Solving Problem by Searching

1. **Problem-solving Agents**

* Note: agent for simplification, we only consider state representation.
* A problem-solving agent can be understood as solving a problem
* Must be so simple that we can install it.
* If we want it to be simplified, then we have to consider the properties of the environment:
* Observable
* Discrete
* Known
* Deterministic
* Problem-solving agents will have two steps:
* Search
* Execution phase
* To define a problem for an agent, we must have 5 components:
* Initial state
* Actions
* Transition model
* Goal test
* Path cost
* The solution to a problem is a sequence of actions leading from the initial state to a target state

1. **Example Problems**
2. **Searching for Solutions**

* All sort algorithms are based on the data structure of the search tree
* The search tree will start in the initial state
* Each node of the search tree corresponds to a state of the problem
* Branches corresponds to actions
* Root node of search tree corresponds to the initial state (state + information)
* Performing an operation with a search tree is expanding the current state, which generates more child nodes
* Frontier: set of leaf nodes
* Note:
* *Same*: Search algorithms all share this basic structure as above
* *Different*: Choose any node in the split so that it expands - search strategy
* Loopy path cause repeated states in the search tree
* Redundant paths exist whenever there is more than one way to get from one state to another, some cases create loops.
* Graph search use the explored set, a set or data structure used to store passed or opened states
* Properties of graph search: the frontier separates the state-space graph into 2 region
* The explored region
* The unexplored region
* If we want the initial state to an unexplored state, we have to pass through a state in the frontier
* Each node n has a structure that contains 4 components:
* n.State
* n.Parent
* n.Action
* n.Path-Cost
* There are 4 criteria to evaluate an algorithm’s performance:
* Completeness
* Optimality
* Time complexity
* Space complexity
* All reviews are based on search tree
* Time and space complexity are based on:
* b: maximum branching factor of the search tree
* d: depth of the least-cost/shallowest solution
* m: maximum depth of the state space (can be infinite)