SHOW N TELL ACTIVITY

TOPIC

PATH FINDING ALGORITHMS

MEMBERS

Ali Amir

Jaffar Kazmi

Ibrahim Salman

Unzila Ahsan

Rania -------

Adan --------

OVERVIEW

1. Introduction to Graph Theory
2. BFS and DFS Algorithm
3. Djikstra’s Algorithm
4. A* Algorithm
5. Optimization Techniques
6. Applications and Real world examples
7. Path finding Demo
8. Thought Provoking Questions
9. Q/A

INTRODUCTION TO GRAPH THEORY

Graph has coordinates in the form G(V,E)

V-->Set of vertices(V1,V2...Vn)

E-->Set of edges(E1,E2...En)

Vertex=Nodes(1,2,3..)

Edges=line or path(use to connect two vertex)

|V|=Total number of vertex in a graph(Order of graph)

|E|=Size of graph(Total number of edges).

Adjacent vertex:

Two vertex are adjacent vertex if they have common edges

TYPES OF GRAPH

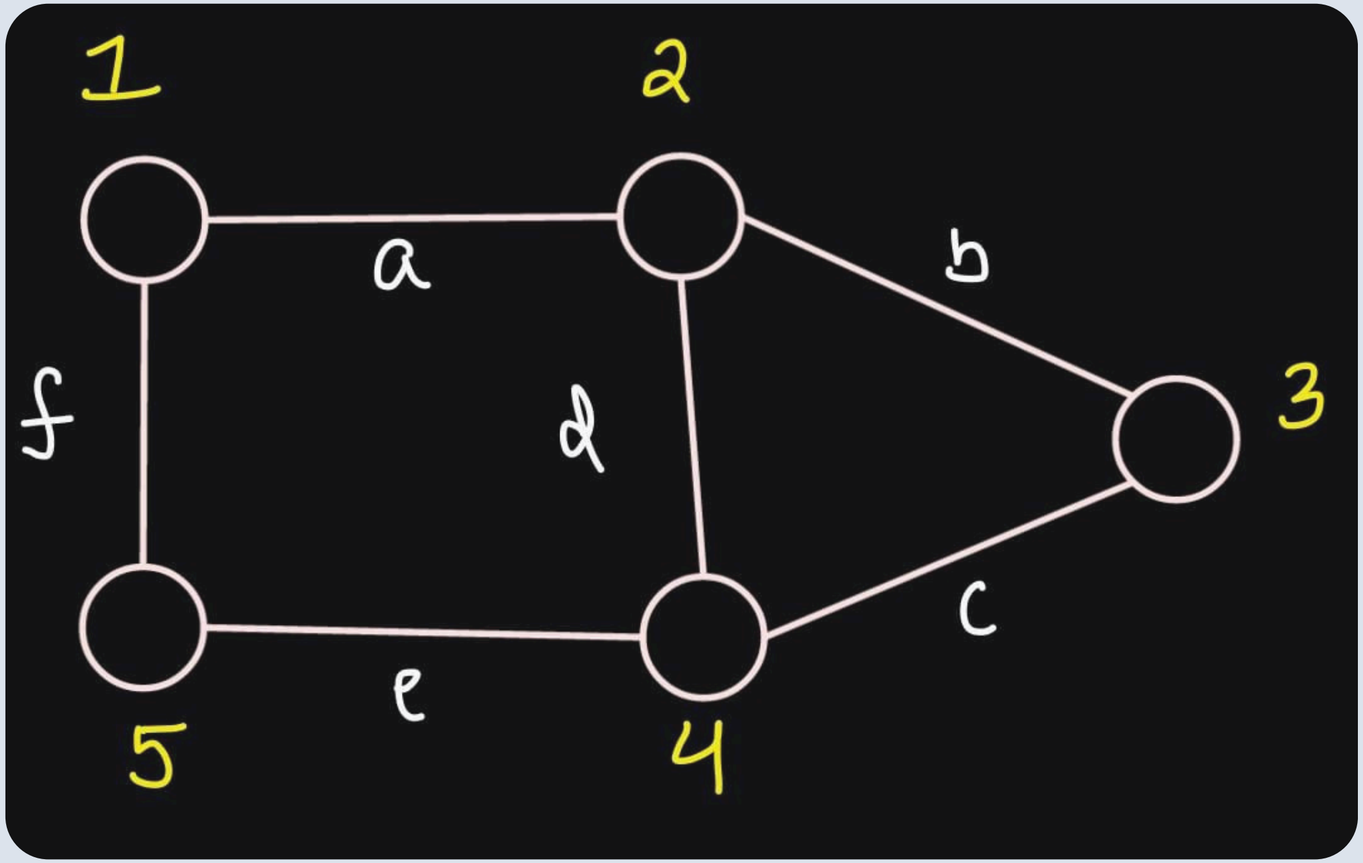
- **Directed graph(Diagraph):**
  Edges have directions
- **Undirected graph:**
  Edges have no direction
- **Weighted graph:**
  Edges have weights representing any value
- **Unweighted graph:**
  All edges are equal

