Assignment No: 2

1. Title of Assignment:

Implement A star Algorithm for 8 puzzle game search problems.

2. Prerequisite:

Basic knowledge of Graph, Tree , informed search, uninformed search, best first search etc.

3. Objective:

In this experiment, we will be able to do the following:

- To understand Informed Search Strategies.
- To make use of Graph and Tree Data Structure for implementation of Informed Search strategies.
- Study how A star Algorithm is useful for implementation of 8 puzzle game search problems.

4. Outcome: Successfully able to implement 8 puzzle game search problem using A star Algorithm

5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

6. Relevant Theory / Literature Survey:

Informed search

- Informed search algorithm contains an array of knowledge such as how far we are from the goal, path cost, how to reach the goal node, etc.
- This knowledge helps agents to explore less of the search space and find the goal node.
- The informed search algorithm is more useful for large search spaces.
- Informed search algorithms use the idea of heuristic, so it is also called Heuristic search

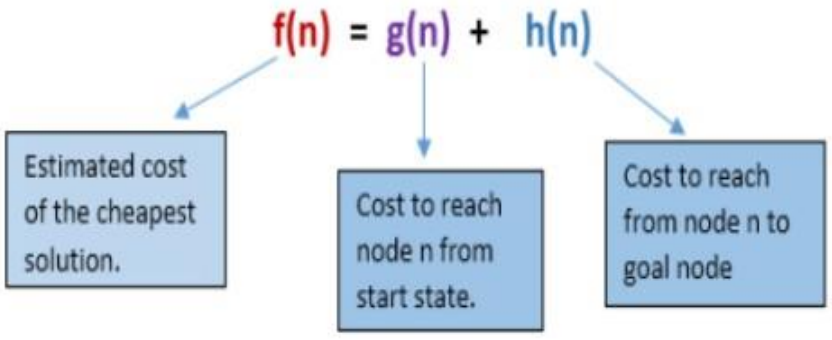
Heuristics function:

- Heuristic is a function which is used in Informed Search, and it finds the most promising path.
- It takes the current state of the agent as its input and produces the estimation of how closethe agent is from the goal.
- The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time.
- Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states.
- The value of the heuristic function is always positive.

A* Search Algorithm:

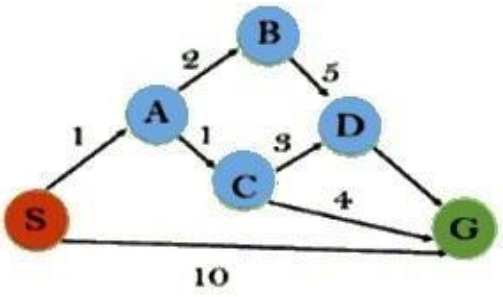
- A* search is the most commonly known form of best-first search.
- It uses the heuristic function h(n), and costs to reach the node n from the start state g(n).
- It has combined features of UCS and greedy best-first search, by which it solves the problem efficiently.
- A* search algorithm finds the shortest path through the search space using the heuristic function.
- This search algorithm expands less search tree and provides optimal results faster.
- A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.

Here we will use OPEN and CLOSED list.



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Initialization: {(S, 5)}

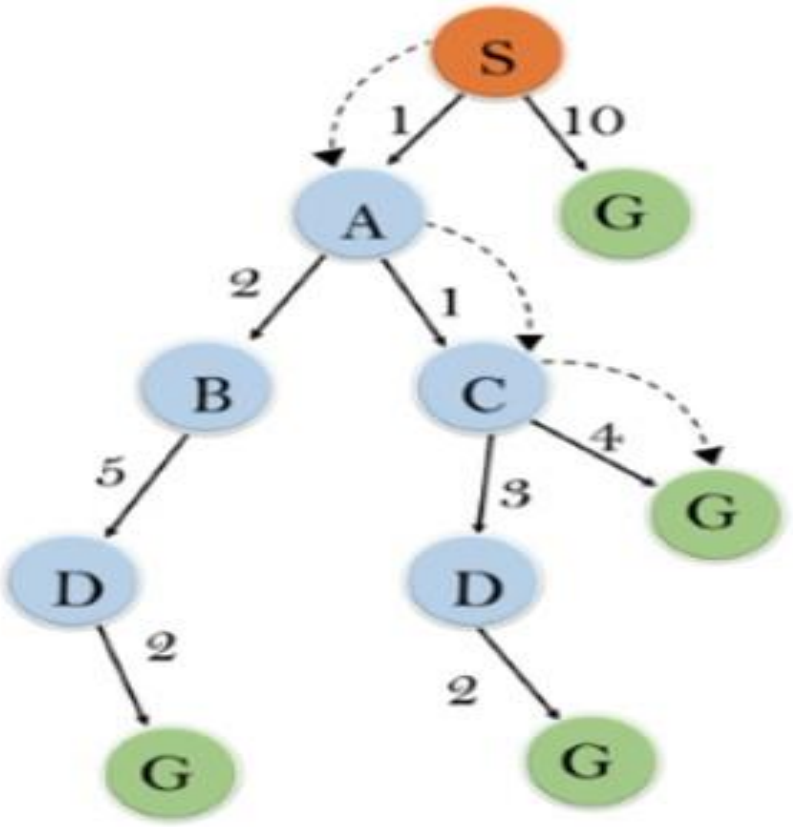
Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

Solution:



A* search Algorithm Advantages:

- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

A* search Algorithm Disadvantages:

- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Complete: A* algorithm is complete as long as:

- Branching factor is finite.
- Cost at every action is fixed.

Optimal: A* search algorithm is optimal if it follows below two conditions:

- **Admissible:** the first condition requires for optimality is that h(n) should be an admissible heuristic for A* tree search. An admissible heuristic is optimistic in nature.
- **Consistency:** Second required condition is consistency for only A* graph-search.

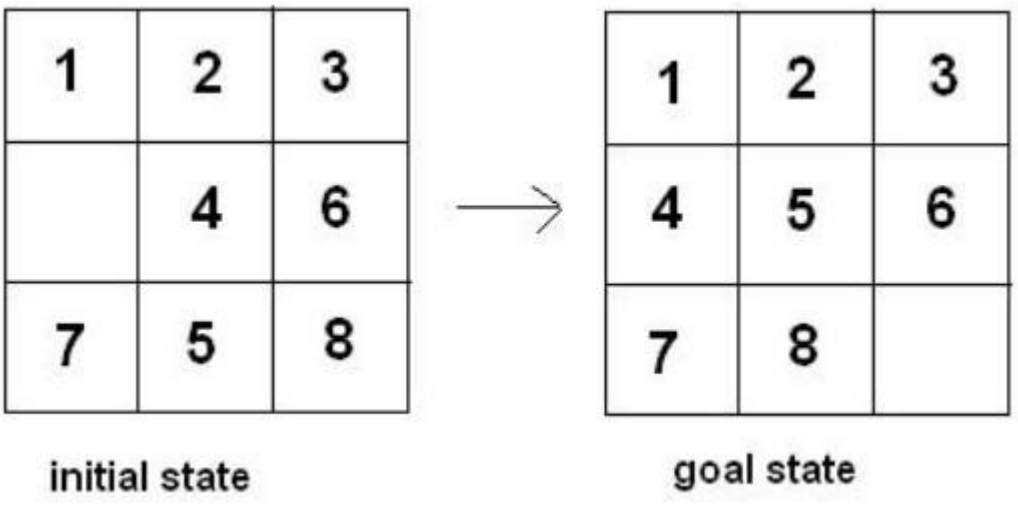
If the heuristic function is admissible, then A* tree search will always find the least cost path.

Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **O(b^d)**

8 Puzzle Algorithm:-

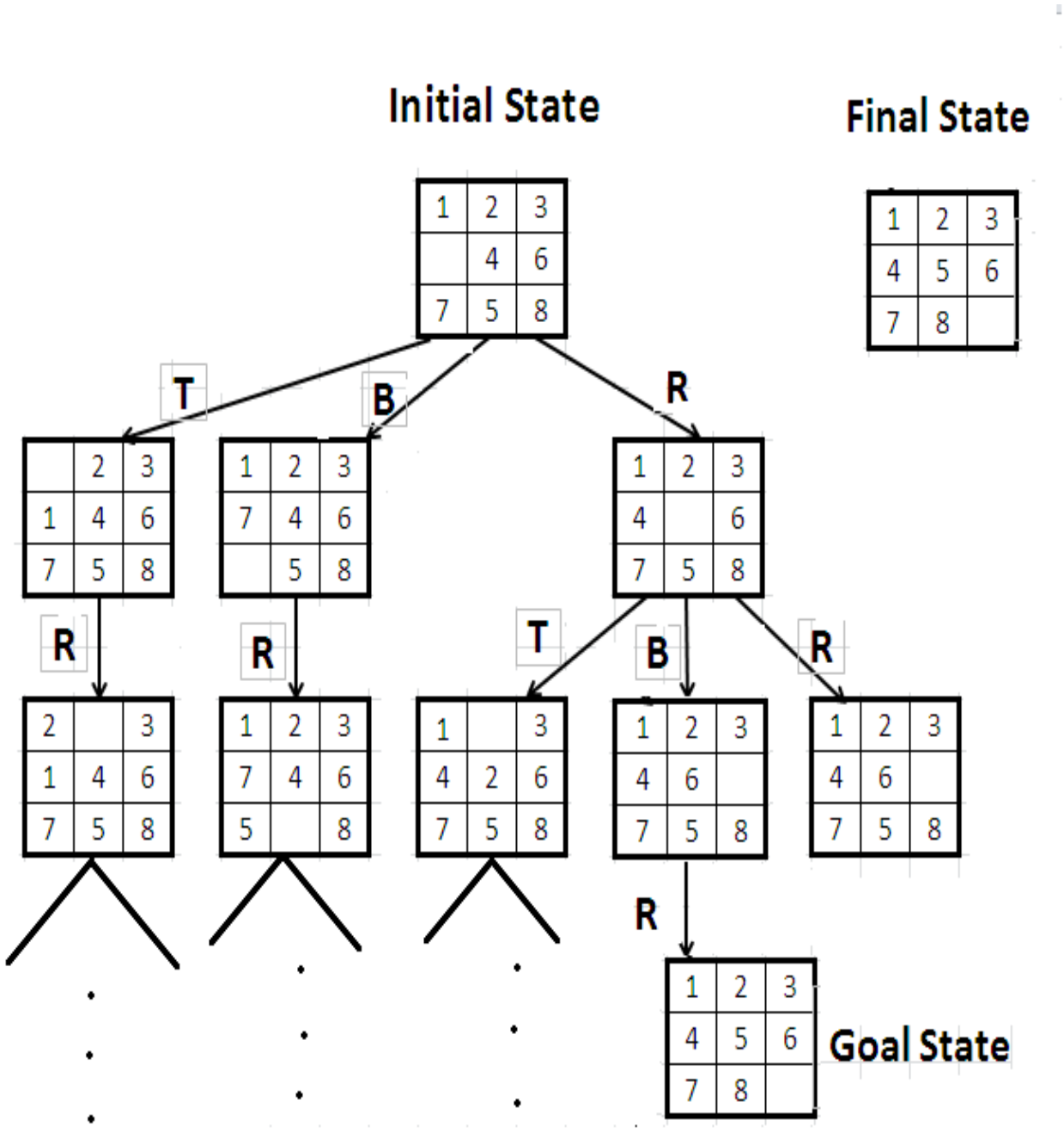
The 8-puzzle problem is a puzzle invented and popularized by Noyes Palmer Chapman in the 1870s. It is played on a 3-by-3 grid with 8 square blocks labeled 1 through 8 and a blank square. Your goal is to rearrange the blocks so that they are in order. You are permitted to slide blocks horizontally or vertically into the blank square.



There are a number of ways by which we can solve 8 puzzle problems.

- Solution without Heuristic Function
- Solution A* Algorithm

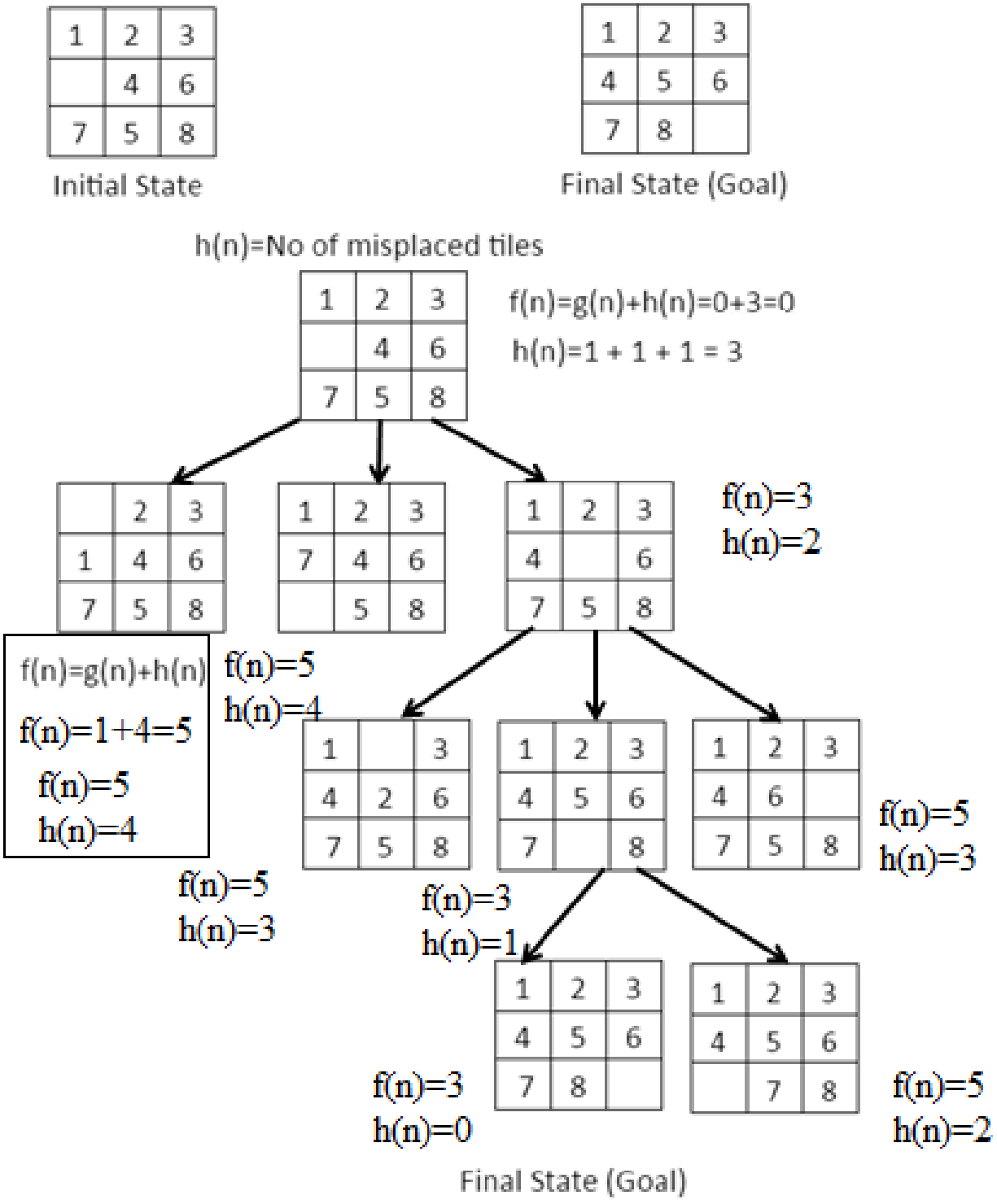
Solution without Heuristic Function



Disadvantages

need to explore each node and in case of failure need to generate its child which is a very time consuming as well as space consuming process.

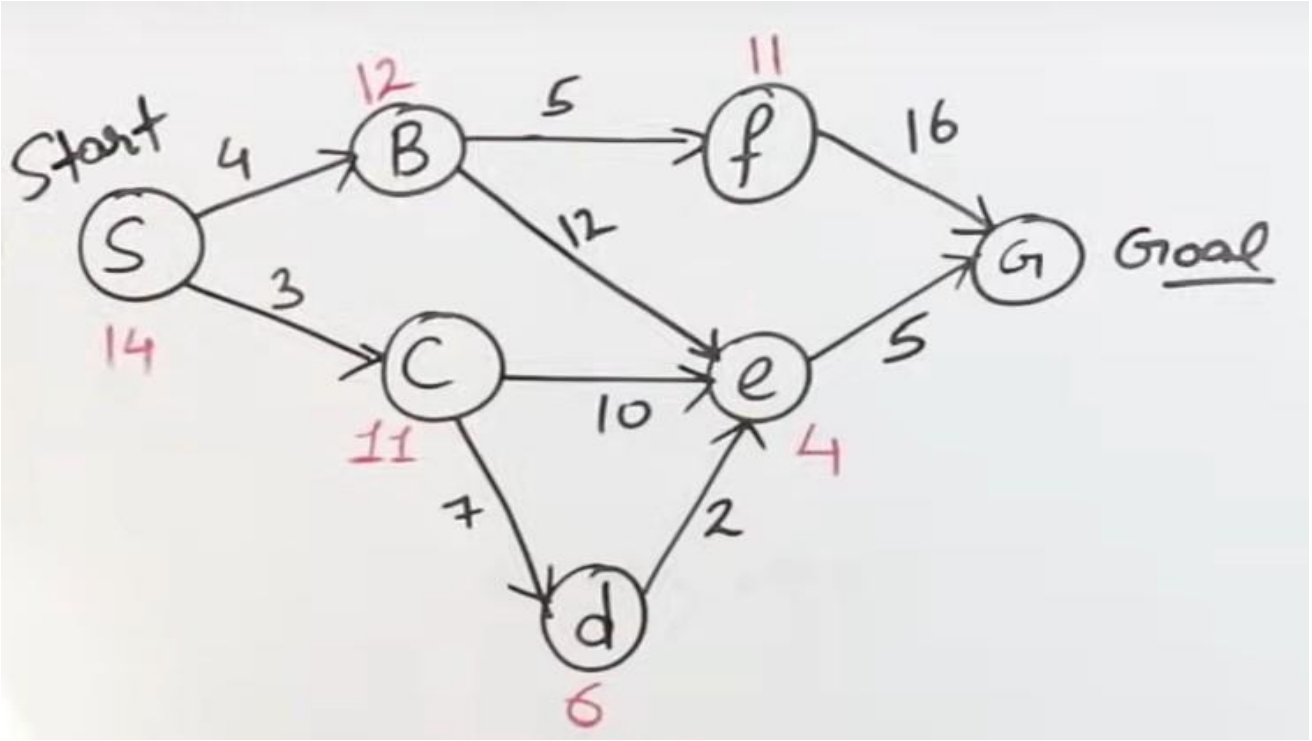
Solution A* Algorithm



7. Questions:

Q 1: Differentiate between Best first search and A star algorithm.

