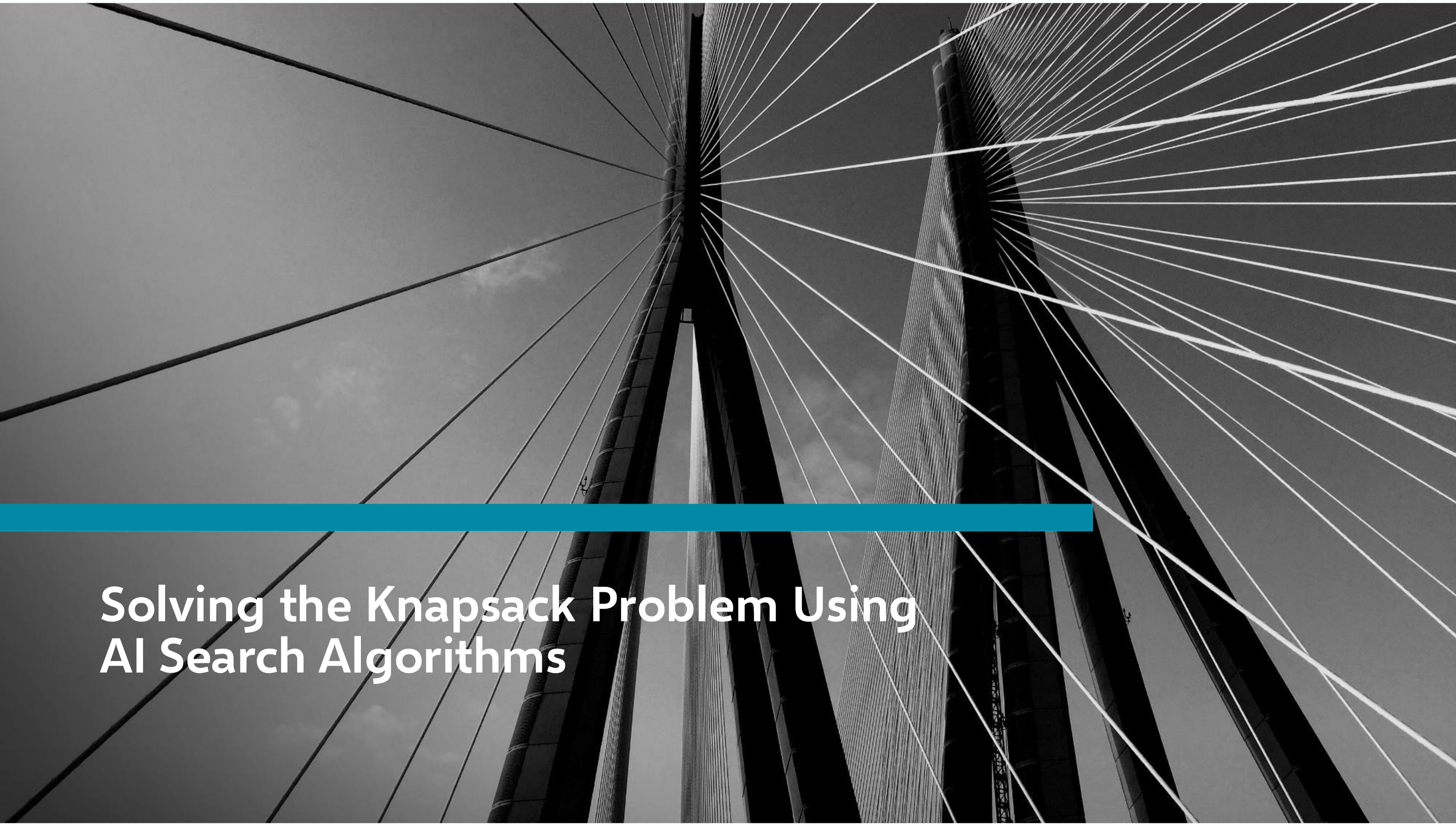
