A graph $G(V,E)$ consists of a set of vertices $V$ (also called nodes) and a set of edges $E$ connecting these vertices.
A simple graph $G(V,E)$ is a graph which contains no multi-edges and no loops.
Path

G contains only edges that can be consecutively traversed
• Node degree $\deg(x)$  The number of edges connected to this node
A directed graph (digraph) is a graph that discerns between the edges \( A \to B \) and \( A \leftarrow B \).
Additional Terminology

**Independent Set**
G contains no edges

**Clique**
G contains all possible edges
**Tree**
G contains no cycles

**Network**
G contains cycles
Binary Tree

Contains no nodes, or
Is comprised of three disjoint sets of nodes:
  a root node,
  a binary tree called its left subtree, and
  a binary tree called its right subtree
Solving by search

• Search is a goal-based agent

• Uninformed search algorithms
  • Given just the problem formulation

• Informed search algorithms
  • Given some guidance
Route finding
Definition of a problem

- Initial State
- Actions (s) -> \{a1, a2, a3, ...\}
- Result (s,a) -> s' (transition model)
- Goaltest(s) -> T | F
- Pathcost(s (a) -> s (a) -> s) -> n
- Stepcost(s,a,s') -> n
Practice once...

states??
actions??
goal test??
path cost??
states??: integer locations of tiles (ignore intermediate positions)
actions??: move blank left, right, up, down (ignore unjamming etc.)
goal test??: = goal state (given)
path cost??: 1 per move
Search strategies

A strategy is defined by picking the order of node expansion.

Strategies are evaluated along the following dimensions:
- **completeness**—does it always find a solution if one exists?
- **time complexity**—number of nodes generated/expanded
- **space complexity**—maximum number of nodes in memory
- **optimality**—does it always find a least-cost solution?

Time and space complexity are measured in terms of
- \( b \)—maximum branching factor of the search tree
- \( d \)—depth of the least-cost solution
- \( m \)—maximum depth of the state space (may be \( \infty \))
**Breadth first search**

- Expand the shallowest unexpanded node
Breadth first search

- Expand the shallowest unexpanded node
Breadth first search

- Expand the shallowest unexpanded node
Breadth first search

- Expand the shallowest unexpanded node
Route finding
Uniform cost search

• Find the pass with cheapest total cost…
Route finding
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node

![Depth First Search Diagram](image-url)
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

• Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Depth First search

- Expand the deepest unexpanded node
Route finding
## Comparisons of algorithms

<table>
<thead>
<tr>
<th></th>
<th>Breadth First</th>
<th>Uniform Cost</th>
<th>Depth First</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimal?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Comparisons of algorithms

<table>
<thead>
<tr>
<th></th>
<th>Breadth First</th>
<th>Uniform Cost</th>
<th>Depth First</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complete?</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Optimal?</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>$O(b^d)$</td>
<td>$O(b^{(1 +[C/e])})$</td>
<td>$O(b^m)$</td>
</tr>
<tr>
<td><strong>Space</strong></td>
<td>$O(b^d)$</td>
<td>$O(b^{(1 +[C/e])})$</td>
<td>$O(bm)$</td>
</tr>
</tbody>
</table>
Figure 3.2  A simplified

<table>
<thead>
<tr>
<th>City</th>
<th>Distance to Bucharest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
</tr>
<tr>
<td>Drobeta</td>
<td>242</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
</tr>
<tr>
<td>Fagaras</td>
<td>176</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsova</td>
<td>151</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
</tr>
<tr>
<td>Neamet</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>100</td>
</tr>
<tr>
<td>Rimnicu Vilcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiui</td>
<td>253</td>
</tr>
<tr>
<td>Timisoara</td>
<td>329</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>

Figure 3.22  Values of $h_{SLD}$—straight-line distances to Bucharest.