Previous lecture

frontier (list)

"node"

extract the solution just a state will not give us

the solution

store the path in every node.

completeness  \( \square \)  BFS  \( \times \)  DFS

will not be able to

yield a solution if

at least one branch

before the goal's

branch is infinite.

BFS time complexity

worst case

how many children for a node

maximum (worst)

branching factor \( b \)

\( \square \) goal at the far right

("worst" location)

at depth \( d \)

how much time is needed to get to the

goal on \( \bigl\{ \text{a tree (with } b, \text{ and } d) \bigr\} \)

BFS  \( \sqrt{\text{ }} \)

time to visit a node

how many nodes are visited.
\[ b^0 + b^1 + b^2 + \ldots + b^{d-1} \]

# of nodes generated

\[ b^0 + b^2 + \ldots + b^{d-1} + b^d \]

0 \( \left( b^d \right) \) exponential!

DFS time complexity 

(assume all branches are finite with a maximum depth of \( m \))
Space complexity of BFS

How many nodes are stored (maximum) in the frontier?

\[
\begin{align*}
\text{(a)} & \quad \text{b} \\
\text{(b)} & \quad \text{b} \\
\text{(c)} & \quad \text{b} \\
\text{X} & \quad n-3
\end{align*}
\]

previous moves:

\[
\begin{align*}
\text{b} \\
\text{b} \\
\text{b} \\
\text{b} \\
\text{b}
\end{align*}
\]

goal test:

When generality

When expensiveness

\[
\begin{align*}
\text{(b)} & \quad ? \quad \text{b} \\
\text{(c)} & \quad ? \quad \text{b} \\
\text{d+1} & \quad ? \quad \text{b}
\end{align*}
\]
Summary of algorithms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time</td>
<td>$O(b^d)$</td>
<td>$O(b^{1+\lfloor C^*/\epsilon \rfloor})$</td>
<td>$O(b^m)$</td>
<td>$O(b')$</td>
<td>$O(b^d)$</td>
<td>$O(b^{d/2})$</td>
</tr>
<tr>
<td>Space</td>
<td>$O(b^d)$</td>
<td>$O(b^{1+\lfloor C^*/\epsilon \rfloor})$</td>
<td>$O(bm)$</td>
<td>$O(bl)$</td>
<td>$O(bd)$</td>
<td>$O(b^{d/2})$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>Yes*</td>
<td>Yes*</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
New ideas: have a search technique with a linear space complexity (DFS) and optimality (BFS).

Iterative Deepening Search (IDS)

\[ l = 0 \quad 0 \]
\[ \downarrow \text{erase everything} \]
\[ l = 1 \]
\[ \downarrow \text{erase everything} \]
\[ l = 2 \]

Nodes at level \( l \) and lower are not allowed to have children.

Has repeat repetition but has linear space complexity.

Why? Finds the optimal solution why?