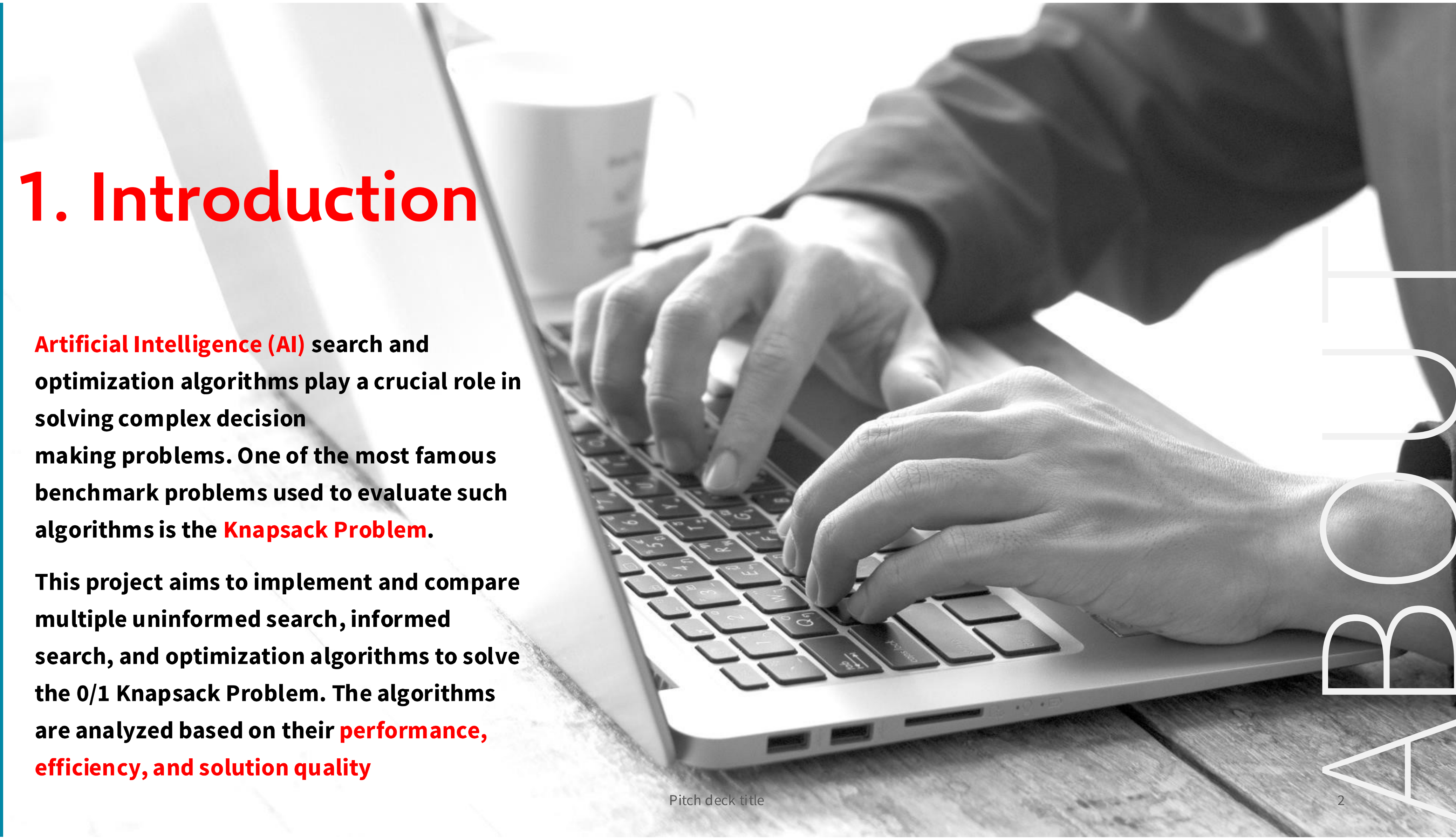


