**[Week 0](https://cs50.harvard.edu/ai/2024/weeks/0/" \l "week-0-search)**[**Search**](https://cs50.harvard.edu/ai/2024/weeks/0/#week-0-search)

Search Problems. Depth-First Search. Breadth-First Search. Greedy Best-First Search. A\* Search. Minimax. Alpha-Beta Pruning.

* **agent** – entity that perceives its environment and acts upon that environment
* **state** – configuration of the agent and ist environment. (node of a graph)
* **initial state** – the state in which the agent begins
* **actions** - choices that can be made in a state. *Actinions(s)* – returns the set of functions that can be executed in a state s. (arrow of a graph)
* **transition model** – a description of what state results from performing any applicable action in any state. *Result(s, a)* returns the state resulting from performing action *a* in state *s*.
* **state space** – the set of all states reachable from the initial state by any sequece of actions
* **goal test** – way to determine wether a given state is a goal state
* **path cost** – numerical cost assosiated with a given path

SEARCH PROBLEM

* **initial state**
* **actions**
* **transition model**
* **goal test**
* **path cost function**

SOLUTION

* **solution** – sequence of actions that leads from the initial state to the goal state
* **optimal solution** – a solution that hast he minimal path cost among all solutions
* **node** – data structure that keeps track of:
* **state**
* **parent** (node that generated this node)
* **action** (applied to parent to get node)
* **path cost** (from initial state to node)

APPROACH

* starts with **frontier** (data structure) that contains the initial state
* start with an **empty set of explored nodes**
* repeat:
* if frontier is empty then no solution
* remove a node from the frontier
* if node contains goal state, return the solution
* add the node tot he explored set
* expand node (look at all neighbours, consider all possible actions), add resulting nodes to the frontier if they are not already in the frontier or the explored set

DEPTH-FIRST SEARCH (DFS)

Search algorithm that always expands the deapest node in the frontier

when the frontier is a stack (LIFO)

BREADTH-FIRST SEARCH (BFS)

Search algorithm that always expands the shallowest node in the frontier

when the frontier is a queue (FIFO)

**Uninformed search** -- search strategy that uses no problem specific knowledge (DFS; BFS)

I**nformed search** -- search strategy that uses problem-specific knowledge to find solutions more efficiently

* **greedy best-first search (GBFS) –** search algorithm that expands the node that is closest to the goal , as estimated by heuristic function *h(n)*
* **A\* search –** search algorithm that expands the node with lowest value of g(n) + h(n),

g(n) – costs to reach node

h(n) – estimated cost to goal

optimal if:

* *h(n)* is admisible (never overestimates the true cost) and
* *h(n)* is consistent (for every node *n* and successor *n‘* with step cost *c* *h(n) <= h(*n‘*) + c* )

**ADVERSARIAL SEARCH**

I the agent that is trying to find the solution and there is someone who is trying to stop me (e.g. tic-tac-toe)

* **Minimax**
* **MAX(X)** aims to maximize the score
* **MIN(O)** aims to minimize the score

Given a state *s*:

* MAX picks action a in Actions(s) that produces highest value of MIN-VALUE(Result(s, a))
* MIN picks action a in Actions(s) that produces smallest value of MAX-VALUE(Result(s, a))

*Alpha-Beta Pruning* – optimization approach for minimax, keep track of the best i can do (alphy) and the worst i can achieve (beta) and optimize by not searching through everything (by removing some ot the nodes)

* **Depth-Limited Minimax**
* **evaluation function –** function that estimates the expected utility oft he game from a gaven state
* **h**