Heuristics and A* Search

CS 4300 — Fall 2025

Uniform-Cost Search (UCS): Reintroduction

- **UCS** expands the node with lowest path cost $g(n)$.
- Frontier is a **priority queue** ordered by $g(n)$.
- Goal test occurs when a node is *popped*, ensuring optimality.
- Appropriate when step costs are **non-uniform** and **positive**.

UCS (Graph Search)

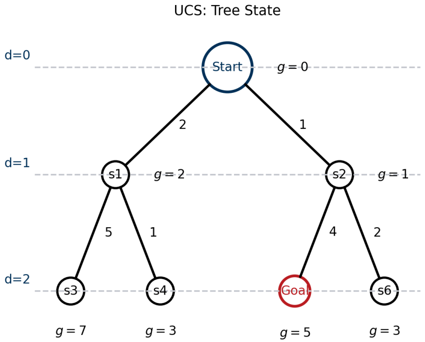
Algorithm 1 *

Uniform-Cost Graph Search (UCS)

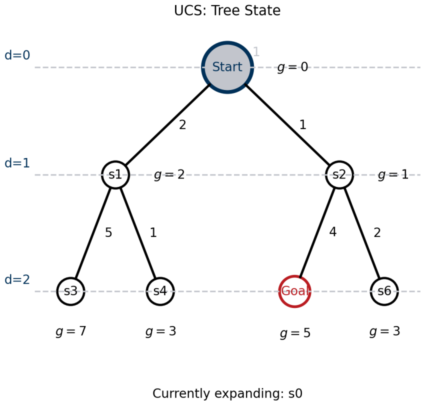
Require: initial state s_0; A(s); T(s, a); GOAL-TEST(s); step cost $c(s, a, s') > 0$

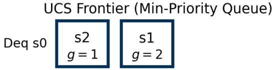
Initialize the frontier as an empty **min-priority queue** (keyed by g)
$g(s_0) \leftarrow 0$; PUSH(frontier, s_0, key $= g(s_0)$)
best_g $\leftarrow$ empty map from state $\rightarrow$ best known path cost; best_g[s_0] $\leftarrow 0$
while frontier is not empty **do**
    $n \leftarrow$ POP-MIN(frontier) ▷ state with lowest $g(n)$
    if GOAL-TEST($n.state$) **then**
        return solution path from s_0 to $n.state$
    end if
    for each $a \in$ A$(n.state)$ **do**
        $s' \leftarrow$ T$(n.state, a)$
        $g' \leftarrow g(n) + c(n.state, a)$
        if $s' \notin$ best_g **or** $g' <$ best_g[s'] **then**
            best_g[s'] $\leftarrow g'$
            PUSH(frontier, s', key $= g'$)
        end if
    end for
end while
return failure