Solving the Knapsack Problem Using AI Search Algorithms

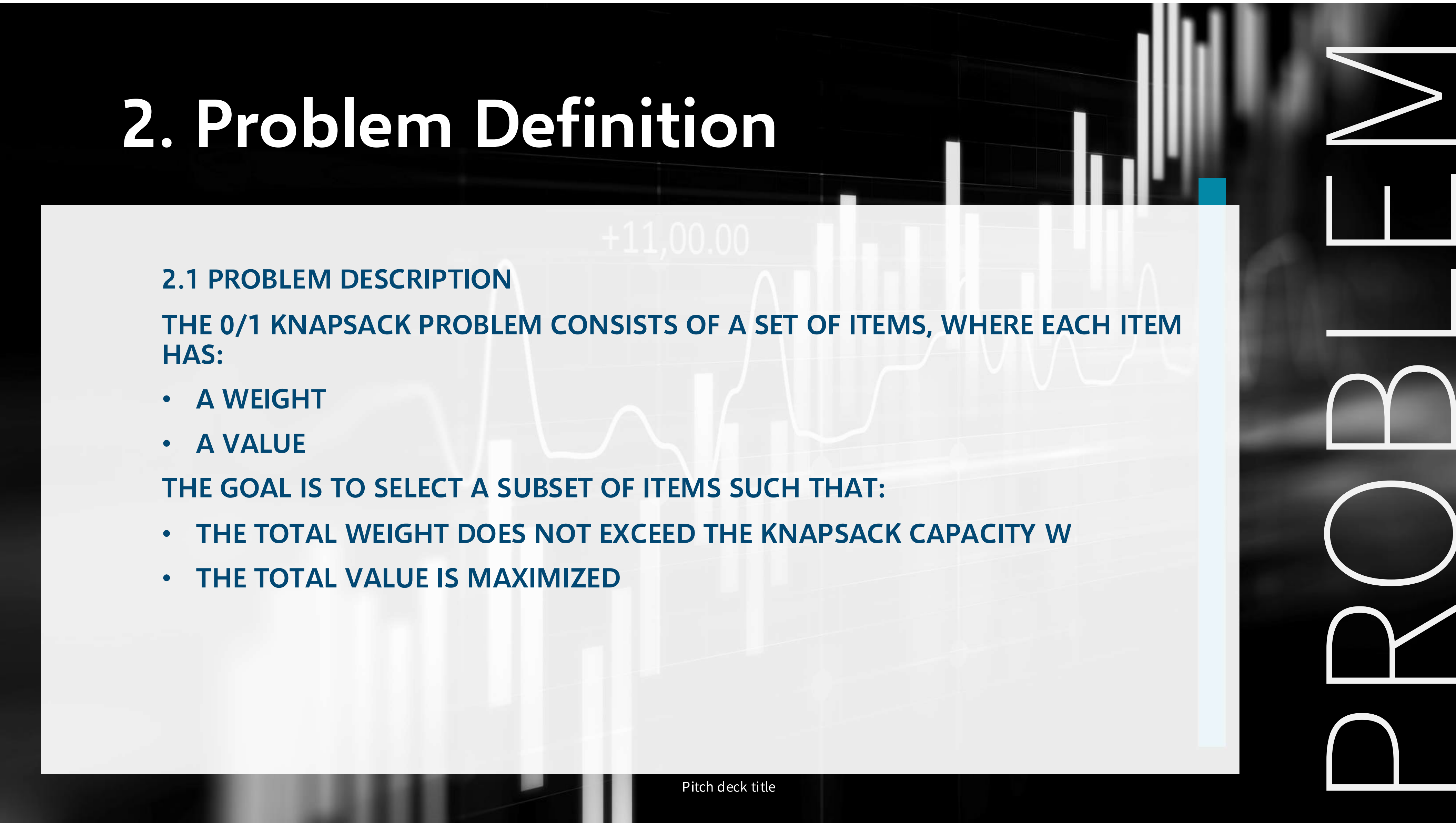


1. Introduction

Artificial Intelligence (AI) search and optimization algorithms play a crucial role in solving complex decision making problems. One of the most famous benchmark problems used to evaluate such algorithms is the Knapsack Problem.