Q 2: Solve this problem using A star algorithm



Q 3: What is the drawback to solve 8 Puzzle problem with a non heuristic method ?

8. Conclusion:

In This way we have studied informed search strategy, how to calculate heuristic function and implementation of 8 puzzle game search problems using A star Algorithm.

Assignment No: 3

1. Title of Assignment:

Implement Greedy search algorithm for Selection Sort.

2. Prerequisite:

Basic knowledge of Greedy algorithm and Sorting concept.

3. Objective:

In this experiment, we will be able to do the following:

- Study how selection sort works under greedy search algorithm.

4. Outcome: Successfully able to sort , unsorted list of numbers.

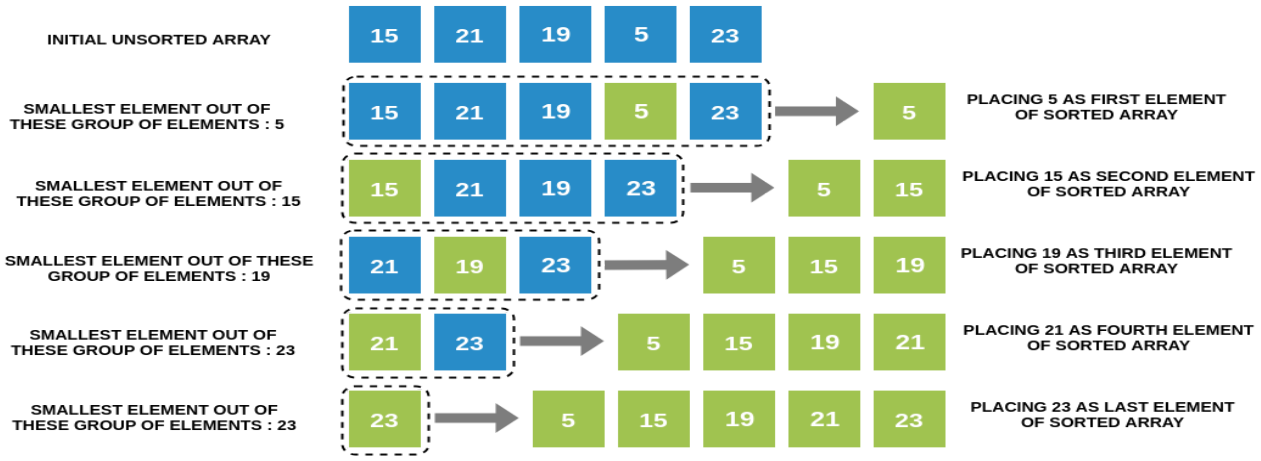
5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

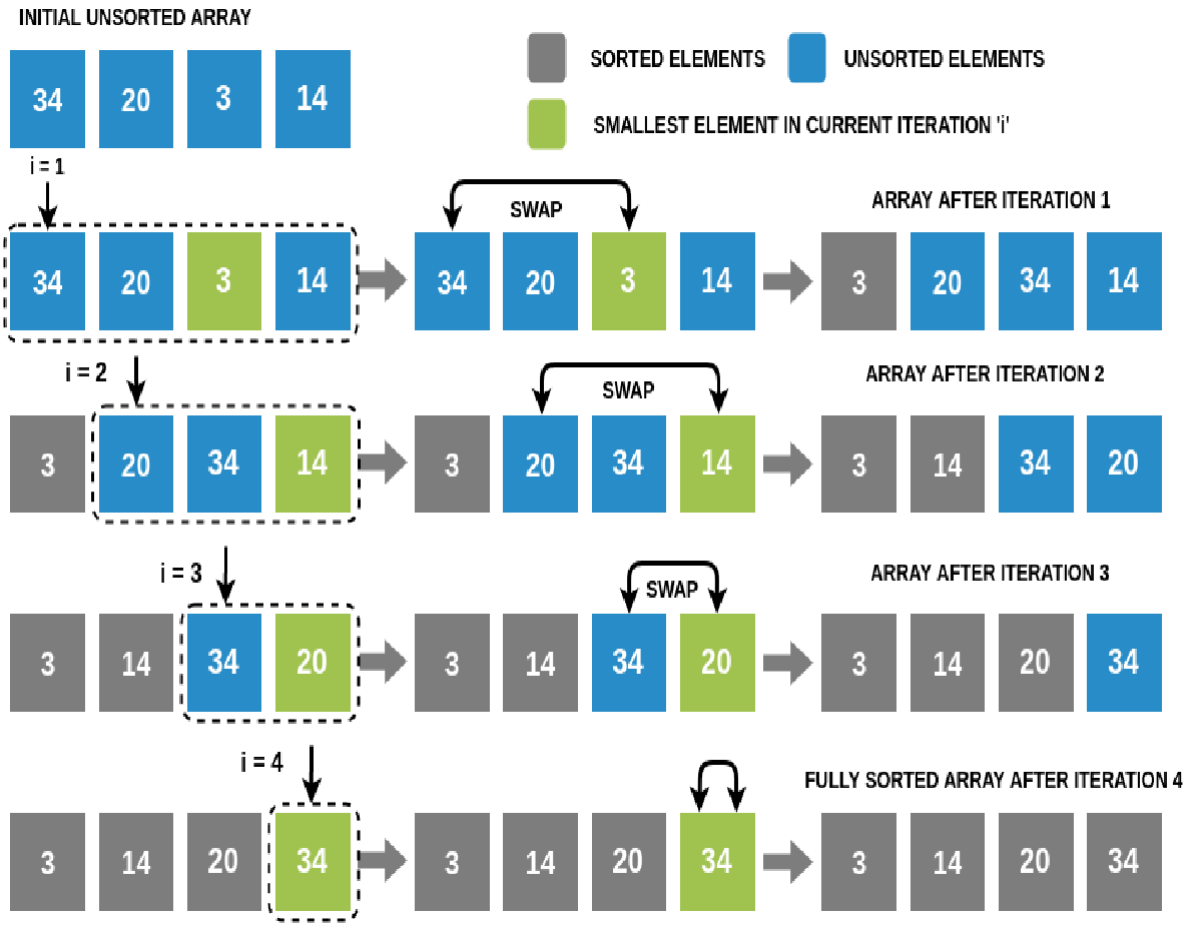
6. Relevant Theory / Literature Survey:

In Selection Sort, we take the simplest, most intuitive approach to sort an array. Choose the smallest number, place it in the first position. Then choose the next smallest number out of the remaining elements, and place it in the second position and so on till the end.

Intuition Behind the Algorithm



Selection Sort Algorithm



We will perform N-1 iterations on the array (N is the number of elements in the array). In iteration i ($1 \le i \le N-1$):

- We will traverse the array from the ith index to the end and find the smallest number among these elements. Note that if there are two smallest elements of the same value, we choose the one with the lower index.
- We will swap this smallest element with the ith element.
- Hence at the end of the ith iteration, we have found the ith smallest number, and placed it at the ith position in the array.

In the (N-1)th iteration, we will place the (N-1)th smallest element, which is the 2nd largest element in the array, at the second last position. This will leave us with one element which would already be at its correct place.This completes our sorting! Selection Sort Algorithm.

Running Time of Selection Sort

Let's assume that we are sorting N elements of a given array using Selection Sort.

- To complete one iteration, we traverse a part of the array (from index i to the end) exactly once (while keeping track of the smallest element encountered so far). Since the longest length we ever traverse in any given iteration is N (in the first iteration when i=1 -> from first to last element), time complexity of completing one iteration is O(N).
- In Selection Sort, we run N iterations, each of which takes O(N) time. Hence overall time complexity becomes O(N*N).

- Note that even if the array is fully sorted initially, Selection Sort will take O(N^2) time to complete, just as it will take for a reverse sorted or randomly sorted array.

