
CIS 471/571 (Fall 2020): Introduction to Artificial Intelligence

Assignment Project Exam Help

<https://powcoder.com>
Lecture 3: Informed Search
Add WeChat powcoder

Thanh H. Nguyen

Most slides are by Pieter Abbeel, Dan Klein, Luke Zettlemoyer, John DeNero,
Stuart Russell, Andrew Moore, or Daniel Lowd
Source: <http://ai.berkeley.edu/home.html>

Reminder

- Homework 1: Search
 - Deadline: Oct 10, 2020
[Assignment](#) [Project](#) [Exam](#) [Help](#)
- Project 1: Search <https://powcoder.com>
 - Deadline: Oct 13, 2020 [Add WeChat powcoder](#)

Today

- Uninformed Search

- Uniform Cost Search

Assignment Project Exam Help

- Informed Search

- Heuristics

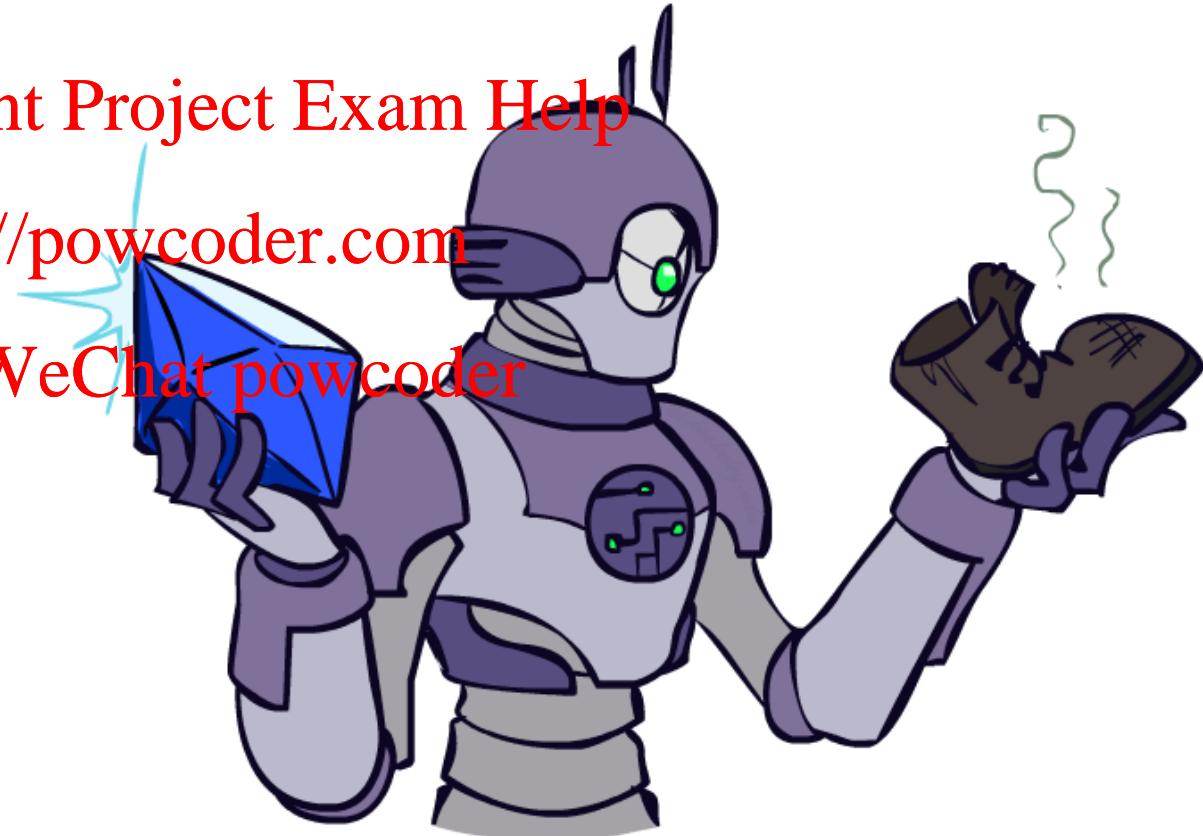
- Greedy Search

- A* Search

- Graph Search

<https://powcoder.com>

Add WeChat powcoder



Recap: Search

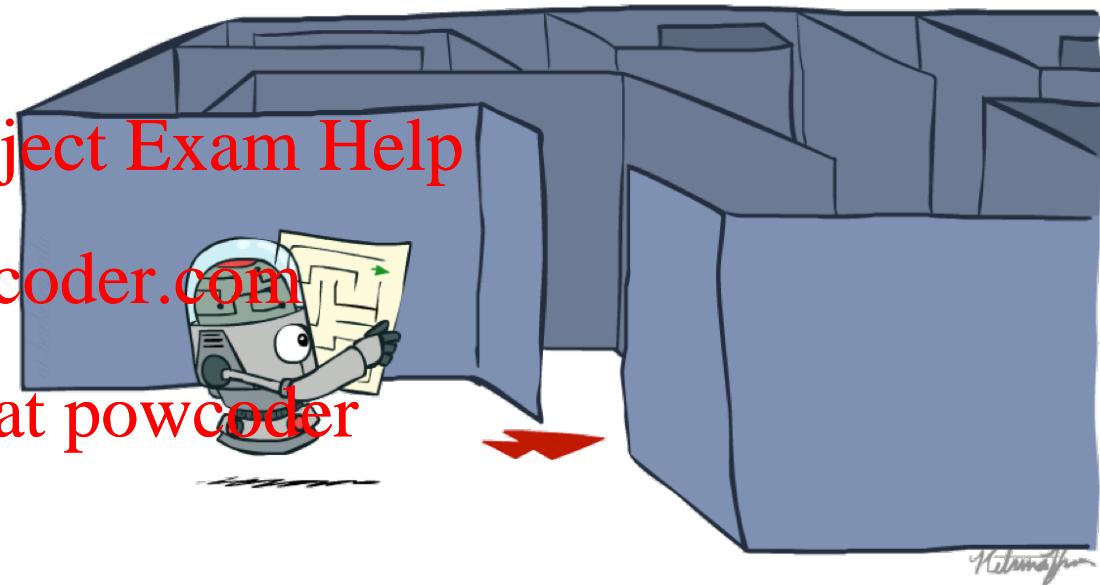
- Search problem:

- States (configurations of the world)
- Actions and costs
- Successor function (world dynamics)
- Start state and goal test

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



- Search tree:

- Nodes: represent plans for reaching states
- Plans have costs (sum of action costs)

- Search algorithm:

- Systematically builds a search tree
- Chooses an ordering of the fringe (unexplored nodes)
- Optimal: finds least-cost plans