This project aims to implement and compare multiple uninformed search, informed search, and optimization algorithms to solve the 0/1 Knapsack Problem. The algorithms are analyzed based on their performance, efficiency, and solution quality

ABOUT



2. Problem Definition

2.1 PROBLEM DESCRIPTION

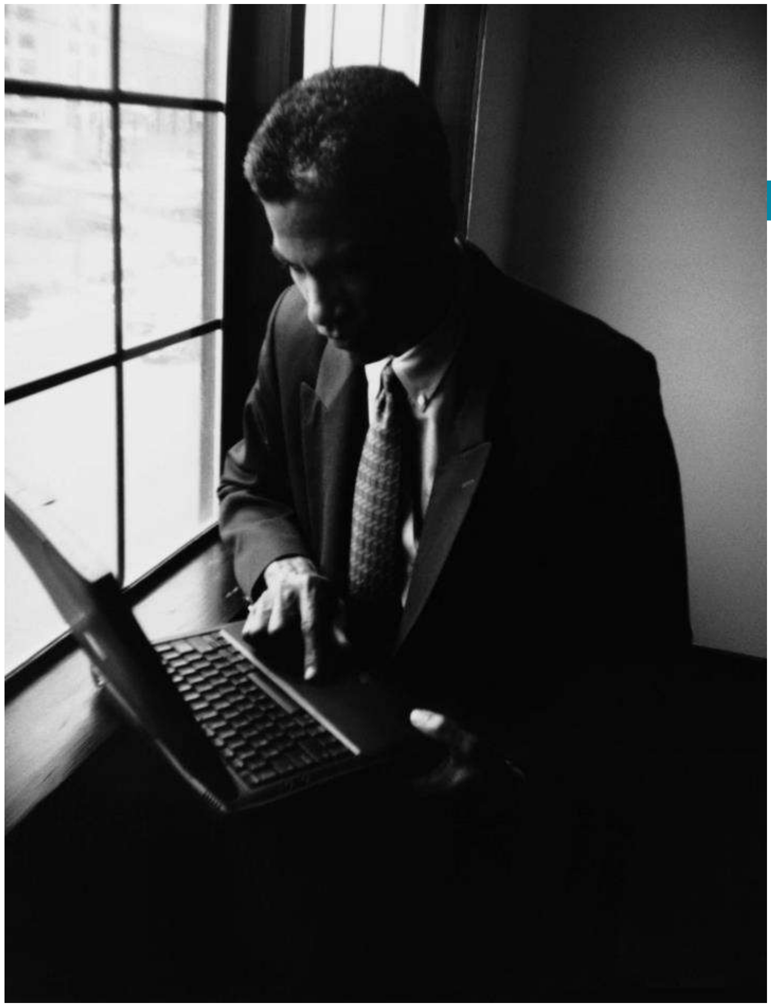
THE 0/1 KNAPSACK PROBLEM CONSISTS OF A SET OF ITEMS, WHERE EACH ITEM HAS:

- A WEIGHT
- A VALUE

THE GOAL IS TO SELECT A SUBSET OF ITEMS SUCH THAT:

