← an empty set
  loop do
      if EMPTY?(frontier) then return failure
      node ← POP(frontier)  /* chooses the lowest-cost node in frontier */
      if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
      add node.STATE to explored
      for each action in problem.ACTIONS(node.STATE) do
          child ← CHILD-NODE(problem, node, action)
          if child.STATE is not in explored or frontier then
              frontier ← INSERT(child, frontier)
          else if child.STATE is in frontier with higher PATH-COST then
              replace that frontier node with child
```

Figure 3.15: Pseudo-code of Uniform-cost Search

Algorithm that follows the problem-solving heuristic of making the locally optimal option at each point is known as a greedy algorithm. A greedy search may deliver locally optimized solution that resemble a higher level of accuracy in a reasonable amount of time in many problems, but it does not always produce an optimal result.

```
set Greedy (Set Candidate){
  solution= new Set( );
  while (Candidate.isNotEmpty()) {
     next = Candidate.select(); //use selection criteria,
            //remove from Candidate and return value
     if (solution.isFeasible( next)) //constraints satisfied
               solution.union( next);
               if (solution.solves()) return solution}
  //No more candidates and no solution
  return null
}
```

Figure 3.16: Pseudo-code of Greedy Search

Due to its completeness, optimal and reliability, A* is a graph traversal and path search algorithm that is frequently used in many fields of computer science. Its space complexity is one of its major functional weaknesses, as it caches all generated nodes in memory.

```
A* search {

 closed list = [ ]
 open list = [start node]

   do {
         if open list is empty then {
               return no solution
         }
         n = heuristic best node
         if n == final node then {
               return path from start to goal node
         }
         foreach direct available node do{
               if current node not in open and not in closed list do {
                     add current node to open list and calculate heuristic
                     set n as his parent node
               }
               else{
                     check if path from star node to current node is
                     better;
                     if it is better calculate heuristics and transfer
                     current node from closed list to open list
                     set n as his parrent node
               }
         delete n from open list
         add n to closed list
   } while (open list is not empty)

}
```

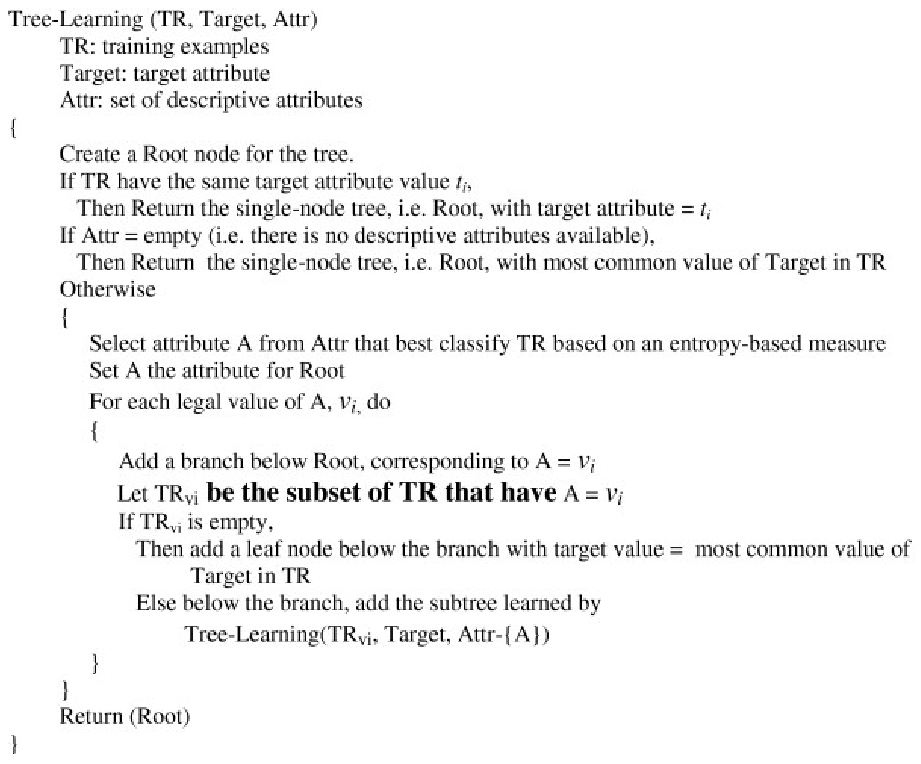
Figure 3.17: Pseudo-code of A* Search

3.4.2.5 Classification Algorithms

Classification is the method of dividing a collection of data into categories. It can be done on both structured and unstructured data. Predicting the class of provided data points is the first step in the process.. Here we considered the visualization of 5 classification techniques-

- Decision Tree
- Support Vector Machine
- Random Forest
- Naive Bayes
- Logistic Regression

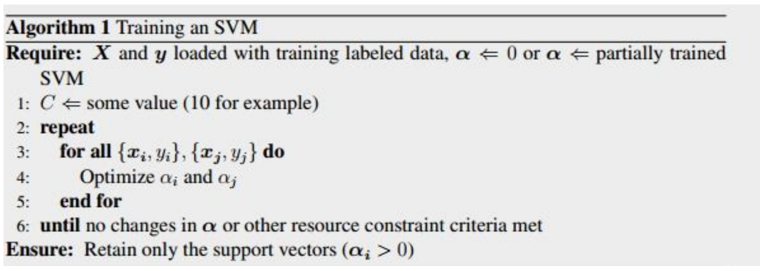
Decision Trees are a version of supervised machine learning in which data is continually split based on a parameter. Two entities, decision nodes and leaves, can be used to illustrate the tree.



```
Tree-Learning (TR, Target, Attr)
    TR: training examples
    Target: target attribute
    Attr: set of descriptive attributes
{
    Create a Root node for the tree.
    If TR have the same target attribute value $t_i$,
      Then Return the single-node tree, i.e. Root, with target attribute = $t_i$
    If Attr = empty (i.e. there is no descriptive attributes available),
      Then Return  the single-node tree, i.e. Root, with most common value of Target in TR
    Otherwise
    {
        Select attribute A from Attr that best classify TR based on an entropy-based measure
        Set A the attribute for Root
        For each legal value of A, $v_i$, do
        {
            Add a branch below Root, corresponding to A = $v_i$
            Let $TR_{vi}$ be the subset of TR that have A = $v_i$
            If $TR_{vi}$ is empty,
              Then add a leaf node below the branch with target value =  most common value of
                    Target in TR
              Else below the branch, add the subtree learned by
                    Tree-Learning($TR_{vi}$, Target, Attr-{A})
        }
    }
    Return (Root)
}
```

Figure 3.18: Pseudo-code of Decision Tree

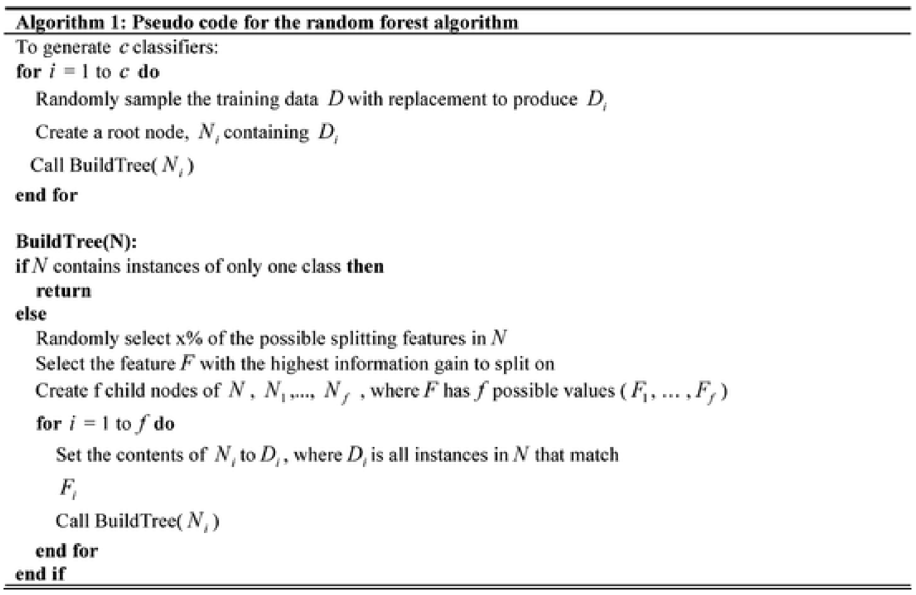
SVM (Support Vector Machine) is a supervised machine learning algorithm that can be used to solve regression and classification tasks. It is, however, mostly used to solve classification problems. Unique observation co-ordinates are what Support Vectors are.



