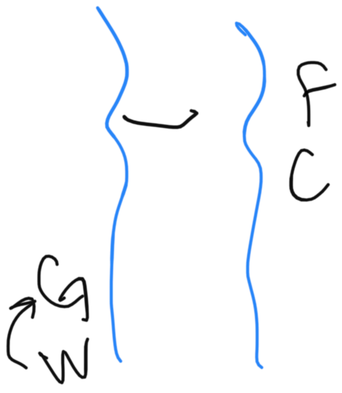
answer in chat

Question

A **farmer** wants to get his **cabbage**, **goat**, and **wolf** across a river. He has a boat that only holds two. He cannot leave the cabbage and goat alone or the goat and wolf alone. How many river crossings does he need?



- 4
- 5
- 6
- 7
- no solution

activate deactivate reset report

Paradigm

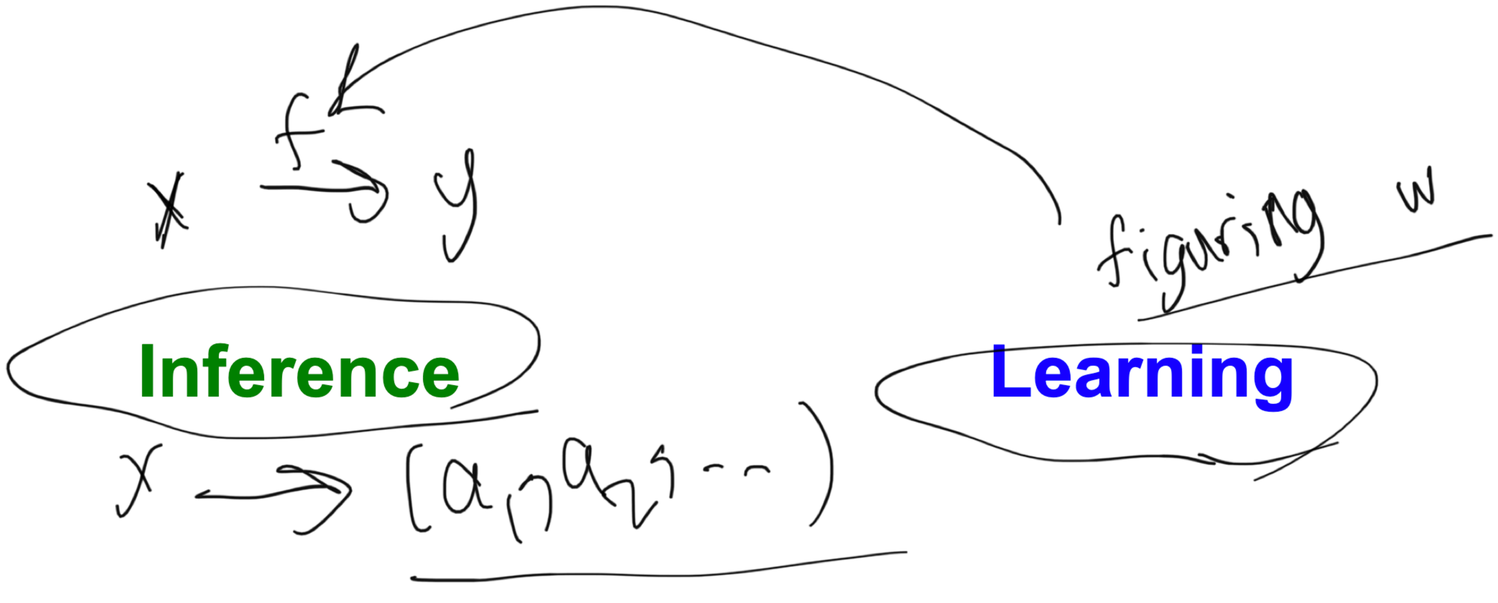
search

