上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Lecture 4: Heuristic Search & A*

Tao Huang

John Hopcroft Center, School of Computer Science, Shanghai Jiao Tong University

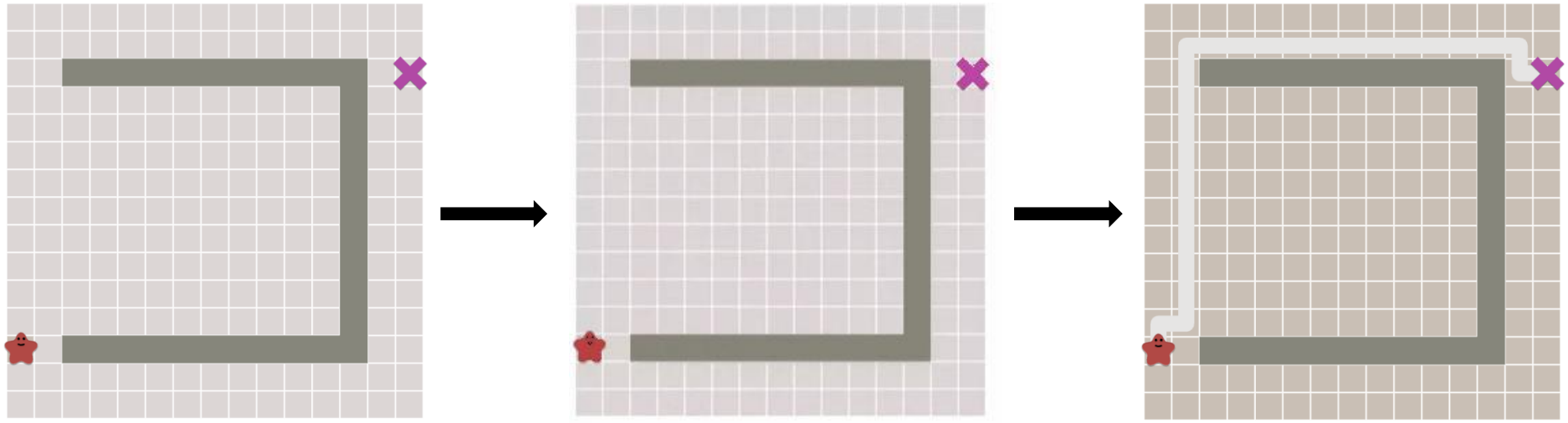
https://taohuang.info/cs3317

https://oc.sjtu.edu.cn/courses/89538

Part of slide credits: Stanford CS221

Review: BFS

- **Breadth-first search** (BFS): Finds a path between two nodes by taking one step down all paths.
- BFS always returns the shortest path between the start and the end nodes.

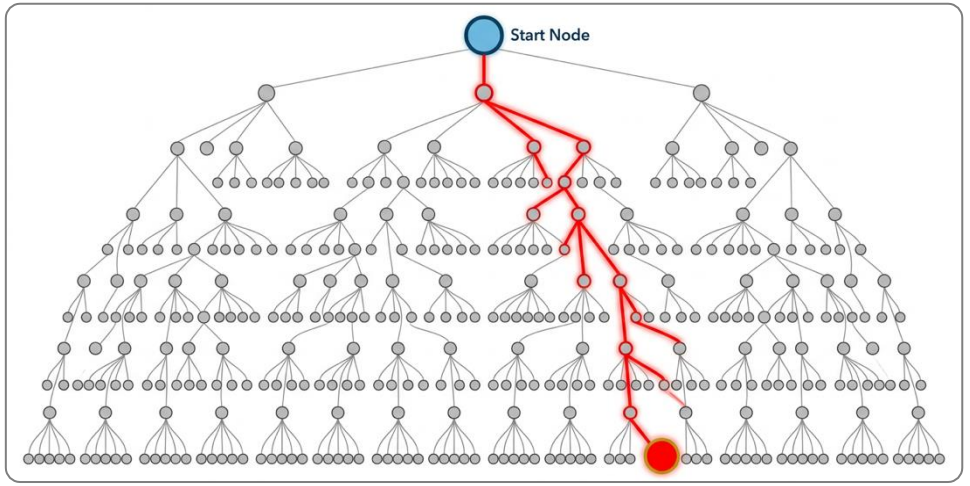


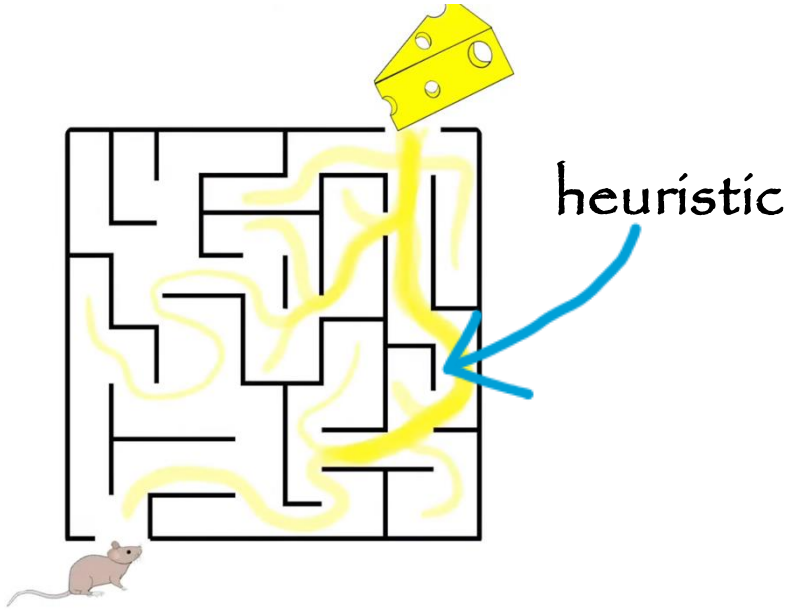
https://www.redblobgames.com/pathfinding/a-star/introduction.html

- BFS is costly

Why Do We Need Heuristics?