```
Algorithm 1 Training an SVM
Require: $X$ and $y$ loaded with training labeled data, $\alpha \Leftarrow 0$ or $\alpha \Leftarrow$ partially trained
    SVM
1: $C \Leftarrow$ some value (10 for example)
2: repeat
3:     for all $\{x_i, y_i\}, \{x_j, y_j\}$ do
4:         Optimize $\alpha_i$ and $\alpha_j$
5:     end for
6: until no changes in $\alpha$ or other resource constraint criteria met
Ensure: Retain only the support vectors ($\alpha_i > 0$)
```

Figure 3.19: Pseudo-code of SVM

The random forest is a classification algorithm that uses several decision trees to classify data. When constructing each individual tree, it employs bagging and feature randomness in order to establish an uncorrelated forest of trees whose committee prediction is more effective than that of any single tree.

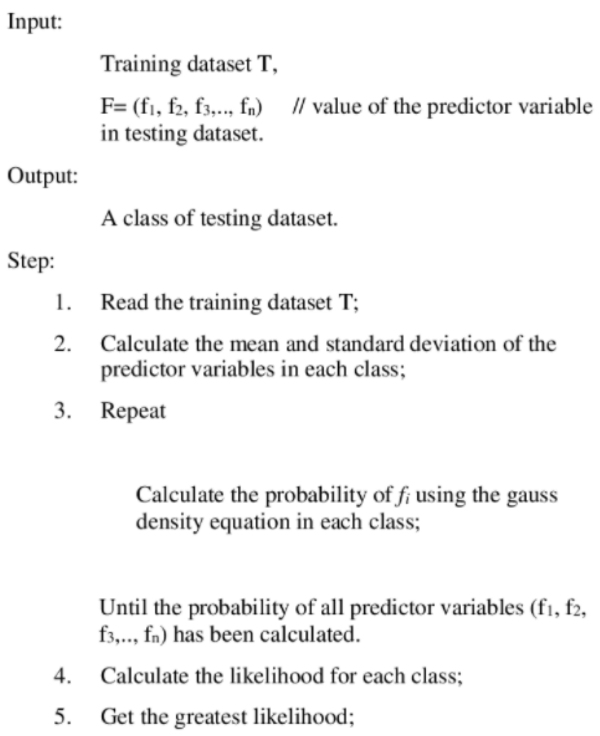


```
Algorithm 1: Pseudo code for the random forest algorithm
To generate $c$ classifiers:
for $i = 1$ to $c$ do
    Randomly sample the training data $D$ with replacement to produce $D_i$
    Create a root node, $N_i$ containing $D_i$
    Call BuildTree( $N_i$ )
end for

BuildTree(N):
if $N$ contains instances of only one class then
    return
else
    Randomly select x% of the possible splitting features in $N$
    Select the feature $F$ with the highest information gain to split on
    Create f child nodes of $N$ , $N_1$,..., $N_f$ , where $F$ has $f$ possible values ( $F_1$, ... , $F_f$ )
    for $i = 1$ to $f$ do
        Set the contents of $N_i$ to $D_i$ , where $D_i$ is all instances in $N$ that match
        $F_i$
        Call BuildTree( $N_i$ )
    end for
end if
```

Figure 3.20: Pseudo-code of Random Forest

Naive Bayes is a classification algorithm. Instead of depending on big data, it is a faster way to classify information about customers based on chance. This helps keep the workload manageable and efficient while reducing the demand on the AI system.



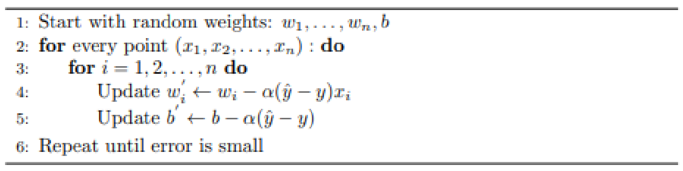
```
Input:
        Training dataset T,
        F= ($f_1$, $f_2$, $f_3$,.., $f_n$)    // value of the predictor variable
        in testing dataset.
Output:
        A class of testing dataset.
Step:
    1.  Read the training dataset T;
    2.  Calculate the mean and standard deviation of the
        predictor variables in each class;
    3.  Repeat

            Calculate the probability of $f_i$ using the gauss
            density equation in each class;

        Until the probability of all predictor variables ($f_1$, $f_2$,
        $f_3$,.., $f_n$) has been calculated.
    4.  Calculate the likelihood for each class;
    5.  Get the greatest likelihood;
```

Figure 3.21: Pseudo-code of Naive Bayes

Under the Supervised Learning technique, one of the most common Machine Learning algorithms is logistic regression. It's a technique for assessing a categorical dependent variable from a number of independent variables.



```
1: Start with random weights: $w_1, \ldots, w_n, b$
2: for every point $(x_1, x_2, \ldots, x_n)$ : do
3:     for $i = 1, 2, \ldots, n$ do
4:         Update $w_i^{'} \leftarrow w_i - \alpha(\hat{y} - y)x_i$
5:         Update $b^{'} \leftarrow b - \alpha(\hat{y} - y)$
6: Repeat until error is small
```

Figure 3.22: Pseudo-code of Logistic Regression

3.4.2.6 Planning Algorithms

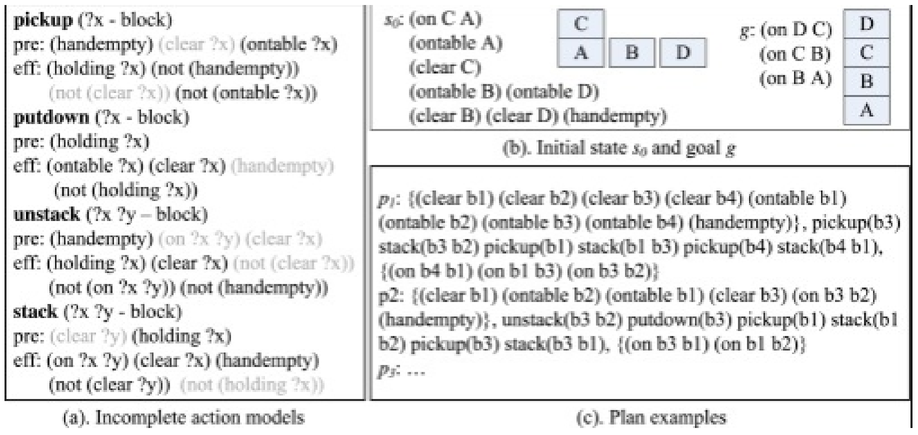
In Artificial Intelligence, preparation refers to the decision-making activities carried out by robots or computer programs in order to accomplish a particular objective. The execution of preparation entails selecting a set of actions that will most likely result in the mission being completed. Here we considered the visualization of 3 types of planning techniques-

- Block World Planning

- Goal Stack Planning

One of the most well-known planning domains of artificial intelligence is the block world planning. The algorithm is similar to a table full of different-shaped and colored wooden blocks. The purpose is to construct one or more vertical block layers.

Goal stack planning is one of the most basic planning algorithms for dealing with problems with multiple targets. For plan generation, this method employs a Stack. Sub-goals and behavior listed with predicates can be found in the stack. Each of the sub-goals can be performed in any order.