sequences of actions

Modeling

reflex

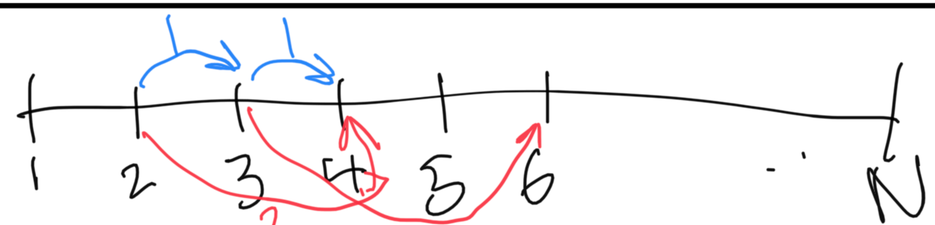
linear classifier
neural net



Transportation example

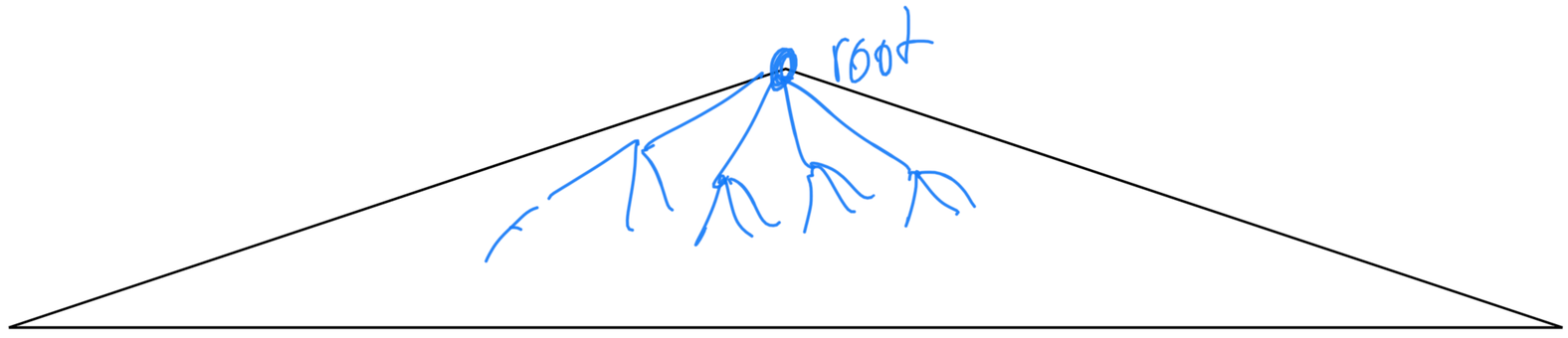
Example: transportation

Street with blocks numbered 1 to n.
Walking from s to $s+1$ takes 1 minute.
Taking a magic tram from s to $2s$ takes 2 minutes.
How to travel from 1 to n in the least time?



[semi-live solution: `TransportationProblem`]