BASIC EXAMPLE

Representing edges:
A(1,2)- A is between 1 and 2
F(1,5)-F is between 1 and 5

1 and 4 are not adjacent vertex

1 and 2 are adjacent vertex



BFS & DFS ALGORITHMS

Breath First Search(BFS)

Approach:

- Explores all nodes layer by layer
- uses a queue to keep the track of visited nodes

Time complexity: O(V+E)

- V: Each vertex is visited once during the exploration.
- E: Each edge is checked once while visiting neighbors of each node.

Explanation: BFS visits every node and edge in the graph, leading to a time complexity of O(V+E).

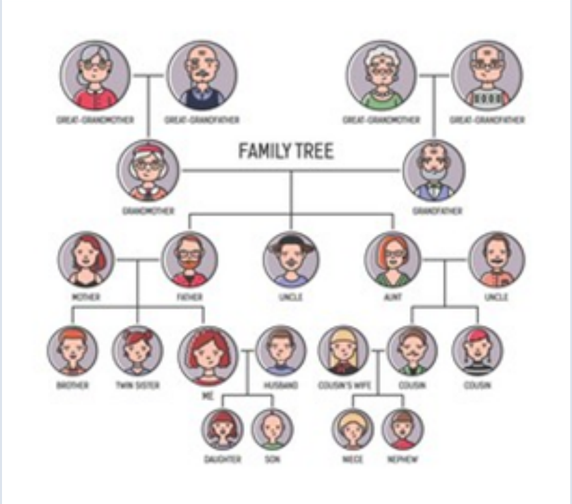
Examples:

- social network
- path finding

DFS(Depth first search)

Approach:

- it goes in depth to reach the final goal
- it uses backtrack technique (goes back to the node when get stuck)
- uses a stack to keep the track of visited nodes



Recurrence:

The recurrence for DFS represents the recursive exploration of nodes where, for each node, we recursively visit its neighbors until no more unexplored neighbors are left. This reflects the depth-first nature of DFS where exploration continues down a branch before backtracking.

Example:

- Family tree
- Exploring a file system on computer

DJIKSTRA'S ALGORITHM

A graph search algorithm used to find the shortest path between nodes in a graph.

Key Points :

- Dijkstra Algorithm uses greedy approach.Moreover,it uses the relaxation formula to find shortest path between two points.

- The relaxation formula is :

  If
  D(V)>D(U) + C(U,V)
  Then
  D(V)=D(U)+C(U,V)

- where D(V) is our destination,D(U) is our starting point and C(U,V) is the cost from U to V.

Calculation of relaxation formula :

- Initialization: Assign to every node a tentative distance. Set the initial node's distance to 0 and all other nodes' distances to infinity. Mark all nodes as unvisited.The initial node becomes the current node.

- Relaxation: For each unvisited neighbour of the current node, calculate the tentative distance through the current node. If this distance is smaller than thecurrent assigned distance of that neighbour, update the tentative distance.

