GROUP 8 – BYTE  
CCS 246 – INTRODUCTION TO ARTIFICIAL INTELLIGENCE

BSCS 2-A AI  
  
TAMBALO, SELWYN

BALOGO, REGINO JR.

GALVEZ, KHEE JAY

TACUEL, ALLAN ANDREWS

**What is an Uninformed Search Algorithm?**

**Uninformed Search Algorithm** is a class of search algorithm used in artificial intelligence and computer science to explore a search space without any prior knowledge or information about the problem other than the initial state and the possible actions to take. It lacks domain-specific heuristics or prior knowledge about the problem. Uninformed search algorithms systemically explore the search space by applying predefined rules to generate successor states until a goal state is found or the search is exhausted. These algorithms are typically less efficient than informed search algorithms but can be useful in certain scenarios or as a basis for more advanced search techniques. (<https://www.analyticsvidhya.com/blog/2021/02/uninformed-search-algorithms-in-ai>)

**What is Bi-Directional Search Algorithm?**

**Bi-Directional Search Algorithm** is a form of **Uninformed Search Algorithm** where we find the shortest path between the Starting Node from and to the End Node. Bi-Directional Search Algorithm is particularly useful when searching for paths in large graphs or networks as it can significantly reduce the number of nodes explored compared to a unidirectional search, making it suitable for optimization problems were finding the shortest route is essential.

*> A simplified step-by-step overview of the algorithm:*  
**1.** **Initialization:** We create two processes, one for the forward search (from the Starting Node A), one for the backward search (from the End Node B).  
**2.** **Searching Process:**

Forward Search – We do this by starting a Breadth-First Search Process from the Point A going to Point B.  
Backward Search – We do another Breadth-First Search Process from Point B to Point A.  
Meeting Point – The desired goal state of this search algorithm is for the two search processes to reach the same node.

**3.** **Path Reconstruction:** Once the Meeting Point is found, we can reconstruct the shortest path by combining the paths found by the forward and backward search processes from their respective starting points. This path will be known as the shortest path between the two points.  
**4.** **Termination:** This step is taken after the algorithm either accomplished the goal state or the search space is exhausted.

**When do we use Bi-Directional Search Algorithm?**

We use **Bi-Directional Search Algorithm** when the branching factor for the forward and reverse direction is the same. Additionally, it is also applicable when both the source and goal vertices are specified and distinct from each other. (<https://www.educative.io/answers/how-to-use-bidirectional-search-implementation-in-python>)

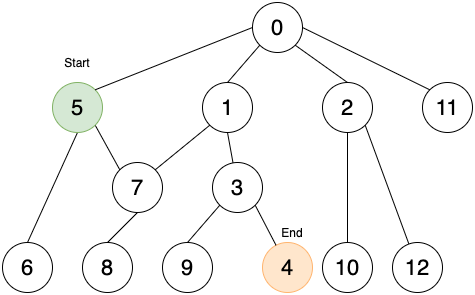
**Where do we see applications of Bi-Directional Search Algorithm?**

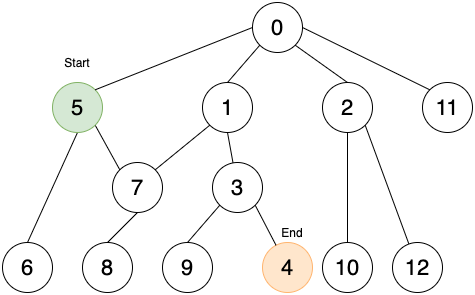
* **Route Planning and Navigation:** Navigation apps like Google Maps and Waze use bidirectional search to find the shortest path between two locations efficiently, especially in complex road networks.
* **Web Crawling:** Search engines like Google employ bidirectional search to index and discover web pages efficiently, crawling the internet from both ends.
* **Social Network Connectivity:** In social networks, bidirectional search helps find the shortest path between two users, which is useful for friend recommendations and connection analysis.
* **Protein Folding:** In bioinformatics, bidirectional search aids in finding the optimal folding structure of proteins, which is essential for understanding disease mechanisms and drug design.
* **Robotics Path Planning:** Robots and autonomous vehicles use bidirectional search to plan collision-free paths in dynamic environments.
* **Database Query Optimization:** In databases, bidirectional search can be used to optimize query execution plans by simultaneously searching for results from both the data source and the query goal.
* **Game AI:** In board games like chess or checkers, bidirectional search helps AI agents find the shortest path to victory by exploring moves from both sides.
* **Puzzle Solving:** In puzzle-solving applications, such as solving mazes or Rubik's cubes, bidirectional search can find the shortest path to a solution efficiently.

