Date: Day:

BAHRIA UNIVERSITY ISLAMABAD CAMPUS

Assignment No 2

Assignment on: Biderectional Search Algorithm

Course Title: Artificial Intelligence

Course Code: AIC 201

Submitted By:

Name: Ammar Jamil

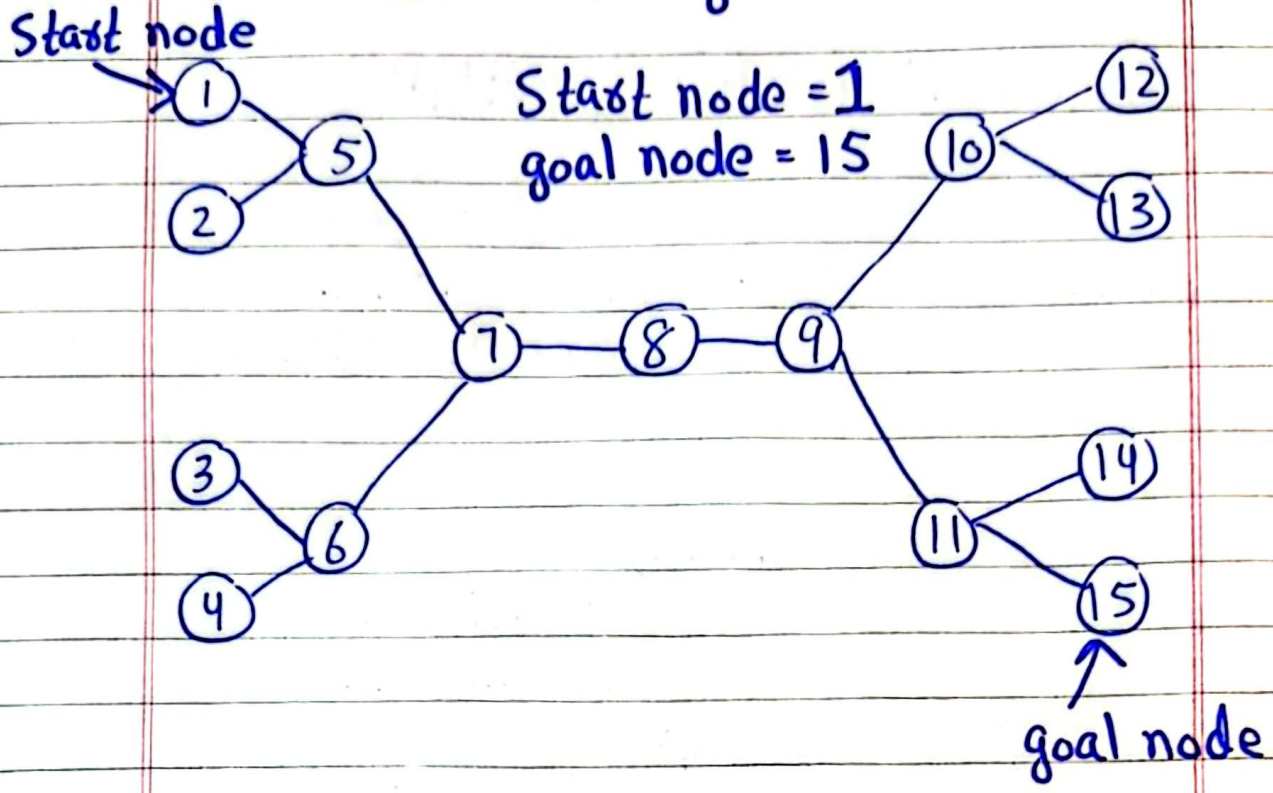
Enrollment: 01-134231-010

Submitted To:

DR. ARSHAD FARHAD

Description

Clearly design a tree/graph with start and final goal.



1 Overview and Conceptual Foundation

1.1 Provide a concise description of the Bidirectional Search Algorithm.

Sol:

Bidirectional search algorithm is a graph/tree search algorithm that finds a shortest path from an initial node to a goal node by running two **simultaneous searches**: one forward from the initial state, and one backward from the goal state, stopping when the two meet. Thus, it greatly reduces the number of nodes explored compared to the BFS. It is an uninformed search

algorithm because it does not use any heuristic or have any knowledge about the goal location.