- Uninformed search expands nodes indiscriminately
- Search space grows exponentially (b^d)
- We need **guidance** toward the goal





Heuristic Search

- **Heuristic:** An estimate of distance/cost to the goal
- Example: Straight-line distance (SLD) in route planning
- Guided Search: Prefer states that **look closer** to the goal
- Goal: Reduce the effective branching factor

The Heuristic Function (h(n))

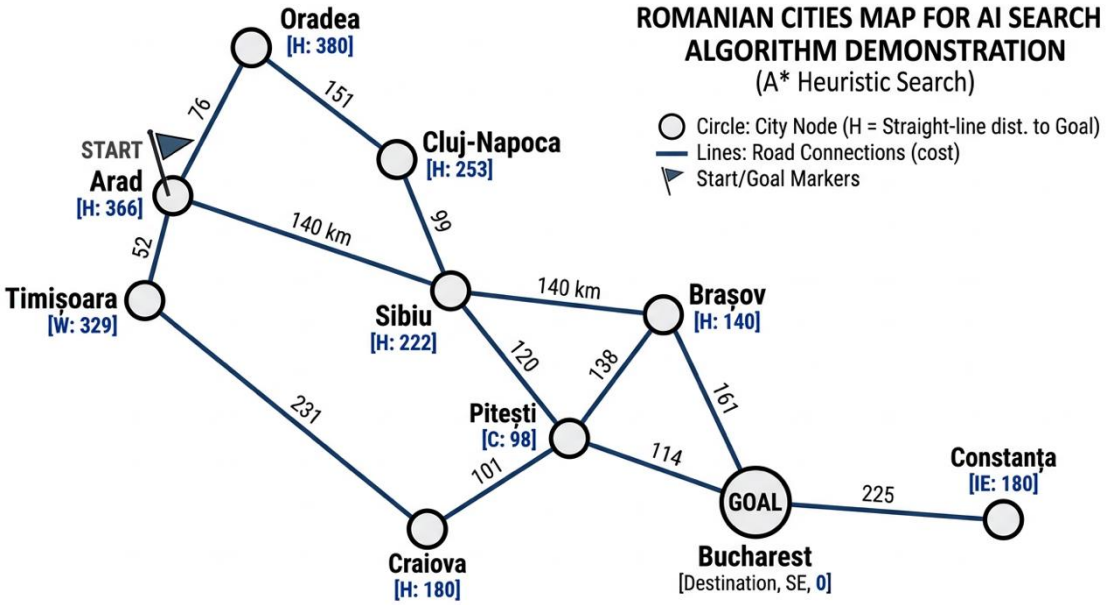
- **Notation:** h(n)
- **Meaning:** Estimated cost from node n to a goal
- **Informative:** Accurate estimate of true cost
- **Efficient:** Computationally cheap
- Note: h(n) = 0 if n is a goal

Example: Route Planning

- **Goal:** Reach Bucharest
- **Heuristic:** Straight-line distance (SLD) to Bucharest

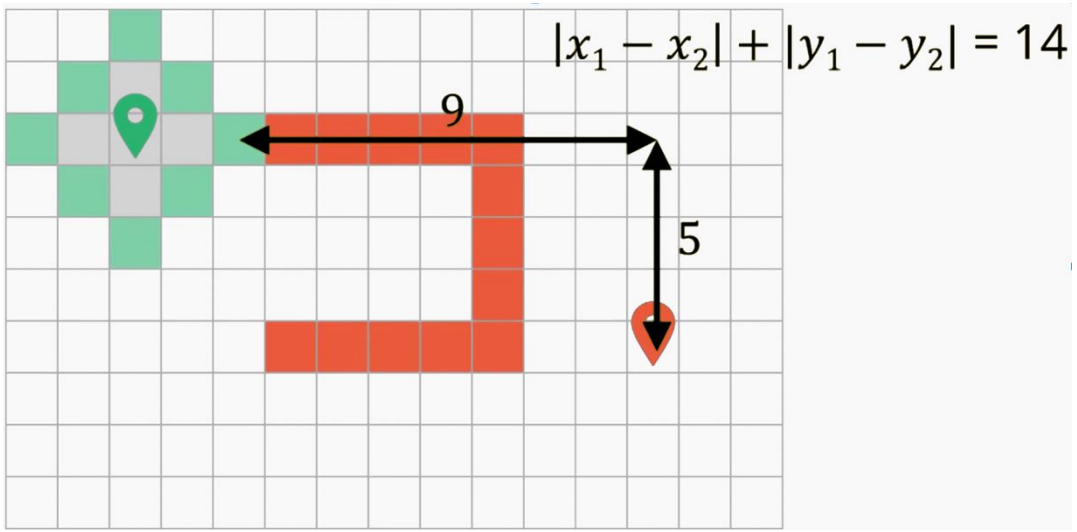
  e.g., h("Sibiu") = 222

- Guided exploration significantly reduces expanded nodes.



Greedy Best-First Search

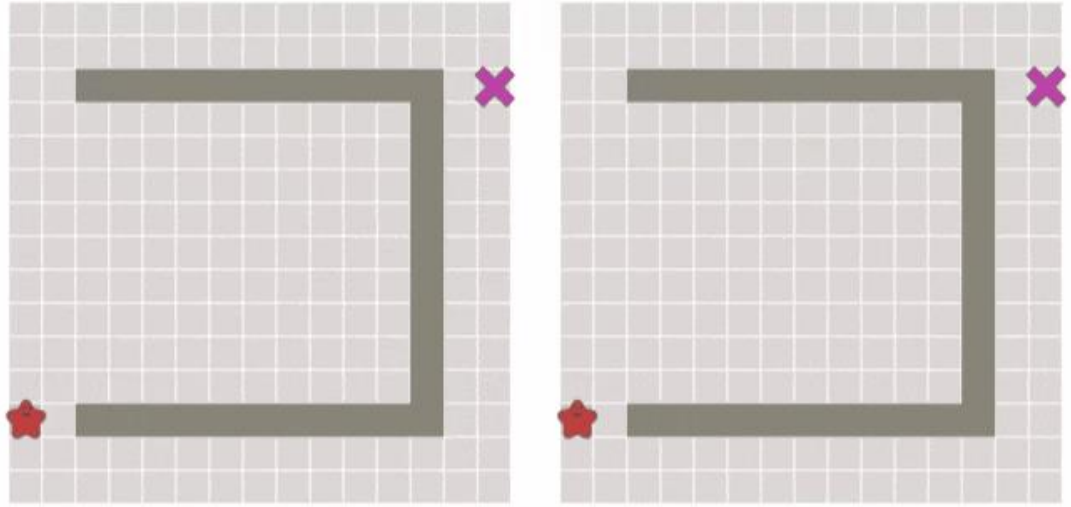
- **Best-first search:** Explores a graph by expanding the most promising node chosen according to a specified rule f(n).
- Greedy best-first search: nodes which are closest to the goal are extended first. f(n) = h(n)
  - h(n): Straight-line distance, Manhattan distance, etc.