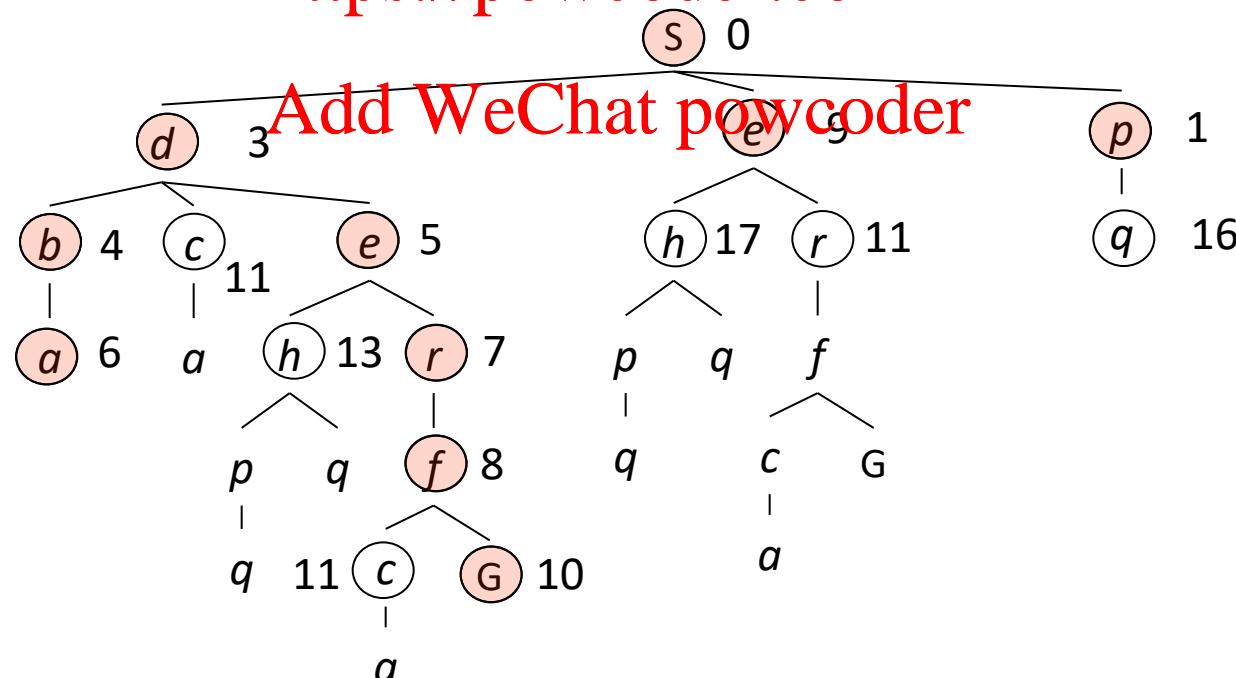
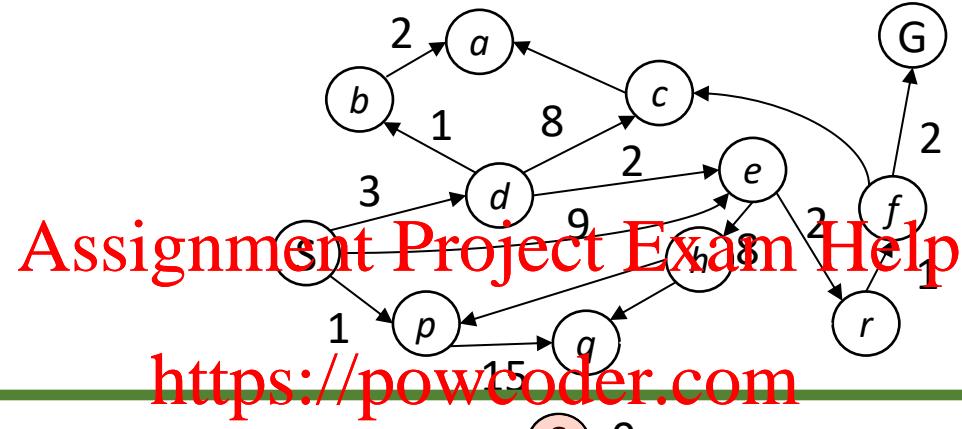
Uninformed Search



Uniform-Cost Search(UCS)

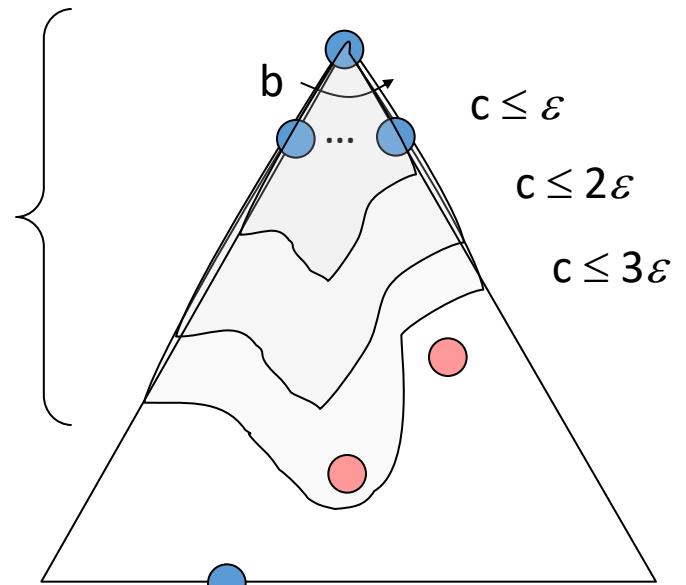
Strategy: expand a cheapest node first:

Fringe is a priority queue
(priority: cumulative cost)



UCS Properties

- What nodes does UCS expand?
 - Processes all nodes with cost less than cheapest solution!
 - If that solution costs C^* and arcs cost at least ε , then the “effective depth” is roughly C^*/ε “tiers”
 - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)
<https://powcoder.com>
- How much space does the fringe take?
 - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
 - Assuming best solution has a finite cost and minimum arc cost is positive, yes!
- Is it optimal?
 - Yes!