Space Complexity of Selection Sort

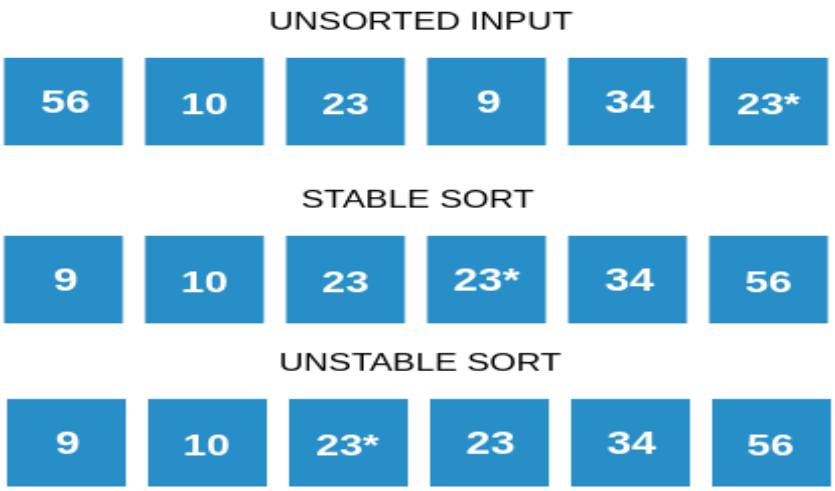
While swapping two elements, we need some extra space to store temporary values. Other than that, the sorting can be done in-place. Hence space complexity is O(1) or constant space.

Comparison with other sorting algorithms

Algorithm Sort	Algorithm Average	Time Best	Time Worst	Features Space	Features Stability
Modified Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Constant	Stable
Modified Selection Sort	$O(n^2)$	O(n)	O(n2)	Constant	Stable
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$	Constant	Stable
Insertion Sort	$O(n^2)$	O(n)	$O(n^2)$	Constant	Stable
Heap Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Constant	Unstable
Merge Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Depends	Stable
Quick Sort	O(n*log(n))	O(n*log(n))	$O(n^2)$	Constant	Stable

What is a Stable Sort Algorithm?

A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appear in the input unsorted array.



Is Selection Sort Stable?

Yes, Selection Sort is a stable sorting algorithm. When looking for the smallest element, we choose the element with lower index in case there are two or more equal elements that are the smallest elements in the array. This makes sure that we preserve the relative ordering between equal elements.

7. Questions:

Q 1: What is the time and space complexity of selection sort?

Q 2: If an array is [6,1,9,10] , sort the list by selection sort, step wise.

Q 3: What is the maximum number of comparisons in one iteration for an array of size N?

Q 4: Draw Comparison chart with other sorting techniques with respect to time and space complexity.

Q 5: What is a Stable Sort Algorithm? whether selection sort is a stable algorithm?

8. Conclusion:

In This way we have studied how to sort , unsorted list of numbers using selection sort.

Assignment No: 4

1. Title of Assignment:

Implement a solution for a Constraint Satisfaction Problem using Branch and Bound / Backtracking for n-queens problem

2. Prerequisite:

Basic knowledge of CSP, Backtracking

3. Objective:

In this experiment, we will be able to do the following:

- Study how to place N queens on board with non attacking mode using backtracking.
- What is CSP Problem

4. **Outcome:** Successfully able to place N queens on board with non attacking mode using backtracking.

5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

6. Relevant Theory / Literature Survey:

Constraint Satisfaction Problem

CSP means solving a problem under certain constraints or rules

CSP depends on three components X: It is a set of variables

D: It is a set of domains where the variables reside There is a specific domain for each variable.

C: It is a set of constraints which are followed by the set set of variables

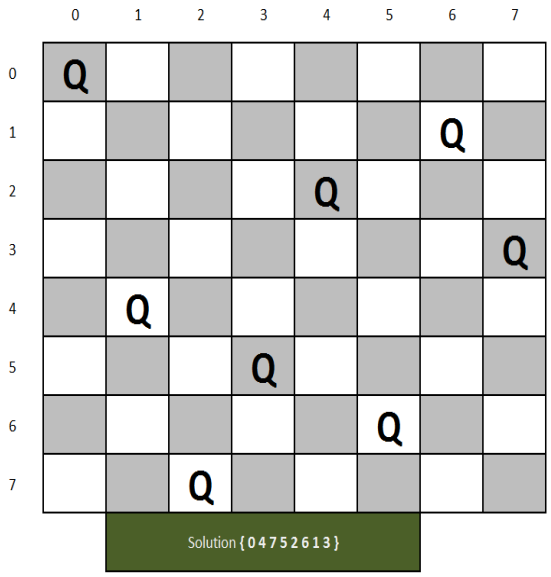
Backtracking Algorithm

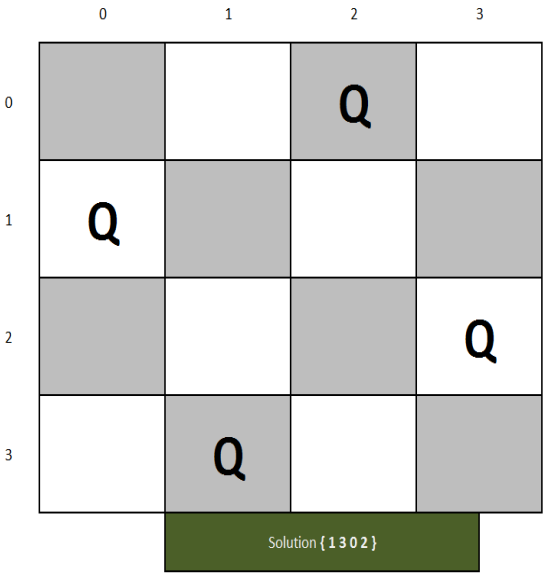
Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

N-Queens problem

The N-Queens problem is a puzzle of placing exactly N queens on an NxN chessboard, such that no two queens can attack each other in that configuration. Thus, no two queens can lie in the same row,column or diagonal.

The idea is to place queens one by one in different columns, starting from the leftmost column. When we place a queen in a column, we check for clashes with already placed queens. In the current column, if we find a row for which there is no clash, we mark this row and column as part of the solution. If we do not find such a row due to clashes then we backtrack and return false.





N-Queens problem Algorithm

1) Start in the leftmost column
2) If all queens are placed return true
3) Try all rows in the current column. Do the following for every tried row.
   a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing the queen here leads to a solution.
   b) If placing the queen in [row, column] leads to a solution then return true.
   c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.
4) If all rows have been tried and nothing worked, return false to trigger backtracking.

Branch and Bound

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These problems are typically exponential in terms of time complexity and may require exploring all possible permutations in the worst case. The Branch and Bound Algorithm technique solves these problems relatively quickly.

The branch and bound solution is somehow different, it generates a partial solution until it figures that there's no point going deeper as we would ultimately lead to a dead end.
In the backtracking approach, we maintain an 8x8 binary matrix for keeping track of safe cells (by eliminating the unsafe cells, those that are likely to be attacked) and update it each time we place a new queen. However, it required O(n^2) time to check the safe cell and update the queen.

Applying the branch and bound approach : The branch and bound approach suggests that we create a partial solution and use it to ascertain whether we need to continue in a particular direction or not.

Parameter	Backtracking	Branch and Bound
Approach	Backtracking is used to find all possible solutions available to a problem. When it realizes that it has made a bad choice, it undoes the last choice by backing it up. It searches the state space tree until it has found a solution for the problem.	Branch-and-Bound is used to solve optimization problems. When it realizes that it already has a better optimal solution that the pre-solution leads to, it abandons that pre-solution. It completely searches the state space tree to get an optimal solution.
Traversal	Backtracking traverses the state space tree by DFS(Depth First Search) manner.	Branch-and-Bound traverse the tree in any manner, DFS or BFS.
Function	Backtracking involves a feasibility function.	Branch-and-Bound involves a bounding function.
Problems	Backtracking is used for solving Decision Problems.	Branch-and-Bound is used for solving the optimization Problem.
Searching	In backtracking, the state space tree is searched until the solution is obtained.	In Branch-and-Bound as the optimum solution may be present anywhere in the state space tree, so the tree needs to be searched completely.
Efficiency	Backtracking is more efficient.	Branch-and-Bound is less efficient.
Applications	Useful in solving N-Queen Problem, Sum of subset.	Useful in solving Knapsack Problem, Traveling Salesman Problem.

7. Questions:

Q 1: Which are the constraints required to solve the N Queen Problem?

Q 2: Compare Backtracking and branch and bound methods.

Q 3: What do you mean by Constraint satisfaction problem?

8. Conclusion:

In This way we have studied how to solve the CSP problem and how to place N queens on board with non attacking mode using backtracking.

Assignment No: 5

1. Title of Assignment:

Develop an elementary chatbot for any suitable customer interaction application.

2. Prerequisite:

Basic knowledge of AI and customer interaction etc.

3. Objective:

- To create a chatbot to interact with customers.

4. Outcome:

Successfully able to implement Chatbot for different applications

5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

6. Relevant Theory / Literature Survey:

What is a chatbot?

A chatbot is a computer program designed to have a conversation with human beings over the internet. It's also known as conversational agents, which communicate and collaborate with human users, through text messaging, in order to accomplish a specific task.