Backtracking search



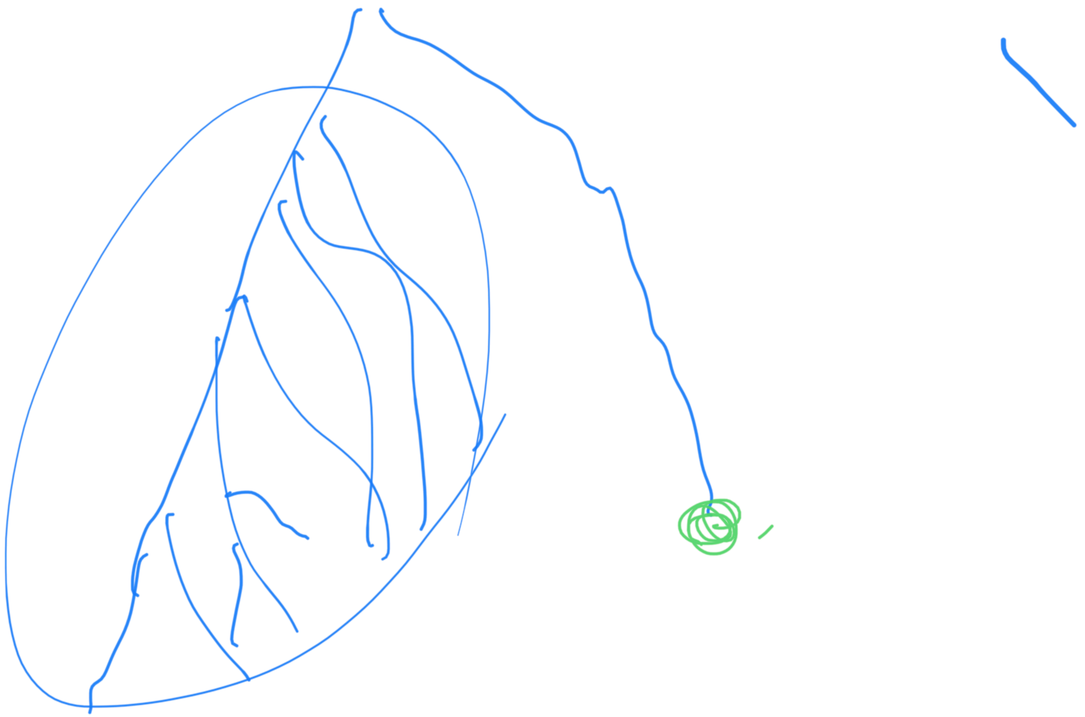
[whiteboard: search tree]

If b actions per state, maximum depth is D actions:

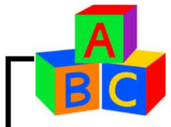
Breadth-first search

Assumption: constant action costs

Assume action costs $\text{Cost}(s, a) = c$ for some $c \geq 0$.



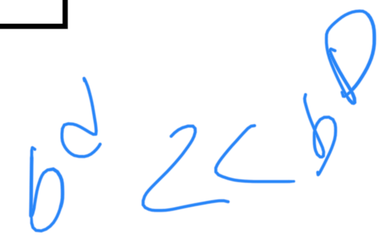
Breadth-first search



Assumption: constant action costs

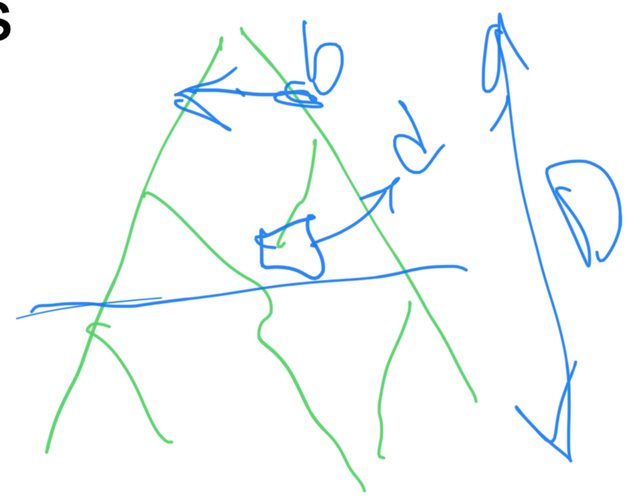
Assume action costs $\text{Cost}(s, a) = c$ for some $c \geq 0$.

Idea: explore all nodes in order of increasing depth.

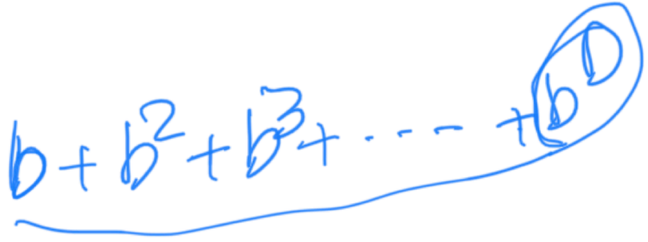


Legend: b actions per state, solution has d actions

- Space: now $O(b^d)$ (a lot worse!)