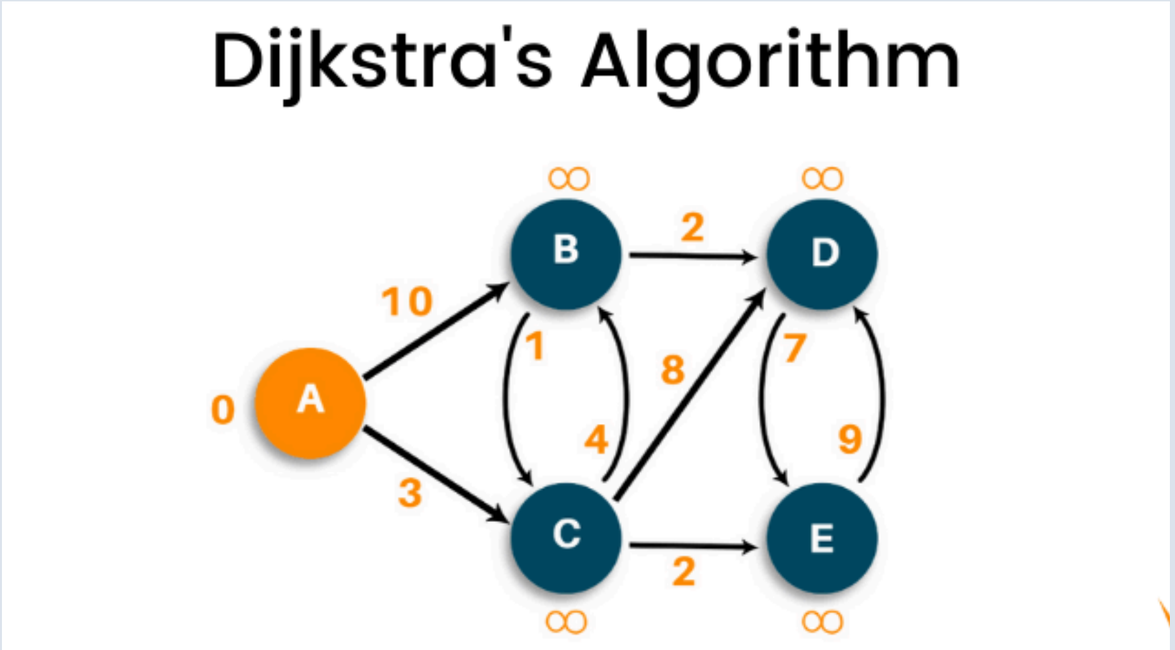
- Mark visited: Once all neighbours of the current node have been considered, mark the current node as visited. A visited node will not be checked again.

- Selection of the next node: Select the unvisited node with the smallest tentative distance, and make it the new current node

- Repeat: Repeat steps 2–4 until the destination node has been visited or all possible nodes have been processed.

Understanding with example :

The given figure is an illustration of Dijkstra's Algorithm.Here, A is our starting point.Suppose our destination is node E.



Possible Paths :

- From node A to node B,from node B to node D and node D to E.
- From node A to node B,from node B to node C and from node C to node E.
- From node A to node B,from node B to node C,from node C to node D and from node D to node E
- From node A to node C,from node C to node D and from node D to node E
- From node A to node C and from node C to node E

Dijsktra’s Algorithm will calculate all possible paths

Dijkstra Algorithm Answer :

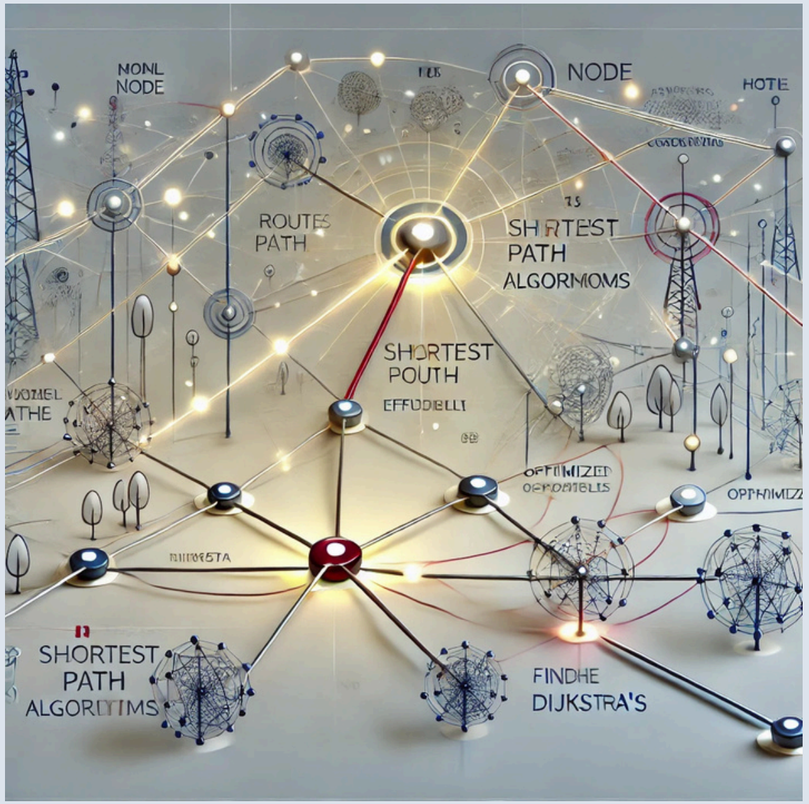
- From node A to node C and from node C to node E.