Basically, there are two types of chatbots. The one that uses Artificial Intelligence, and another one is based on multiple choice scripts.

Both types of chatbots aim to create a more personalized content experience for the users, whether that's while watching a video, reading articles or buying new shoes.

These Chatbots hold the promise of being the next generation of technology that people use to interact online with business enterprises. These Chatbots offer a lot of advantages, one of which is

that, because Chatbots communicate using a natural language, users don't need to learn yet another new website interface, to get comfortable with the unavoidable quirks.

Chatbots are capable of interpreting human speech, and decide which information is being sought. Artificial intelligence is getting smarter each day, and brands that are integrating Chatbots with the artificial intelligence, can deliver one-to-one individualized experiences to consumers.

Why chatbot?

Chatbots can be useful in many aspects of the customer experience, including providing customer service, presenting product recommendations and engaging customers through targeted marketing campaigns. If a customer has an issue with a product, she can connect with a chatbot to explain the situation and the chatbot can input that information to provide a recommendation of how to fix the product. On the recommendation side, chatbots can be used to share popular products with customers that they might find useful and can act as a sort of personal shopper or concierge service to find the perfect gift, meal or night out for a customer with just a few basic questions. Brands are also using chatbots to connect their customers with thought leaders and add personality to their products. In all cases, brands seem to be having great success and experiencing increased engagement and revenue.

Chatbots are easy to use and many customers prefer them over calling a representative on the phone because it tends to be faster and less invasive. They can also save money for companies and are easy to set up.

Chatbots are relatively new and most companies haven’t implemented them yet, it’s only natural that users are interested in them. Hence, people want to discover what chatbots can and cannot do. The number of businesses using chatbots has grown exponentially. Chatbots have increased from 30,000 in 2016 to over 100,000 today. Every major company has announced their own chatbot and 60% of the youth population uses them daily.

These statistics prove that chatbots are the new-gen tech. No more waiting for the right time to incorporate them into your business. The time is now. By the year 2020, nearly 80% of businesses will have their own chatbot.

Billions of people are already using chatbots, so it’s time your business did too.

Benefits of chatbots?

Chatbots are being made to ease the pain that the industries are facing today. The purpose of chat bots is to support and scale business teams in their relations with customers.

Chatbots may sound like a futuristic notion, but according to Global Web Index statistics, it is said that 75% of internet users are adopting one or more messenger platforms. Although research shows us that each user makes use of an average of 24 apps a month, where in 80% of the time would be in just 5 apps. This means you can hardly shoot ahead with an app, but you still have high chances to integrate your chatbot with one of these platforms.

Types of Chatbots

Chatbots are categorized into two different types. Let us look at both and see how they function.

Rule-based chatbots

Chatbots follow a set of established rules or flows to respond to questions posted by a user. All your simple applications contain rule-based chatbots, which respond to queries based on the rules they are trained on. For instance, a weather application, where you ask for a weather forecast and it fetches the data from different sources and responds with the information.

Rule-based chatbots may not be able to hold complex conversations. It can only accomplish the tasks it is programmed to perform unless more improvements are made by the developer.

Machine Learning-based chatbots

Chatbots that are based on machine learning can hold more complex conversations as they try to process the question and understand the meaning behind the question. It learns from the previous conversation and enables itself to handle more complex questions in the future.

Now let's go through some of the benefits that chatbots provide:

1. Available 24*7: I’m sure most of you have experienced listening to the boring music playing while you’re kept on hold by a customer care agent. On an average people spend 7 minutes until they are assigned to an agent. Gone are the days of waiting for the next available operative. Bots are replacing live chat and other forms of contact such as emails and phone calls. Since chatbots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break. This improves your customer satisfaction and helps you rank highly in your sector.

2. Handling Customers: We humans are restricted to the number of things we can do at the same time. A study suggests that humans can only concentrate on 3–4 things at the same time. If it goes beyond that you are bound to meet errors. Chatbots on the other hand can simultaneously have conversations with thousands of people. No matter what time of the day it is or how many

people are contacting you, every single one of them will be answered instantly. Companies like Taco Bell and Domino's are already using chatbots to arrange delivery of parcels.

3. Helps you Save Money: If you are a business owner you are bound to have a lot of employees who need to be paid for the work they do. And these expenses just keep adding up as business grows. Chatbots are a one time investment which helps businesses reduce down on staff required. You could integrate a customer support chatbot in your business to cater to simple queries of customers and pass on only the complex queries to customer support agents.

4. Provides 100% satisfaction to customers: Humans react to others based on their mood and emotions. If an agent is having a good attitude or is in a good mood he will most probably talk to customers in a good way. In contrast to this the customer will not be satisfied. Whereas chatbots are bound by some rules and obey them as long as they're programmed to. They always treat a customer in the most polite and perfect way no matter how rough the person is. Also, in the travel and hospitality industry where travelers do not speak the same language, a bot can be trained to communicate in the language of the traveler.

5. Automation of repetitive work: Let's be honest, no one likes doing the same work again and again over a brief period of time. In the case of humans, such tasks are prone to errors. Chatbots now help automate tasks which are to be done frequently and at the right time. Also, now there are numerous slack bots which automate repetitive tasks. This helps people save time and increase productivity. For example, there are new items bought from your eCommerce site or there is a bug reported then it sends a short summary to a slack channel.

6. Personal Assistant: People could use Bots as a fashion advisor for clothing recommendations, or ask trading tips from a finance bot, suggest places to visit from a travel bot and so forth. This would help the users get a more personal touch from the chatbot. Also, the chatbot will remember all your choices and provide you with relevant choices the next time you visit it.

To create your own chatbot:

1 Identify your business goals and customer needs.

2 Choose a chatbot builder that you can use on your desired channels. 3 Design your bot

conversation flow by using the right nodes.4 Test your chatbot and collect messages to get more insights. 5 Use data and feedback from customers to train your bot.

How can chatbots drive revenue for you? Below we have compiled reasons why chatbots are important for your business and how can they help in increasing revenues:

a. Higher user customer engagement

Most businesses these days have a web presence. But with being on the internet, boundaries of day and night, availability and unavailability have changed, so have user expectations. This is probably the biggest reason to use them. Bots give the user an interactive experience. It makes customers feel they are working with someone to help resolve their issue. If done right, bots can help customers find what they are looking for and make them more likely to return.

Customer Engagement

Clearance Sale : Notify users about on-going clearance sale of products relevant to the users at their nearest outlets. Product Finder : Enable consultative selling without the need of a call center It offer Notification : Notify users about offers, product launches on products/ services they've shown interest in, and products that's back in stock

b. Mobile-ready and immediate availability

Along with a web presence, it has also become increasingly important for brands to have a mobile presence - mobile apps, mobile-optimized websites. Considering how chat has been around on the mobile for ages, most chatbot implementations don't need you to work on tweaking their UI, they are ready to implement and so available to your customers immediately.

You might argue that you have an app for that. Having an app for your brand is great, but having users discover that app, download it and use it to stay engaged is not an easy deal. Instead, implementing a chatbot - which works on the mobile browser or a messaging-app which the user regularly uses - makes it all the more reason for a customer to be engaged with the brand

c. It can drive sales

Chatbots can be intelligent. Depending on a user's preferences or purchases, it can send products to customers which are more likely to convert into sales. Or it can send coupons to users for in-store purchases/discounts. Bots can also be used to link the user to your mCommerce site/app so they can buy the product directly from the convenience of their phones.

Sell Intelligently

Product Recommendations : Push proactive recommendations to users based on their preferences and search and order history.

Enable order booking over chat.

d. **Minimal cost -** Maximum return The best part about bots is they are cheap. Chatbot provides the necessary infrastructure and APIs for creating these bots. They require minimal maintenance and since it is automated, there is no labor-intensive work that goes in there.

e. **Customer Service Track Order :** Keep users up to date with order status. Schedule or reschedule delivery to a provided address or request to pick it up at any other Best Buy outlet. Stock outs : Notify users when desired product is available and place order over a chat. Returns and Replacements : No waiting time to reach customer care. Customers can instantly place a request to replace or return an order. Seek Reviews : Reach out to users to seek reviews on the products recently bought

Gift Recommendations

- Recommend relevant gifting options to users, accessing calendar events and understanding the likes and style of beneficiary.
- Opportunity to upsell gift cards for the users for every

occasion. Application across Industries

According to a new survey, 80% of businesses want to integrate chatbots in their business model by 2020. So which industries can reap the greatest benefits by implementing consumer-facing chatbots? According to a chatbot, these major areas of direct-to-consumer engagement are prime:

Chatbots in Restaurant and Retail Industries

Famous restaurant chains like Burger King and Taco Bell have introduced their Chatbots to stand out from competitors of the Industry as well as treat their customers quickly. Customers of these restaurants are greeted by the resident Chatbots, and are offered the menu options- like a counter order, the Buyer chooses their pickup location, pays, and gets told when they can head over to grab

their food. Chatbots also work to accept table reservations, take special requests and go the extra step to make the evening special for your guests.

Chatbots are not only good for the restaurant staff in reducing work and pain but can provide a better user experience for the customers.