- THE TOTAL WEIGHT DOES NOT EXCEED THE KNAPSACK CAPACITY W
- THE TOTAL VALUE IS MAXIMIZED

PROBLEM

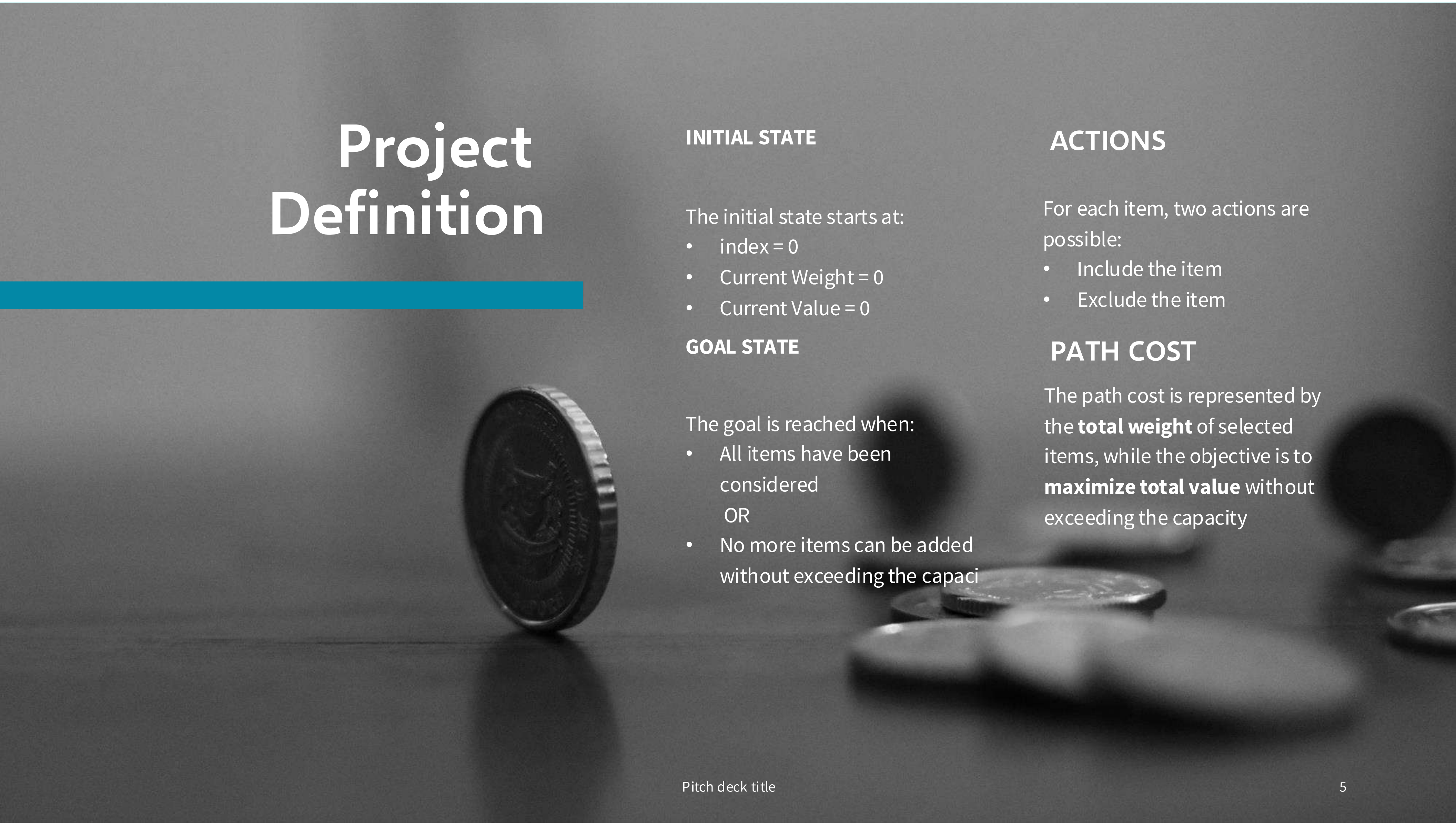


State Representation

A state is represented by:

- The current index of the item being considered
- The accumulated weight
- The accumulated value

State = (index, currentWeight, currentValue)



Project Definition

INITIAL STATE

The initial state starts at:

- index = 0
- Current Weight = 0
- Current Value = 0

GOAL STATE

The goal is reached when:

- All items have been considered
  OR
- No more items can be added without exceeding the capaci