Uniform Cost Search

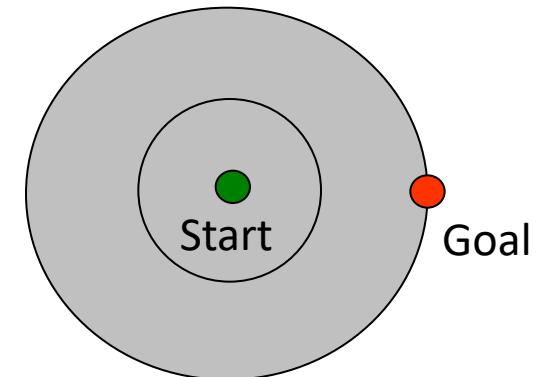
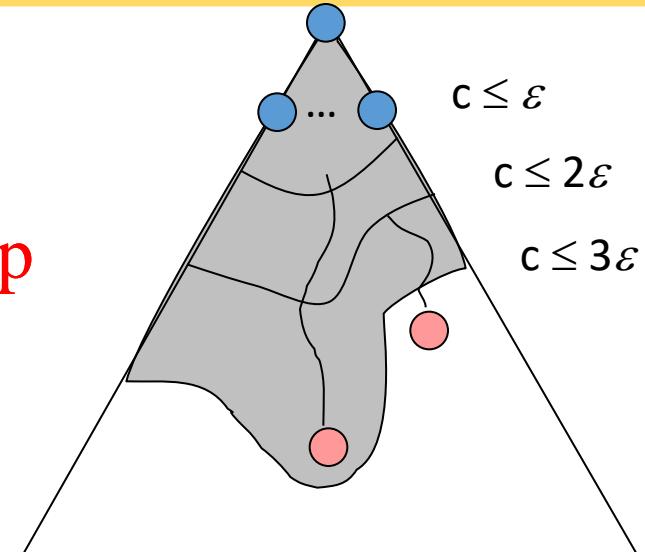
- Strategy: expand lowest path cost

Assignment Project Exam Help

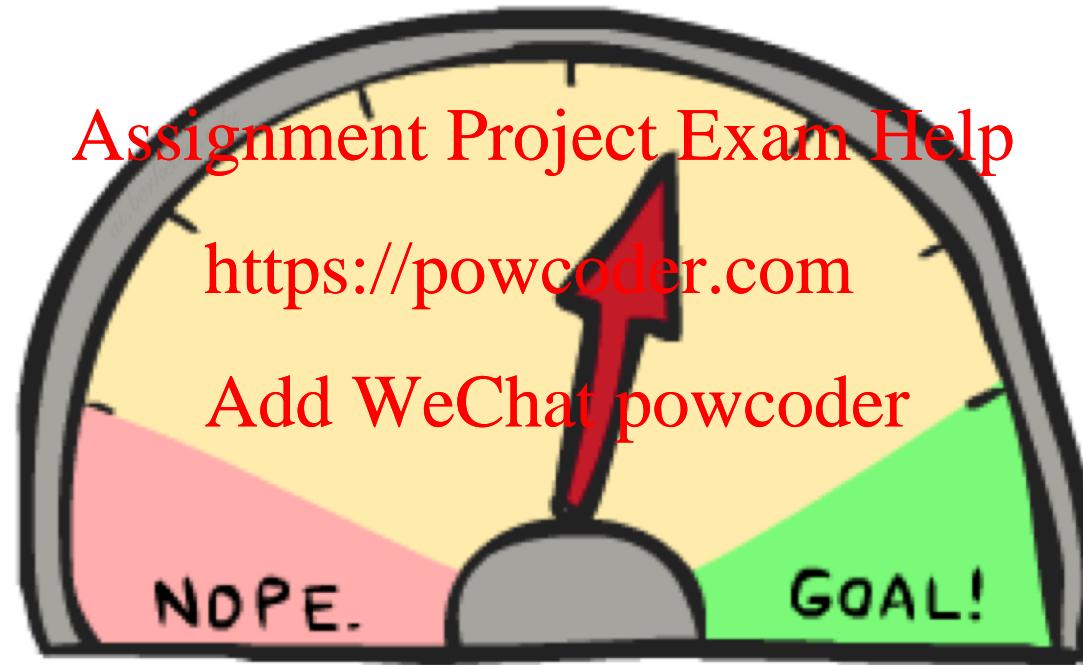
- The good: UCS is complete and optimal!