Applications Of Dijkstra’s Algorithms

Telecommunication Networks :

- **Optimizing Data Flow:**

Dijkstra's algorithm calculates the shortest route for data packets to travel, minimizing delays and reducing the chance of congestion by avoiding crowded or slow paths.



Game Development :

- ***NPC Pathfinding :***

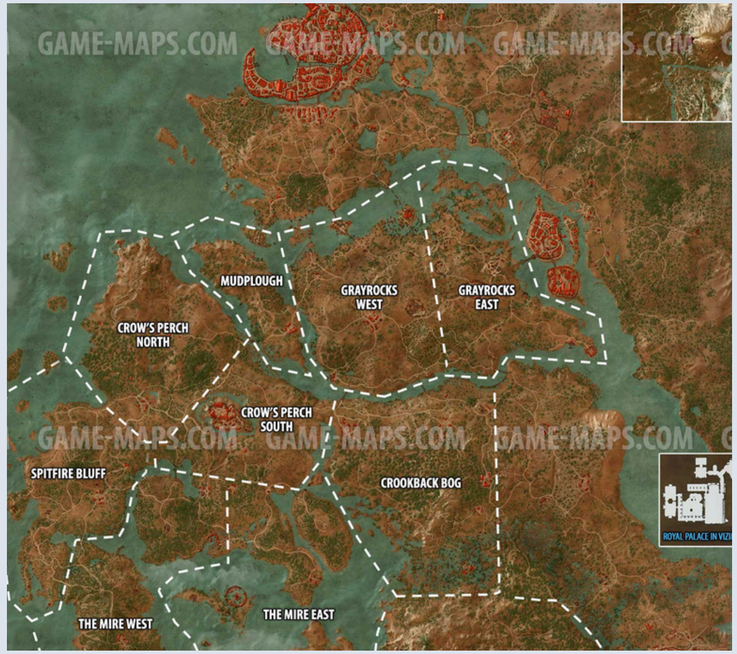
Non-player characters (NPCs) need to navigate from one point to another, avoiding obstacles and reaching goals efficiently. Dijkstra’s algorithm allows NPCs to calculate the shortest path to their destination dynamically, making them seem more intelligent and responsive.

- ***Dynamic Environments:***

Many games have changing environments (like doors opening/closing or new obstacles). Dijkstra's algorithm can adapt, recalculating paths to respond to new obstacles or changes

Role-Playing Games (RPGs) :

- In RPG like The Witcher or Skyrim, non-playable characters (NPCs) move around the game world, interacting with objects and other characters. Dijkstra's algorithm can help these NPCs find paths to specific points, like a shop or a quest location, avoiding obstacles and taking the quickest route.



This is the map of Witcher 3, if you want to go from one place to another it will use Dijkstra’s Algorithm to provide you the shortest route to that place.

RECCURENCE

A recurrence relation defines each term of a sequence in terms of previous terms, creating a relationship that can be solved step-by-step.

Example:

Fibonacci Sequence:

$F(n)=F(n-1)+F(n-2)$

with initial conditions $F(0)=0$ and $F(1)=1$

Types Of Reccurence Relation :

- Linear: Involves linear combinations of previous terms.
- Non-linear: Includes non-linear operations on previous terms.

A*
ALGORITHM

An Introduction to Pathfinding and Search algorithms

A* (pronounced ‘A-star’) is a search algorithm that finds the shortest path between some nodes S and G in a graph.

Key Points

- A* is a popular search and pathfinding algorithm used in computer science.
- Combines aspects of Dijkstra's Algorithm and Greedy Best-First Search.
- Efficient for finding the shortest path from a start point to a goal.