```
pickup (?x - block)
pre: (handempty) (clear ?x) (ontable ?x)
eff: (holding ?x) (not (handempty))
     (not (clear ?x)) (not (ontable ?x))
putdown (?x - block)
pre: (holding ?x)
eff: (ontable ?x) (clear ?x) (handempty)
     (not (holding ?x))
unstack (?x ?y – block)
pre: (handempty) (on ?x ?y) (clear ?x)
eff: (holding ?x) (clear ?x) (not (clear ?x))
     (not (on ?x ?y)) (not (handempty))
stack (?x ?y - block)
pre: (clear ?y) (holding ?x)
eff: (on ?x ?y) (clear ?x) (handempty)
     (not (clear ?y)) (not (holding ?x))
```

(a). Incomplete action models

s_0: (on C A)
(ontable A)
(clear C)
(ontable B) (ontable D)
(clear B) (clear D) (handempty)

g: (on D C)
(on C B)
(on B A)

(b). Initial state s_0 and goal g

p_1: {(clear b1) (clear b2) (clear b3) (clear b4) (ontable b1) (ontable b2) (ontable b3) (ontable b4) (handempty)}, pickup(b3) stack(b3 b2) pickup(b1) stack(b1 b3) pickup(b4) stack(b4 b1), {(on b4 b1) (on b1 b3) (on b3 b2)}
p2: {(clear b1) (ontable b2) (ontable b1) (clear b3) (on b3 b2) (handempty)}, unstack(b3 b2) putdown(b3) pickup(b1) stack(b1 b2) pickup(b3) stack(b3 b1), {(on b3 b1) (on b1 b2)}
p_3: …

(c). Plan examples

Figure 3.23: Pseudo-code of Goal Stack Planning

3.4.3 Working Module

Actually the working module is divided into 2 sections. They are-

3.4.3.1 Front End

For the development of front end we use unity as tool for our work. We have created scenes, created animations and created gameplay in front end.

- **Create Scene**

  Create scene is nothing but a fresh page in unity where we will built our graph/scenery for algorithms. All sorts of nodes, boxes, texts are create here, they are all like static without any working function in this part.

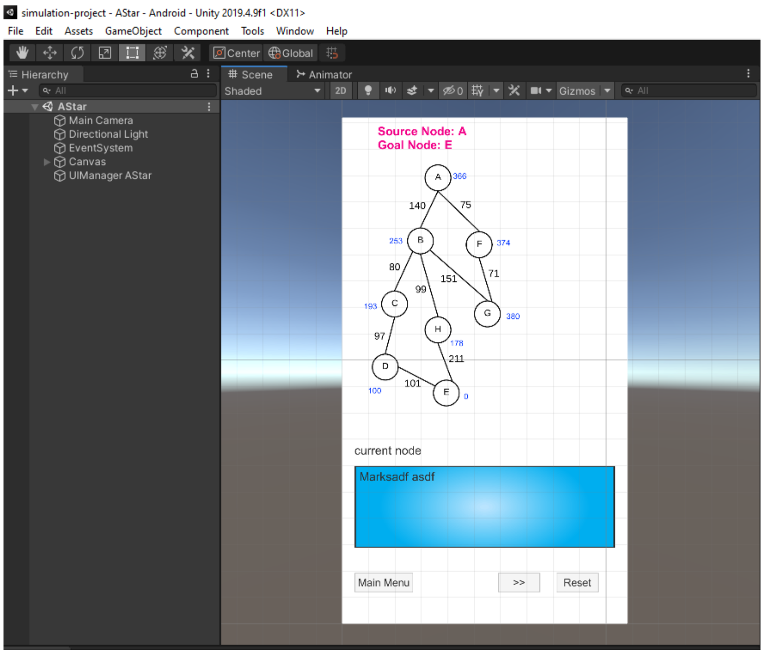


Figure 3.24: Create Scene

- **Create Animation**

The animation is basically the points which works in dynamic state. When we go from one node to another then the circles becomes bigger to smaller and smaller to bigger or the boxes unstacks from a packet and keeps on the ground. There is a movement of graphical change. This is actually animations of this project.

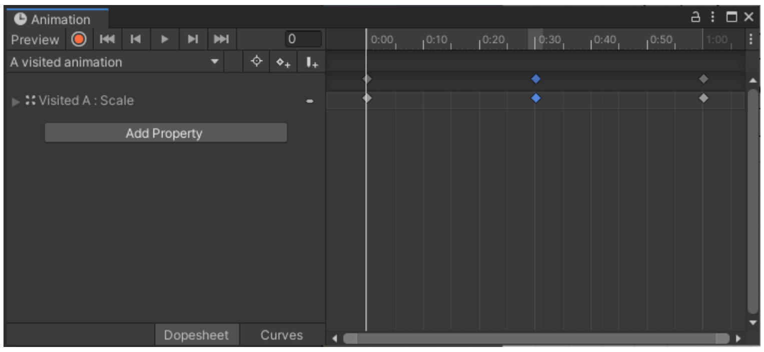
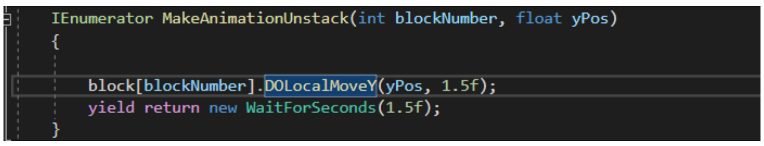


Figure 3.25: Create Animation



```
IEnumerator MakeAnimationUnstack(int blockNumber, float yPos)
{
    block[blockNumber].DOLocalMoveY(yPos, 1.5f);
    yield return new WaitForSeconds(1.5f);
}
```

Figure 3.26: Animation working function

- **Create Gameplay**

  The gameplay is basically the play button which indicates any scene to start its working procedure. Each scene has a gameplay button which actually insist to trigger the animation to be played.

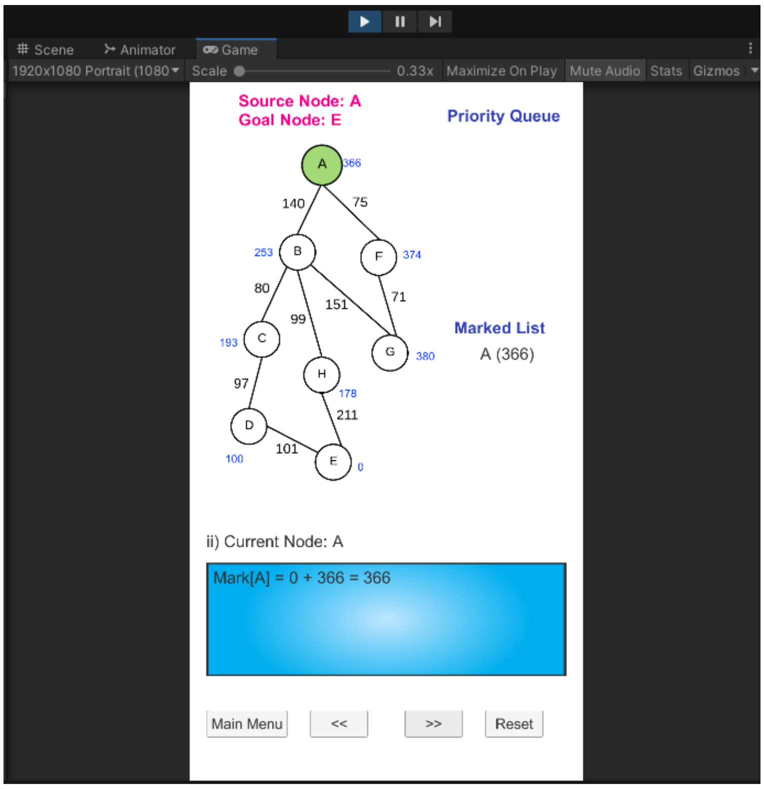


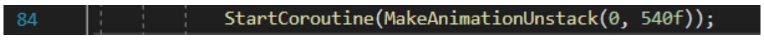
Figure 3.27: Create Gameplay

3.4.3.2 Back End

For the back end development, we used visual studio as tool. Visual studio mainly used to write coding scripts and to triggerd the animations.

- **Triggered Animation**

  It is mainly to call the function of a selected class of algorithm. As an example- if we want to visualize dfs, then the dfs scene will be triggered for visualization

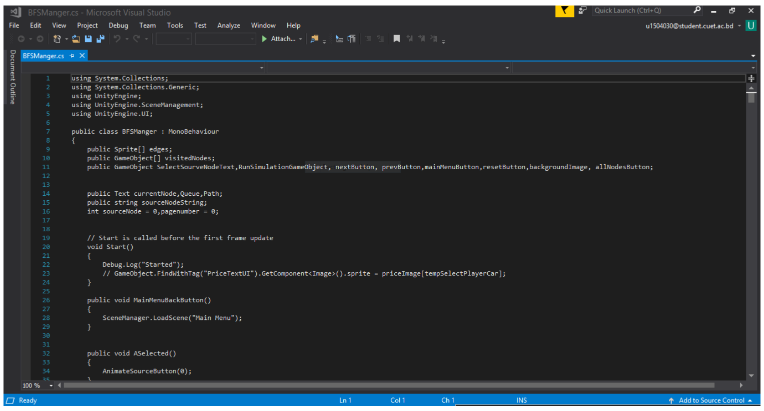