(ginGoogle, di ko run matrace ang sources.)

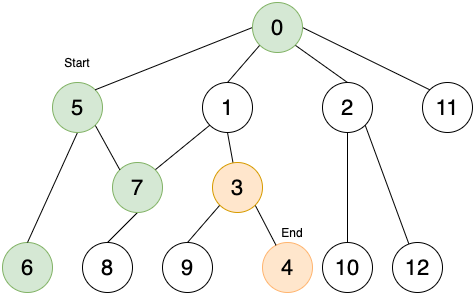
*>Here is a better visual aid from an online website explaining the process behind Bi-Directional Search Algorithm (*[*https://iq.opengenus.org/bidirectional-search/*](https://iq.opengenus.org/bidirectional-search/)*):*

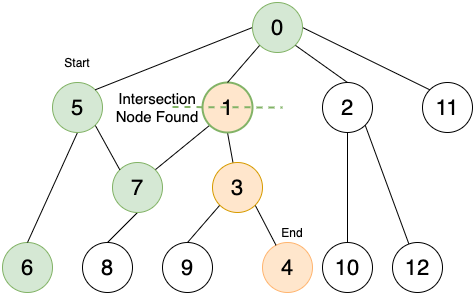
**Figure 1:**  
The start node is 5 and the end node is 4.  
Aim: To find the shortest path from 5 to 4 using bidirectional search.

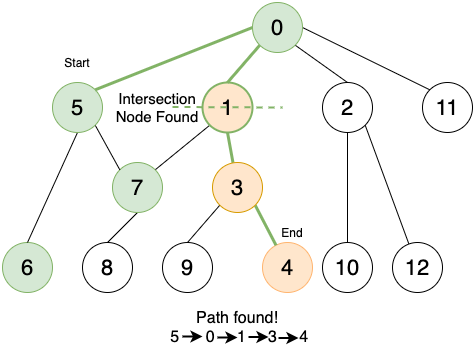


**Do BFS from both directions.  
Figure 2:**Start moving forward from start node (Green) and backwards from end node (Orange).  


**Do BFS from both directions.**

**Figure 3:**  
Similar to BFS, at every point explore the next level of nodes till you find an intersecting node.

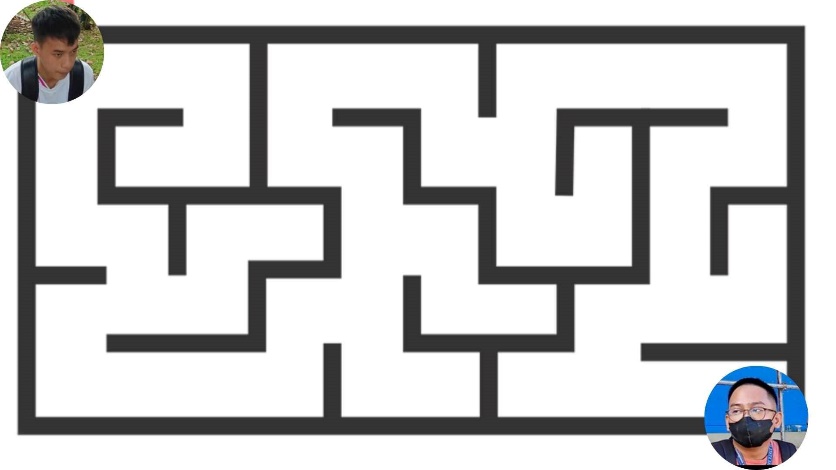
**Figure 4:**  
Stop on finding the intersecting node. ****

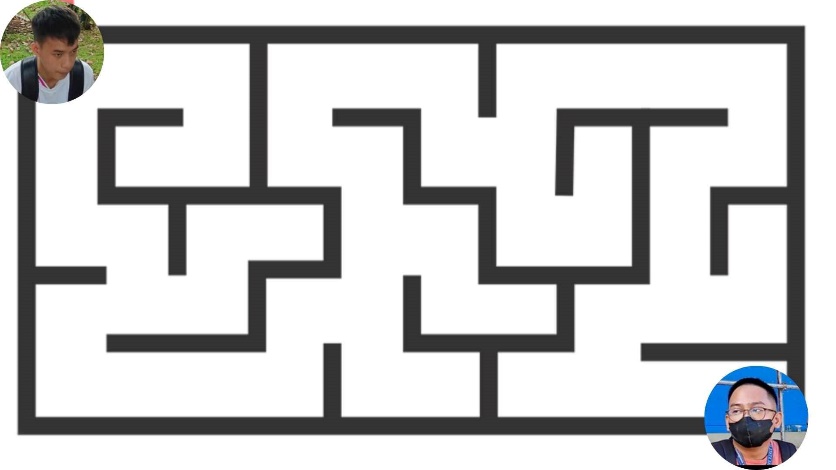
**Figure 5:**  
Trace back to find the path ****