A*

WORKING

It uses a cost function f(n)=g(n)+h(n).

- g(n): The cost from the start node to the current node.
- h(n): The heuristic estimate of the cost to reach the goal node from the current node.

Note :) A* expands the most promising nodes by considering both the cost so far and an estimate of future cost.

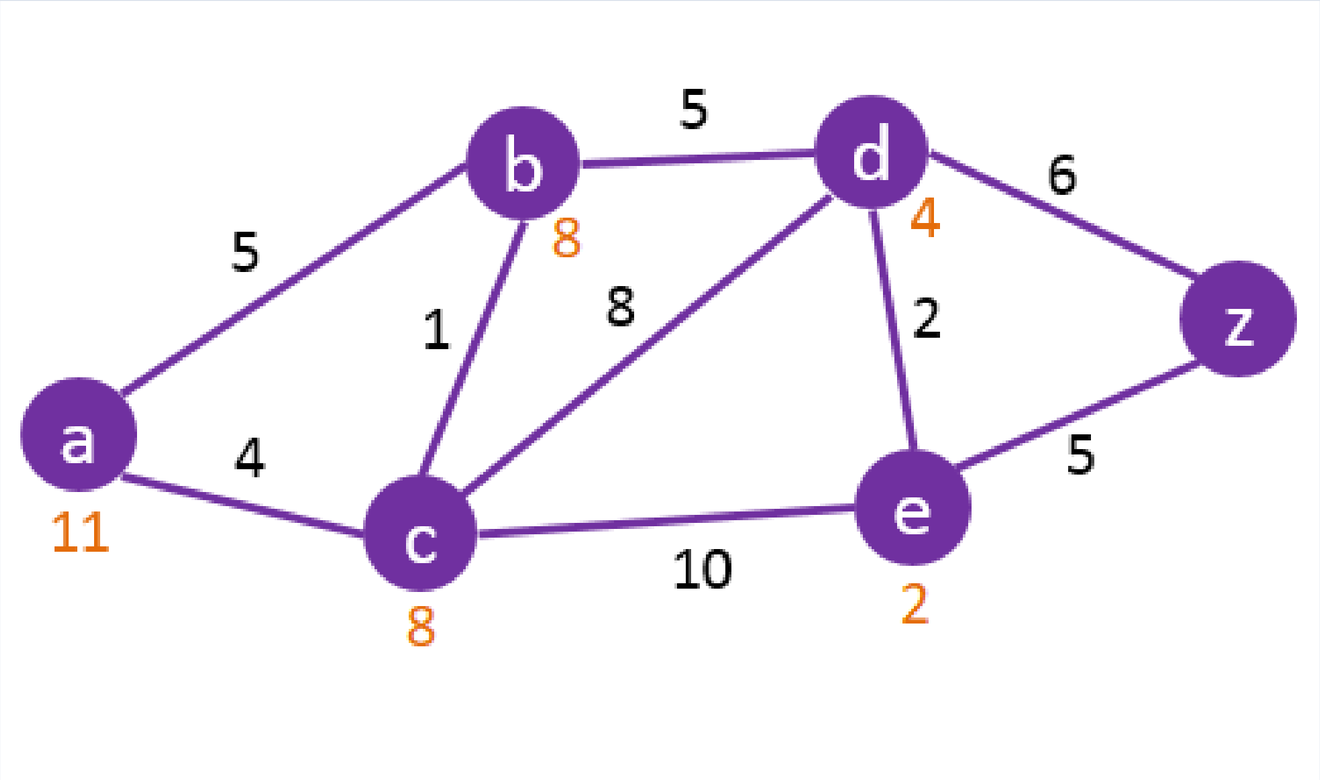
Heuristic Function:

Suppose we want to get to node G, and we are currently at node n. **Informally, a heuristic function h(v) is a function that 'estimates' how v is away from G.**

Example:

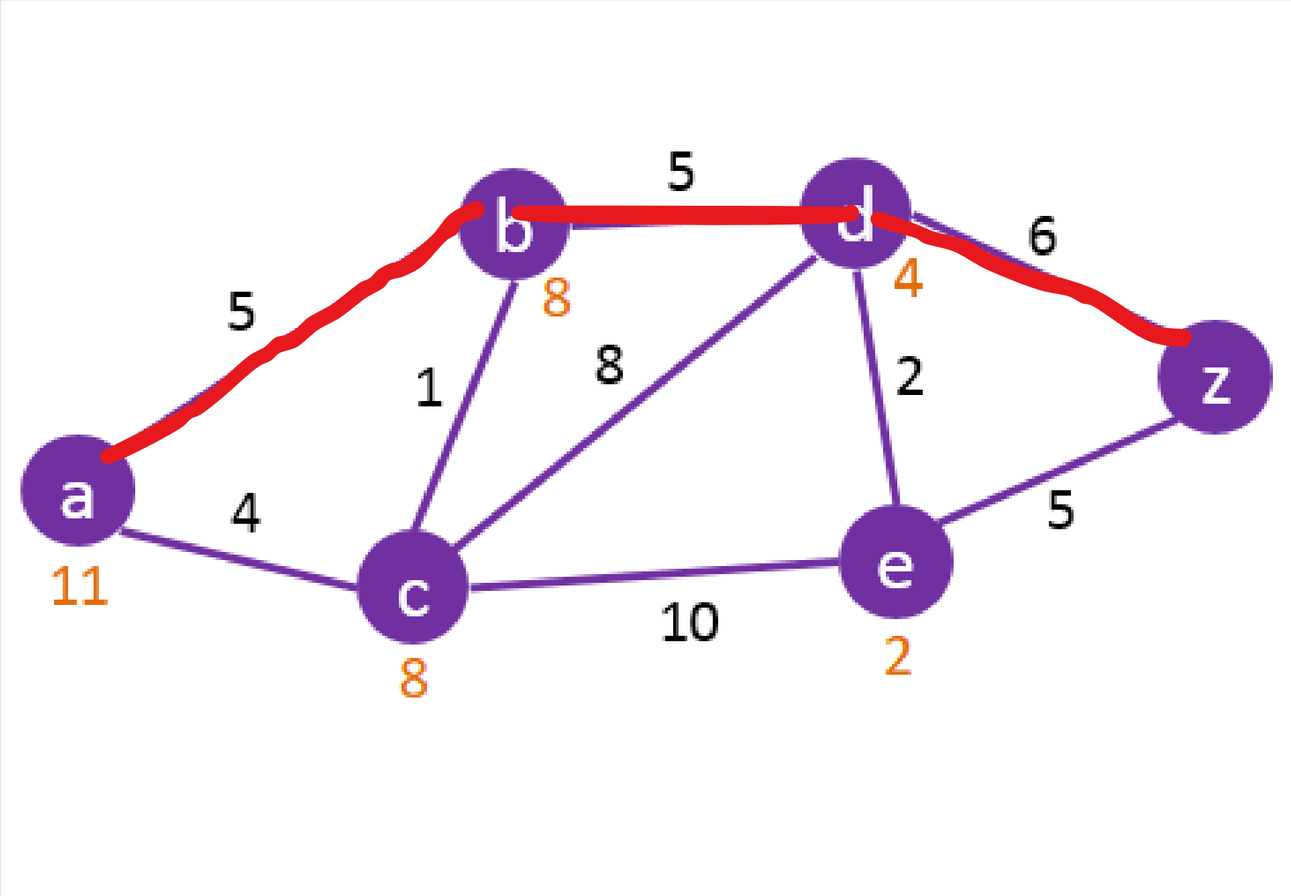
Suppose I am driving from Lahore to Hunza. A heuristic function would tell me approximately how much longer I have to drive

- The orange values represent the heuristic values h(n).
- The black values represent the actual value from start node to required node.



The shortest path from a to z found by A* is:

a→b→d→z with a total cost of 16.



What is Cost ?

Cost in A*

In the A* algorithm, cost can represent distance, time, price, energy, traffic, or risk—essentially any factor to minimize in a pathfinding problem.

Example

In GPS navigation, cost often represents time to reach the destination, while in logistics, it might represent fuel expenses or delivery fees.

Applications of A*

Network Routing:

A* is used in network routing to find the most efficient path for data packets to travel across a network.

Cost:

Here, **cost** can represent things like **delay** (how long it takes for data to travel), **network load** (how busy a route is), or **number of stops** (hops)

Heuristic:

heuristic estimates the remaining time or hops to reach the destination in this case.