1.2 Explain the core intuition behind it: how simultaneous forward and backward searches can lead to greater efficiency.

Sol:

The main idea behind Bidirectional search is to reduce the number of nodes explored by meeting in the middle instead of searching the whole state space.

In a normal BFS, the search space grows rapidly upto a factor of $O(b^d)$; However in Bidirectional search, two smaller searches run at the same time, one forward from the start node and the other from the goal state

How it is efficient?

If each search explores roughly half the distance, the total number of nodes visited is much smaller because the growth of the tree is exponential So,

Time complexity of Bidirectional is $\Rightarrow O(b^{d/2} + b^{d/2})$

which is a huge improvement. Hence by searching from both directions

Date: Day:

and stopping when the two frontiers meet, the algorithm meets greater efficiency in both time and space

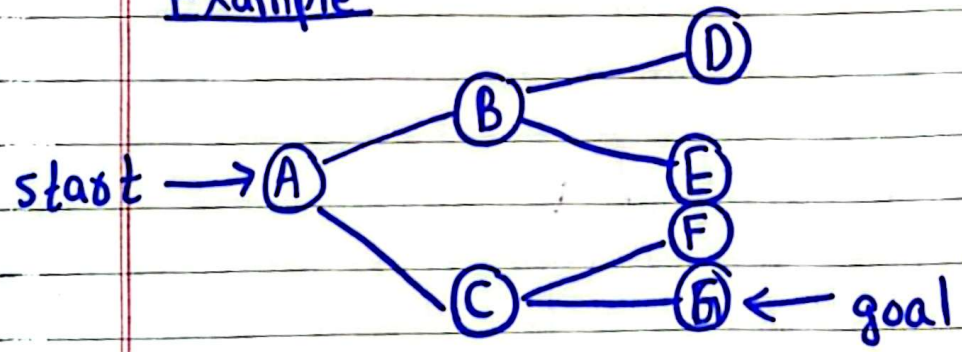
1.3 List and briefly describe the two primary variants of Bidirectional search.

Sol:

- **Bidirectional BFS:**

In Bidirectional BFS, both the forward and backward searches uses the BFS method. Each side explores the graph level by level until the two frontiers meet. It guarantees finding the shortest path in terms of the number of edges and is complete.

Example



In first step A is expanded into {B, C} and G is expanded into {C}.

Both searches meet at {C}

Hence path found ⇒ A → C → G

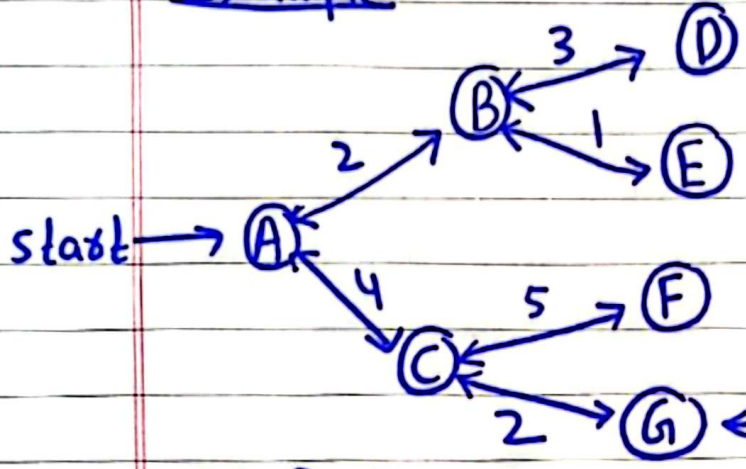
Each search explored only part of the tree, so it saved time and space.

Date: Day:

- **Bidirectional UCS:**

In Bidirectional UCS, both searches expand nodes based on path cost rather than level. It prioritizes nodes based on lower cumulative costs. In this, two seperate UCS searches expand outwards, one from the start and one from the goal. The search ends when the two expanding frontiers meet.

Example:



At first, A is expanded to {B(2), C(4)} and G is expanded int {C(2)}. The two searches meet at C

Least cost path ⇒ A → C → G

4 + 2 = 6

Hence it also ensures mimum total cost.