Depth-first search



Assumption: zero action costs

Assume action costs $\mathrm{Cost}(s, a) = 0$.

Idea: Backtracking search + stop when find the first end state.

If b actions per state, maximum depth is D actions:

- Space: still $O(D)$
- Time: still $O(b^D)$ worst case, but could be much better if solutions are easy to find

DFS with iterative deepening

Assumption: constant action costs

Assume action costs $\text{Cost}(s, a) = c$ for some $c \geq 0$.

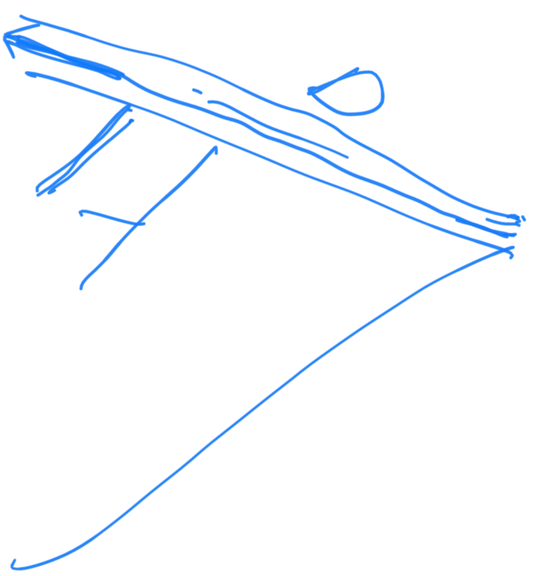
Idea:

- Modify DFS to stop at a maximum depth.
- Call DFS for maximum depths $1, 2, \ldots$.

  DFS on d asks: is there a solution with d actions?

Legend: b actions per state, solution size d

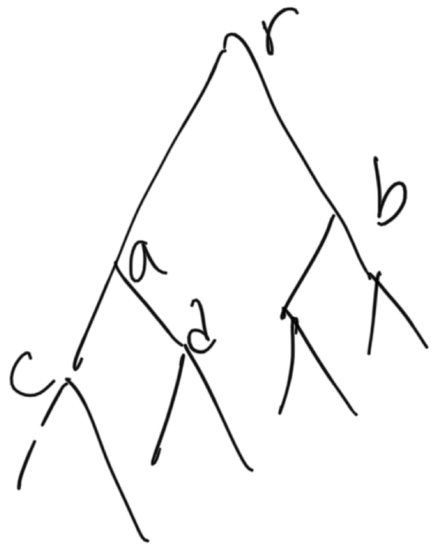
- Space: $O(d)$ (saved!)
- Time: $O(b^d)$ (same as BFS)



