SEARCH TECHNIQUES

Dr. Krishnendu Guha
Assistant Professor (On Contract)
National Institute of Technology (NIT), Jamshedpur
Email: krishnendu.ac@nitjsr.ac.in

OVERVIEW OF TODAY'S CLASS

- Production Systems
- Breadth First Search (BFS)
- Depth First Search (DFS)

PRODUCTION SYSTEMS

- Since search forms the core of many intelligent processes, it is useful to structure AI programs in a way that facilitates describing and performing the search process

- Production systems provides such structu

OR

- A **production system in AI** is a type of computer program that provides **artificial intelligence** based on a set of rules

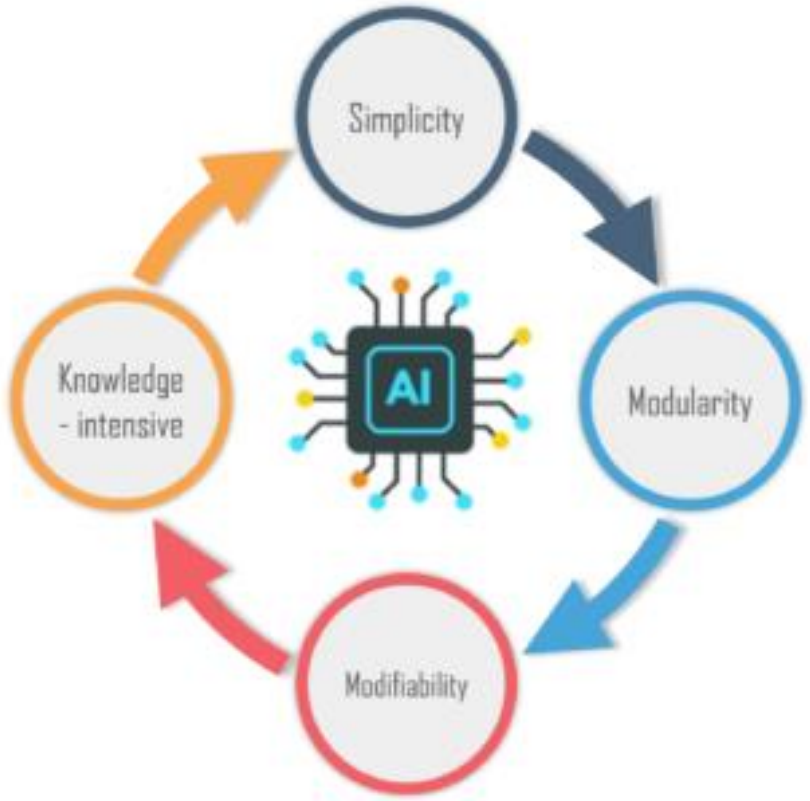
COMPONENTS OF A PRODUCTION SYSTEM

- Thus, a production system consists of

- A set of rules
- One or more knowledge or databases that contain whatever information is appropriate for the particular task
- A control strategy that specifies the order of application of the rules (and resolving the conflicts when more than one rules match)
- A rule applier

FEATURES OF PRODUCTION SYSTEMS IN AI

- **Simplicity:** The structure of each sentence in a production system is unique and uniform as they use the "IF-THEN" structure. This structure provides simplicity in knowledge representation. This feature of the production system improves the readability of production rules.
- **Modularity:** This means the production rule code the knowledge available in discrete pieces. Information can be treated as a collection of independent facts which may be added or deleted from the system with essentially no deleterious side effects.
- **Modifiability:** This means the facility for modifying rules. It allows the development of production rules in a skeletal form first and then it is accurate to suit a specific application.
- **Knowledge-intensive:** The knowledge base of the production system stores pure knowledge. This part does not contain any type of control or programming information. Each production rule is normally written as an English sentence; the problem of semantics is solved by the very structure of the representation



CONTROL STRATEGY

- We need to decide which rule to apply next during the process of searching for a solution to a particular problem
- Need to satisfy spatio-temporal requirements
- REQUIREMENTS ARE:
  - The first requirement for a good control strategy is that it should **cause motion**.
    - Eg: If the first rule is applied again and again, then we would never reach the solution or no motion is involved
  - The second requirement for a good control strategy is that it should be **systematic**.
    - Eg: Just choosing a random rule will take a longer time to reach the result
  - Finally, it must be **efficient** in order to find a good answer
    - Eg: Rules chosen must aid us to reach the result in minimum time and space