*(Backup example:* [*https://www.educba.com/bidirectional-search/*](https://www.educba.com/bidirectional-search/)*)*

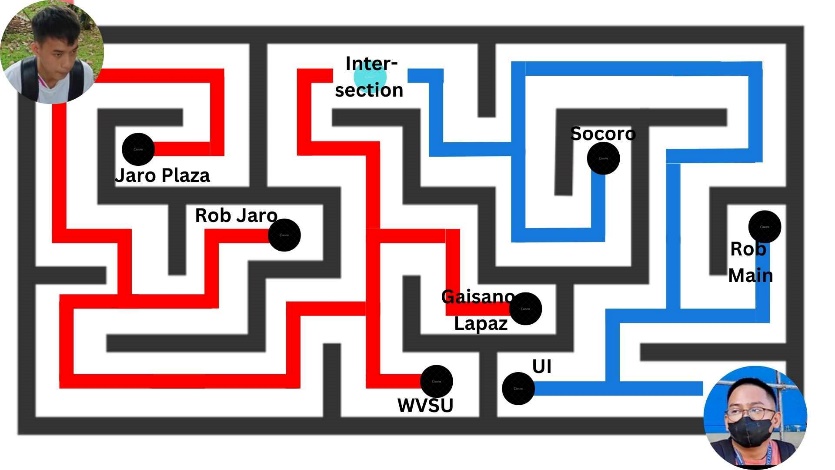
*(Source para sa pros and cons:* [*https://www.educba.com/bidirectional-search/*](https://www.educba.com/bidirectional-search/) *(amo man sa dalom lang))*

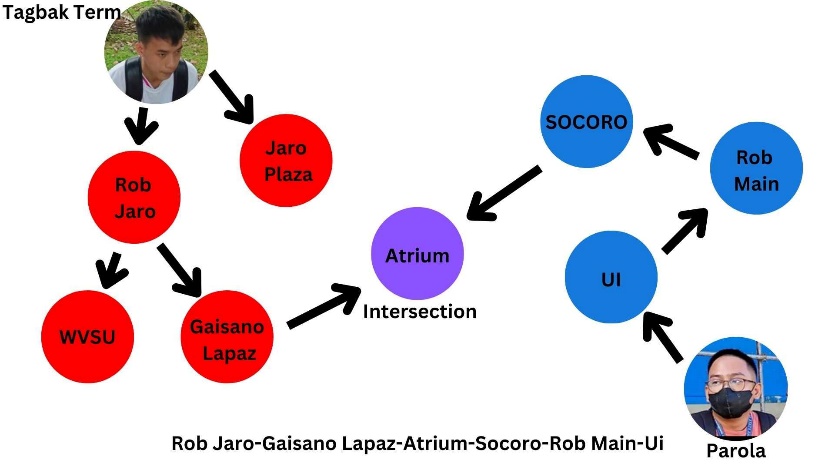
**Group Activity Questions:**

[1] *Come up with a simple problem from your (daily or weekly) routine or your favorite game features that you want to improve.*  
**Problem:** KJ wants to go to Allan at Parola but is not familiar with the path from Tagbak terminal to Parola. Allan on the other hand does not know the path from Parola to Tagbak terminal either. They need to meet as soon as possible because they have an appointment at Parola and they can’t be late.  


[2] *From this problem, justify why it is considered a problem.*  
**Response:** This is considered a problem because there are multiple ways for KJ to reach Parola and so does Allan to Tagbak. They are running out of time and they have to work together to simultaneously explore the possible paths from their end and when they meet, they will discuss how to get from Tagbak (KJ) to Parola (Allan).  
  
[3] *Use the uninformed search to solve your problem and choose what search category you’ll use (Space search or Search Tree).*

**Solution:** We will be using a Search Tree.  
**Step 1:**



**Step 2:**  


**Step 3:**  