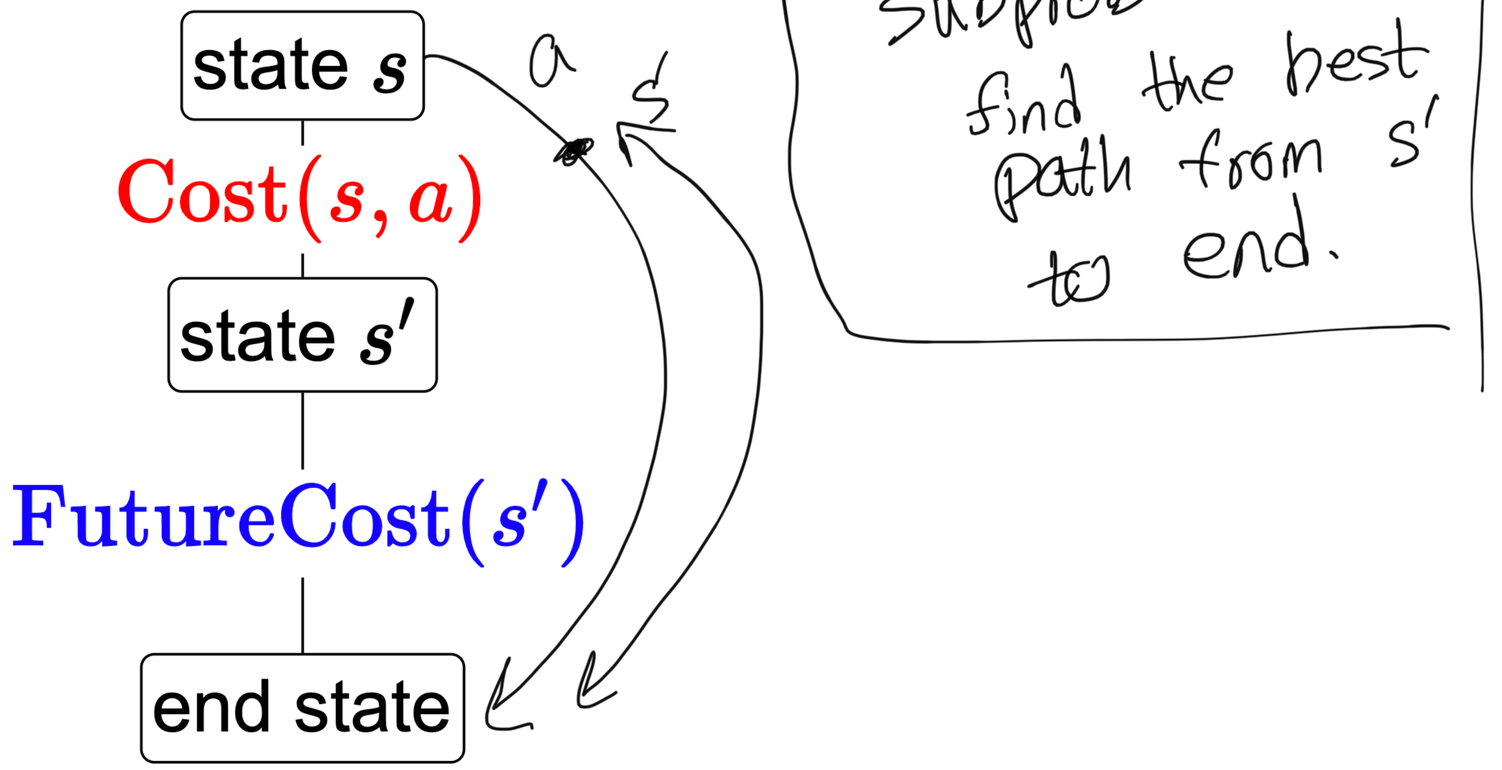
DFS with iterative deepening

Assumption: constant action costs

Assume action costs $\text{Cost}(s, a) = c$ for some $c \geq 0$.



Dynamic programming



Dynamic programming

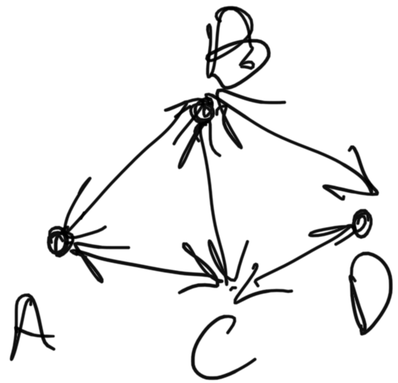
Algorithm: dynamic programming

def $\mathrm{DynamicProgramming}(s)$:

If already computed for s, return cached answer.

If $\mathrm{IsEnd}(s)$: return solution

For each action $a \in \mathrm{Actions}(s)$: ...