ACTIONS

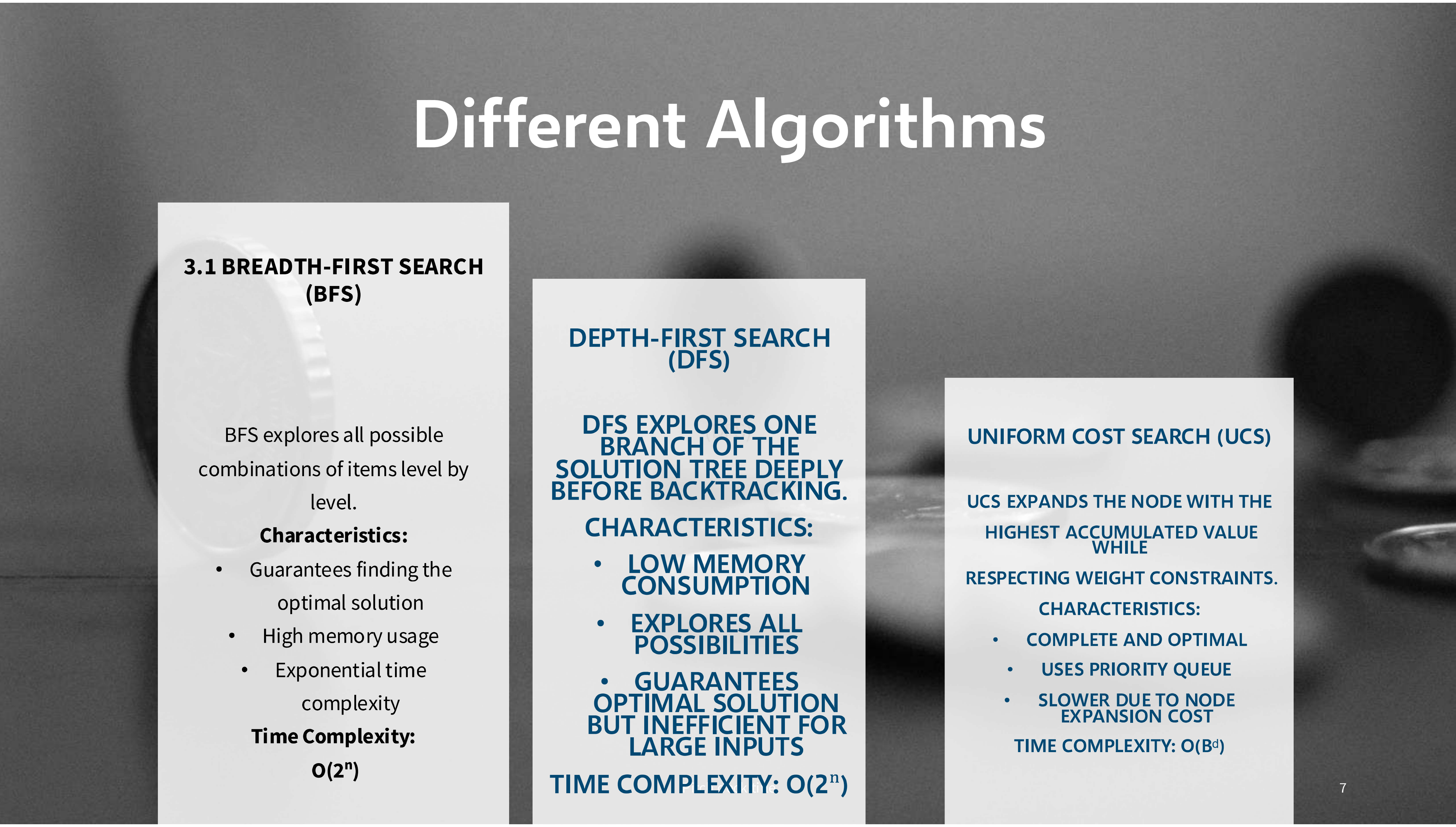
For each item, two actions are possible:

- Include the item
- Exclude the item

PATH COST

The path cost is represented by the **total weight** of selected items, while the objective is to **maximize total value** without exceeding the capacity

Implemented Algorithms



Different Algorithms

3.1 BREADTH-FIRST SEARCH (BFS)

BFS explores all possible combinations of items level by level.

Characteristics:

- Guarantees finding the optimal solution
- High memory usage
- Exponential time complexity

Time Complexity:

$O(2^n)$

DEPTH-FIRST SEARCH (DFS)

DFS EXPLORES ONE BRANCH OF THE SOLUTION TREE DEEPLY BEFORE BACKTRACKING.

CHARACTERISTICS:

- LOW MEMORY CONSUMPTION
- EXPLORES ALL POSSIBILITIES
- GUARANTEES OPTIMAL SOLUTION BUT INEFFICIENT FOR LARGE INPUTS

TIME COMPLEXITY: $O(2^n)$

UNIFORM COST SEARCH (UCS)

UCS EXPANDS THE NODE WITH THE HIGHEST ACCUMULATED VALUE WHILE RESPECTING WEIGHT CONSTRAINTS.

CHARACTERISTICS:

- COMPLETE AND OPTIMAL
- USES PRIORITY QUEUE
- SLOWER DUE TO NODE EXPANSION COST

TIME COMPLEXITY: $O(B^d)$

Different Alograthims

IDS combines the benefits of BFS and DFS by increasing depth limits gradually.