BREADTH FIRST SEARCH

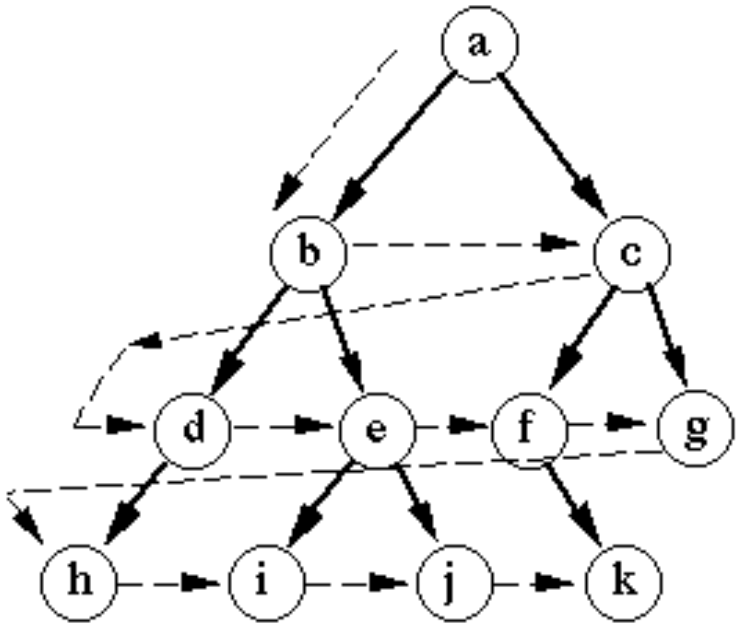
- **Breadth-first search** (**BFS**) is an algorithm for traversing or searching tree or graph data structures
- It starts at the root , and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level

- HISTORY
- BFS and its application in finding connected components of graphs were invented in 1945 by Konrad Zuse, in his (rejected) Ph.D. thesis on the Plankalkül programming language, but this was not published until 1972
- It was reinvented in 1959 by Edward F. Moore, who used it to find the shortest path out of a maze and later developed by C. Y. Lee into a wire routing algorithm (published 1961)

BFS ALGORITHM

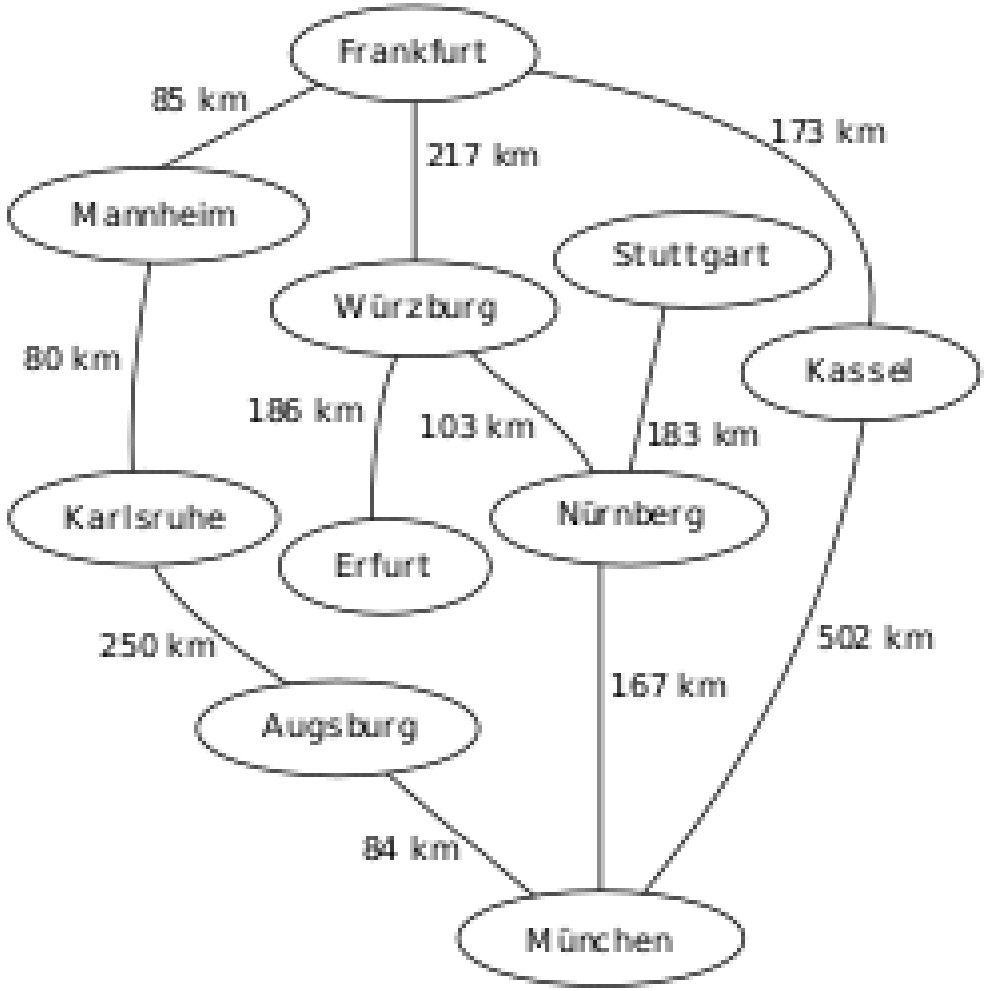
- Create a variable called NODE-LIST and set it to the initial state
- Until a goal state is found or NODE-LIST is empty:
  - Remove the first element from NODE-LIST and let us call it E; If NODE-LIST was empty, quit
  - For each way that each rule can match the state described in E; do
    - Apply the rule to generate a new state
    - If the state is a goal state, quit and return this state
    - Otherwise, add the new state to the end of the NODE-LIST