[semi-live solution: `Dynamic Programming`]

Assumption: acyclicity

The state graph defined by $\mathrm{Actions}(s)$ and $\mathrm{Succ}(s, a)$ is acyclic.

Handling additional constraints

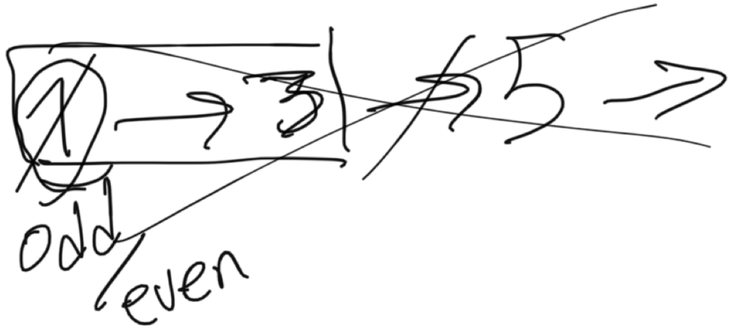
Example: route finding

Find the minimum cost path from city 1 to city n, only moving forward. It costs c_{ij} to go from i to j.

Constraint: Can't visit three odd cities in a row.

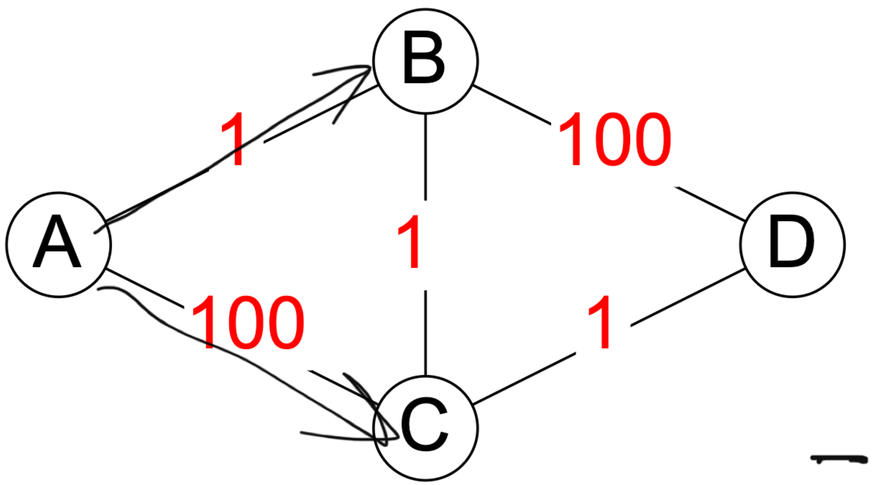
State: (previous city, current city)

$O(n^2)$

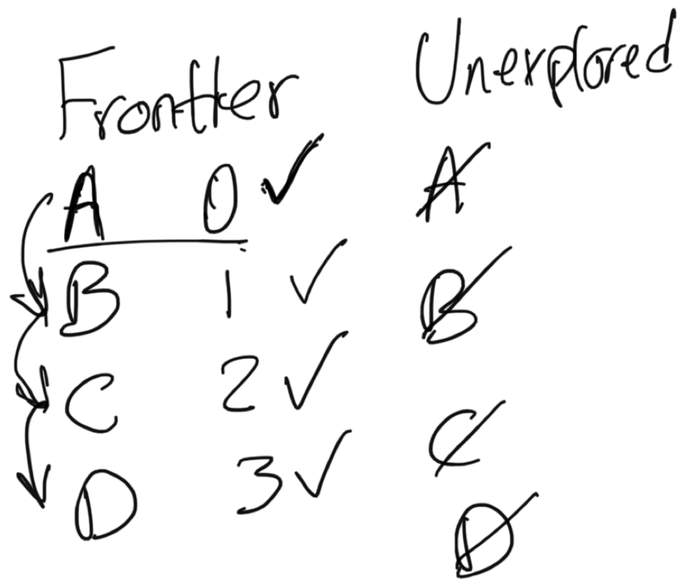


Uniform cost search example

Example: UCS example



Start state: A, end state: D



[whiteboard]

