Searching for solutions

How do we solve above kinds of problems?
We will use problem to create a search tree. Root of tree is initial state. (vertices of tree called sarch nodes)
General strategy at a search node – apply goal test to see if node satisfies goal. If yes done. If no, apply successor function to get all nodes reachable from current node in one step (expanding a node). Add these nodes to a list of nodes we need to consider. Pick some node from this list and repeat.

Strategies for picking which node to expand next.

- Depth First Search (always expand left most node that still can be expanded)
- Breadth first search (Expand root, then expand all children of root until reach goal.)

Uninformed Search Strategies
(don't have any way of telling if getting close to a solution)

Time Complexity of Breadth First Search
Suppose each node expands into b children
Then to search for a goal of depth d takes time proportional to
$1 + b + b^2 + \ldots + b^d = b^{d+1}-1 = O(b^d)$
Space complexity is also $O(b^d)$

Depth First Search – Always expand the deepest node that can be expanded (if tie choose left most node)

Time complexity of DFS

If tree has a solution of depth m and this bounds length of any path, let's say branching factor b. Then time takes is $O(b^m)$

Only need to remember path to use algorithm so space complexity is O(b*m)

Problem can get stuck on infinite branches and never find a solution.

Depth limited search upto L search

- does depth first search to some fixed depth L

(i.e. not allowed to expand node to depth >= L)
Problem might never find solution because solution has depth > L

Iterative deepening search

```
Do      DLS(0)
        DLS(1)
        DLS(2)
        …
        Until find a solution
```

Space Complexity O(bm)
Time Complexity $O(b^m)$