Manhattan distance

Greedy Search: Analysis

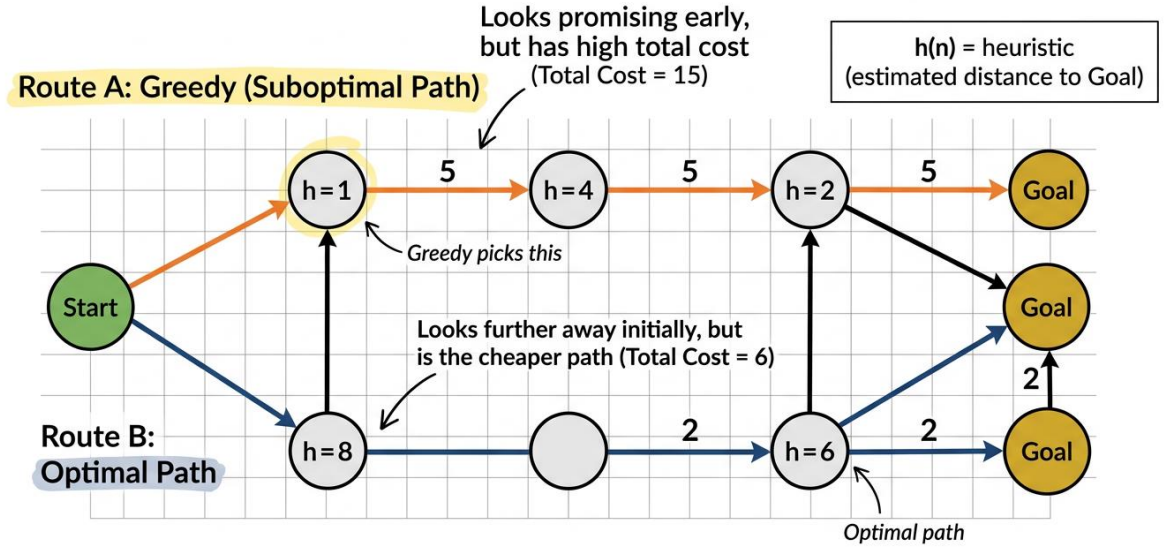
- Pros: Often very fast in finding a solution
- Cons: **Not optimal** — can choose poor paths
- Cons: **Incomplete** — can get stuck in infinite loops
- Problem: It ignores the path cost already incurred (\(g\))



Breadth-first search Greedy Best-first search

Why Greedy Can Fail

- Greedy focuses only on the "future" estimate (h)
- A "short-looking" path might be extremely expensive
- To fix this, we need to balance:
  1. Cost already traveled: g(n)
  2. Estimated cost remaining: h(n)



A* Search

- **Motivation:** we want to still be able to generate the path with minimum cost

- A* is an algorithm that:
  - Uses heuristic to guide search
  - While ensuring that it will compute a path with minimum cost

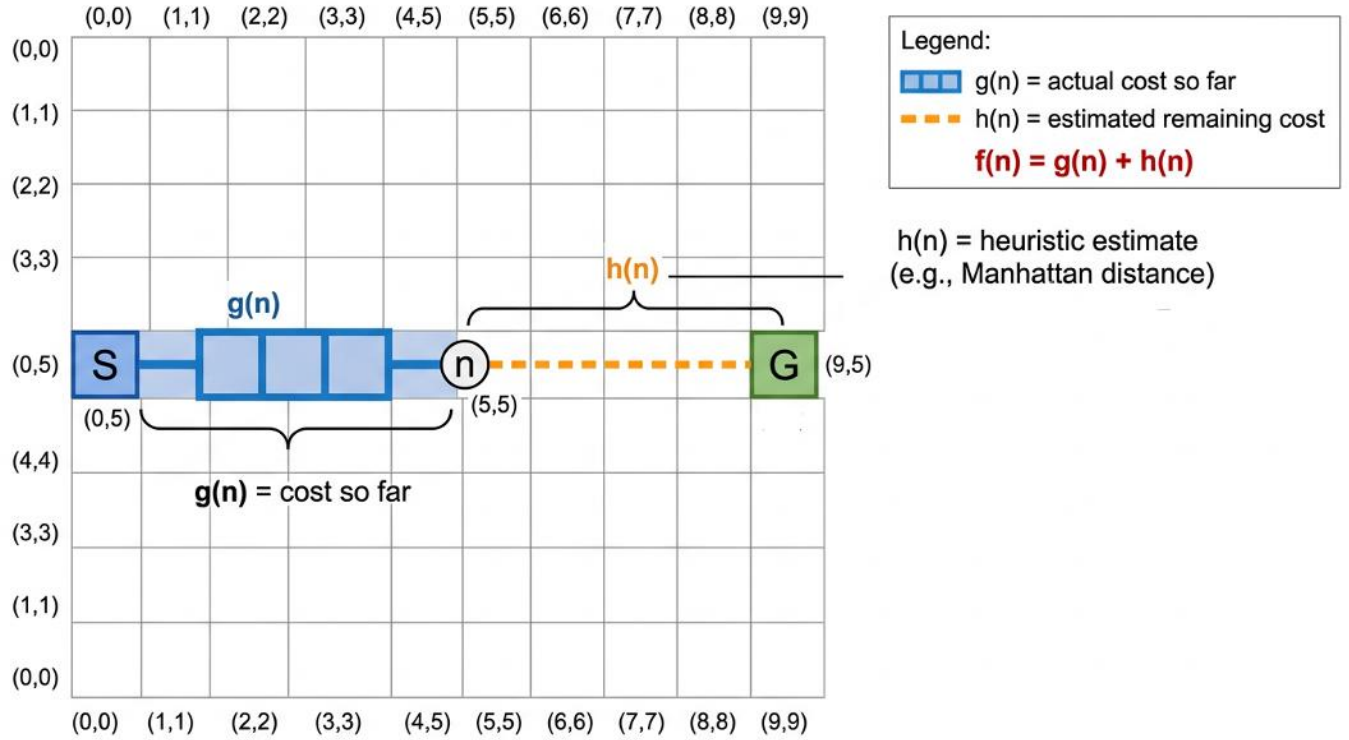
- A* computes the function $f(n) = g(n) + h(n)$

"estimated cost"

"actual cost"

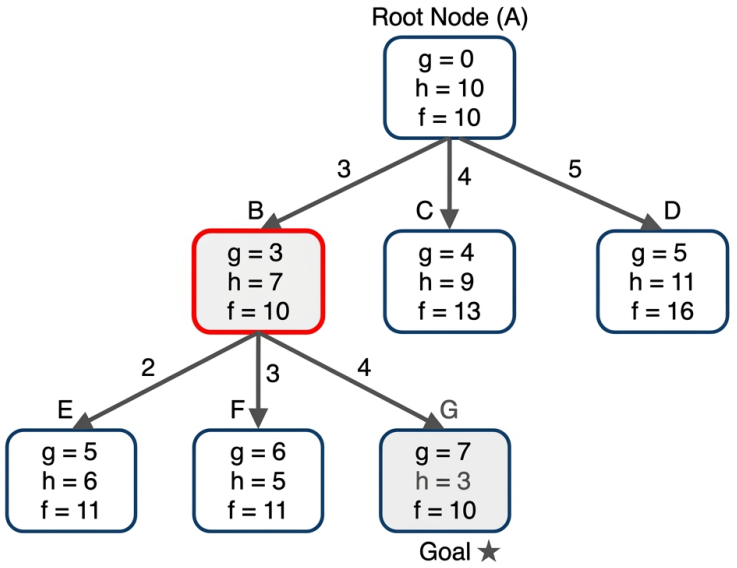
Balancing Past and Future

- **UCS:** Minimizes g(n) (past cost)
- **Greedy:** Minimizes h(n) (future cost)
- **A*:** Minimizes the sum g(n) + h(n)