```
StartCoroutine(MakeAnimationUnstack(0, 540f));
```

Figure 3.28: Triggered Animation

- **Program in C#**

In this project, we actually code in C# for each scene. We did it in visual studio which collaboratively works with unity.


BFSManger.cs - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
u1504030@student.cuet.ac.bd
Attach...
BFSManger.cs
Document Outline
100 %
Ready
Ln 1
Col 1
Ch 1
INS
Add to Source Control

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class BFSManger : MonoBehaviour
{
    public Sprite[] edges;
    public GameObject[] visitedNodes;
    public GameObject SelectSourveNodeText,RunSimulationGameObject, nextButton, prevButton,mainMenuButton,resetButton,backgroundImage, allNodesButton;


    public Text currentNode,Queue,Path;
    public string sourceNodeString;
    int sourceNode = 0,pagenumber = 0;


    // Start is called before the first frame update
    void Start()
    {
        Debug.Log("Started");
        // GameObject.FindWithTag("PriceTextUI").GetComponent<Image>().sprite = priceImage[tempSelectPlayerCar];
    }

    public void MainMenuBackButton()
    {
        SceneManager.LoadScene("Main Menu");
    }


    public void ASelected()
    {
        AnimateSourceButton(0);
```

Figure 3.29: Program in C#

3.4.4 Output Module

Basically after getting all the inputs, selections the app will start to simulate/visualize the algorithms, which is represented as animation. In this module the visualization shows how the algorithms are working step by step. After visualizing the selected class, the app shows the optimal solution of an algorithm. This is how the app works.

3.5 Conclusion

In this chapter, we have discussed the overall methodology of our proposed framework. We have discussed that our methodology works in four different module. Each sections have their individual tasks. We have discussed about the algorithm classes in short. Actually before simulation it is important to know the algorithm works. Then we showed every step with image files. That indicates the four module actually works with each other.

Chapter 4

Results and Discussions

4.1 Introduction

In this chapter, we will discuss about the final result of our work. We will discuss about the result we got from our proposed work and the impacts of this framework.

4.2 Impact Analysis

The impact analysis has divided into two parts and they are briefly discussed in section 4.2.1 and section 4.2.2.

4.2.1 Social and Environmental Impact

Our educational system is improving day-by-day. So modern methods of learning techniques are being applied in every sector of education. Our framework provides a good number of impacts on social and environmental perspectives-

- Marketing strategies depend on such kind of works in a large scale [12].
- A good strategy on education to solve the problems in animation is revolutionary on education system [13].
- Developing an effective and suitable educational system for the future which will be design for achieving highest excellence [14] [15].
- Making a qualitiful platform for higher studies [16].

4.2.2 Ethical Impact

In an educative site the ethical impact is very important to learn some interesting things. An effortlessness system must be work in quest to gain popularity and spread a good vive among learners [17]. Otherwise the system will not work properly as we have to choose between the right or wrong. When you are using a technology for something there must be a good and a bad side. It's on us that which path should we choose. So that every person should realize about ethical impact of a thing and bring the best from that [18].

4.3 Experimental Result

This section provides the experimental settings, as well as evaluates the performance of our methodology.

4.3.1 Experiment Setup

We set up our system with given specifications:

1. Number of commodity PCs: 2
2. Memory of each PC: 8
3. Operating System: Windows 10
4. Unity version: 2019.4.24f1
5. Visual Studio version: Community 2017
6. UI interface build: Lucid chart

4.3.2 Beginning Interface

The beginning interface is the section that pop-up when we enter into the app. It contains the main sections of classes in forms of button.

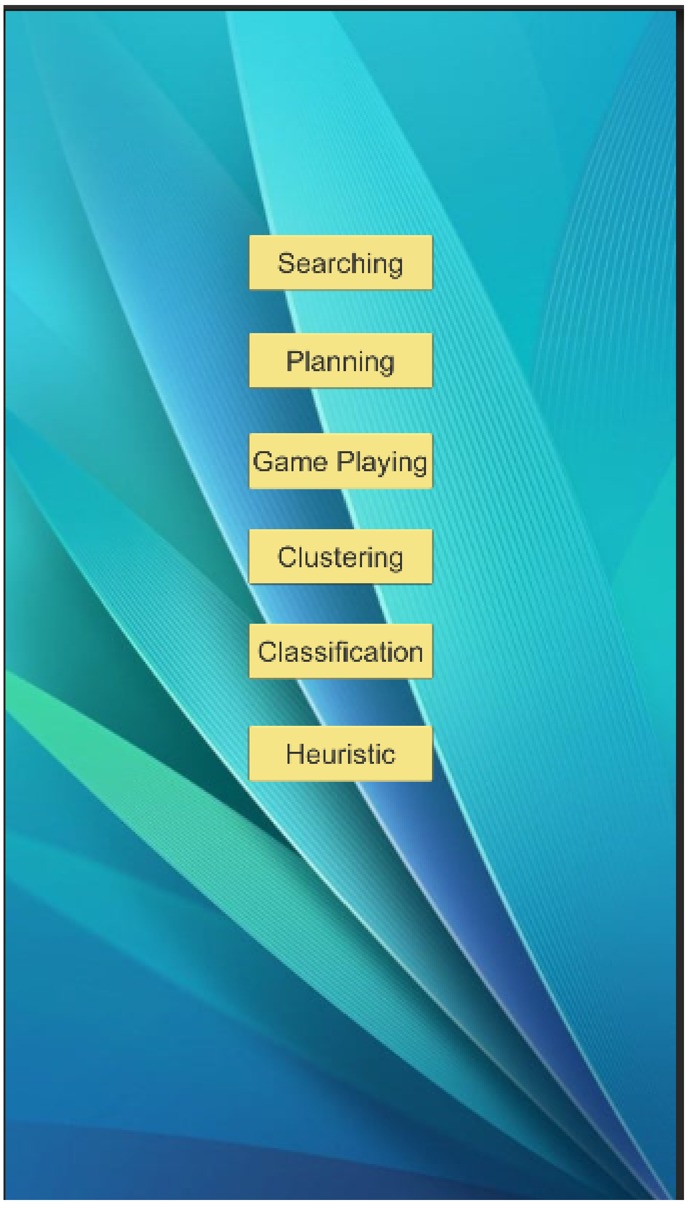


Figure 4.1: Beginning Interface

4.3.3 Selection Buttons

It is a section of beginning interface. Actually the classes are present in this section. If we consider planning class the the internal section of planning class will pop-up in front of the screen and we can select the planning sub-classes according to our needs.