UCS Walkthrough (0/6): Init

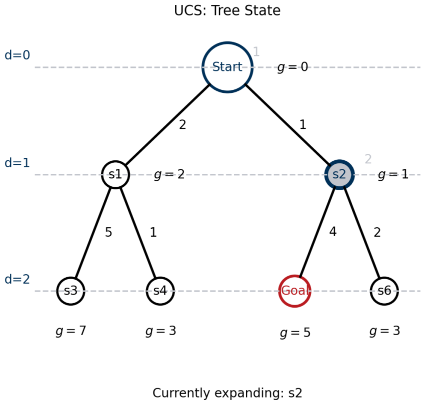


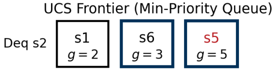
UCS Walkthrough (1/6): Dequeue Start



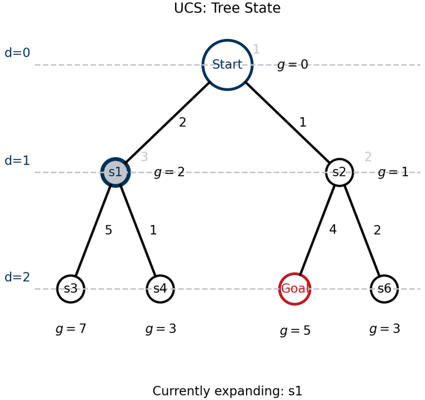


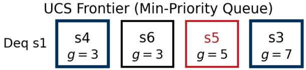
UCS Walkthrough (2/6): Dequeue s2



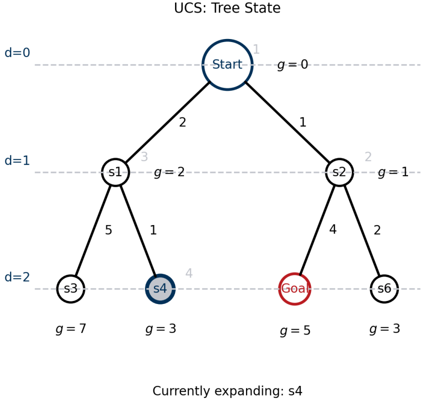


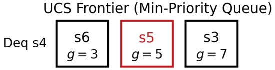
UCS Walkthrough (3/6): Dequeue `s1`



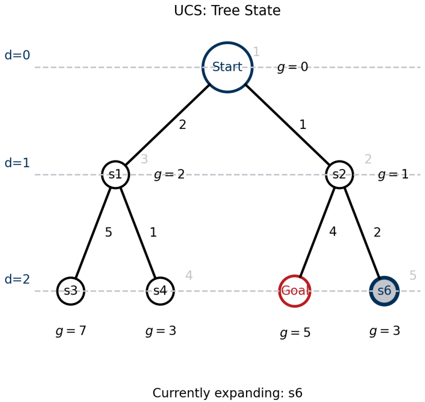


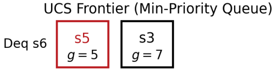
UCS Walkthrough (4/6): Dequeue s4



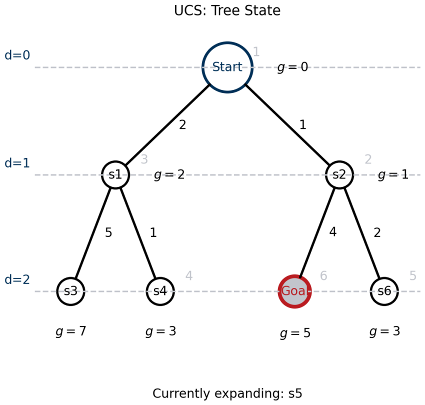


UCS Walkthrough (5/6): Dequeue s6





UCS Walkthrough (6/6): Dequeue s5 = Goal



UCS Frontier (Min-Priority Queue)

Deq s5 = GOAL

s3 $g = 7$

Motivation for Heuristics

- UCS only considers **cost so far**, $g(n)$.
- This can cause many unnecessary expansions if the goal is deep or far.
- Idea: add an estimate of the **remaining cost**.
- Define a **heuristic function** $h(n)$:
  - $h(n) \approx$ estimated cost from n to a goal.
  - Example: Manhattan distance in a grid world.

From UCS to A*

- Define the evaluation function:

$$f(n) = g(n) + h(n).$$

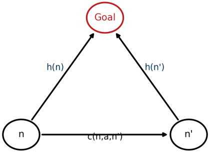
- $g(n)$ = cost so far, $h(n)$ = estimated cost-to-go.
- With a good heuristic, search focuses on promising areas.
- Guarantees:
  - **Admissibility:** $h(n)$ never overestimates true cost.
  - **Consistency:** heuristic obeys triangle inequality.

Consistent Heuristics

- A heuristic h is **consistent** (or monotone) if for every node n, successor n' by action a:

$$h(n) \leq c(n, a, n') + h(n')$$

- This is the *triangle inequality*: estimated cost at n is never more than the step cost plus estimate from n'.
- Consistency implies admissibility.
- With a consistent heuristic:
  - $f(n) = g(n) + h(n)$ is non-decreasing along a path.
  - Once a node is expanded, the best path to it has been found (no re-expansions needed).



A* (Tree Search)

Algorithm 2 A* (Tree Search)

Require: Problem with initial state s_0, actions $A(s)$, transition $\mathrm{T}(s, a)$, GoalTest(s, a), step cost $c(s, a, s') > 0$, heuristic $h(s) \geq 0$

frontier $\leftarrow$ min-priority queue by $f(n) = g(n) + h(n)$
push node(s_0) with $g(s_0) = 0$, $f(s_0) = g(s_0) + h(s_0)$
while frontier not empty **do**
    $n \leftarrow$ pop_min(frontier) ▷ lowest f
    if GoalTest(n.state) **then**
        return solution by following n's parents
    end if
    for all $a \in A(n.\text{state})$ **do**
        $s' \leftarrow \mathrm{T}(n.\text{state}, a)$
        $g' \leftarrow g(n) + c(n.\text{state}, a, s')$
        push node(s') with parent n and key $f' = g' + h(s')$
    end for
end while
return failure