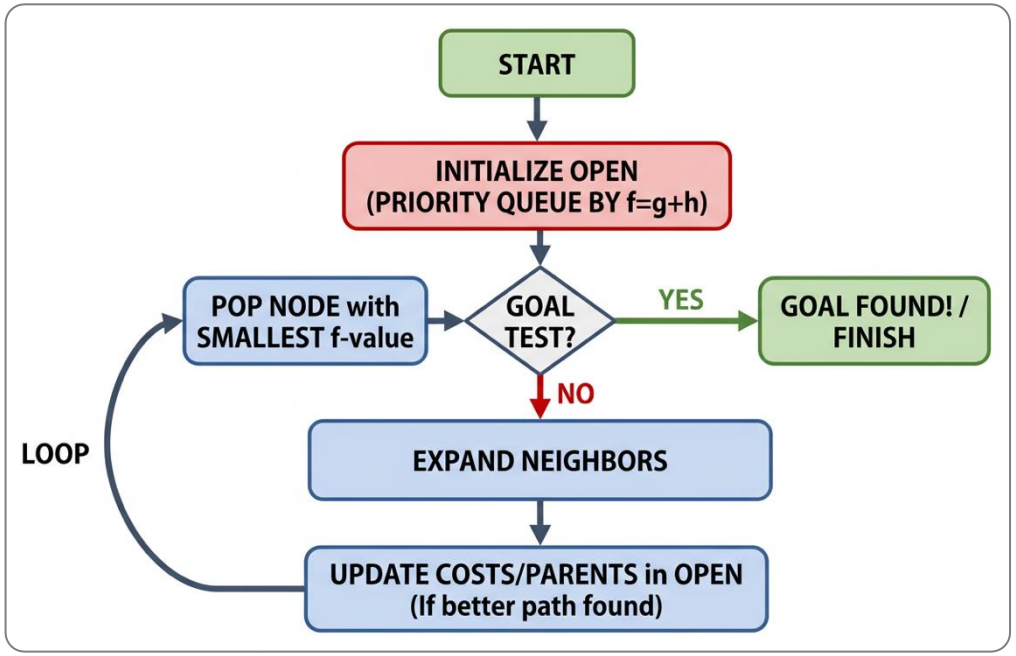
A* Exploration Strategy

- Frontier is prioritized by f(n)
- Expanded nodes show: g, h, f
  - g: accumulated actual cost
  - h: heuristic estimate to goal
  - f: total priority value
- Minimum f node is expanded first



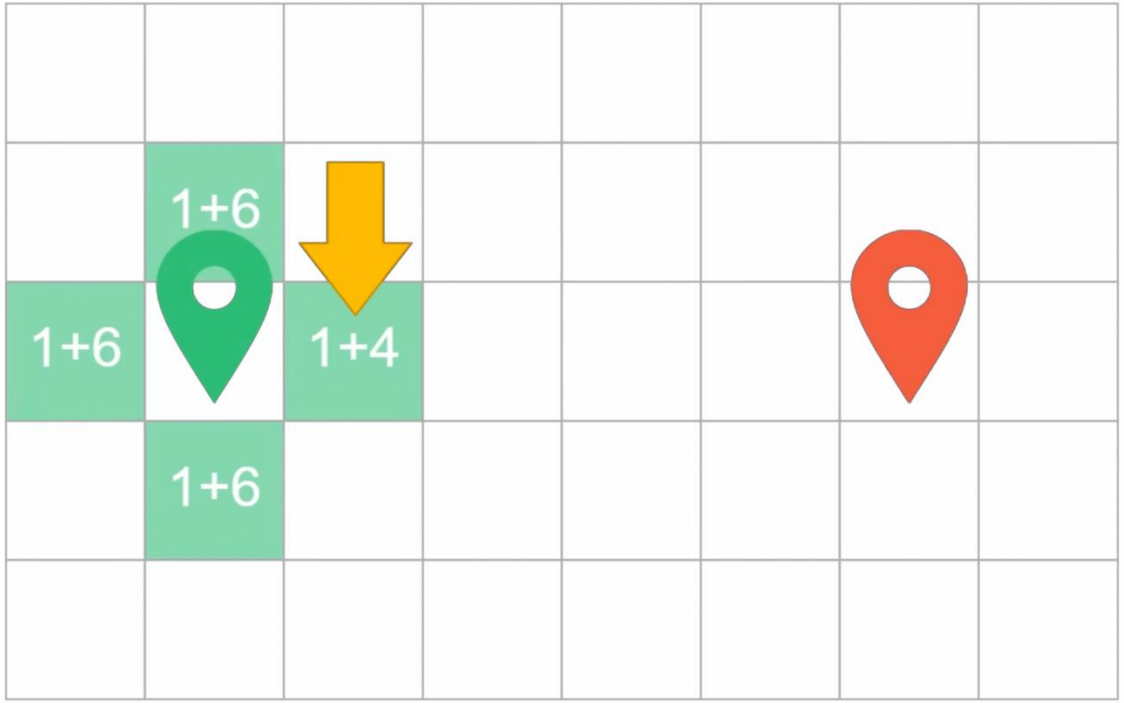
The A* Algorithm

- Initialize Frontier with start node
- Repeat while Frontier is not empty:
    1. Pick node n with smallest f(n)
    2. If n is goal, return solution
    3. Expand n, add neighbors to Frontier
- Note: Update neighbors if a better path is found.