Chatbots in Hospitality and Travel

For hoteliers, automation has been held up as a solution for all difficulties related to productivity issues, labour costs, a way to ensure consistently, streamlined production processes across the system. Accurate and immediate delivery of information to customers is a major factor in running a successful online Business, especially in the price sensitive and competitive Travel and Hospitality industry. Chatbots particularly have gotten a lot of attention from the hospitality industry in recent months.

Chatbots can help hotels in a number of areas, including time management, guest services and cost reduction. They can assist guests with elementary questions and requests. Thus, freeing up hotel staff to devote more of their time and attention to time-sensitive, critical, and complicated tasks. They are often more cost effective and faster than their human counterparts. They can be programmed to speak to guests in different languages, making it easier for the guests to speak in their local language to communicate.

Chatbots in Health Industry

Chatbots are a much better fit for patient engagement than Standalone apps. Through these Health-Bots, users can ask health related questions and receive immediate responses. These responses are either original or based on responses to similar questions in the database. The impersonal nature of a bot could act as a benefit in certain situations, where an actual Doctor is not needed. Chatbots ease the access to healthcare and industry has favourable chances to serve their customers with personalised health tips. It can be a good example of the success of Chatbots and Service Industry combo.

Chatbots in E-Commerce

Mobile messengers- connected with Chatbots and the E-commerce business can open a new channel for selling the products online. E-commerce Shopping destination “Spring” was the early adopter. E-commerce future is where brands have their own Chatbots which can interact with their customers through their apps.

Chatbots in Fashion Industry

Chatbots, AI and Machine Learning pave a new domain of possibilities in the Fashion industry, from Data Analytics to Personal Chatbot Stylists. Fashion is such an industry where luxury goods can only be bought in a few physical boutiques and one to one customer service is essential. The Internet changed this dramatically, by giving the customers a seamless but a very impersonal experience of shopping. This particular problem can be solved by Chatbots.

Customers can be treated personally with bots, which can exchange messages, give required suggestions and information. Famous fashion brands like Burberry, Tommy Hilfiger have recently launched Chatbots for the London and New York Fashion Week respectively. Sephora a famous cosmetics brand and H&M– a fashion clothing brand have also launched their Chatbots.

Chatbots in Finance

Chatbots have already stepped into the Finance Industry. Chatbots can be programmed to assists the customers as Financial Advisor, Expense Saving Bot, Banking Bots, Tax bots, etc. Banks and Fintech have ample opportunities in developing bots for reducing their costs as well as human errors. Chatbots can work for customer's convenience, managing multiple accounts, directly checking their bank balance and expenses on particular things. Further about Finance and Chatbots have been discussed in our earlier blog: Chatbots as your Personal Finance Assistant.

Chatbots in Fitness Industry

Chat based health and fitness companies using Chatbot, to help their customers get personalized health and fitness tips. Tech based fitness companies can have a huge opportunity by developing their own Chatbots offering a huge customer base with personalized services. Engage with your fans like never before with news, highlights, game-day info, roster and more.

Chatbots and the Service Industry together have a wide range of opportunities and small to big all size of companies using chatbots to reduce their work and help their customers better.

Chatbots in Media

Big publisher or small agency, our suite of tools can help your audience chatbot experience rich and frictionless. Famous News and Media companies like The Wall Street Journal, CNN, Fox news, etc have launched their bots to help you receive the latest news on the go.

Chatbot in Celebrity:

With a chatbot you can now have one-on-one conversations with millions of fans. Chatbot in Marketing

SMS Marketing

- Why promote just a coupon code that the customer does not know how to use?

- Improve conversions from your existing SMS campaigns.
- Talk to your customers when they want to using “Talk to an Agent” feature.

Email Marketing

- So your eMail has made a solid elevator pitch about your product.
- As a next step, is making customers fill an online form the most exciting way to engage with your customers?
- It’s time to rethink the landing page.
- Instantly engage in a conversation with your customers.
- Address their concerns and queries

Social Media Triage

- How effectively are you addressing the negative sentiment around your brand on social media?
- Addressing queries instantly and effectively can convert even an angry customer into a loyal fan.
- Leverage a chatbot as your first response strategy and comfort that customer.

7. Questions:

Q 1: What is the use of a chat bot?

Q 2: Explain dialog flow in detail.

Q 3: What are the requirements for developing a chatbot?

Q 4: How do you evaluate a chatbot performance?

Q 5: How do I improve my chatbot accuracy?

8. Conclusion:

Smart solutions are important for the success of any business. From providing 24/7 customer service, improving current marketing activities, saving time spent on engaging with users to improving internal processes, chatbots can yield the much-needed competitive advantage. If you are looking to develop a chatbot, the best thing to do is to approach a company that will understand your business needs to develop a chatbot that helps you achieve your business goals.

Assignment No: 1

1. Title of Assignment:

Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.

2. Prerequisite:

Basic knowledge of Arrays, Lists, Stack, Queue, Graph, Tree etc.

3. Objective:

In this experiment, we will be able to do the following:

- To understand Uninformed Search Strategies.
- To make use of Graph and Tree Data Structure for implementation of Uninformed Search strategies.
- Study the various Graph traversal algorithms and the difference between them.
- Understand the BFS Graph traversal using queues.
- Demonstrate knowledge of time complexity and space complexity of performing a BFS on a given graph.

4. Outcome: Successfully able to find depth first search algorithm and Breadth First Search algorithm.

5. Software and Hardware Requirement:

Open Source C++ Programming tool like G++/GCC, python,java and Ubuntu.

6. Relevant Theory / Literature Survey:

Uninformed (Blind search) search

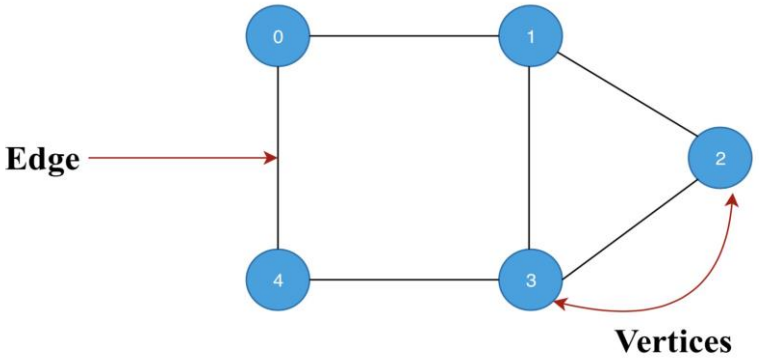
- Uninformed search is a class of general-purpose search algorithms which operates in brute force-way.
- It examines each node of the tree until it achieves the goal node.

- Uninformed search algorithms do not have additional information about state or search space other than how to traverse the tree and how to identify leaf and goal nodes, so it is also called blind search
- Eg. DFS, BFS Search algorithm

Graphs Definition

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges. Pictorial Representation of Graph.

Fig: Pictorial Representation of Graph



Types of Graphs

- Undirected Graph:- Directions are not given to edges
- Directed Graph:- Directions are given to edges wit

Theory of Graph Traversal Techniques

In computer science, graph traversal (also known as graph search) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Tree traversal is a special case of graph traversal.

Techniques of Graph Traversal

- **DFS** - A depth-first search (DFS) is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices; that is, it traverses the depth of any particular path before exploring its breadth. A stack (often the program's call stack via recursion) is generally used when implementing the algorithm.
- **BFS** - A breadth-first search (BFS) is another technique for traversing a finite graph. BFS visits the neighbor vertices before visiting the child vertices, and a queue is used in the search process. This algorithm is often used to find the shortest path from one vertex to another.

DFS

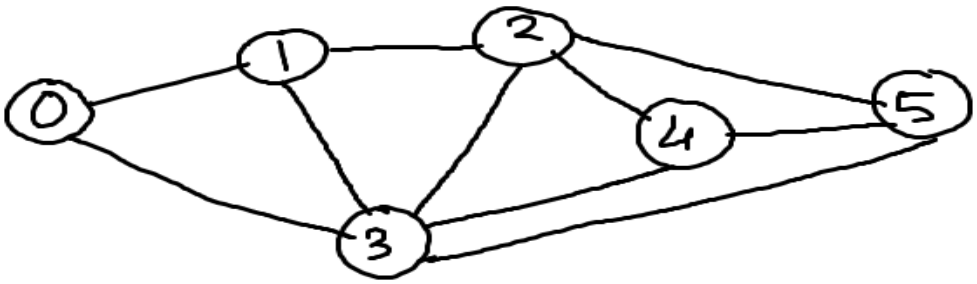
- Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.
- Depth First Search algorithm is a recursive algorithm that uses the idea of backtracking.