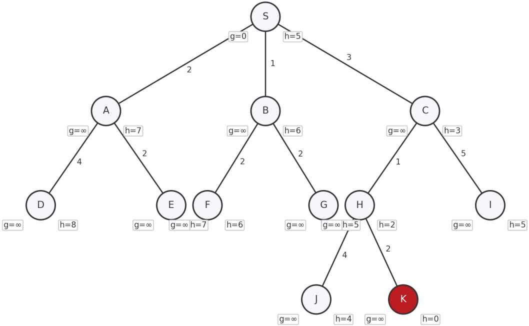
A* (Graph Search): with best-known g-costs

Algorithm 3 A* (Graph Search)

Require: s_0, $A(s)$, T(s, a), GoalTest(s, a), $c(s, a, s') > 0$, $h(s)$

```
frontier ← min-PQ by f = g + h; push node(s0) with g = 0, f = h(s0)
best_g ← map (state → best known g); best_g[s0] ← 0
while frontier not empty do
    n ← pop_min(frontier)
    if GoalTest(n.state) then return solution by following n's parents
    end if
    for all a ∈ A(n.state) do
        s' ← T(n.state, a)
        g' ← g(n) + c(n.state, a, s')
        if s' ∉ best_g or g' < best_g[s'] then
            best_g[s'] ← g'
            push/DecreaseKey node(s') with parent n and f' = g' + h(s')
        end if
    end for
end while
return failure
```

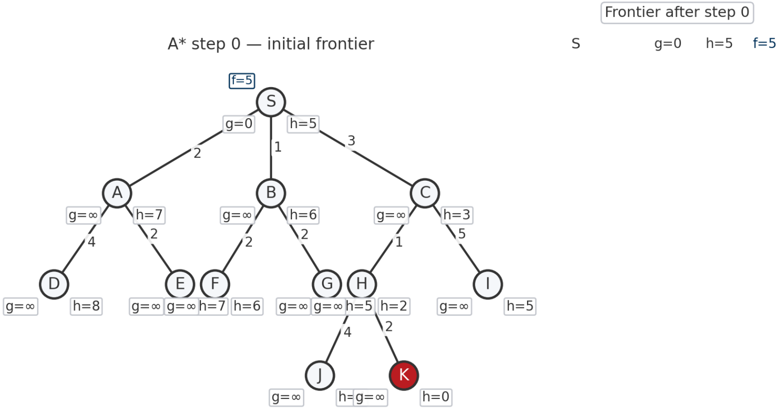
A* Search (Tree) — Intuition



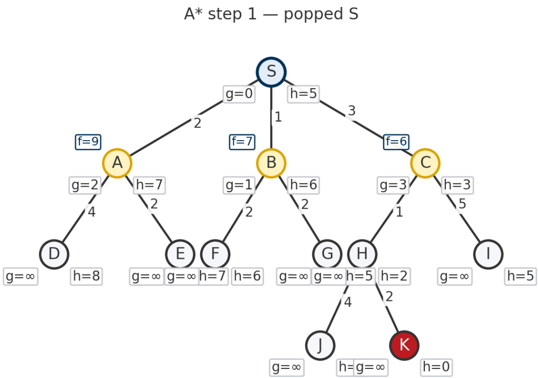
Key ideas

- Score nodes with $f(n) = g(n) + h(n)$.
- Pop the lowest f from a min-PQ.
- g: path cost so far. h: est. cost to goal.
- Goal test on *pop* $\Rightarrow$ optimal if h is admissible & consistent.