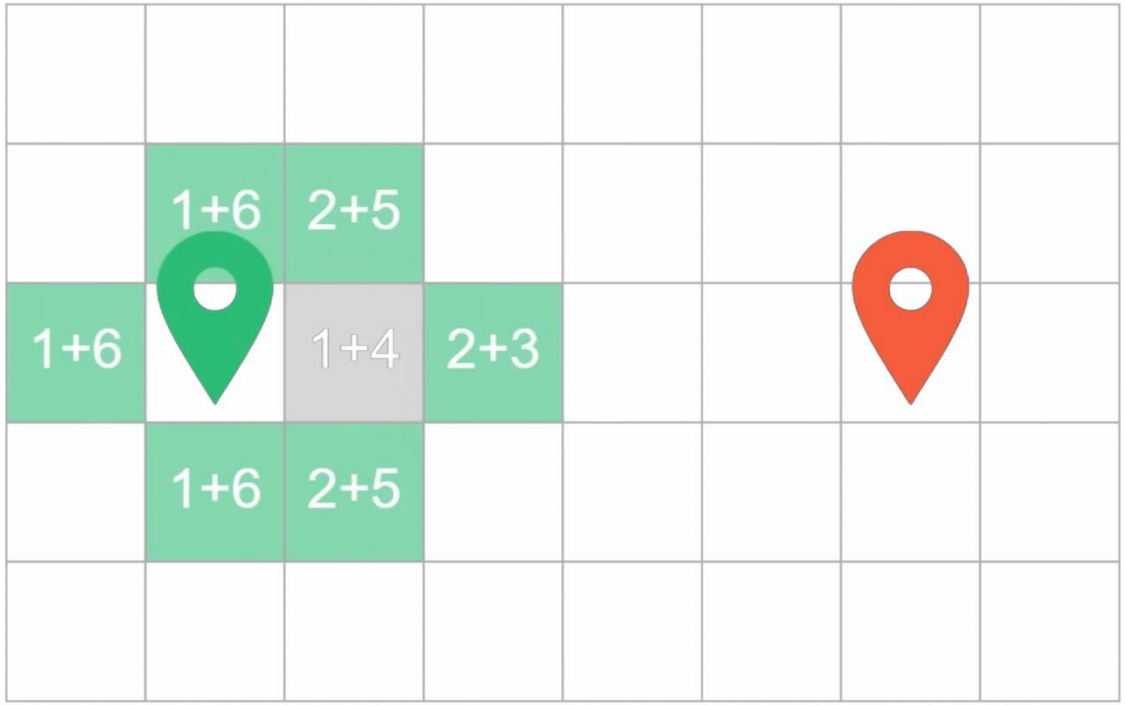
Example

- f(n) = g(n) + h(n)
  - g(n) = “cost from the starting node to reach n”
  - h(n) = “estimate of the cost of the cheapest path from n to the goal node”



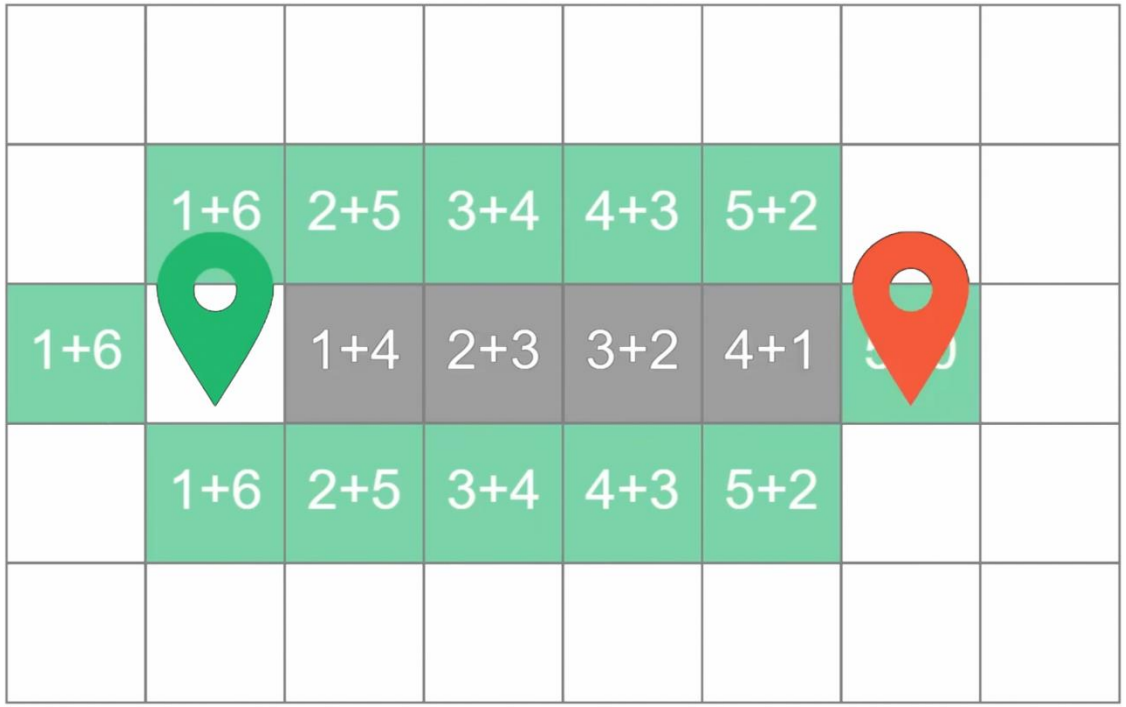
Example

- f(n) = g(n) + h(n)
  - g(n) = “cost from the starting node to reach n”
  - h(n) = “estimate of the cost of the cheapest path from n to the goal node”



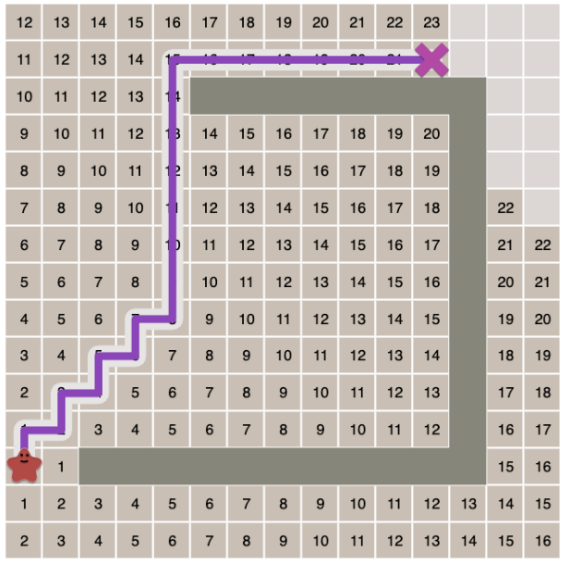
Example

- f(n) = g(n) + h(n)
  - g(n) = “cost from the starting node to reach n”
  - h(n) = “estimate of the cost of the cheapest path from n to the goal node”

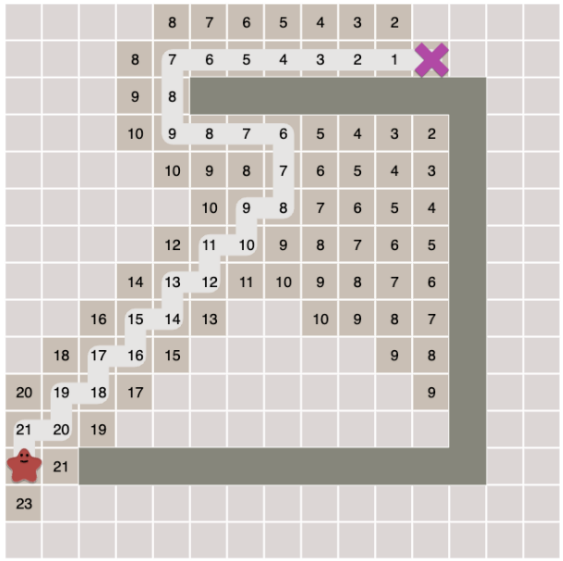


Search Algorithm Comparison

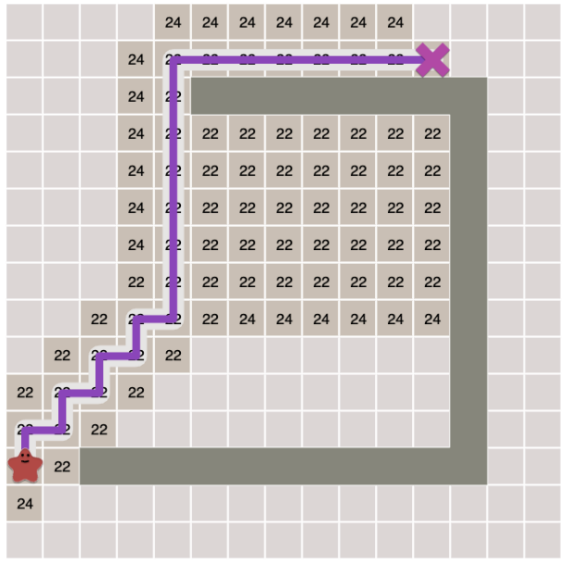
Algorithm	Uses Heuristic?	Optimal?
BFS	No	Yes (if step cost = 1)
Greedy	Yes	No
A*	Yes	Yes (if admissible)