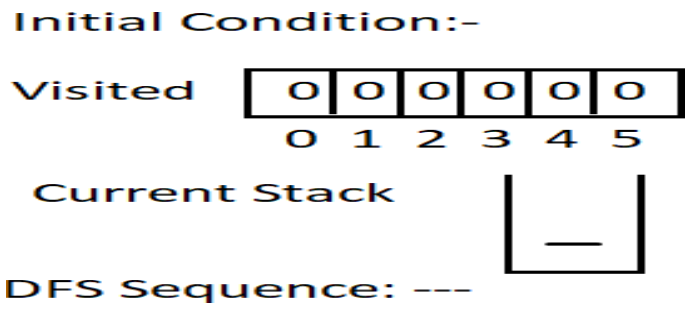
Depth First Search (DFS) Algorithm

Depth first search (DFS) algorithm starts with the initial node of the graph G, and then goes deeper and deeper until we find the goal node or the node which has no children. The algorithm then backtracks from the dead end towards the most recent node that is yet to be completely explored. The data structure which is being used in DFS is stack.

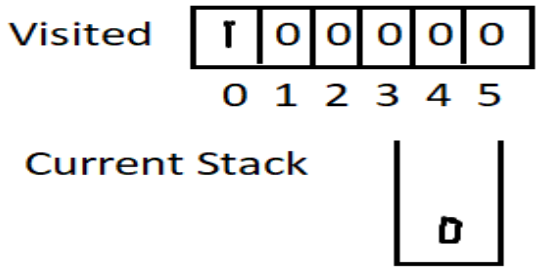
- STEP 1: Start by putting any one of the graph's vertices on top of a stack (acts as source node of DFS).
- STEP 2: Take the top item of the stack and set its visited as 1.
- STEP 3: Create a list of that vertex's adjacent nodes. Add the ones whose visited is 0 to the top of stack.
- STEP 4: Keep repeating steps 2 and 3 until the stack is empty.

An example which explains DFS Algorithm



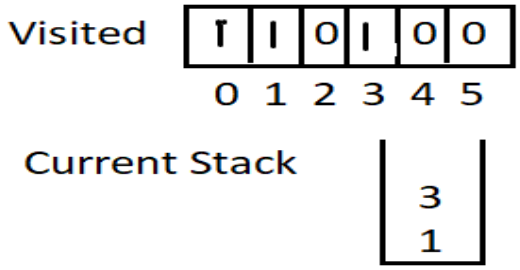


Step 1:
Start with Node 0
Node 0 is pushed into stack and its visited value set to 1



DFS Sequence: ---

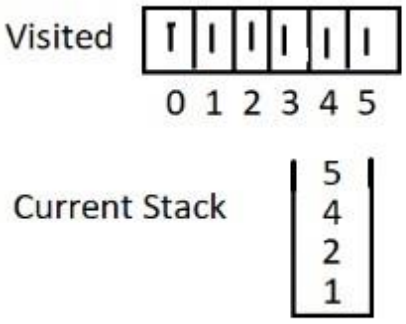
Step 2:
Node 0 is popped from stack and its adjacent nodes 1 and 3 are pushed into stack and visited value set to 1



DFS Sequence: 0

Step 3:

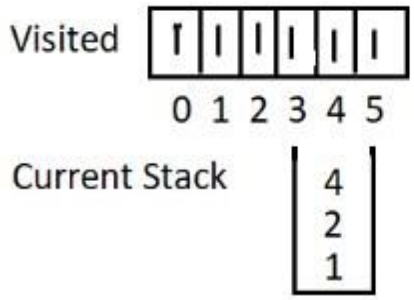
Node 3 is popped from stack and its adjacent nodes 2,4 and 5 are pushed into stack because their visited value is 0,
Set visited value to 1.



DFS Sequence: 0 , 3

Step 4 :

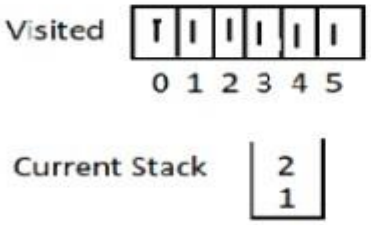
Node 5 is popped from stack. No adjacent node to add in stack



DFS Sequence: 0, 3, 5

Step 5 :

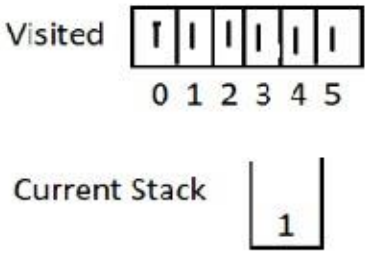
Node 4 is popped from stack.
Adjacent Nodes visited value is already 1 so no need to push it into stack.



DFS Sequence: 0,3,5,4

Step 6:

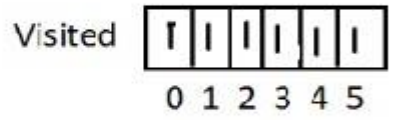
Node 2 is pooped from Stack, Adjacent Nodes Visited Value is Already 1 , So No Need to Push it into Stack



DFS Sequence: 0,3,5,4,2

Step 7:

Node 1 is pooped from stack as no adjacent nodes are remain unvisited.
No Nodes are pushed into stack.
As stack became empty We can stop.



DFS Sequence: 0,3,5,4 ,2 ,1

DFS Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm

DFS Disadvantage:

- There is the possibility that many states keep reoccurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometimes it may go to the infinite loop.

Completeness: DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.

Time Complexity: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:

$T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)$

Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)

Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.

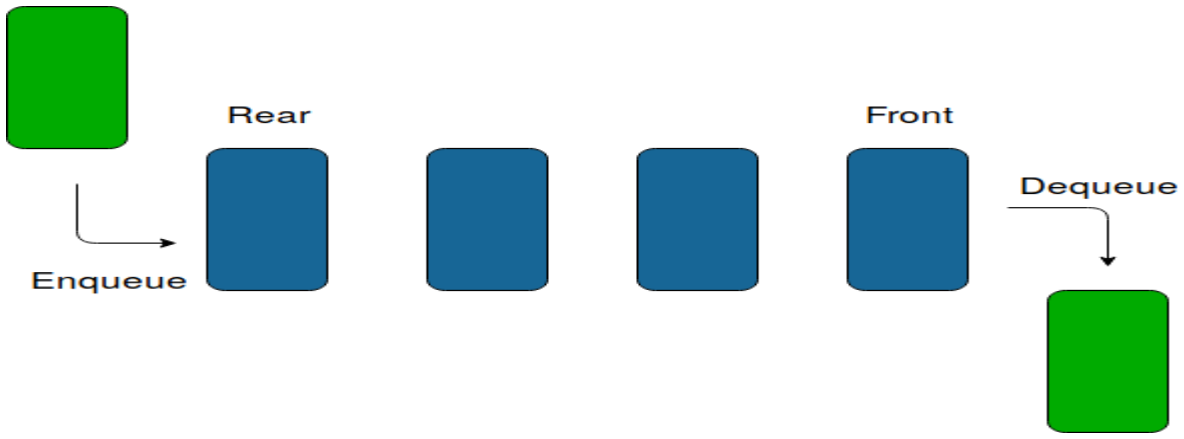
Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

BFS

Queues Definition

A Queue is a linear structure which follows a particular order in which the operations are performed.

The order is First In First Out (FIFO).



- Breadth-first search is the most common search strategy for traversing a tree or graph.
- This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor nodes at the current level before moving to nodes of the next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure

BFS Algorithm

The algorithm starts with examining the source node and all of its neighbors. In the next step, the neighbors of the nearest node of the source node are explored. The algorithm then explores all neighbors of all the nodes and ensures that each node is visited exactly once and no node is visited twice.

STEP 1: Set visited as 0 for all nodes in the Graph.

STEP 2: Enqueue the selected source node into the queue.

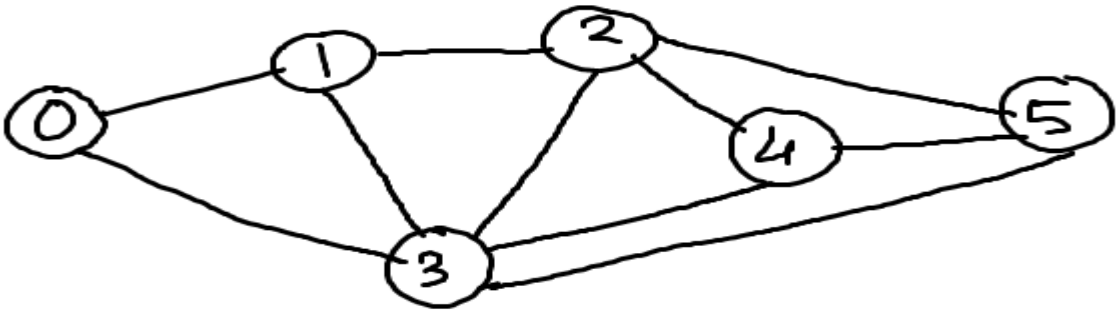
STEP 3: Dequeue a node N from the queue and update its visited as 1.

STEP 4: Enqueue all the neighbours of node N which are not present in the queue and whose visited is 0.