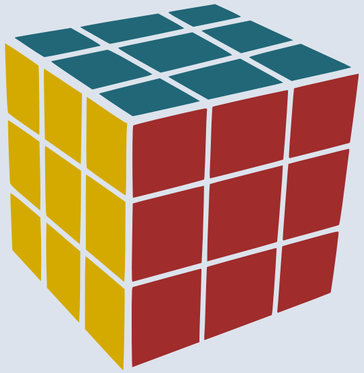
Video Game Pathfinding

A* is a common choice for pathfinding in video games, where characters or NPCs (non-player characters) need to navigate complex maps to reach a goal without bumping into obstacles.

Cost:

Cost here often represents movement difficulty or distance

Heuristic:



The heuristic helps guide characters toward their destination faster by estimating the remaining distance.

OPTIMIZATION TECHNIQUES

Our Goal

Achieve faster, resource-efficient, or near-optimal solutions, especially for complex, real-world problems.

Optimization Techniques are methods that improve an algorithm’s efficiency by reducing computation time, memory usage, or improving solution quality.

1. **Heuristics**
2. **Time Complexity**

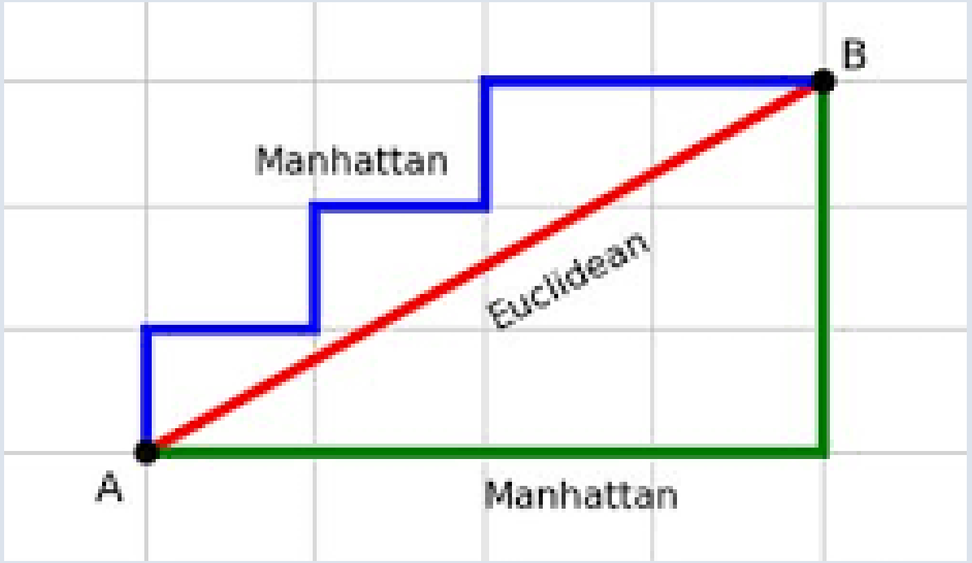
Heuristics

Definition: Heuristics are methods that provide shortcuts to find solutions faster, often trading precision for speed.

For example, In the A* algorithm, heuristics guide the search toward the goal by estimating the distance from the current node to the end.

Heuristics

1. **Manhattan Distance**
2. **Euclidean Distance**



Time Complexity

Definition: Measures the time an algorithm takes to solve a problem based on input size.

Helps select the most efficient algorithm for pathfinding, particularly in large or complex graphs.

Algorihtm	Time Complexity
BFS	O(V + E)
DFS	O(V + E)
Djiksta’s	O((V+E)logV)
A*	O((V+E)logV)

APPLICATION & REAL WORLD EXAMPLES

Applications of BFS

1. Social Networking:
    - Bluetooth
    - Finding Mutual Friends
2. GPS Navigation
3. Broadcasting

Applications of DFS

1. Backtracking Problems
    - N-Queens problem
    - Sudoku solving
2. File System Navigation
3. Family tree

Applications of Djikstra’s

1. Networking routing protocols
2. Game Development

Applications of A*

1. ----------
2. ----------

PATH FINDING DEMO

PATH FINDING DEMO

THOUGHT-PROVOKING QUESTIONS AND DISCUSSION

THANK YOU