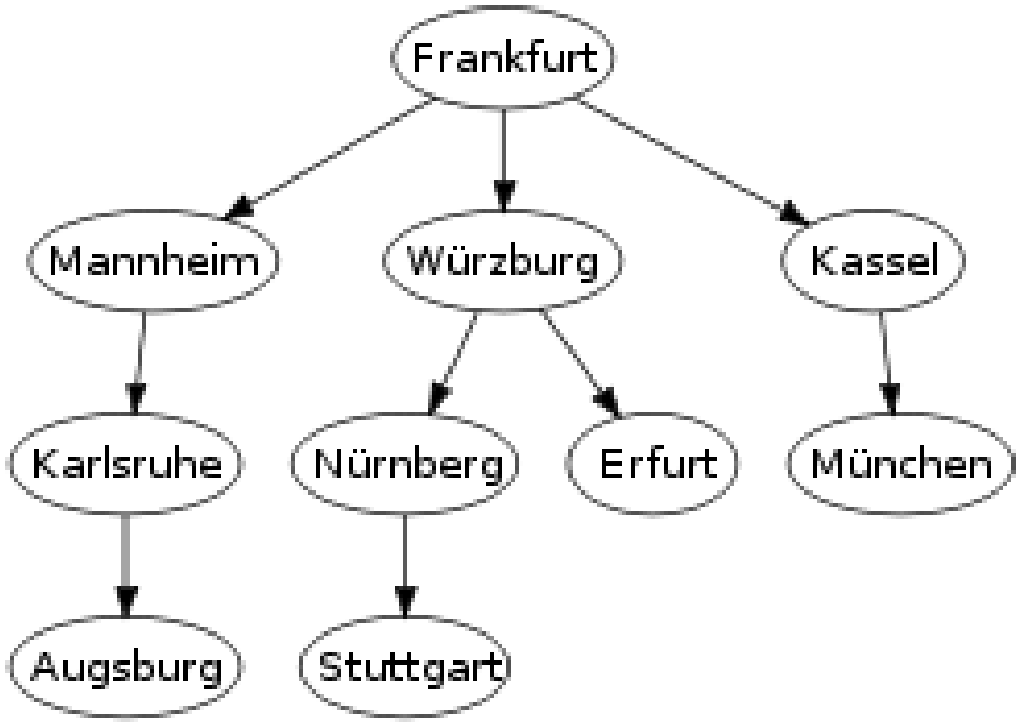
it uses a queue (First In First Out)
it checks whether a vertex has been discovered before enqueueing the vertex rather than delaying this check until the vertex is dequeued from the queue



Breadth-first search

EXAMPLE


Frankfurt
85 km
217 km
173 km
Mannheim
Würzburg
Stuttgart
Kassel
80 km
186 km
103 km
183 km
Karlsruhe
Erfurt
Nürnberg
250 km
502 km
167 km
Augsburg
84 km
München


Frankfurt
Mannheim
Würzburg
Kassel
Karlsruhe
Nürnberg
Erfurt
München
Augsburg
Stuttgart

SPACE AND TIME COMPLEXITY

- Let V be the number of vertices and E the number of edges
- Then time complexity=O(|V|+|E|), as in the worst case, every vertex and edge will be explored

- When the number of vertices in the graph is known ahead of time, and additional data structures are used to determine which vertices have already been added to the queue, the space complexity can be expressed as O(|V|)

- When working with graphs that are too large to store explicitly (or infinite), it is more practical to describe the complexity of breadth-first search in different terms:
  - to find the nodes that are at distance *d* from the start node (measured in number of edge traversals), BFS takes $O(b^{d+1})$ time and memory (i.e. both space and time complexity), where *b* is the "branching factor" of the graph

APPLICATIONS OF BFS

Breadth-first search can be used to solve many problems in graph theory, for example:

- Copying garbage collection, Cheney's algorithm
- Finding the shortest path between two nodes *u* and *v*, with path length measured by number of edges (an advantage over depth-first search)[12]
- (Reverse) Cuthill–McKee mesh numbering
- Ford–Fulkerson method for computing the maximum flow in a flow network
- Serialization/Deserialization of a binary tree vs serialization in sorted order, allows the tree to be re-constructed in an efficient manner.
- Construction of the *failure function* of the Aho-Corasick pattern matcher.
- Testing bipartiteness of a graph.

