**PATH FINDING**

Pathfinding is the computational process of determining the optimal path between two points . This process typically involves navigating through a network ,grid or graph, which consists of nodes (representing locations ) and edges (representing connections or paths between these locations) . The main objective in pathfinding is to find the shortest, least expensive , or most efficient route from a starting point (source) to a destination (goal), taking into account various factors such as distance , obstacles , and transversal costs.

**Real-World Examples:**:

* **Robotics**: Autonomous robots use pathfinding to navigate through environments.
* **Navigation Systems**: GPS systems find the best routes for vehicles.
* **Video Games**: Characters navigate through game worlds using pathfinding algorithms.

**Introduction to Path Finding Algorithms :**

Pathfinding algorithms are techniques used to determine the best path from a starting point to a destination within a given environment. These algorithms are essential in various fields such as robotics, navigation systems, and video games.

**Key Concepts:**

* **Graph Representation:** The environment is often represented as a graph, where nodes are points and edges are the connections between them.
* **Optimization:** The goal is to find the most efficient path, which could mean the shortest distance, the least time, or the lowest cost.

**What is GraphTheory?**

Graph theory is a part of mathematics that studies graphs. A graph consists of nodes (vertices) and edges (connections between nodes). Graph theory is used to represent and solve problems related to networks, paths, connections, and relationships in various domains.

**Types of graphs:**

**Undirected Graphs:** Edges have no direction, meaning if you can travel from node A to node B, you can also travel from B to A.

**Directed Graphs:** Edges have a direction, meaning you can travel from node A to node B, but not necessarily the other way around (e.g., one-way streets).

**Weighted Graphs:** Each edge has a weight or value that represents some cost, distance, or time (e.g., travel time, distance between locations).

**Unweighted Graphs:** All edges are equal, meaning no specific weight or value is assigned to the edges.

**What Does GraphTheory Say?**

Graph theory provides ways to model and solve problems involving connections and paths.

**Some key things that graph theory addresses include**

1. Connectivity

2. Shortest Path

3. Traversal

4. Graph Cycles

5. Graph Coloring

**Example in Detail:**

**1. Google Maps Navigation:**

Let’s take an example of Google Maps to illustrate Graph Theory more clearly:

**● Suppose you want to go from Point A (Home) to Point D(Work).**

**○ Nodes(Vertices):**These are the intersections or locations on the map (e.g., A, B, C, D).

**○ Edges:**Thesearetheroads connecting the intersections (e.g., A → B, B → C, C →D).

**○ Weightsonedges:**Eachroadhasaweight,which could represent the distance, travel time, or traffic conditions.

**● The Google Maps algorithm would:**

**○** Start at Point A.

○ Checkallpossible roads (edges) leading to the next points.

○ Foreachroad, it calculates the cost (distance or time).

○ Continue exploring until it reaches Point D.

○ Ateachstep, it chooses the shortest or fastest path based on the weights (travel time or distance) on the edges.

**Types of Algorithms:**

**1. Dijkstra algorithm:**

Find the shortest path from a starting node to all other nodes in a weighted graph with non-negative edge weights. The algorithm stops when all nodes are visited and the shortest path to each node is found.

**⦁ Applying Dijkstra:**

**⦁** Initialize distances.

⦁ Visit neighbors.

⦁ Move to the next closest node.

⦁ Continue (until the final node is not reached).

**⦁ Advantages in pathfinding:**

**⦁** Finds shortest path as long as all edge weights are non-negative.

⦁ Works with different types of grids such as 2D grids, road networks, etc.

⦁ Abstract graphs in computer networks.

⦁ Limitations:

⦁ Can be slow on large graphs, as its time complexity is O((V + E) log V) when using a priority queue.

⦁ No Heuristic (which can make it slower where the goal is far away).

2.Breath First Search(BFS) Algorithm:

The **Breadth-First Search (BFS)** algorithm is a graph traversal algorithm that explores the nodes of a graph level by level, starting from the source node. BFS is commonly used for pathfinding in an unweighted graph, where the goal is to find the shortest path from a start node to a target node.

**How BFS Works for Pathfinding:**

**Initialize the Search:**

* Start at the source node.
* Place the source node in a queue (FIFO).
* Mark the source node as visited.
* Initialize a parent or predecessor map (optional) to keep track of the path.
* **Explore the Graph:**
* Dequeue a node from the front of the queue.
* Explore all unvisited neighbors of the node.
* For each unvisited neighbor:
* Mark it as visited.
* Add it to the queue.
* Set the current node as its parent (for path reconstruction later).
* **Check for the Goal:**
* If the current node is the target node, the search is complete, and the shortest path can be reconstructed by tracing back through the parent map.
* If there are no more nodes to explore and the target hasn't been found, then there is no path from the source to the target.
* **Reconstruct the Path:**
* If the goal node is found, you can reconstruct the path by starting from the goal node and following the parent pointers back to the source.

**3. A\* Algorithm :**

A\* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost (least distance travelled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until the goal node is reached.

At each iteration of its main loop, A\* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A\* selects the path that minimizes

**f(n)=g(n)+h(n)**

where n is the next node on the path, g(n) is the cost of the path from the start node to n, and h(n) is a heuristic function that estimates the cost of the cheapest path from n to the goal.

**IMPLEMENTATION AND APPLICATION**

**⦁ Introduction:**

Pathfinding algorithms like Dijkstra's, A\*, and Bellman-Ford are essential in computer science and artificial intelligence. They are designed to determine optimal routes in graphs or networks, addressing shortest-path problems with precision. These algorithms are foundational for dynamic decision-making in real-time environments, driving innovations in technologies like GPS navigation, robotics, gaming, and logistics optimization. By balancing computational efficiency, accuracy, and adaptability, path finding algorithms are indispensable in addressing complex, evolving challenges.

**Application of Path Finding Algorithms:**

**⦁ Robotics and Automation**

**⦁ Driverless Vehicles**

**⦁ Human Navigation**

**⦁ In Games and Virtual Tour**

**Pathfinding System: Key Functions:**

**Path finding system will do the following jobs:**

⦁ Collecting the location information and the selected destination address from the user.

⦁ Calculating the shortest path between the starting and final points.

⦁ Sending the path information to the navigation unit.

**⦁ Flowchart:**

**The system flowchart of the game is shown below**

**Conclusion:**

Pathfinding algorithms play a pivotal role in numerous applications, from robotics and navigation systems to video games. By efficiently navigating through complex environments, these algorithms optimize paths and ensure accuracy and speed in decision-making processes. Whether it's ensuring a robot can maneuver through obstacles or helping a driver find the quickest route, the ability of these algorithms to adapt and provide optimal solutions is indispensable. Understanding these techniques not only enhances our comprehension of computational problem-solving but also paves the way for advancements in technology that can significantly impact various aspects of our daily lives. As we continue to innovate, the principles of pathfinding remain a cornerstone of intelligent system design, driving progress across diverse fields.