2. **Comparative Evaluation and Critical Analysis**

- **Theoretical Properties**: Based on your

Date: Day:

analysis and the lecture material, state and justify whether Bi-directional Search is

Sol:

- **Complete:**

A search algorithm is complete if it is guaranteed to find a solution when one exists.

Bidirectional Search is complete, because if both the forward and backward searches use a complete algorithm like BFS or UCS and the branching factor is finite, then Bidirectional search will eventually make the two frontiers meet and find the path between start and goal.

Example

start → S ↔ A ↔ B ↔ C ↔ G ← goal

In the above simple graph there are 5 nodes and there is a path between start node {S} and the goal node {G}. Both of the frontiers {S, G} will meet at point {B} and that is the solution from S — A — B — C — G.

- **Optimal:**

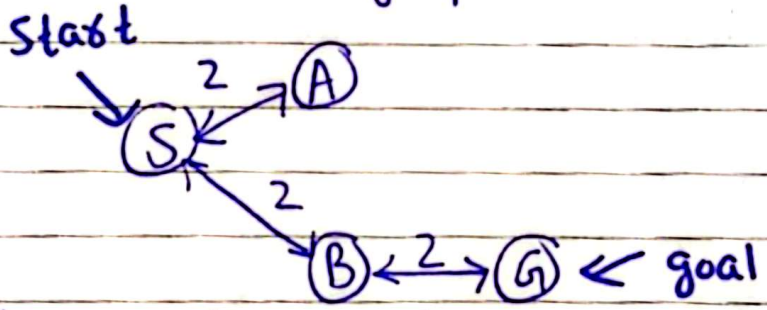
A search algorithm is optimal if it always finds the least cost shortest path.

Bidirectional Search is considered optimal only in certain cases:

With BFS, it is optimal if all edge costs are equal, because BFS explores by shortest number of steps, if the edge costs are not equal, it might find a shorter path in terms of steps but with a higher total cost.

For example:

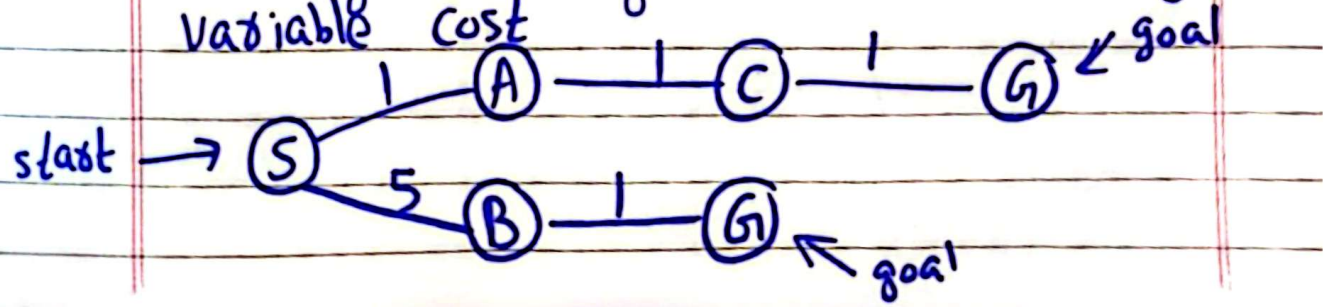
Consider the graph



Here the cost is same so the path is $S \rightarrow B \rightarrow G$

$2 + 2 = 4$

But consider the following graph having two goal states having variable cost



In the above tree, there are two possible paths from S to G.

Path 1: S → A → C → G

cost = 1+1+1 ⇒ 3

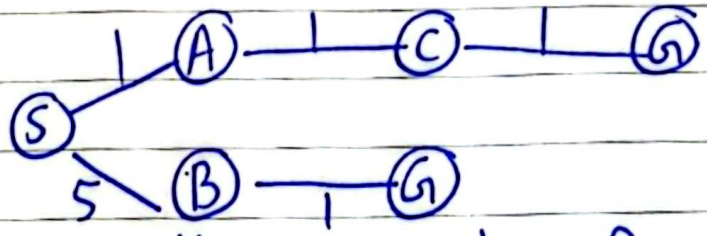
Path 2: S → B → G

cost = 5+1 ⇒ 6

As BFS only look for the number of steps. It will stop when the path from S→B→G because it requires only 1 step although the other path is optimal.