<https://powcoder.com>
Add WeChat powcoder

- The bad:
 - Explores options in every “direction”
 - No information about goal location



Informed Search

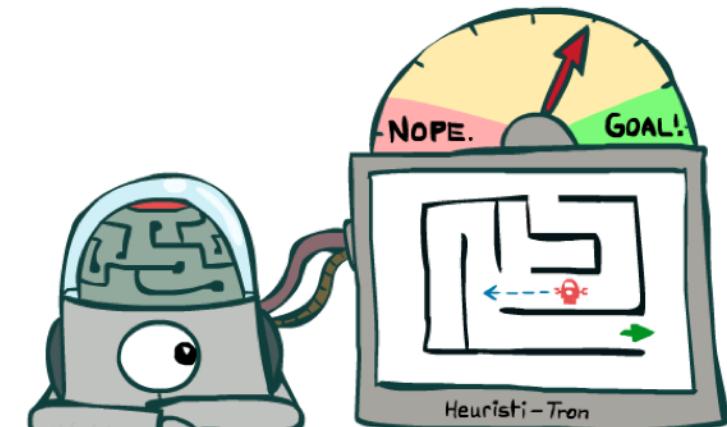
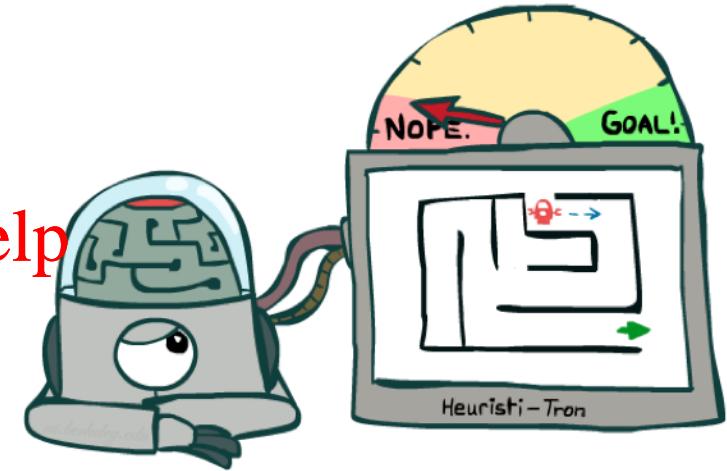
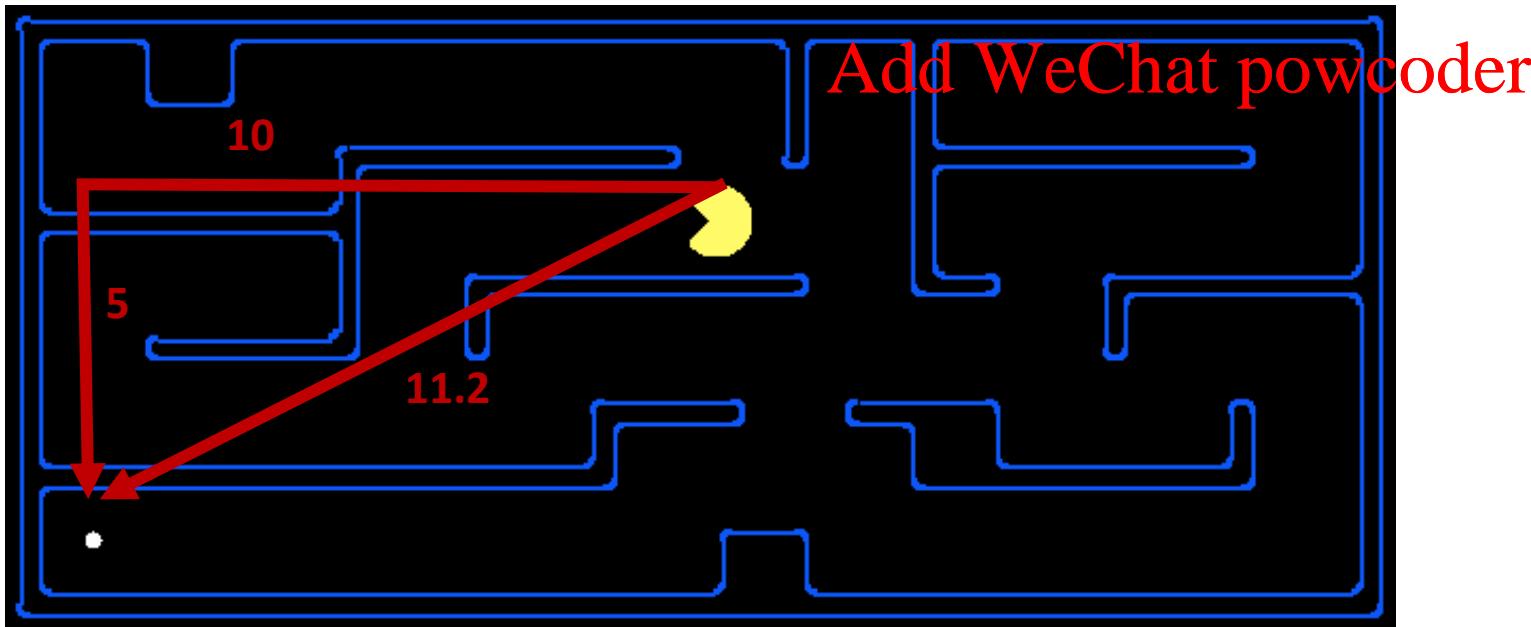