DEPTH FIRST SEARCH

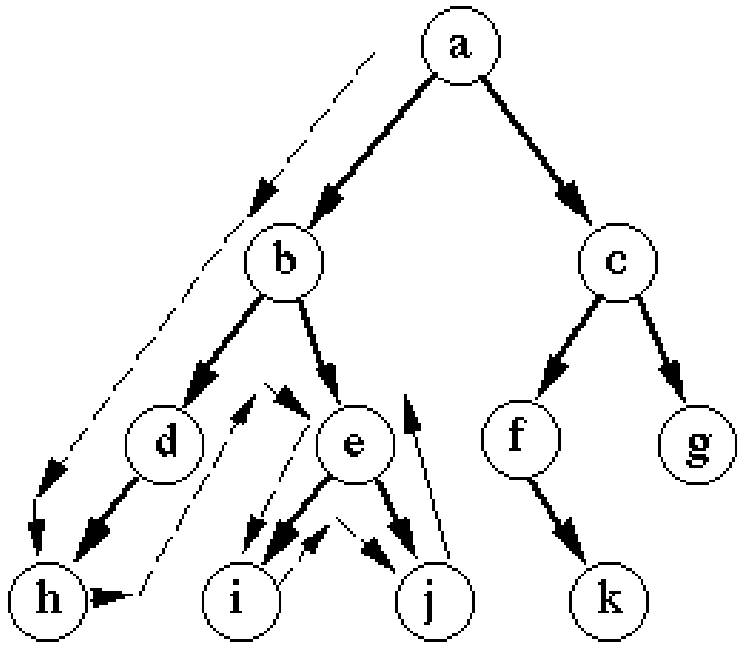
- **Depth-first search** (**DFS**) is an algorithm for traversing or searching tree or graph data structures
- The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking

- HISTORY
- A version of depth-first search was investigated in the 19th century by French mathematician Charles Pierre Trémaux[1] as a strategy for solving mazes

DFS ALGORITHM

- If the initial state is the goal state, quit and return success
- Otherwise, do the following until failure or success is returned:
  - Generate a successor E of the initial state. If there are no more successors, return failure
  - Call depth first search with E as the initial state
  - If success is returned, signal success, otherwise, continue in the loop

It uses a stack data structure



Depth-first search

SPACE AND TIME COMPLEXITY

- Time Complexity:
  - O(|V|+|E|) for explicit graphs without repetition
  - $O(b^d)$ for implicit graphs with branching factor b. searched to depth d

- Space Complexity:
  - O(|V|) if entire graph is traversed without repetition
  - O(longest path searched)=O(bd) for implicit graphs without elimination of duplicate nodes

APPLICATIONS OF DFS

Algorithms that use depth-first search as a building block include:

- Finding connected components.
- Topological sorting.
- Finding 2-(edge or vertex)-connected components.
- Finding 3-(edge or vertex)-connected components.
- Finding the bridges of a graph.
- Generating words in order to plot the limit set of a group.
- Finding strongly connected components.
- Planarity testing.[9][10]
- Solving puzzles with only one solution, such as mazes. (DFS can be adapted to find all solutions to a maze by only including nodes on the current path in the visited set.)
- Maze generation may use a randomized depth-first search.
- Finding biconnectivity in graphs.

ADVANTAGES OF DFS

- Requires less memory, as only the nodes in the current path are stored
- DFS may find a solution quite early (if care is taken in ordering the successor nodes)

ADVANTAGES OF BFS

- BFS will not get trapped exploring a blind alley
- BFS is guaranteed to find a solution if it exists
- If there is multiple solutions, then a minimal solution is guaranteed to be found

COMPLETENESS

- a search method is described as being complete if it is guaranteed to find a goal state if one exists
- Breadth-first search is complete, but depth-first search is not
- When applied to infinite graphs represented implicitly, breadth-first search will eventually find the goal state, but depth first search may get lost in parts of the graph that have no goal state and never return


GRACIAS
THANK
YOU
ARIGATO
SHUKURIA
BOLZÏN
MERCI
TASHAKKUR ATU
SUKSAMA
MEHRBANI
GRAZIE
BIYAN
SHUKRIA
DANK$CHEEN
JUSPAXAR