Search

★ Reflex Agents

→ Choose action based on current percept (and may be memory)

→ May have memory on a model of the world's current state

→ Do not consider the future consequence of their actions.

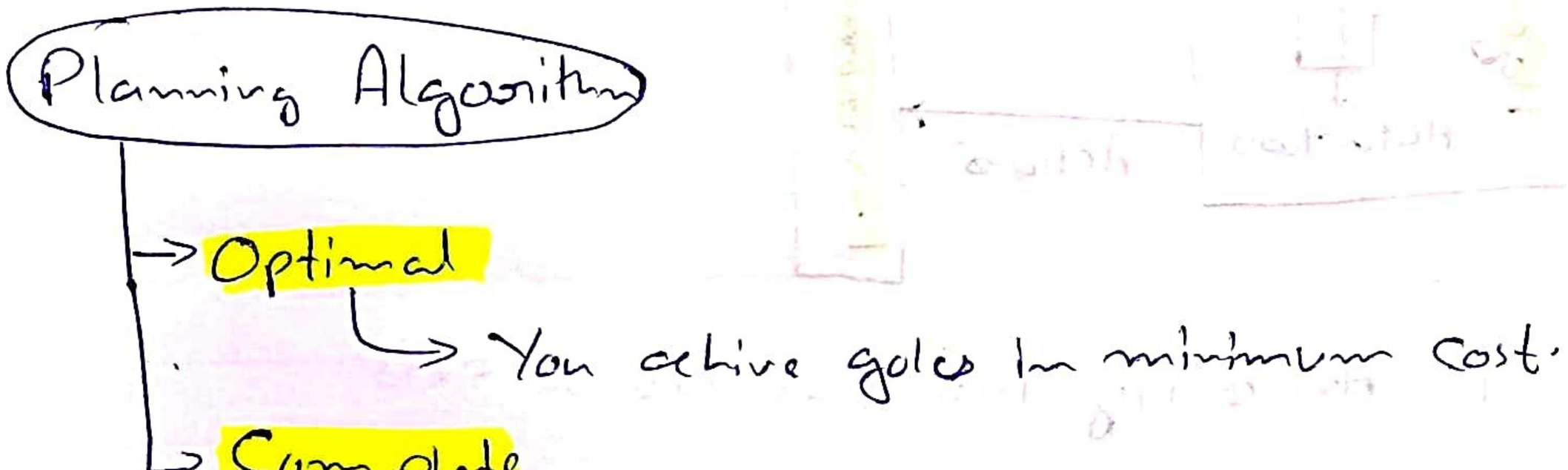
★ Planning Agents

→ Ask "what if"

→ Decision based on consequence of action.

→ Must have a model of how the world evolves in response to actions.

→ Must formulate a goal.



★ Search problems

⇒ A **Search problem** consists of:

① A State Space

② A Successor function
(with actions, costs)

③ A start state & a goal test

⇒ A **Solution** is a sequence of actions (a plan) which transforms the start state to a goal state

★ State Space Graphs

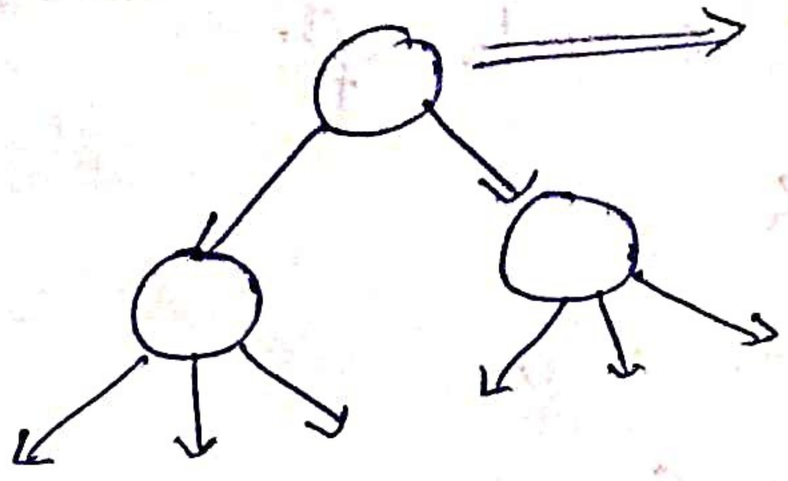
⇒ "A Mathematical representation of a Search Problem"

* Nodes are abstracted world configurations.
* Arcs represent successors
* The goal test is set of goal nodes (may be only one)

⇒ In a state space graph, each state occurs only once.

⇒ We can rarely build the full graph in memory (its too big) but it's useful idea.