Search Heuristics

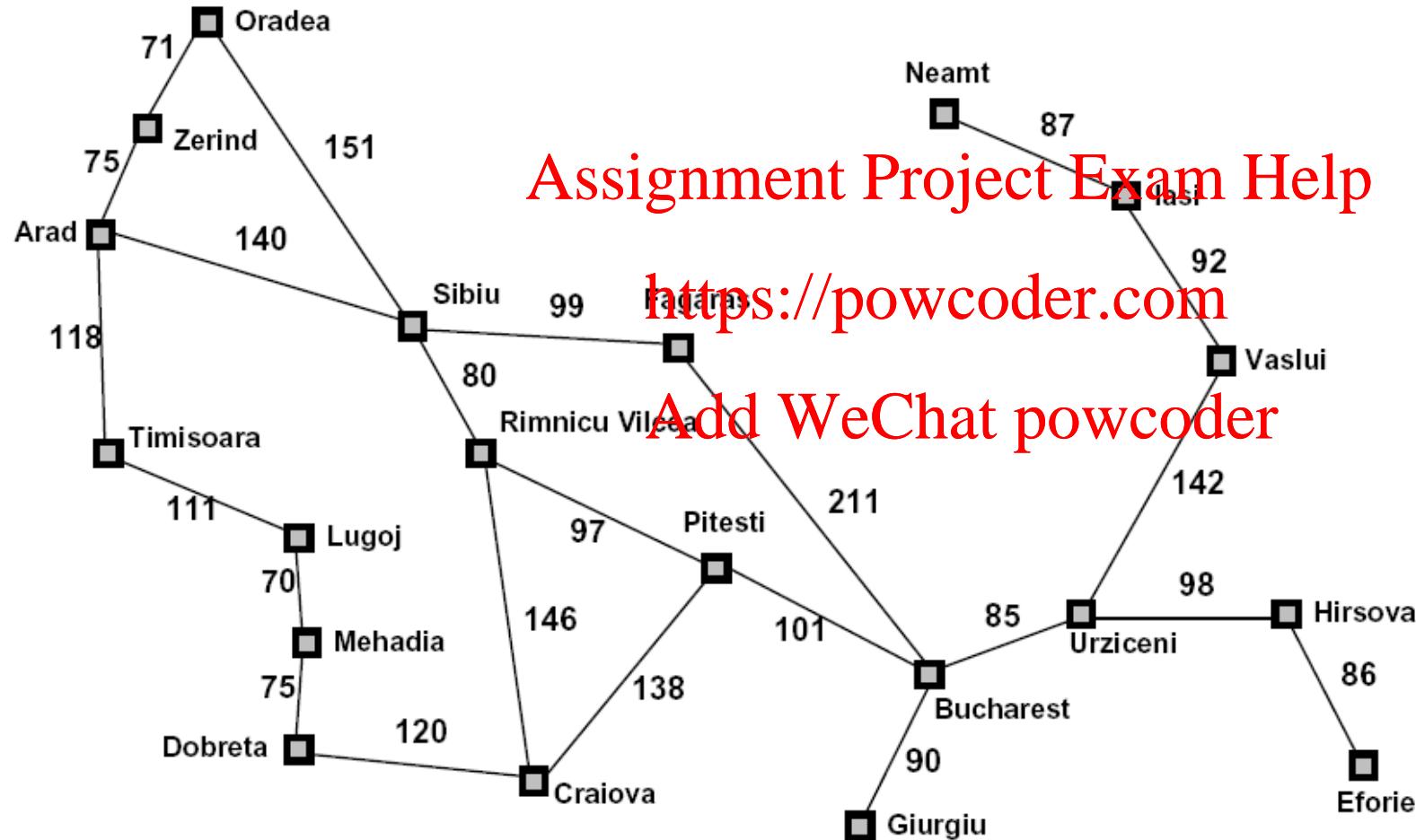
- A heuristic is:
 - A function that *estimates* how close a state is to a goal
 - Designed for a particular search problem
 - Examples: Manhattan distance, Euclidean distance for pathing