tree search: $O(b^D)$

dynamic programming $O(N \cdot A)$

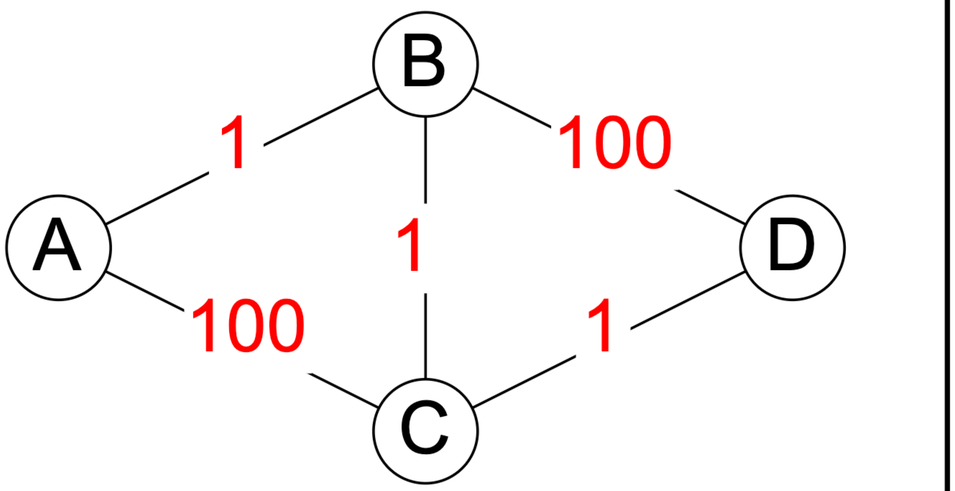
uniform cost search $O(N \cdot A \cdot N \log N)$

N states

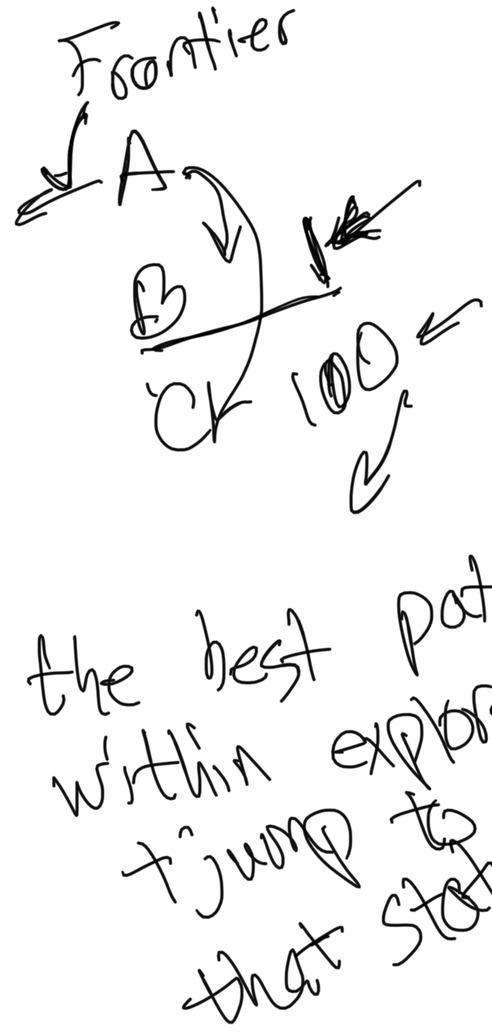
A: actions per state

Uniform cost search example

Example: UCS example



[whiteboard]



Minimum cost path:

$$A \rightarrow B \rightarrow C \rightarrow D \text{ with cost } 3$$