

# Grokking Artificial intelligence Algorithms by Rishal Hurbans

## **Section 2 Search Fundamentals**

When we think about what makes us intelligent, the ability to plan before carrying out actions is a prominent attribute. Before embarking on a trip to a different country, before starting a new project, before writing functions in code, planning happens. Planning happens at different levels of detail in different contexts to strive for the best possible outcome when carrying out the tasks involved in accomplishing goals

### Time Complexity

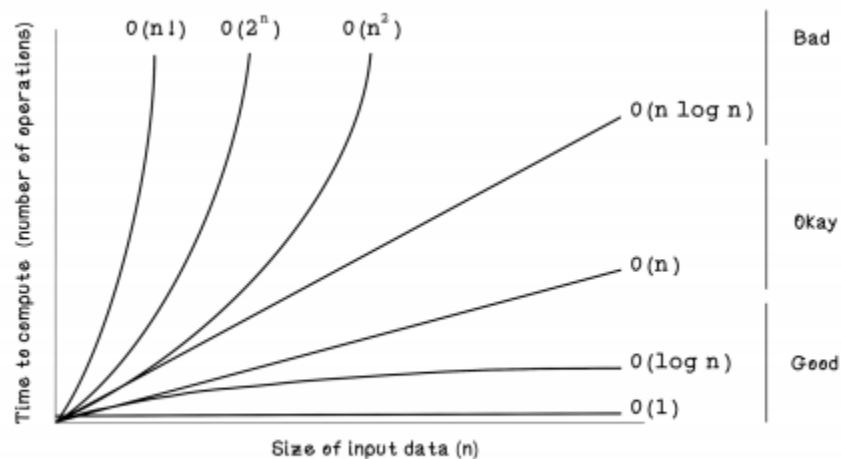


Figure 2.3 Big O complexity

### Data vs. Information

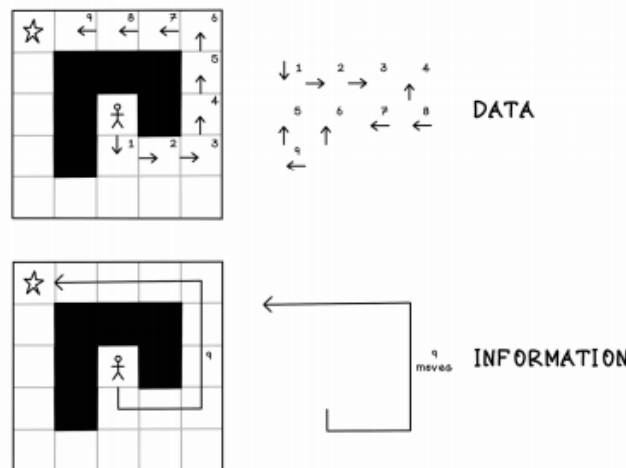


Figure 2.7 Data versus information

## Data Structures

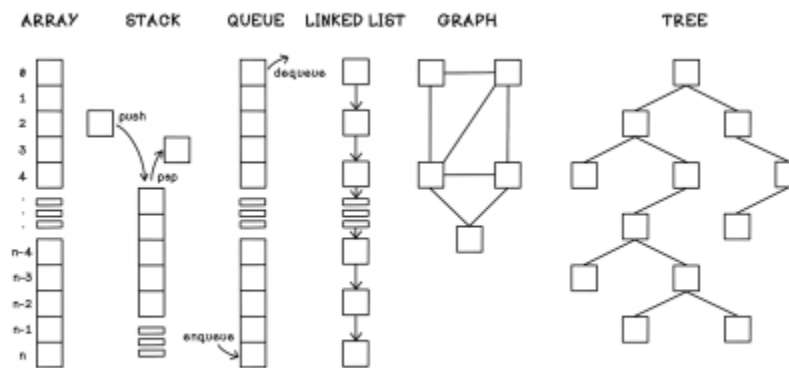


Figure 2.8 Data structures used with algorithms

## Tree Attributes

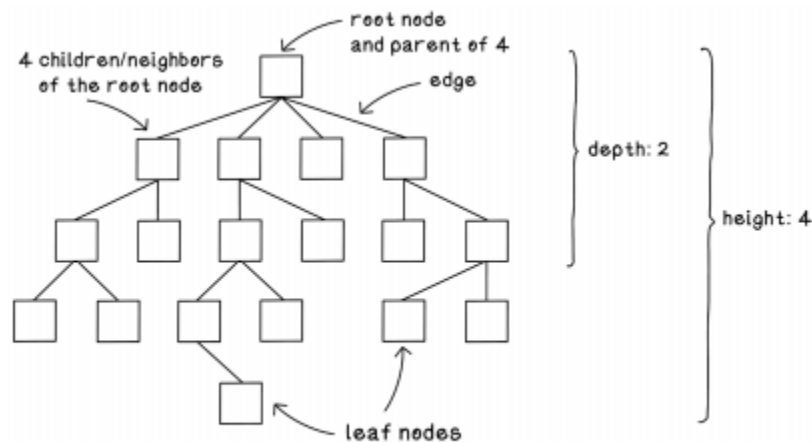


Figure 2.12 The main attributes of a tree

## Breadth First Search pseudocode

```
run_bfs(maze, current_point, visited_points):  
    let q equal a new queue  
    push current_point to q  
    mark current_point as visited  
    while q is not empty:  
        pop q and let current_point equal the returned point  
        add available cells north, east, south, and west to a list neighbors  
        for each neighbor in neighbors:  
            if neighbor is not visited:  
                set neighbor parent as current_point  
                mark neighbor as visited  
                push neighbor to q  
            if value at neighbor is the goal:  
                return path using neighbor  
    return "No path to goal"
```