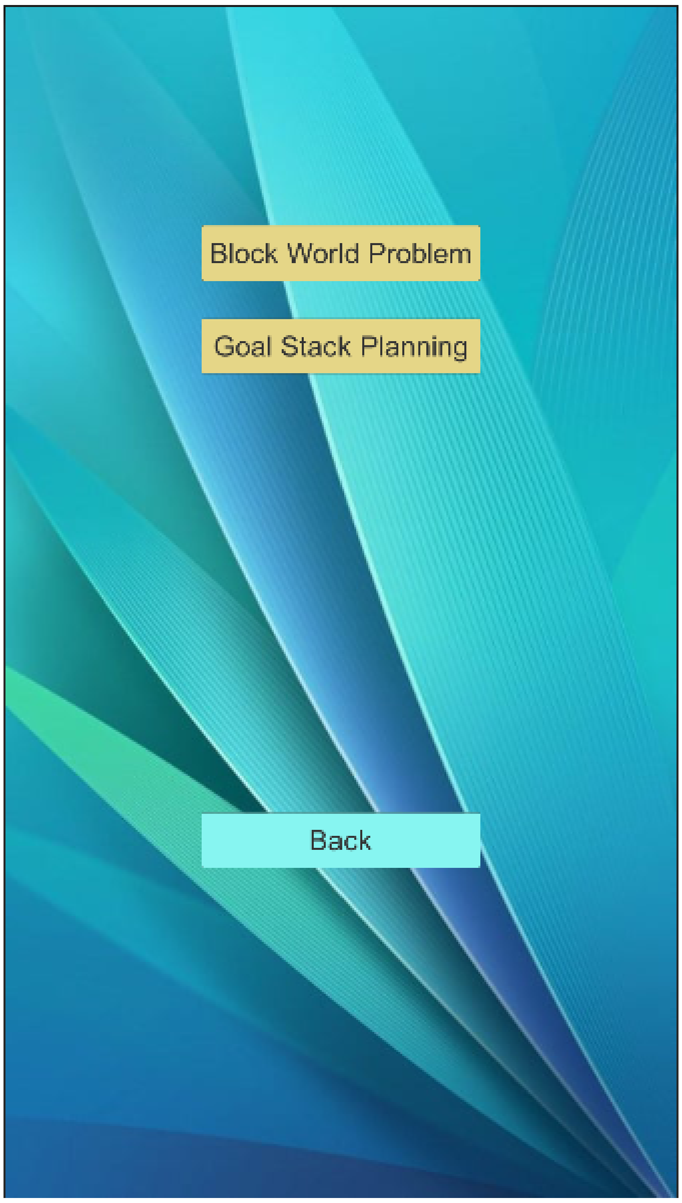
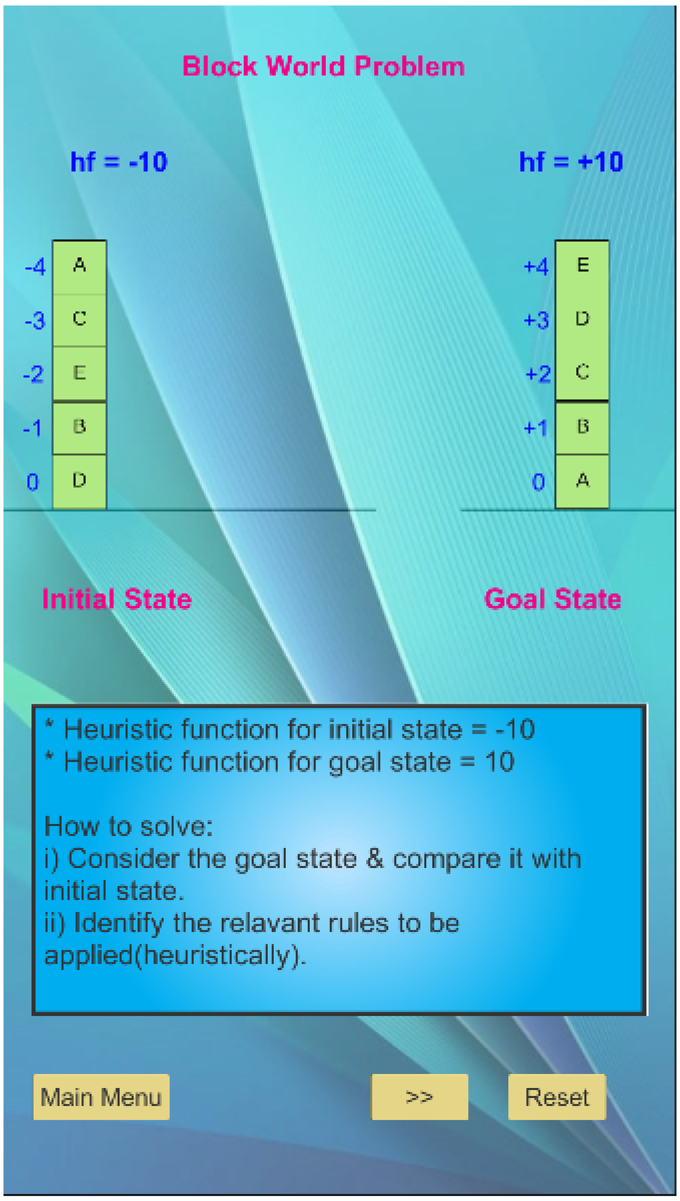


Figure 4.2: Planning Algorithms

4.3.4 Visualization of Selected Class

As an example to examine, we have selected block world problem to visualize. So as a result after entering into the scene there we can see the initial state of the problem and a goal state where the initial state has to reach to complete the scenes. If we look at the block world algorithm we can see- there must be a value of each block in initial state and a heuristic value for both initial state and goal state. The heuristic value of the initial state must be same as the goal state, which satisfies the solution of block world problem. When we put a block on the ground, then its heuristic value becomes 0. And when we stack them together its value begin to increase. So the scenes are going to be described step by step to show how actually the animation works-

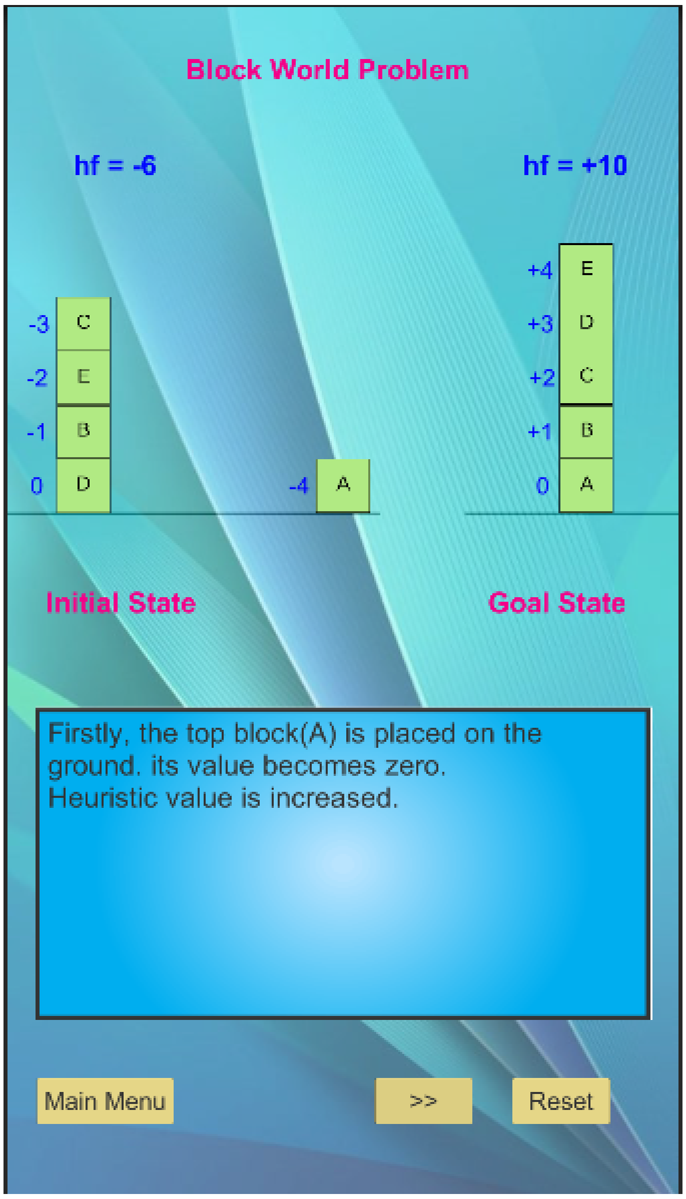


* Heuristic function for initial state = -10
* Heuristic function for goal state = 10

How to solve:
i) Consider the goal state & compare it with initial state.
ii) Identify the relavant rules to be applied(heuristically).

