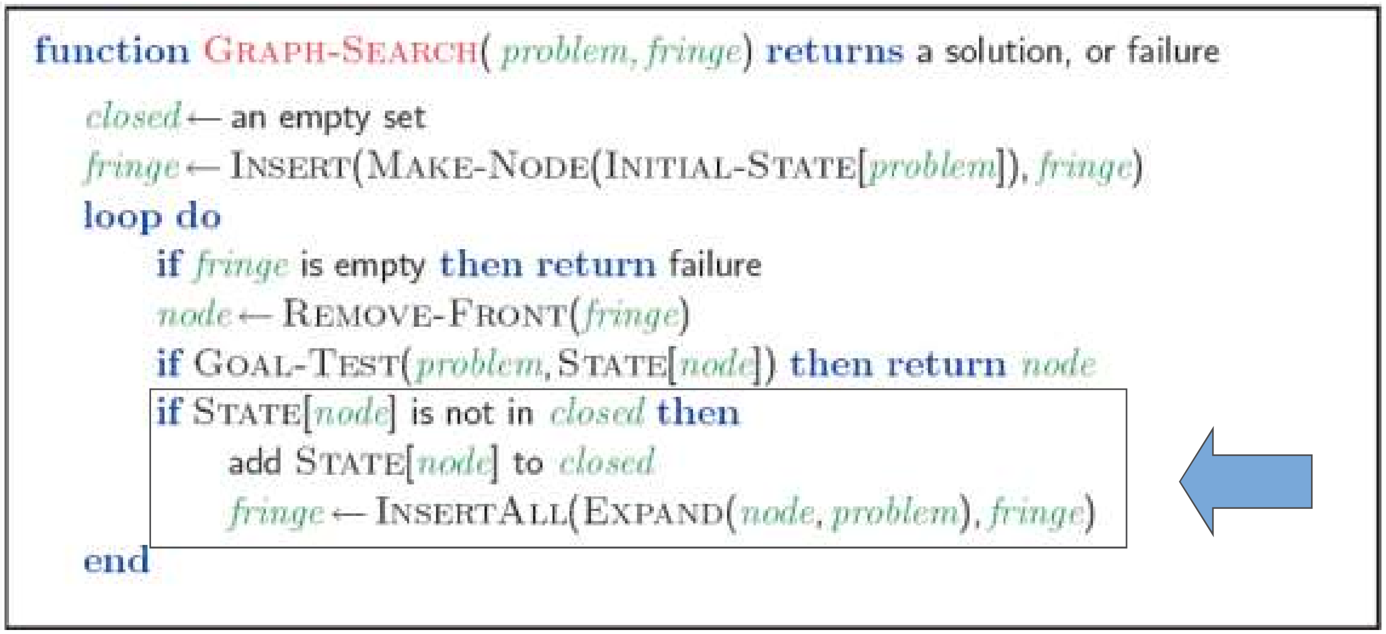
Graph Search

- Very simple fix: never expand a state type twice

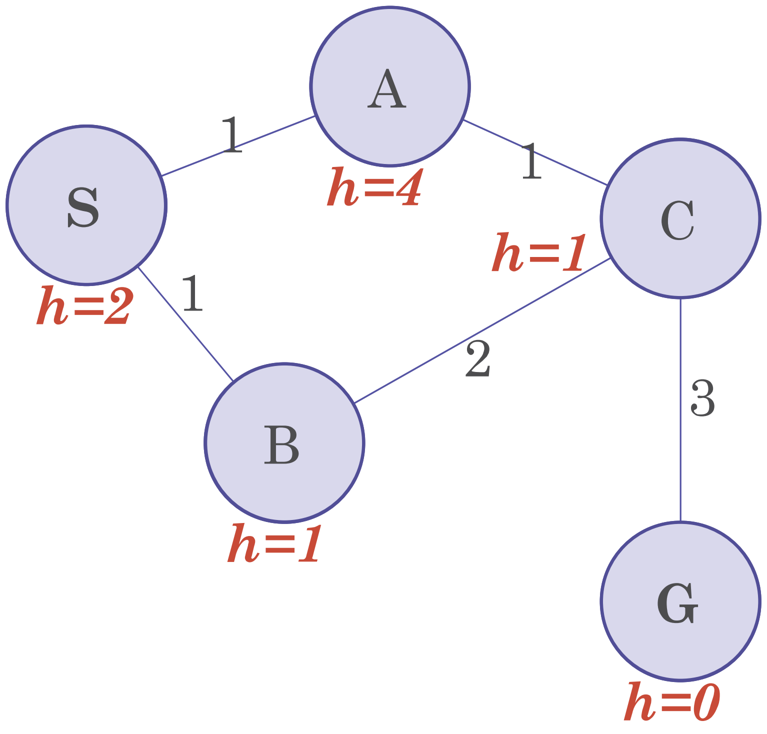


```
function GRAPH-SEARCH( problem, fringe) returns a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            fringe ← INSERTALL(EXPAND(node, problem), fringe)
    end
```

- Can this wreck completeness? Why or why not?
- How about optimality? Why or why not?

A* Graph Search Gone Wrong

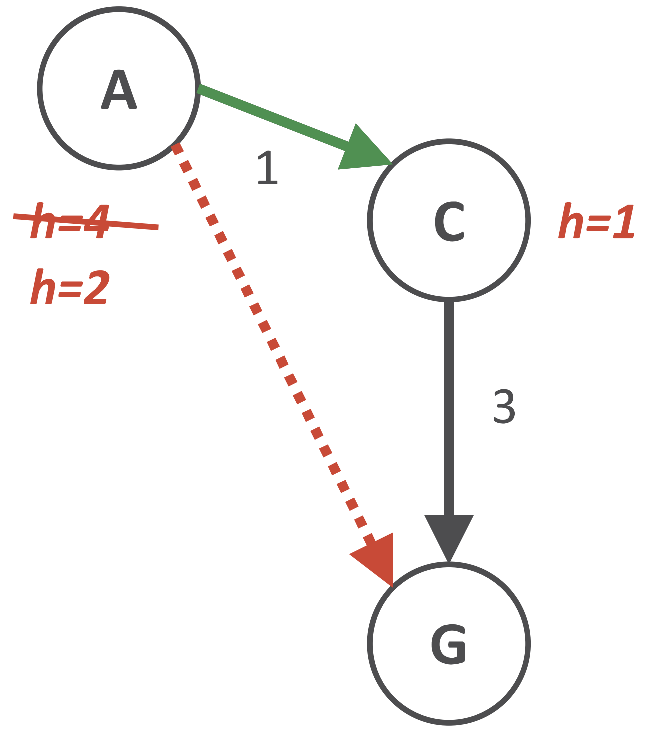
State space graph



Search tree

$$\begin{tikzcd}
 & S\ (0+2) \arrow[dl] \arrow[dr] & \\
A\ (1+4) \arrow[d] & & B\ (1+1) \arrow[d] \\
C\ (2+1) \arrow[d] & & C\ (3+1) \arrow[d] \\
G\ (5+0) & & G\ (6+0)
\end{tikzcd}$$

Consistency of Heuristics



- Main idea: estimated heuristic costs ≤ actual costs
  - Admissibility: heuristic cost ≤ actual cost to goal

    h(A) ≤ actual cost from A to G

  - Consistency: heuristic “arc” cost ≤ actual cost for each arc

    h(A) – h(C) ≤ cost(A to C)

- Consequences of consistency:
  - The f value along a path never decreases

    h(A) ≤ cost(A to C) + h(C)

    f(A) = g(A) + h(A) ≤ g(A) + cost(A to C) + h(C) ≤ f(C)

  - A* graph search is optimal