(3) space? Also $O(b^d)$

(4) Optimal assuming for all actions cost = 1? Yes! Level-by-level so shallowest goal returned

DEF:
(1) complete?

Finite state space: Yes (if no cycles in paths)
Infinite: No (not always)

(2) time?

Even if goal is high up, still have to search up to max depth

$\Rightarrow O(b^m)$
(3) Space? Typically DFS implemented w/o Explored:

Back to exp

Frontier:

Storing at most

\( \text{b nodes for every node along branch from top to bottom of tree.} \Rightarrow O(bn) \)

Just store one path on tree at a time.

Notice: by the time we traverse back up to (A), to explore (A), the ABEF, ABEC, and ABET paths are no longer stored.

\( \Rightarrow O(bm) \)

Problem w/ not using Explored:

Redundant paths.

(4) Optimal? No, may find deeper goal first
So far assumed all actions have cost = 1

With varying costs, BFS + DFS would not be optimal
ex: if moving blank left cost more than U, D, R

How do we search for cheapest solution?

-> Use shortest path algorithm

Uniform cost search - name explain later

costs of actions are positive, non-identical

- Frontier is a priority queue with key = cost of action that yielded the node

- Always removing node with cheapest path

Suppose C is goal

A: cost = 0
all others = ∞

Current Tree Search:

<table>
<thead>
<tr>
<th>Frontier</th>
<th>Explored</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>E A G</td>
<td>E F</td>
<td>A</td>
</tr>
<tr>
<td>G B D E F</td>
<td>F G A</td>
<td>B &lt; NOT C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E</td>
</tr>
</tbody>
</table>

Update cost! (2)
Add to TreeSearch:

if n not in Frontier:

if (n in Frontier with higher path cost):
  update n's cost

Back to example:

\[
\begin{array}{c}
S \\
C \leftarrow \text{Goal! so done}.
\end{array}
\]

Larger Example:

Goal: (G)

\[
\begin{array}{cccc}
\text{Frontier} & \text{Exp.} & S \\
\{A\} & \{A\} & A \\
\{F, B, E\} & \{A\} & B \\
\{E, H, G\} & \{A, B, F\} & E \\
\{H, G\} & \{A, B, F, E\} & G \leftarrow \text{goal!}
\end{array}
\]
Notice: if costs are identical, UCS = BFS.

Evaluate UCS:

(i) complete?

Finite state space? Infinite?

![Diagram of search trees with goal nodes and cheapest paths indicated.]

(Yes as long as costs ≥ 0)

(ii) Time? # nodes in Frontier?

Previously, depended on:
- b: branching factor
- d: depth of shallowest goal
- n: maximum depth

Now? Don't care about shallowest goal, care about cheapest goal.

Max (worst-case) depth of cheapest goal?

Assume:
- c* : optimal cost
- k : min cost of an action
Max depth of cheapest goal? \( \frac{c^*}{k} \)

\[
\begin{array}{c}
\text{depth} \\
0 \\
1 \\
2 \\
\vdots \\
\ast \\
\hline
0 \leftarrow \text{Optimal Goal.} \\
\frac{c^*}{k}
\end{array}
\]

\# nodes \( \approx 1 + b + b^2 + b^3 + \cdots + b^\frac{c^*}{k} = O(b^\frac{c^*}{k}) \).

(3) Space is also \( \approx O(b^\frac{c^*}{k}) \).

(4) Optimal? Yes. Won't do formal proof but \( \sim \) Dijkstra's.

Intuition: since no negative costs, updates always improve costs.

Summary:

<table>
<thead>
<tr>
<th></th>
<th>Complete?</th>
<th>Time</th>
<th>Space</th>
<th>Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Finite State Space)</td>
<td>(Infinite State Space)</td>
<td>Uniform</td>
<td>Variate</td>
</tr>
<tr>
<td><strong>BFS</strong></td>
<td>✓</td>
<td>✓</td>
<td>( O(b^d) )</td>
<td>( O(b^d) )</td>
</tr>
<tr>
<td><strong>DFS</strong></td>
<td>✓</td>
<td>X</td>
<td>( O(b^m) )</td>
<td>( O(b^m) )</td>
</tr>
<tr>
<td>(w/o Explored)</td>
<td>(no loops)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>UCS</strong></td>
<td>✓</td>
<td>✓</td>
<td>( O(b^\frac{c^*}{k}) )</td>
<td>( O(b^\frac{c^*}{k}) )</td>
</tr>
<tr>
<td>Iterative Deep.</td>
<td>✓</td>
<td>✓</td>
<td>( O(b^d) )</td>
<td>( O(b \cdot d) )</td>
</tr>
</tbody>
</table>

How to combine optimality + completeness of BFS + UCS

w/ space of DFS?
Ways to improve DFS?

Problem is that depth may be infinite so we won’t find optimal or even any solution.

Set a limit on depth and search only to the limit?
Not complete if goal is beyond that limit.

So what’s a good limit? Try all! One at a time.

Iterative Deepening

Try limit = 0, 1, 2, 3

ex:

\[
\begin{array}{c}
\infty \\
\uparrow \\
\text{infinite branch}
\end{array}
\]

\[
\begin{array}{c}
\text{limit = 0} \\
\circ \\
\text{Seems wasteful since we are generating nodes but only } \frac{1}{2} \\
\text{of the nodes are regenerated from each previous limit.}
\end{array}
\]

\[
\begin{array}{c}
\text{limit = 1} \\
\circ \\
\end{array}
\]

\[
\begin{array}{c}
\text{limit = 2} \\
\circ \\
\end{array}
\]

(i) Complete?
Finite Space: ✓
Infinite: ✓
(2) time?

\[ \text{# nodes generated} = 1 + (1+b) + (1+b+b^2) + (1+b+b^2+b^3) + \ldots + (1+b+b^2+\ldots+b^d) \approx O(b^d) \ (\text{vs} \ 0(b^m) \ \text{for DFS}) \]

(3) space? Will search to depth at most d.

\[ O(b \cdot d) \ (\text{vs} \ 0(b \cdot m) \ \text{for DFS}) \]

(4) optimal?

Yes if costs are uniform.

No if not.

⇒ Add to table ⇒