Figure 4.3: Scene Initialization

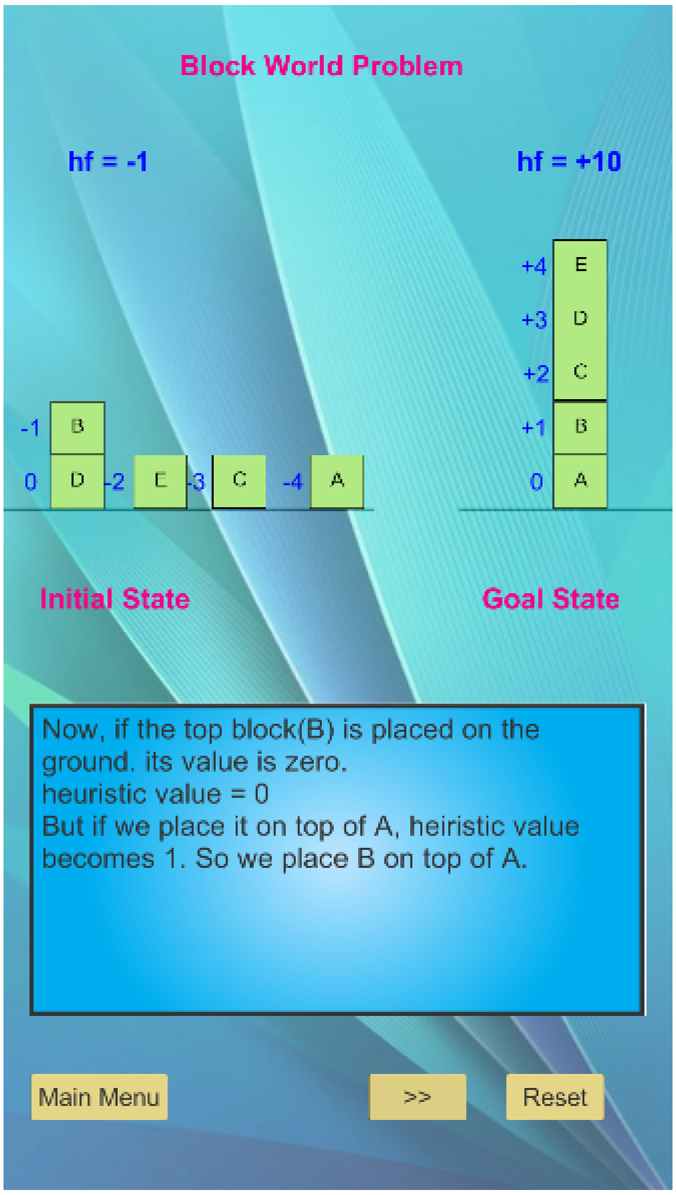
The part of adjusting or changing heuristic functional value comes next. The values are changing here when the blocks are being on the ground.



Firstly, the top block(A) is placed on the ground. its value becomes zero.
Heuristic value is increased.

Figure 4.4: Initializing Values

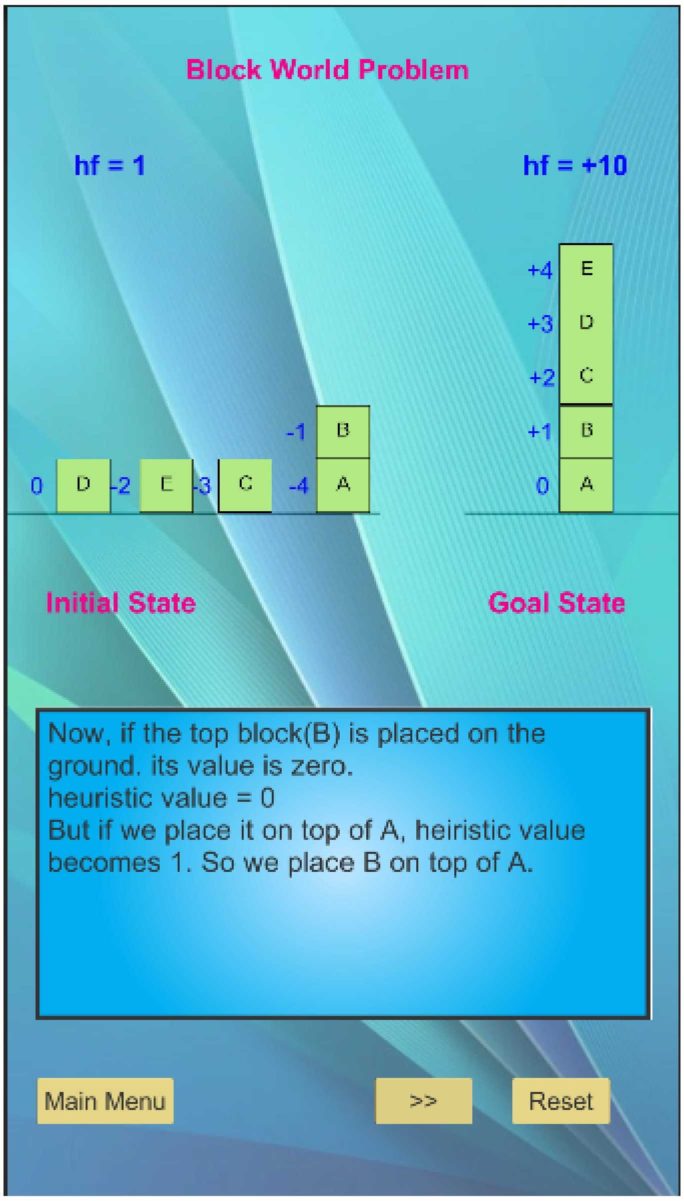
We can see there are five blocks and each of the block has to step on the ground so that their functional values will be changed. When a block steps on the ground then their functional values become 0.



Now, if the top block(B) is placed on the ground. its value is zero.
heuristic value = 0
But if we place it on top of A, heiristic value becomes 1. So we place B on top of A.

Figure 4.5: Unstacking Blocks

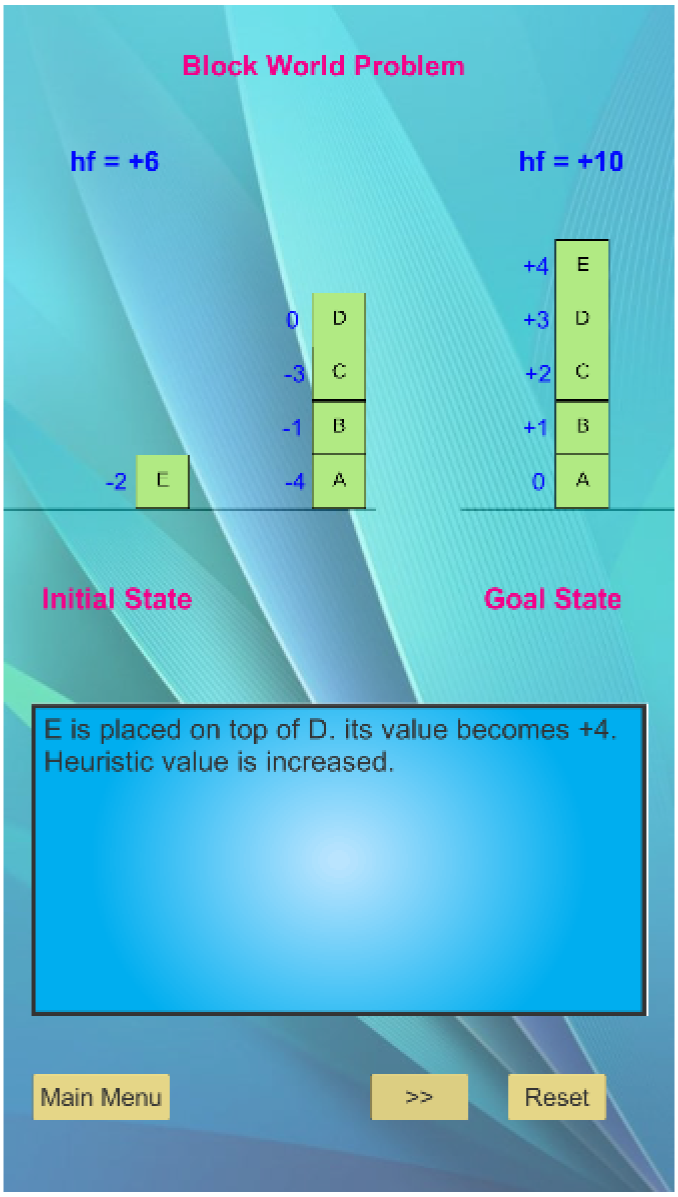
To reach to the final state, the blocks have to be stack together again by maintaining the sequence of goal state. Otherwise we cannot get the desired functional values.



