**#recording5**

Edges = actions

If edge costs are all 1, BFS finds the closest goal.

The problem is needed **memory** will increase exponentially.

However, DFS stores nodes root to node path.

#You should memorize this algorithm!

Function tree-search(problem)

Frontier = initial state

Loop forever:

If frontier is empty, then fail

N = remove a node from frontier

If n is a goal, return n

For each child c of n:

Add c to frontier

#if frontier is implemented in a stack: DFS

#if frontier is implemented in a queue: BFS

**#recording6**

BFS

DFS

Depth-Limited Search

-nodes at depth limit L from the root are treated as if they have no children.

-> This is good when lack of memory or time.

Iterative Deepening DFS

Good: Like BFS, will find nearest goal node (assuming al edge weights are the same cost)

Good: only uses memory proportional to depth of search. Much better than BFS

Bad: Visits some nodes multiple times

If b = branching factor (number of children/node), d= depth

Then ID-DFS generates O(b^d) nodes. (Same as BFS)

Intuition: most of the nodes in a tree are deep and ID-DFS mostly re-expands nodes near the root of tree.

(leaves generated once ~50% of all nodes)

Heuristic Search

Let f(n) be a scoring function for node n

Let g(n) be the cost of the path edges from the root to n

Let h(n) be the ‘estimated’ cost from node n to the nearest goal.

(h is the heuristic function)