Assignment Project Exam Help

<https://powcoder.com>



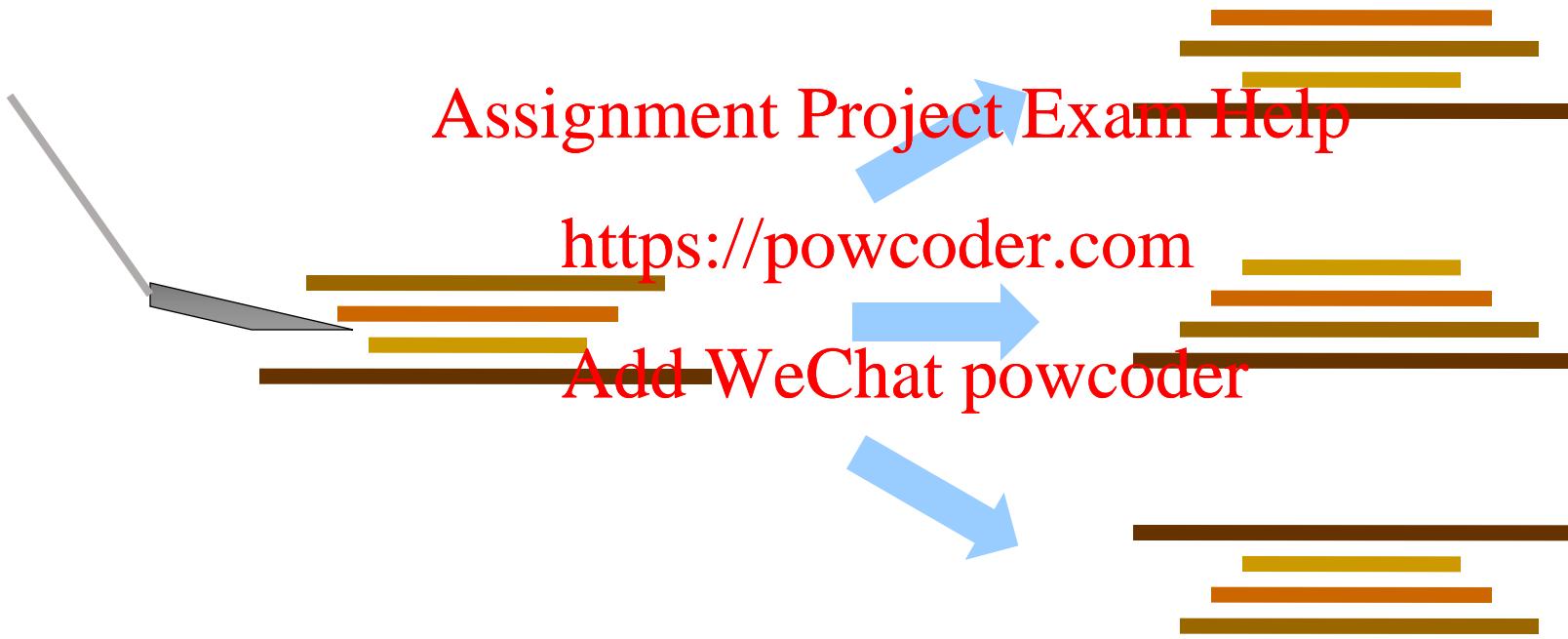
Example: Heuristic Function



