

Problem: State Space Problem

Mathematical Graph

- ↳ nodes particular states of the world
- ↳ arcs actions that take world from one state to another.

Solution: Plan.

- A sequence of actions
- A path in the graph

• GENERIC GRAPH SEARCH:

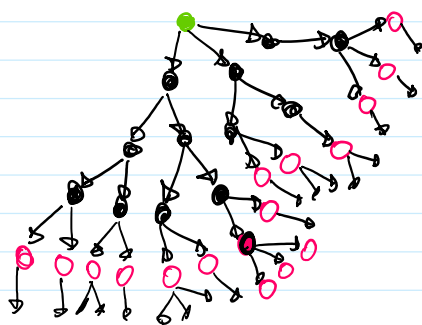
- Input: A Graph
 - explicitly
 - implicitly
 - start s
 - $\text{next}(s, a) = s'$
 - $\text{neighbors}(n) \rightarrow [n_0, n_1, n_2, \dots, n_k]$
 - $\langle n, n_i \rangle \in A$

• The start state/node

• A function $\text{goal}(s)$ to recognize whether a state satisfies the goal conditions.

• Intuition:

Suppose this search has already been running.



- the algorithm has already considered a collection of paths.
 - closed paths.

- the algorithm maintains extensions of those close paths.
 - the frontier.

At every iteration:

- select one path from the frontier.
- test whether the end node of that path satisfies the goal

- select one path from the frontier.
- test whether the end node of that path satisfies the goal
- if not:
 - close selected path
 - add extensions of the path to the frontier

• Pseudo-Code.

FUNCTION GenericGraphSearch()

INPUT: a Graph

a start node s

a boolean function goal(n)

BEGIN

frontier := { [s] : s is the start node }

WHILE frontier is not empty DO

select and remove a path $p=[n_0, n_1, n_2, \dots, n_k]$ from the frontier

IF goal(n_k) THEN

RETURN p

FOR every neighbor n of n_k DO

add $[n_0, n_1, n_2, \dots, n_k, n]$ to the frontier

END

RETURN Fail 😞

END.

Notes:

- RETURN p :- does not have to be "final", and could be called again
 - the frontier captures the progress
- goal(n) :- is used when a path is selected (not when a path is added)
 - goal() could be an expensive function
 - there may be more than one path that leads to the goal. we want the selection criteria to break any ties.
- Different selection criteria will lead to different algorithms
 - ↳ "Search Strategies"
- Possible enhancements:
 - instead of sequence of nodes:- sequence of actions.
 - store only relevant information, instead of complete path.

—o—o—EOF