Search in AI + Production Systems

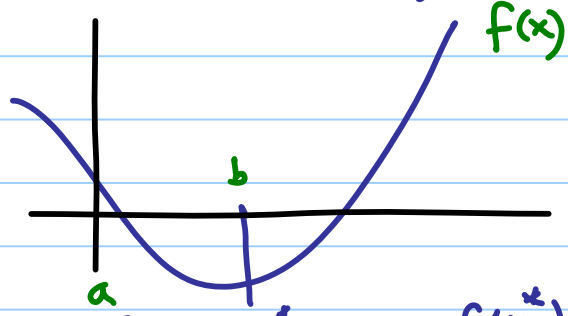
Note Title

Classical Search

Linear Search
Binary Search

Given $x_0, x_1, \ldots, x_{N-1}$
Find i s.t. $x_i = key$ or -1 if not found

"Root-Finding Methods"

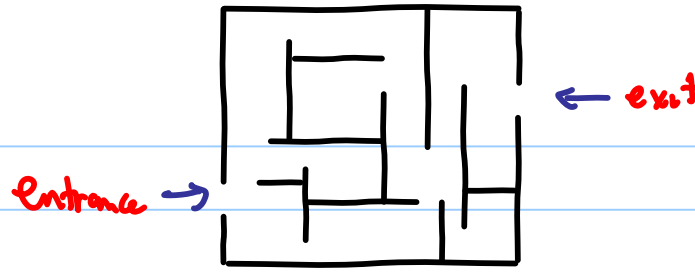


Find x^* s.t. $f(x^*) = 0$

a method:
Bisection → given $[a, b] \in \mathbb{R}$
where it is known that $f(x) = 0$ for some $x \in [a,b]$ [$f(a)$ + $f(b)$ have opposite signs]

⋮
⋮
⋮
⋮ ⋯⋯→ compute $m \leftarrow (a+b)/2$
if $f(a)$ and $f(m)$ have same sign, $a \leftarrow m$
else $b \leftarrow m$

Problem Solving



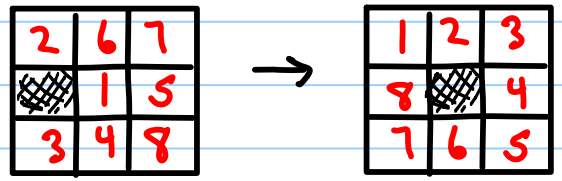
(a) Given a maze
Entrance, exit, walls

Find a sequence of moves that can get you from entrance to exit.

(b) A farmer is taking a fox, goose, and bag of corn to market.
He must cross a river and the boat is only large enough to transport himself + one item at a time. If left unattended, the fox will eat the goose and/or goose will eat the corn. Plan a way for the farmer to get them across.

(c) Given two jugs – one w/ 5 quart capacity, another with 2-qt capacity.
The 5 quart jug is full. Find a way of pouring water from jug to jug (or to ground) which leaves 1 qt. in the 2qt. jug.

(d) "8-puzzle" – given a sliding-block puzzle in an initial configuration, find a sequence of moves that yields another desired configuration



What is a "problem" of this nature?

- Knowledge Base = some set of information that completely describes the problem
  - State = particular current configuration of the system
  - goal condition = true if state satisfies desired configuration
- Actions / Moves / Rules = "legal moves"

We want to find a way of choosing the best move

rule = (precondition, action)

↓ is the move applicable to current state?

↓ "move to be applied"

applying action to current state produces new state

When several are applicable

- "Search strategy"

A very simple (not very intelligent) search strategy —

given state,

While not goal (state)

- Find all applicable rules
- Try one at random
- update state

"FLAIL WILDLY" strategy

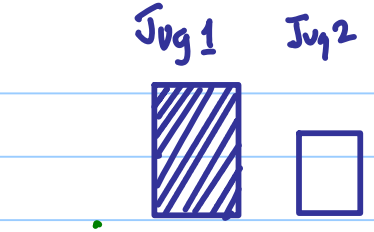
note: doesn't always work

e.g. pour all water out of 5 gt. jug

or take corn across river first

	Maze	Water Jugs	Fox - Goose - Corn
Knowledge Base	Maze configuration: positions of walls, entrance, exit	# of jugs capacities of jugs	
State	where the user is	contents of each jug	• position of boat/farmer • contents of left bank • contents of right bank
rules/ moves	action: ↑ ↓ ← → precond: not blocked + on board; not blocked; not blocked; not blocked	action: pour all water out of jug i precond: jug i not empty action: pour enough water from jug i to fill jug j precond: amt(i) + amt(j) ≥ capacity(j) action: pour all water from jug i to jug j precond: amt(i) + amt(j) ≤ capacity(j)	Farmer → Farmer, Goose → Farmer, Fox → Farmer, Corn → Farmer ← Farmer, Goose ← Farmer, Fox ← Farmer, Corn ← Feast condition
goal	state == exit	amt(2) == 1	all occupants on right

A CLOSER LOOK:



action	precond	
pour all water out of jug i	jug i not empty	
pour enough water from jug i to fill jug j	amt(i) + amt(j) ≥ capacity(j)	Jug i not empty; Jug j not full
pour all water from jug i → jug j	amt(i) + amt(j) ≤ capacity(j)	Jug i not empty; Jug j not full

pour all from 1 to 2 ✗
Pour all from 2 to 1 ✗

state: (5, 0) amt(1) = 5 amt(2) = 0

Pour all water out of Jug 1
(-5, 0) → (0, 0)
Pour all water out of Jug 2
✗ not applicable

Pour enough water from Jug 1 to Jug 2 to fill Jug 2
precond OK: (-2 2)
→ (3, 2)
Pour enough water from Jug 2:
✗ not applicable

```
state = initialstate
While not goal (state)
    Rules ← ApplicableRules(state)
    choose r ∈ Rules
    state = ApplyRule(r, State)
```

```
result = null
for each rule
    if precond(rule, state)
      add rule to result
    return result
```

"FLAILING" Strategy

Finding a valid sequence of rules efficiently is the "intelligent" part.

"GPS" — Generalized Problem Solver
(Newell & Simon, late 1950's)