Breadth-first search



Greedy best-first search



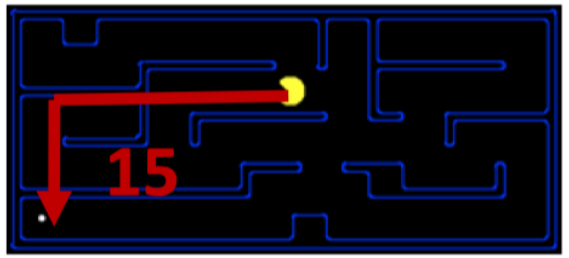
A* search

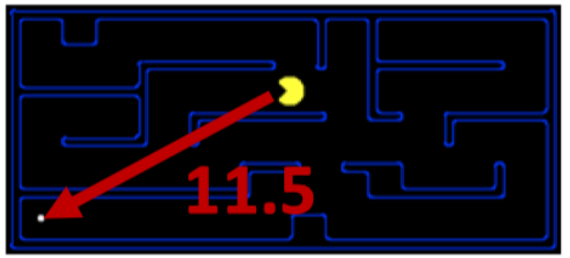
Admissible Heuristics

- Definition: A heuristic h is **admissible（可采纳的）** if it never overestimates the true cost to the goal.

  $h(n) \leq h^*(n)$ for all n, where $h^*(n)$ is true cost.

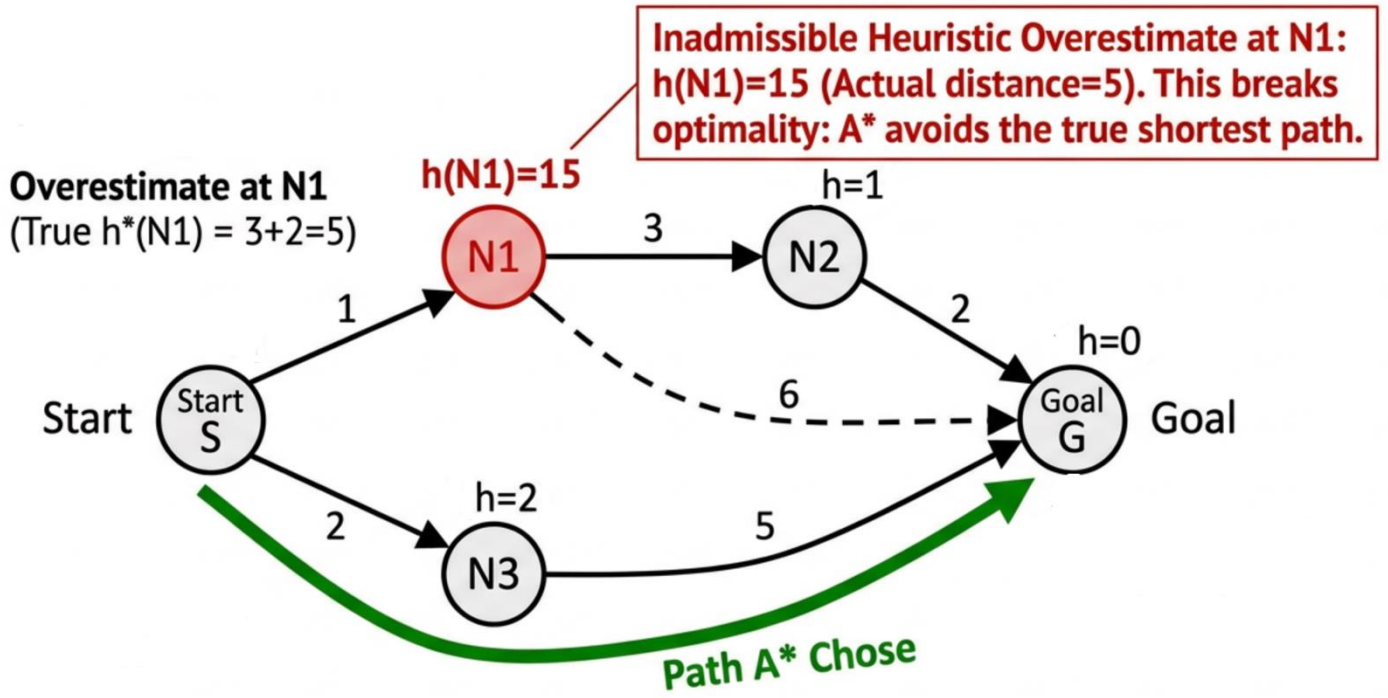
- Example: Straight-line distance in route planning
- Example: Manhattan distance in grid navigation





Is A* Search Optimal?

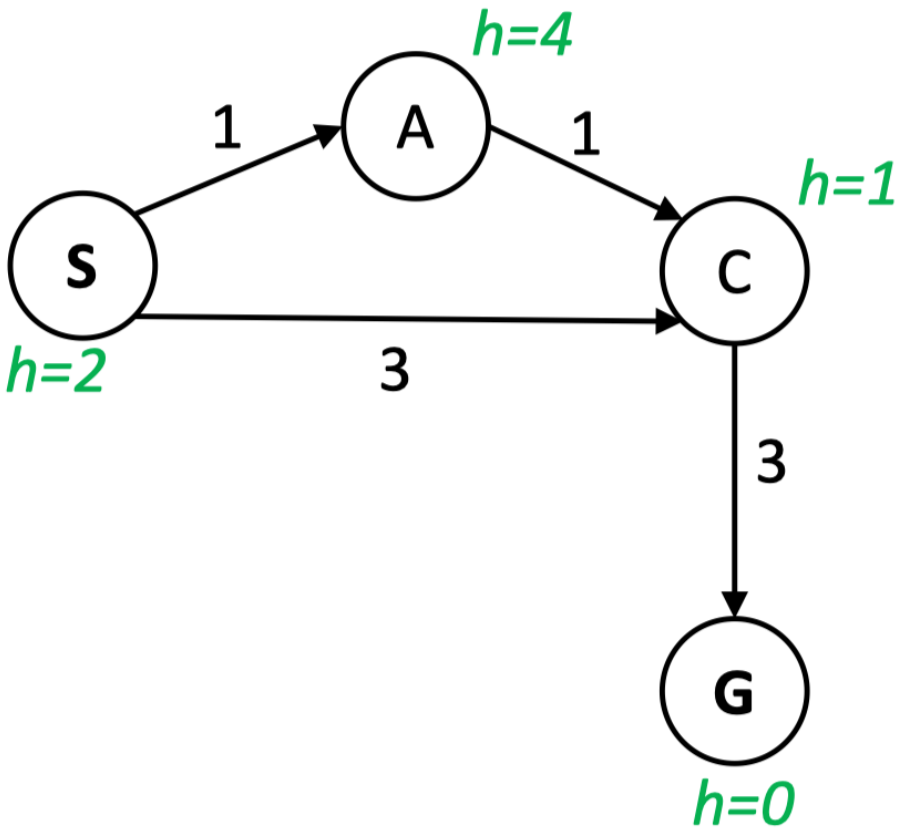
- A* optimality relies on **admissibility** $h(n) \leq h^*(n)$
- If h(n) **overestimates** the cost, A* might find a suboptimal path.