Now, if the top block(B) is placed on the ground. its value is zero.
heuristic value = 0
But if we place it on top of A, heiristic value becomes 1. So we place B on top of A.

Figure 4.6: Stacking Blocks

If we see at the change rate of heuristic values, we can see by moving every blocks, the values are changing continuously. And the values will keep changing till we reach the goal state.

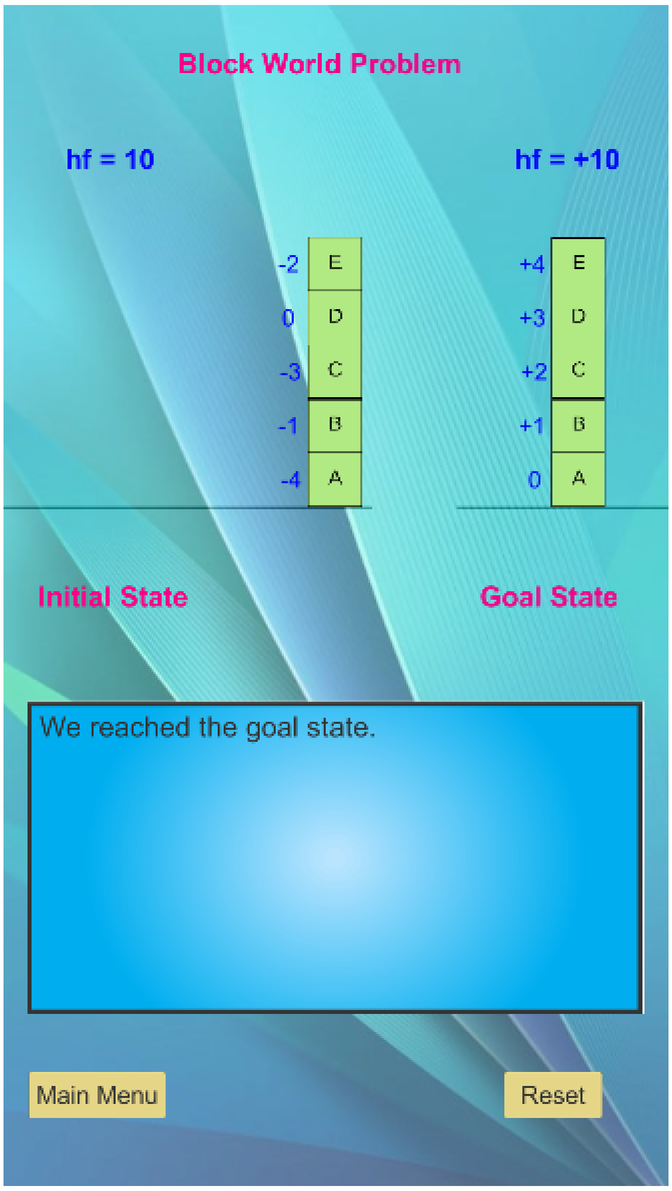


E is placed on top of D. its value becomes +4. Heuristic value is increased.

Figure 4.7: Changing Heuristic Values

4.3.5 Final Output

From the beginning of the scene we saw that there was a initial state which has to obtain a goal state. When our visualization started the heuristic function values started changing by the change of blocks position and after that we can easily obtain our final and desired output.



We reached the goal state.

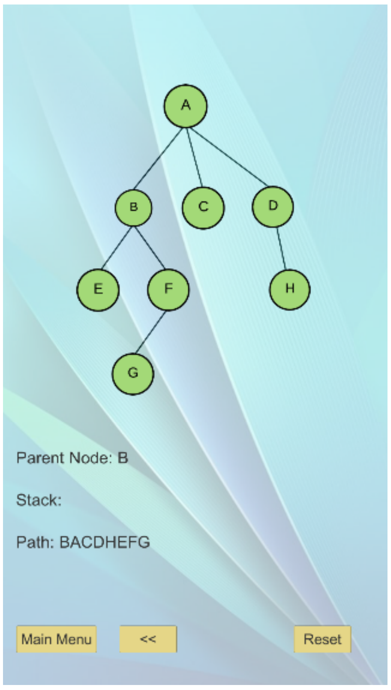
Figure 4.8: Final Output

4.4 Comparison Analysis

In literature review section , we have mentioned some works that was very close and related to our work. We actually developed a framework where we try to explain the new features of algorithms. In this section we will compare our framework to the other related works.

In [7] we saw they developed a framework on basis of searching techniques. If we compare our works with them, we can get an overview on that.

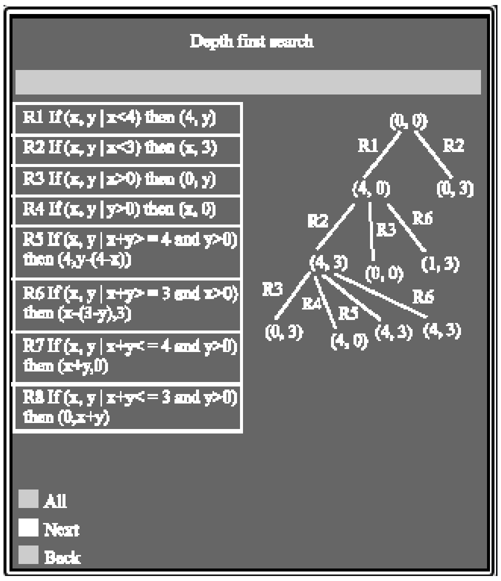
The main comparisons are mentioned below:


Parent Node: B

Stack:

Path: BACDHEFG

(a) DFS solving on our framework


Depth first search

R1 If (x, y | x<4) then (4, y)

R2 If (x, y | x<3) then (x, 3)

R3 If (x, y | x>0) then (0, y)

R4 If (x, y | y>0) then (x, 0)

R5 If (x, y | x+y> = 4 and y>0) then (4,y-(4-x))

R6 If (x, y | x+y> = 3 and x>0) then (x-(3-y),3)

R7 If (x, y | x+y< = 4 and y>0) then (x+y,0)

R8 If (x, y | x+y< = 3 and y>0) then (0,x+y)

(b) DFS solving on related work

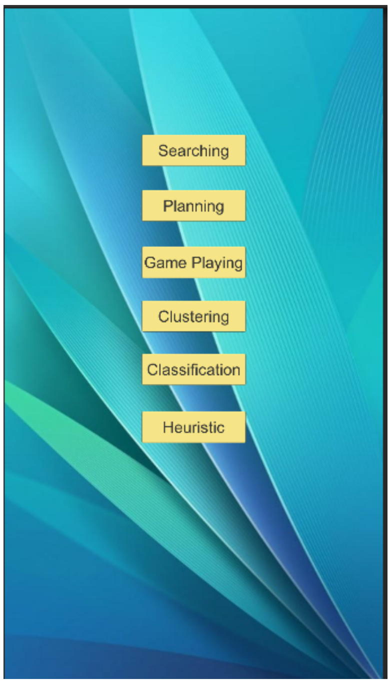
Figure 4.9: Comparison between our framework and related work

- We have a well build interface than the related work.
- Our system is total dynamic as we can select any nodes from here, but their system is static.
- They only set parameters and described a screenshot of solved problem but our system shows how the algorithm works with good visualization.
- In the related work, the paths are hard to described as we cannot identify where the nodes are working. But in our system it can easily be understood where the nodes will move to the next steps.