★ Search Trees



* A "what if" tree of plans & their outcomes.
* The start state is the root node
* Children correspond to successors.
* Nodes show states, but correspond to PLANS that achieve those states.
* For most problems, we can never actually build the whole tree

★ General Tree Search

* Expand out potential plans (tree nodes)
* Maintain a fringe of partial plans under consideration
* Try to expand as few tree nodes as possible.

function TREE SEARCH (problem, strategy) returns Solution or failure

1. initialize the search tree using the initial state of problem
2. loop do
3. if there is no candidate for expansion then return false.
4. Choose a leaf node for expansion according to strategy.
5. if the node contains a goal state then return the corresponding Solution
6. Else expand the node and add the resulting nodes to the search tree
7. end

★ Depth first Search

Strategy: Expand deepest node first

Implementation: Fringe is a LIFO stack

★ Search Algorithm Properties

① Complete
↳ Guaranteed to find a solution if one exists.

② Optimal
↳ Guaranteed to find the least cost path

③ **Time Complexity**

↳ How long does the compute takes to find a solution.

④ **Space Complexity**

↳ How much memory do you need to find a solution.

★ **Depth-first Search (DFS) properties**

What nodes does DFS expands

↳ Could process the whole tree

↳ If m is finite, take time $O(b^m)$

How much space does the fringe take?

↳ Only has siblings on path to root, so $O(bm)$

Is it Complete?

↳ m could be infinite, so only if we Prevent Cycles.

Is it Optimal?

↳ No, it finds the "leftmost" solution, regardless of depth or cost.

★ Breadth-First Search

Strategy: Expand a shallowest node, first

Implementation: Fringe is a FIFO Queue.

* What nodes does (BFS) expands?
  - → Process all nodes above shallowest solution
  - → Let depth of shallowest solution be s
  - → Search takes time $O(b^s)$

How much space does the fringe take?
  - → Has roughly the last tier, so $O(b^s)$

Is it Complete?
  - → s must be finite if a solution exists, so Yes!

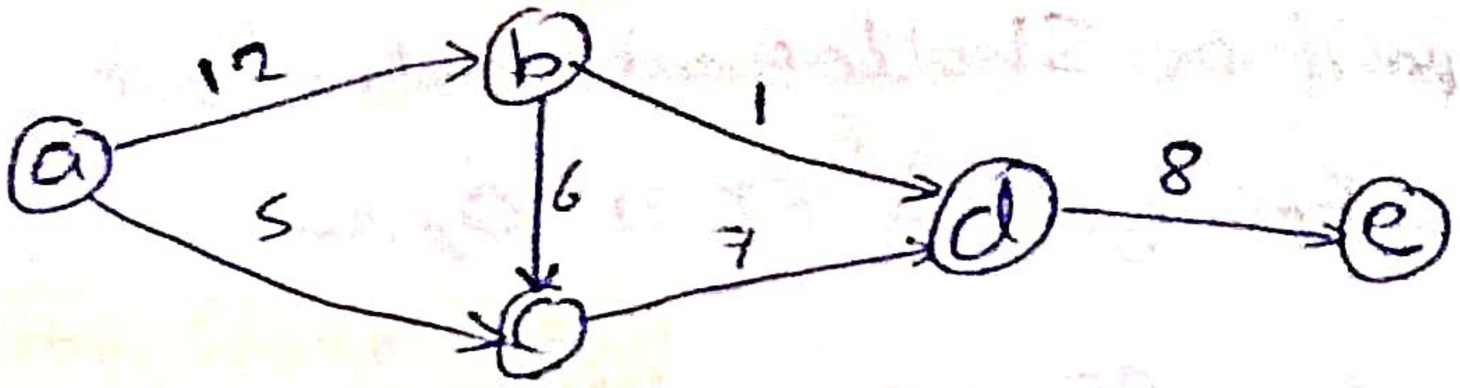
Is it optimal?
  - → Only if Costs are all 1

★ Iterative Deepening

Idea: Get DFS's space advantage with BFS's time/shallow-solution advantages.

Isn't that wasteful & redundant?
  - → Generally most of the work happens in the lowest level searched, so not so bad!

★ Cost-Sensitive Search



★ Uniform Cost Search (UCS)

Strategy: Expand a cheapest node first

Fringe: A Priority queue
(Priority: Cumulative Cost)

What nodes does UCS expands?

⇒ Let C^* be the cost of optimal solution
& Each individual step cost is atleast ε.

⇒ So we might have expanded all the
Plans takes C^*/ε steps.

⇒ Time taken: $O(b^{C^*/\varepsilon})$

How much space does the fringe take?

↳ Has roughly the last tier, so $O(b^{C^*/\varepsilon})$

Is is Complete?

↳ Assuming best Solution has a finite cost &
minimum arc cost is positive Yes!

Is it Optimal?

↳ Yes!

Good

↳ Complet & Optimal

Bad

↳ Explores Options in every direction

↳ No information about goal location