In UCS, the Bidirectional search is optimal in normal case when edges are positive and it is not optimal when some edges are negative.

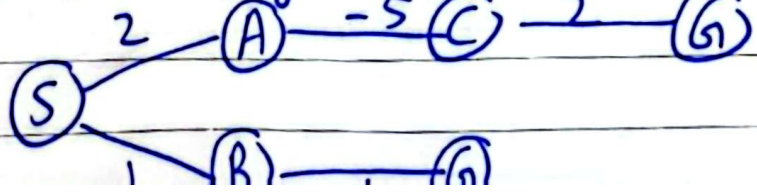
For example



The path will be from

S → A → C → G.

because UCS expands nodes on the increasing order of cost



Day:

consider the above example with -5 edge cost. UCS Bidirectional will find path S → B → G cost = 2 although the other path cost is lesser.

- **Comparative Analysis**

	Bidirectional BFS	BFS	DFS
Completeness	Complete if both the searches meet and graph is finite	Complete if the graph is finite.	Not complete. It can go indefinitely deep or stuck in loop.
Optimality	Optimal if all edges have equal cost	Optimal if all edges have equal cost	Not optimal as it may find the long path first.
Time Complexity	$O(b^{d/2})$	$O(b^d)$	$O(b^d)$
Space Complexity	$O(b^{d/2})$	$O(b^d)$	$O(b^d)$
Practical scenarios use case	Route finding in maps or navigation systems in which both the source and destination are known in advance, searching in this case will be efficient	Finding shortest path in unweighted graph like in network packet routing or social network connection.	Solving puzzles or exploring all possible path such as mazes, Sudoku or backtracking problems.

Date:

3. **Applicability**

- **First Bidirectional Search Application**

Sol:

⇒ **Route Finding in Navigation System**

Bidirectional search is highly effective in applications such as Google Maps, GPS navigation, or transport routing systems

How this search algorithm helps:

In these systems, both the starting location and destination are known, instead of exploring the entire map from one end, the algorithm searches forward from the start and backward from the destination simultaneously. When the two searches meet, the shortest path is found much faster, using far less time and memory than standard BFS or Dijkstra's algorithm

This makes it ideal for large scale routing networks, where searching from only one direction would be too slow and memory-intensive.

In addition to that, network graphs are often symmetrical so searching using Bidirectional is both meaningful and feasible.

⇒ **Social Network Connection Analysis:**

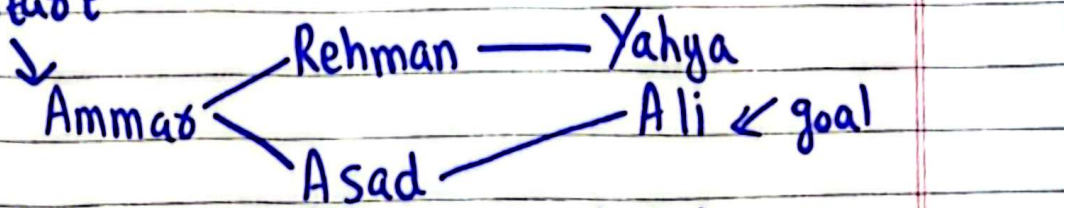
Bidirectional search is used in social network analysis to find the

Date: Day:

shortest connection between two users by running simultaneous forward and backward searches that meet in the middle, making the search more efficient

How this search algorithm helps:

In social media platform like Facebook to find degrees of connection between two users (for example Ammar and Ali have 3 mutual connection. Since both users are known (start and goal state), the search begins from both profiles simultaneously until their friend network overlap. In graphs containing millions of nodes (users) and billions of edges (friend connections), this algorithm is much faster. It reduces computational cost while guaranteeing the shortest connection path.



Both nodes (Ammar and Ali) will form a common connection through Asad node. The meeting point node (Asad) gives the shortest social link path.