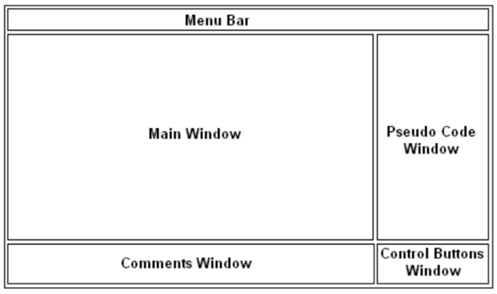
In [9] we saw a web-based platform called visual linprog to solve linear problems. If we compare our works with them, we can get an overview on that.

The main comparisons are mentioned below:

- Our provided framework has better background and clear selection buttons.
- Visual linprog only solves the straight lines in terms of given output, on the other hand our framework gives a clear view of an algorithm.



(a) Interface of our framework



(b) Interface of visual linprog

Figure 4.10: Comparison between our framework and related work

- Our framework provides greater animated views with good visualization techniques, where visual linprog is nothing but a simple view of lines.

4.5 Conclusion

In this chapter, the result and analysis of our proposed framework have been shown. We have seen from the graphs how the visualization works. Also we have seen the output that has shown at the end of the simulation. The main menu button here works as to reset the scenes and get back to the menu again to run another class.

Chapter 5

Conclusion

5.1 Conclusion

In this whole thesis work we actually developed a framework on algorithm visualization. As a core part of computer science education, algorithm is very important for a student. So various medium of learning websites like- Algorithm Visualizer, VisuAlgo etc are very famous and they have been developing the structure since previous decade. And if we search in the google playstore or applestore we can get a dozens of visualizing app. The developers are currently working to give a better representation for the learners. And day-by-day our learning techniques are getting modernized. There are many sites like- Github, Fosshub, SourceForge, F-Droid etc where they provides the experience and source code done by senior or excelled developers which really encourages new learners to develop a brand new framework using this type of experienced work. This process is really interesting and encouraging. In our thesis we tried to provide a new animated framework where some works are done before and some are totally new and we have added some interesting dimension in it. As a result, if my juniors or fellows use this application they will be highly benefited on the way of their learning courses. This type of animated system really slows down our boredom for study. In addition I can say that that type of work actually motivates the others to make the system of education more interesting and effective.

5.2 Future Work

The usage of artificial intelligence in our life is beyond description. By using A.I we are making any impossible things possible. And A.I is getting strong day-by-day by improving technique of algorithms. By using these algorithms we are prospering towards a bright future. So understanding these algorithms are very essential for us. To serve this purpose we developed this framework. For the future, our work can be expanded in many ways. We can consider a lot of scenarios for each of the classes of algorithm so that a learner can easily come to know how to face different sorts of problem in exam or further use. We can also make the animation work more interesting by developing a new gaming framework which can encourage developers to add more dimensions. By improving our ideas and visions, we can maximize our development.

References

[1] C. D. Hundhausen, S. A. Douglas and J. T. Stasko, ‘A meta-study of algorithm visualization effectiveness,’ *Journal of Visual Languages & Computing*, vol. 13, no. 3, pp. 259–290, 2002 (cit. on pp. 1, 5).

[2] T. L. Naps, ‘Jhavé: Supporting algorithm visualization,’ *IEEE Computer Graphics and Applications*, vol. 25, no. 5, pp. 49–55, 2005 (cit. on p. 1).

[3] S. Grissom, M. F. McNally and T. Naps, ‘Algorithm visualization in cs education: Comparing levels of student engagement,’ in *Proceedings of the 2003 ACM symposium on Software visualization*, 2003, pp. 87–94 (cit. on p. 1).

[4] V. Karavirta and C. A. Shaffer, ‘Creating engaging online learning material with the jsav javascript algorithm visualization library,’ *IEEE Transactions on Learning Technologies*, vol. 9, no. 2, pp. 171–183, 2015 (cit. on p. 1).

[5] T. L. Naps and B. Swander, ‘An object-oriented approach to algorithm visualization—easy, extensible, and dynamic,’ in *Proceedings of the twenty-fifth SIGCSE symposium on Computer science education*, 1994, pp. 46–50 (cit. on p. 2).

[6] C. Kann, R. W. Lindeman and R. Heller, ‘Integrating algorithm animation into a learning environment,’ *Computers & Education*, vol. 28, no. 4, pp. 223–228, 1997 (cit. on p. 2).

[7] S. S. Abu-Naser, ‘Developing visualization tool for teaching ai searching algorithms,’ 2008 (cit. on pp. 5, 37).

[8] J. Urquiza-Fuentes and J. Á. Velázquez-Iturbide, ‘A survey of successful evaluations of program visualization and algorithm animation systems,’ *ACM Transactions on Computing Education (TOCE)*, vol. 9, no. 2, pp. 1–21, 2009 (cit. on p. 6).

[9] V. Lazaridis, N. Samaras and A. Sifaleras, ‘An empirical study on factors influencing the effectiveness of algorithm visualization,’ *Computer Applications in Engineering Education*, vol. 21, no. 3, pp. 410–420, 2013 (cit. on pp. 6, 38).

[10] A. Wu, Y. Wang, X. Shu, D. Moritz, W. Cui, H. Zhang, D. Zhang and H. Qu, ‘Survey on artificial intelligence approaches for visualization data,’ *arXiv preprint arXiv:2102.01330*, 2021 (cit. on p. 6).

[11] T. Reis, M. X. Bornschlegl and M. L. Hemmje, ‘Ai2vis4bigdata: A reference model for ai-based big data analysis and visualization,’ in *Advanced Visual Interfaces. Supporting Artificial Intelligence and Big Data Applications*, Springer, 2020, pp. 1–18 (cit. on p. 6).

[12] C. A. Shaffer, M. L. Cooper, A. J. D. Alon, M. Akbar, M. Stewart, S. Ponce and S. H. Edwards, ‘Algorithm visualization: The state of the field,’ *ACM Transactions on Computing Education (TOCE)*, vol. 10, no. 3, pp. 1–22, 2010 (cit. on p. 28).

[13] M. L. Cooper, ‘Algorithm visualization: The state of the field,’ Ph.D. dissertation, Virginia Tech, 2007 (cit. on p. 28).

[14] C. A. Shaffer, T. L. Naps, S. H. Rodger and S. H. Edwards, ‘Building an online educational community for algorithm visualization,’ in *Proceedings of the 41st ACM technical symposium on Computer science education*, 2010, pp. 475–476 (cit. on p. 28).

[15] M. H. Brown and R. Sedgewick, ‘Techniques for algorithm animation,’ *Ieee Software*, vol. 2, no. 1, p. 28, 1985 (cit. on p. 28).

[16] V. Karavirta *et al.*, ‘Facilitating algorithm visualization creation and adoption in education,’ 2009 (cit. on p. 28).

[17] V. Karavirta, A. Korhonen and P. Tenhunen, ‘Survey of effortlessness in algorithm visualization systems,’ in *Proceedings of the Third Program Visualization Workshop*, 2004, pp. 141–148 (cit. on p. 29).

[18] R. Fleischer and L. Kučera, ‘Algorithm animation for teaching,’ in *Software Visualization*, Springer, 2002, pp. 113–128 (cit. on p. 29).