## Breadth Search First Algorithm



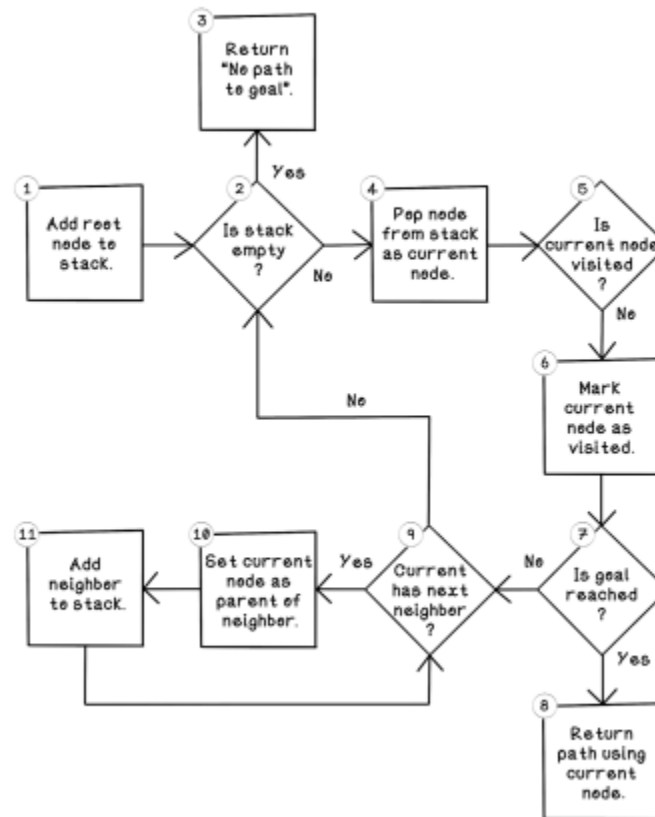
Figure 2.16 Flow of the breadth-first search algorithm

## Depth First Search Pseudocode

```

run_dfs(maze, root_point, visited_points):
    let s equal a new stack
    add root_point to s
    while s is not empty
        pop s and let current_point equal the returned point
        if current_point is not visited:
            mark current_point as visited
            if value at current_node is the goal:
                return path using current_point
            else:
                add available cells north, east, south, and west to a list neighbors
                for each neighbor in neighbors:
                    set neighbor parent as current_point
                    push neighbor to s
    return "No path to goal"
  
```

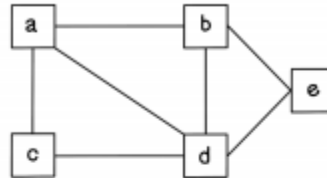
## Depth Search First Algorithm



## Types of Graphs

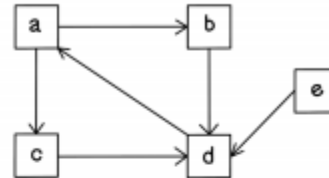
### UNDIRECTED

No edges are directed.  
Relationships between two nodes are mutual.



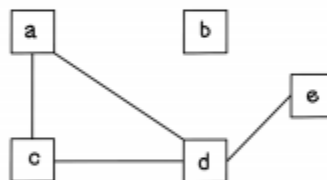
### DIRECTED

Edges indicate direction.  
Relationships between two nodes are explicit.



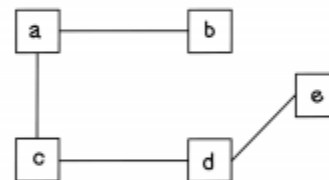
### DISCONNECTED

One or more nodes are not connected by any edges.



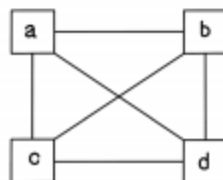
### ACYCLIC

A graph that contains no cycles.



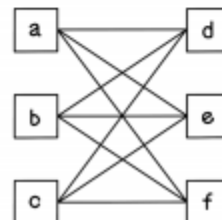
### COMPLETE

Every node is connected to every other node by an edge.



### COMPLETE BIPARTITE

Every node from one partition is connected to every node of the other partition.



### WEIGHTED

A graph where the edges between nodes have a weight.