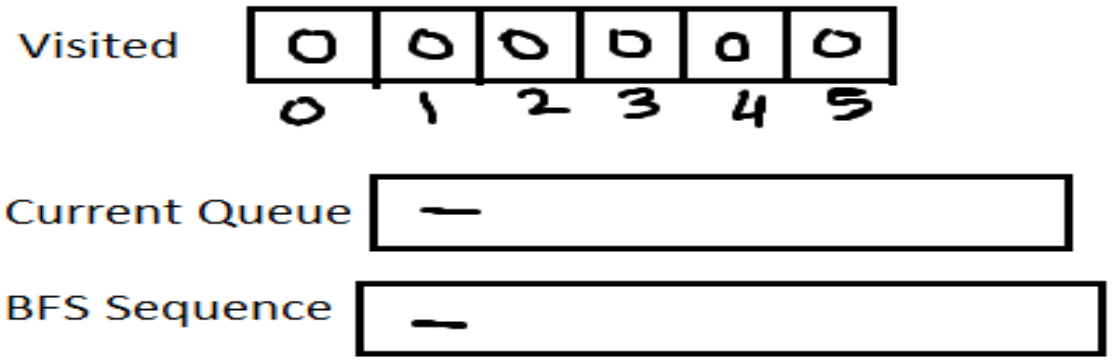
STEP 5: Repeat steps 3 and 4 until the queue is empty.

STEP 6: EXIT

An example which explains BFS Algorithm

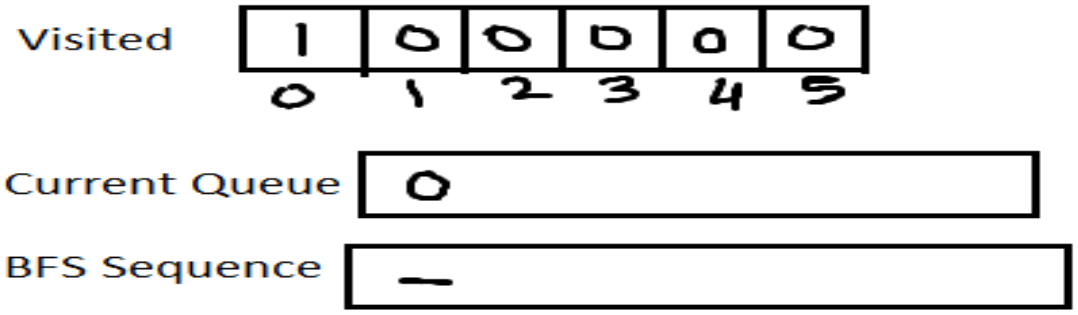


Initial Condition:-



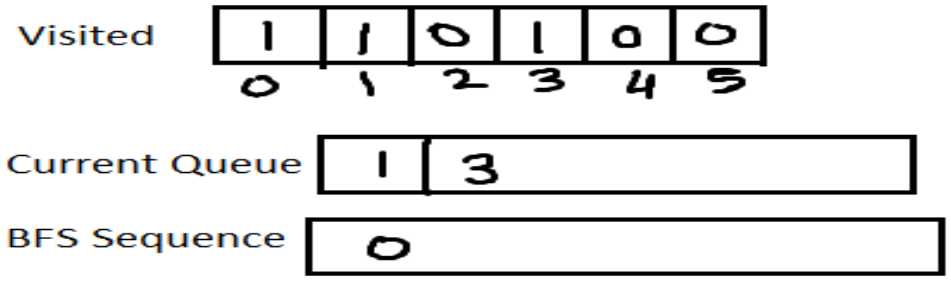
Step 1:-

Node '0' is selcted as source node of BFS So Node '0' is enqueued to Queue



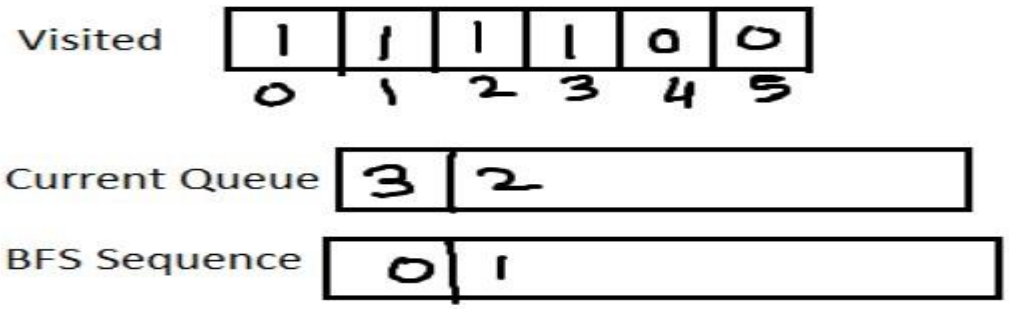
Step 2:-

Node '0' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e. 1,3 by changing visited value to '1'



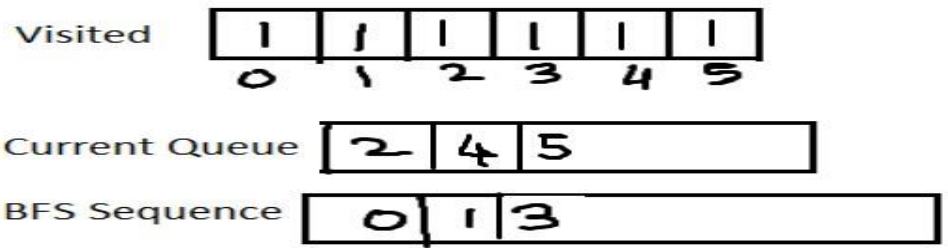
Step 3:-

Node '1' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e.2,3 but only node 2 is having visited value 0 but node 3 is already having visited value 1. we only enqueued node 2 in queue.



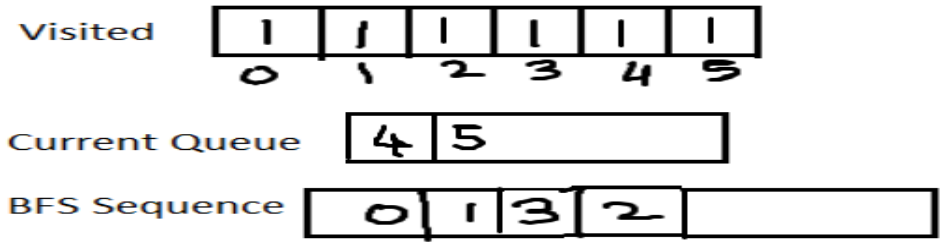
Step 4:-

Node '3 ' is dequeued from Queue and it's adjacent nodes are enqueued into queue i.e. 4,5 and set their visited value to 1



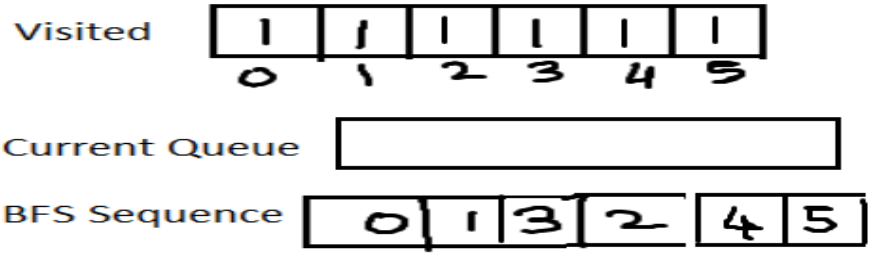
Step 5:-

Node 2 is dequeued from Queue and it's adjacent nodes are already having visited value 1 so No enqueued node 3,4,5 in Queue.



Step 6:-

Simillarly Node 4,5 are dequeued from Queue , No Nodes are enqueued in Queue.
Once Queue is empty u can stop.



As queue is empty , BFS is done on node '0'
Nodes visited in BFS sequence : 0, 1, 3, 2, 4, 5

BFS Applications

- Path and Minimum Spanning Tree for unweighted graph
- In Peer to Peer Networks like BitTorrent, Breadth First Search is used to find all neighbor nodes.

- GPS Navigation systems Breadth First Search is used to find all neighboring locations.
- Cycle Detection in Undirected Graph In undirected graphs, either Breadth First Search or Depth First Search can be used to detect cycle.
  In directed graphs, only depth first search can be used.
- Finding all nodes within one connected component We can either use Breadth First or Depth First Traversal to find all nodes reachable from a given node.

Time Complexity: Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$$T(b) = 1+b^2+b^3+.....+ b^d= O(b^d)$$

Space Complexity: Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness: BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality: BFS is optimal if path cost is a non-decreasing function of the depth of the node.

Differences between BFS and DFS

Key Differences Between BFS and DFS:

- BFS is a vertex-based algorithm while DFS is an edge-based algorithm.
- Queue data structure is used in BFS. On the other hand, DFS uses stack or recursion.
- Memory space is efficiently utilized in DFS while space utilization in BFS is not effective.
- BFS is an optimal algorithm while DFS is not optimal.
- DFS constructs narrow and long trees whereas BFS constructs wide and short tree.

7. Questions:

Q 1: Differentiate between DFS and BFS Algorithms.

Q 2: Write down the Time and Space Complexity of DFS and BFS.

Q 3: Which data structure is used by DFS and BFS?

8. Conclusion:

In This way we have studied uninformed search strategy, how to implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.