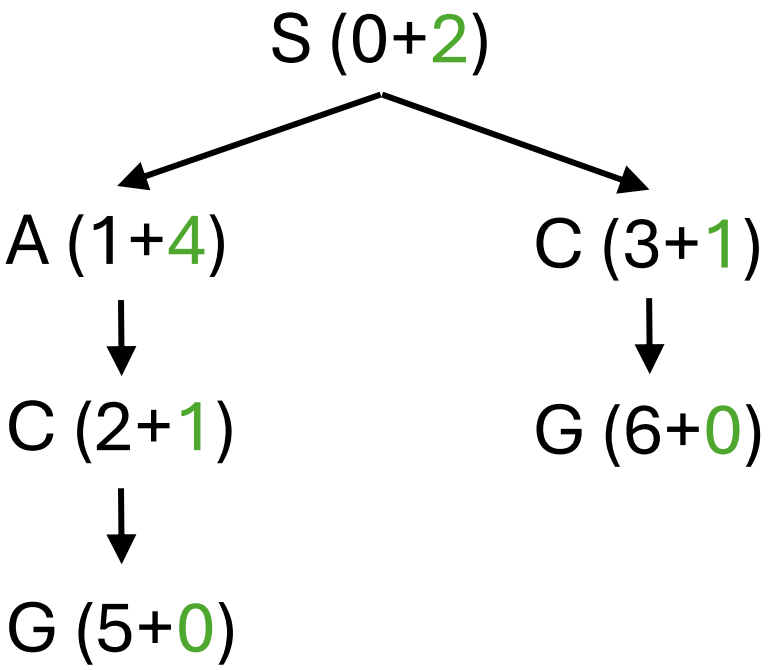
Step 1: Start at S. **Explore N1** f=1+15=16 f=2+2=4)
Step 2: A* explores N3 (lowest f). Expands to G
Step 3: A* finds G via path B, assuming it is **optimal** (f=7)

Quiz: A* Tree Search

State space graph



Search tree

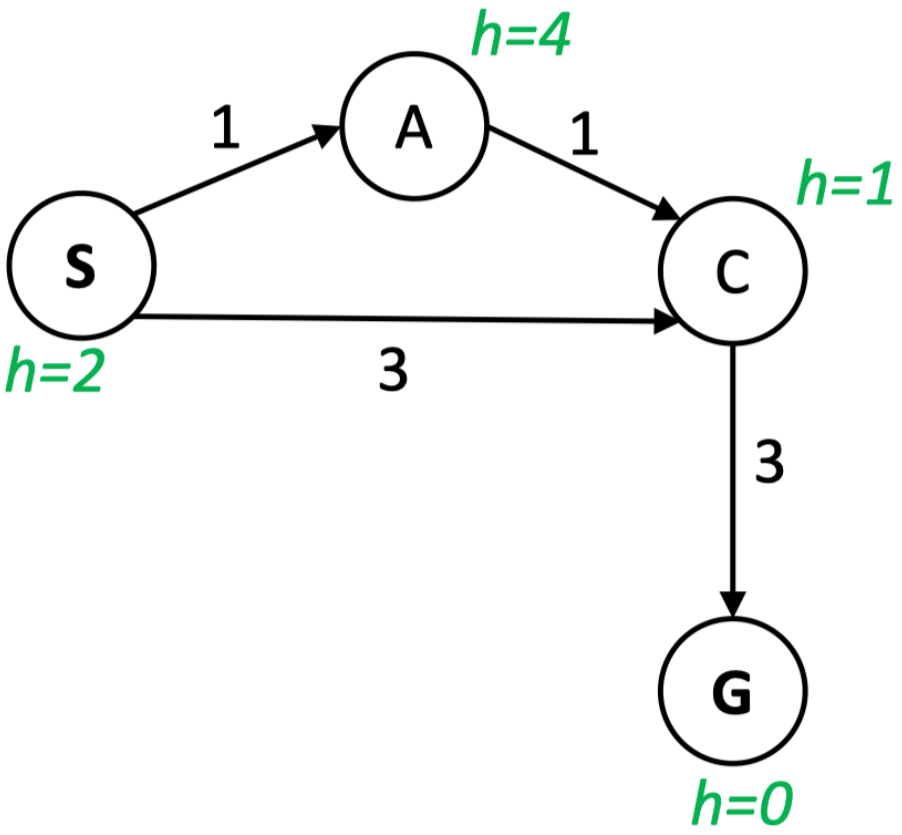


Result: S → A → C → G

Is it optimal?

Quiz: A* Graph Search

- What paths does A* graph search consider during its search?