Characteristics:

- **Complete and optimal**
- **Lower memory usage than BFS**
- **Repeated computations increase runtime**

Time Complexity: $O(b^d)$

The Greedy approach selects items based on the highest **value-to-weight ratio**.

Characteristics:

- Fast and efficient
- Does not guarantee optimal solution

Hill Climbing Algorithm

Hill Climbing is a local search algorithm that improves a solution by making small changes.

Characteristics:

- Fast convergence
- Can get stuck in local maxima
- No guarantee of optimal solution

Other Algrothims

Genetic Algorithm

The Genetic Algorithm simulates natural evolution to find near-optimal solutions.

Representation:

- Chromosome: Binary vector representing item selection
- Fitness Function: Total value under capacity constraint

Operators:

- Selection
- Crossover
- Mutation

Characteristics:

- Good for large search spaces
- Does not guarantee optimal solution
- Depends on parameters such as population size and mutation rat

A * Search Algorithm

A* is an informed search algorithm that uses a heuristic function to guide the search.

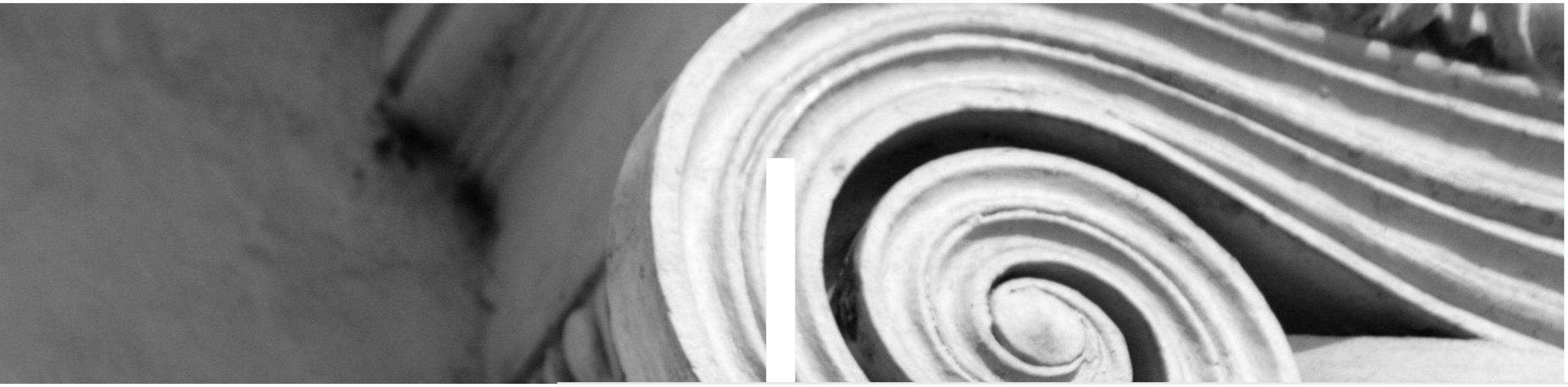
Heuristic Function:

- Estimates remaining value based on greedy filling of remaining capacity

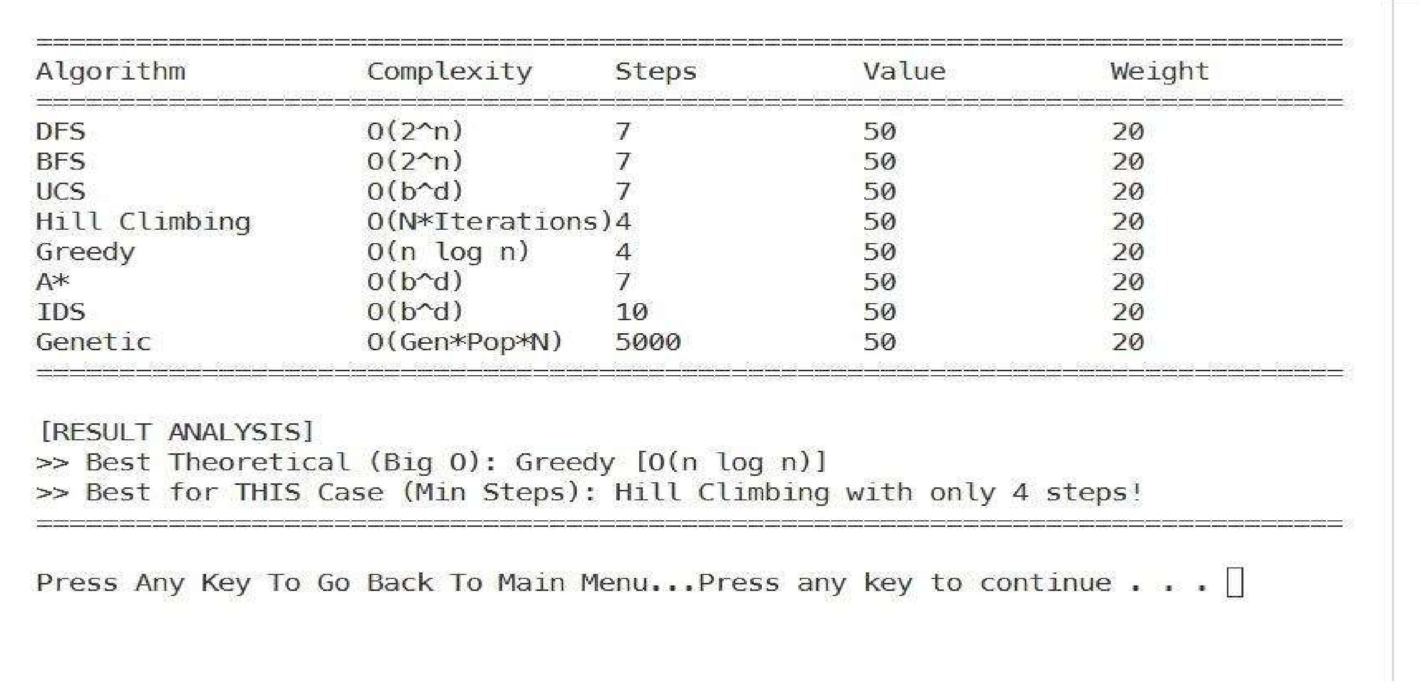
Characteristics:

- Complete and optimal
- More efficient than uninformed search
- Requires admissible heuristic

Time Complexity: $O(b^d)$



Comparison Different Algorithms



```
================================================================================
Algorithm           Complexity      Steps           Value           Weight
================================================================================
DFS                 O(2^n)          7               50              20
BFS                 O(2^n)          7               50              20
UCS                 O(b^d)          7               50              20
Hill Climbing       O(N*Iterations)4                50              20
Greedy              O(n log n)      4               50              20
A*                  O(b^d)          7               50              20
IDS                 O(b^d)          10              50              20
Genetic             O(Gen*Pop*N)    5000            50              20
================================================================================

[RESULT ANALYSIS]
>> Best Theoretical (Big O): Greedy [O(n log n)]
>> Best for THIS Case (Min Steps): Hill Climbing with only 4 steps!
================================================================================

Press Any Key To Go Back To Main Menu...Press any key to continue . . . 
```

"AFTER COMPARING THE DIFFERENT ALGORITHMS SHOWN IN THE TABLE, HILL CLIMBING IS CONSIDERED THE BEST FOR THIS SPECIFIC CASE. THE REASON IS THAT IT ACHIEVED THE OBJECTIVE IN THE MINIMUM NUMBER OF STEPS, REQUIRING ONLY 4 STEPS TO REACH THE SOLUTION, MAKING IT THE MOST EFFICIENT OPTION FOR THIS SCENARIO."

THANK YOU

Team Members :

Ahmed Rebie Ramadan Mohamed Ghazi

Ibrahim Mohammed Ibrahim Sobhy

Amr Khaled Abdelfatah Abdelsalam Zaher

Ahmed Mohammed Abdelqader Khalifa

Amr Mohammed Elsaid Ali Madara

Alaa Mohammed Abdelhamed Hola

Mohamed Reda Mostafa Elneny

Mostafa Sami Mahmoud Haboulna