A* Search — Step 0



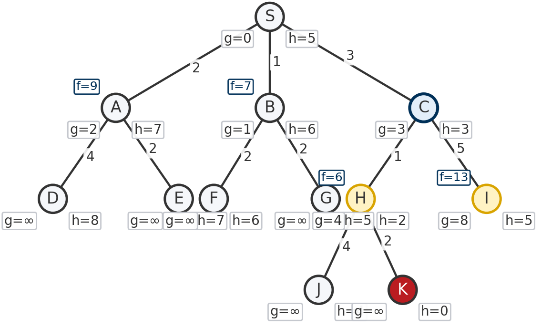
A* Search — Step 1



Frontier after expansion			
C	g=3	h=3	f=6
B	g=1	h=6	f=7
A	g=2	h=7	f=9

A* Search — Step 2

A* step 2 — popped C

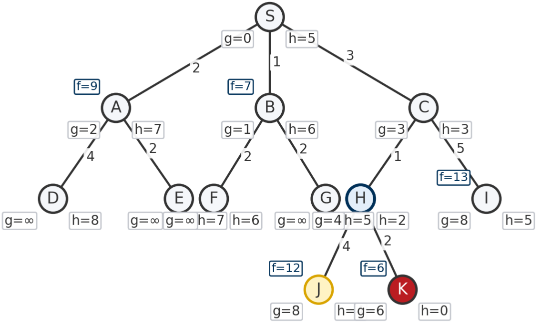


Frontier after expansion

H	g=4	h=2	f=6
B	g=1	h=6	f=7
A	g=2	h=7	f=9
I	g=8	h=5	f=13

A* Search — Step 3

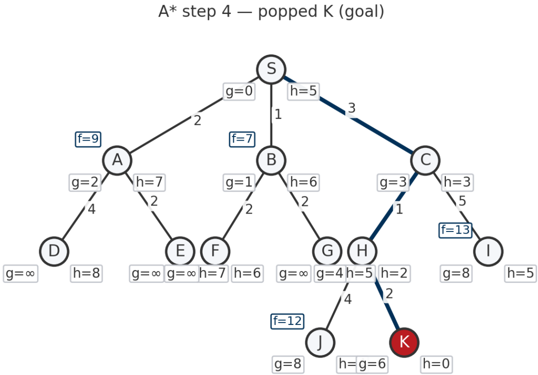
A* step 3 — popped H



Frontier after expansion

K	g=6	h=0	f=6
B	g=1	h=6	f=7
A	g=2	h=7	f=9
J	g=8	h=4	f=12
I	g=8	h=5	f=13

A* Search — Step 4 (Goal Found)



Frontier after expansion

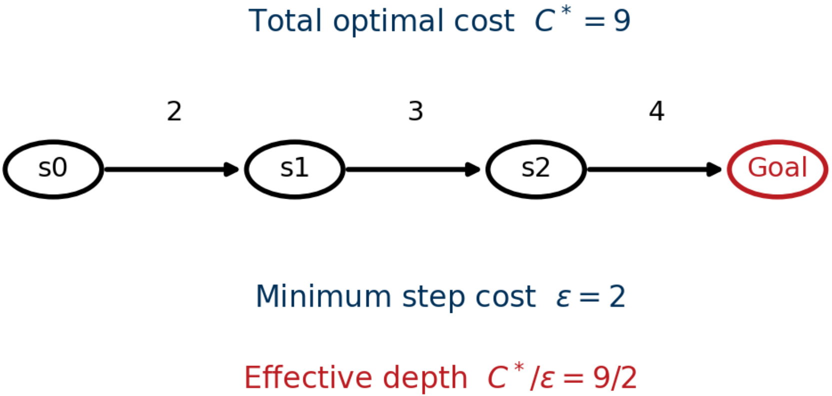
B	g=1	h=6	f=7
A	g=2	h=7	f=9
J	g=8	h=4	f=12
I	g=8	h=5	f=13

Properties of A*

- **Completeness:** Yes, if step costs $\geq \epsilon > 0$ and branching factor $b < \infty$.
- **Optimality:**
  - With **admissible** h, A* (tree search) is optimal.
  - With **consistent** h, A* (graph search) is optimal and never needs to reopen expanded nodes.
- **Time/Space Complexity:**
  - Worst case: $O(b^{C^*/\epsilon})$, exponential in the solution depth bound.
  - C^* = cost of the optimal solution.
  - ϵ = minimum positive step cost.
  - Effective branching factor b^* is reduced with a good heuristic:

$$O\left((b^*)^{C^*/\epsilon}\right)$$

Effective Depth of A*



Designing Heuristics

- **Problem relaxations:** remove constraints to get $h(n)$ from an easier subproblem.
- **Abstractions / pattern databases:** precompute exact distances in abstracted state spaces.
- **Additive heuristics:** when subproblems are independent ($h = h_1 + h_2$ is still admissible).
- **8-puzzle:** h_1=#misplaced tiles; h_2=sum of Manhattan distances; h_2 dominates h_1.
- **Practical tip:** Start with a cheap admissible h, measure node expansions, iterate.

Summary of A*

- **UCS:** expands node with lowest cost so far $g(n)$.
- **A*:** expands node with lowest estimated total cost $f(n) = g(n) + h(n)$.
- With admissible, consistent heuristics:
  - A* is **optimal**.
  - A* is often far more **efficient** than UCS.