What is Iterative Deepening Search?

Iterative Deepening Search (IDS) is an iterative graph searching strategy that takes advantage of the completeness of the Breadth-First Search (BFS) strategy but uses much less memory in each iteration (similar to Depth-First Search).

IDS achieves the desired completeness by enforcing a depth-limit on DFS that mitigates the possibility of getting stuck in an infinite or a very long branch. It searches each branch of a node from left to right until it reaches the required depth. Once it has, IDS goes back to the root node and explores a different branch that is similar to DFS.

Time & space complexity

Let's suppose that we have a tree where each node has `b` children. We will consider this our branching factor and take `d` as the depth of the tree.

Nodes at the bottom-most level, which would be level d, will be expanded exactly once; whereas, nodes on level $d-1$ will be expanded twice. The root node of our tree will be expanded $d+1$ times. If we sum all these terms up, it will be:

$$(d)b+(d-1)b_2+...+(3)b_{d-2}+(2)b_{d-1}+b_d$$

This summation will result in time complexity of $O(b_d)$

The space complexity for IDS is $O(bd)$. Here, we assume that `b` is a constant, and all children are generated at each depth of the tree and stored in a stack during DFS.

Performance analysis

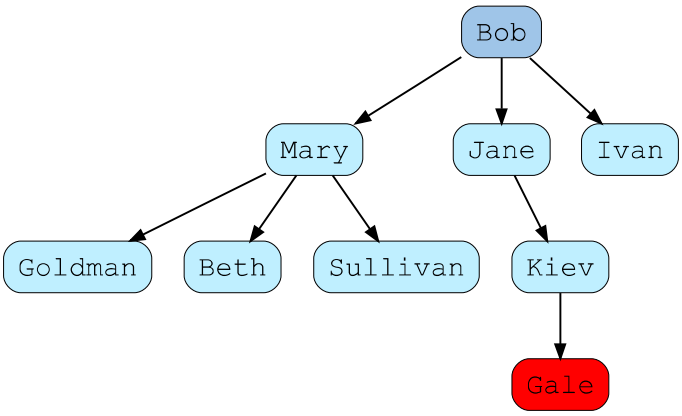
It may look like IDS has a large overhead in the form of repeatedly going over the same nodes again and again, but this turns out to be much of a big deal. This is because most nodes in a tree are at the bottom levels, which are visited only once or twice by the algorithm. Because of this, the cost is kept to a bare minimum since the upper-level nodes do not make up the majority of the nodes in a tree.

Let's see the pseudocode for this strategy and run it on an example:

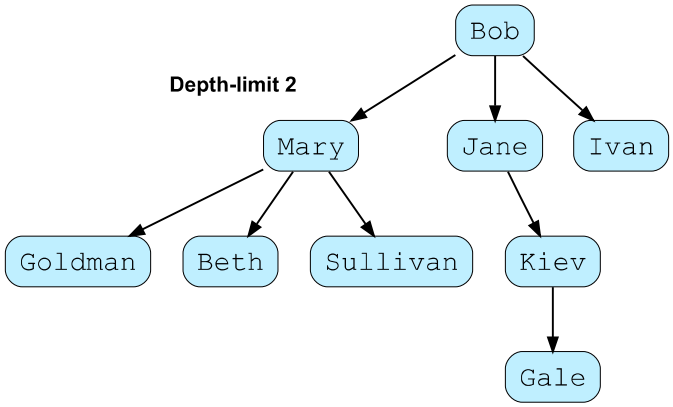
```
IDS(root, goal, depthLimit){
   for depth = 0 to depthLimit
      if (DFS(root, goal, depth))
         return true
```

```
    return false
}

DFS(root, depth){
    if root == goal
        return true
    if depth == 0
        return false
    for each child in root.children
        if (DFS(child, goal, depth - 1))
            return true
    return false
}
```



With a depth-limit of 2, iterative-deepening will never encounter Gale since it is on the 3rd level.



1 of 12

When to use iterative deepening

As a general rule of thumb, we use iterative deepening when we do not know the depth of our solution and have to search a very large state space.

Iterative deepening may also be used as a slightly slower substitute for BFS if we are constrained by memory or space.

RELATED TAGS

general

CONTRIBUTOR

Muhammad Adan

Related Courses

Educative

Recursion for Coding Interviews in Python

Intermediate

Preview

Tariq Rashid

Make Your Own Neural Network in Python

Beginner

Preview

Keep Exploring

What is fuzzy search?

Search insert position problem from LeetCode

What is the numpy.intersect1d() function in Python?

Did you find this helpful?