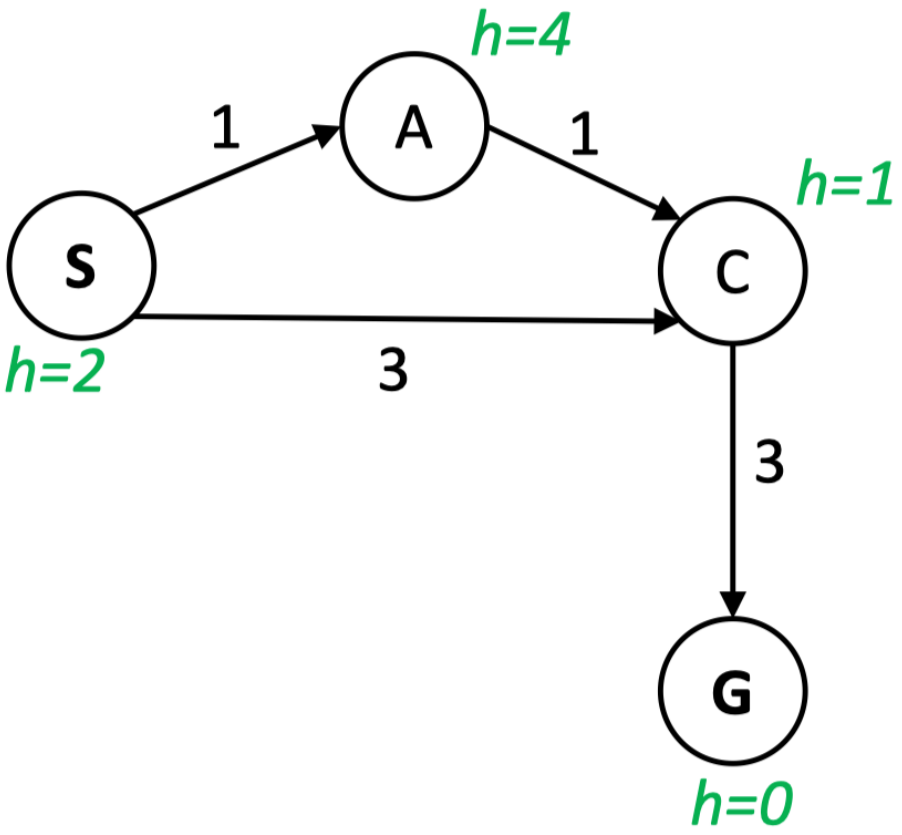


Frontier	Explored set
S	S
S → A (1+4) S → C (3+1)	S A C
S → C → G	S A C G
~~S → A → C~~	S A C G

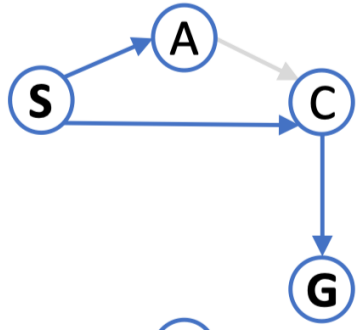
Result: S → C → G

Quiz: A* Graph Search 2

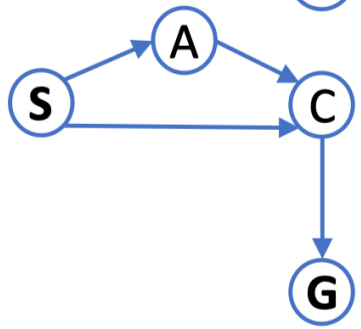
- What does the resulting graph look like?



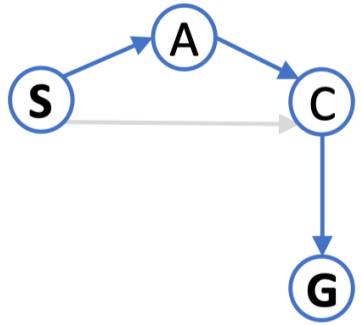
(A)



(B)



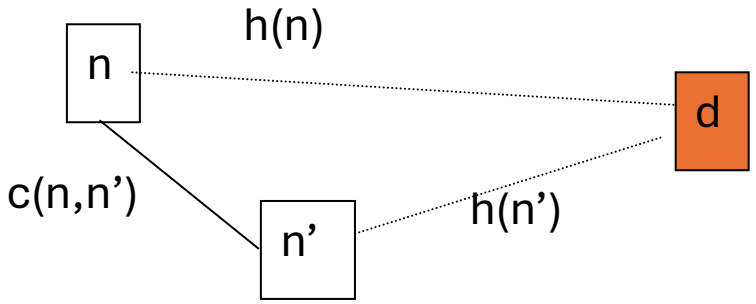
(C)

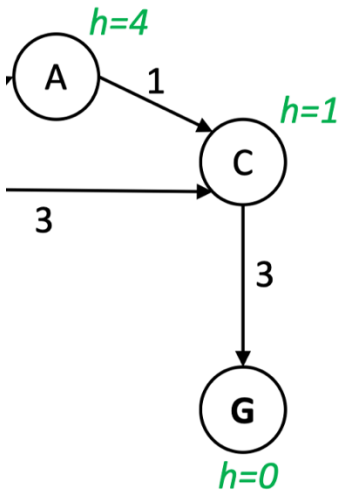


Consistency of Heuristics

- A* generates an optimal solution if h(n) is an **admissible** heuristic and the search space is a **tree**:
  - h(n) is admissible if it never overestimates the cost: $h(n) \leq h^*(n)$
- A* generates an optimal solution if h(n) is a **consistent** heuristic and the search space is a **graph**:
  - h(n) is consistent if for every node n and for every successor node n' of n:

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

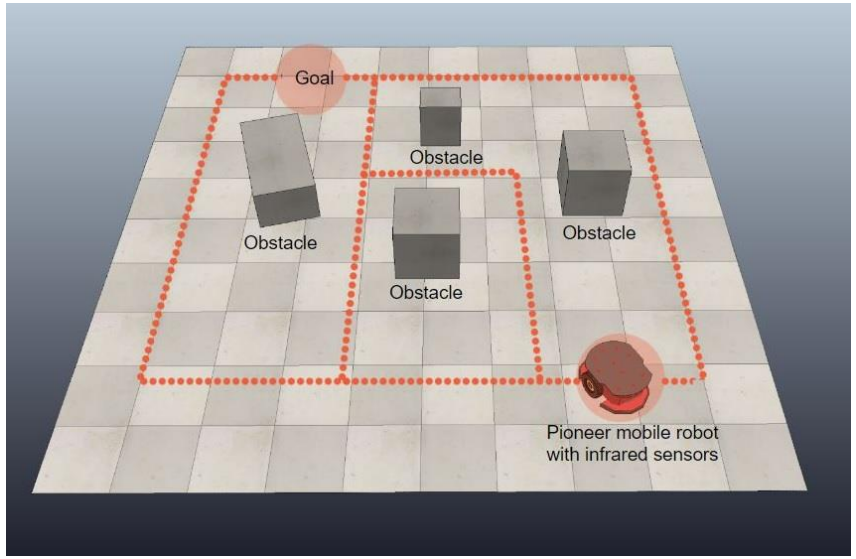




h(A) = 4
c(A, C) + h(C) = 1 + 1 = 2

Applications of A* Search

- **Route Planning:** Maps, airline routing.
- **Robotics:** Motion planning for mobile robots, avoiding physical obstacles.
- **Games:** NPC navigation and pathfinding in real-time strategy games.



Key Takeaways

- Heuristics guide search by estimating distance to the goal
- Greedy Best-First search is fast but not optimal
- A* search combines path cost and heuristic for optimality
- Admissibility is the key property for A* optimality
- Good heuristics dramatically reduce search time

Next Week

Beyond Basic Search

Local Search · Adversarial Search · Constraint Satisfaction

Coursework 1: Route Planning

- Task : Implement DFS, UCS, and A* in Python
- Submission:
  - Implemented code
  - running results
- Due: Next Monday (3.16) 23:59
- Links:
  - https://taohuang.info/cs3317/coursework1/index.html
  - https://oc.sjtu.edu.cn/courses/89538/assignments/396536