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(x)$

Example: Pancake Problem

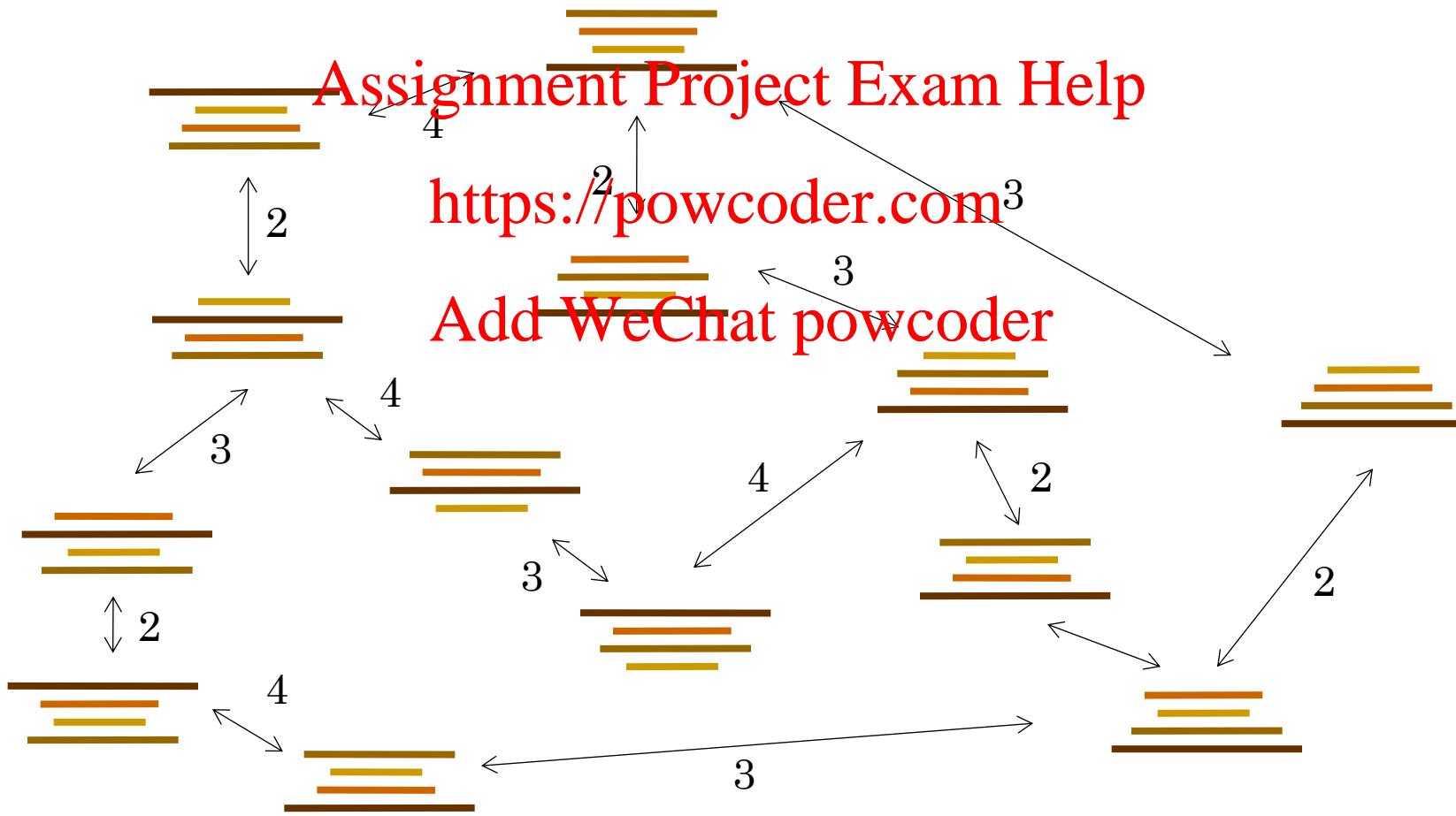


Cost: Number of pancakes flipped



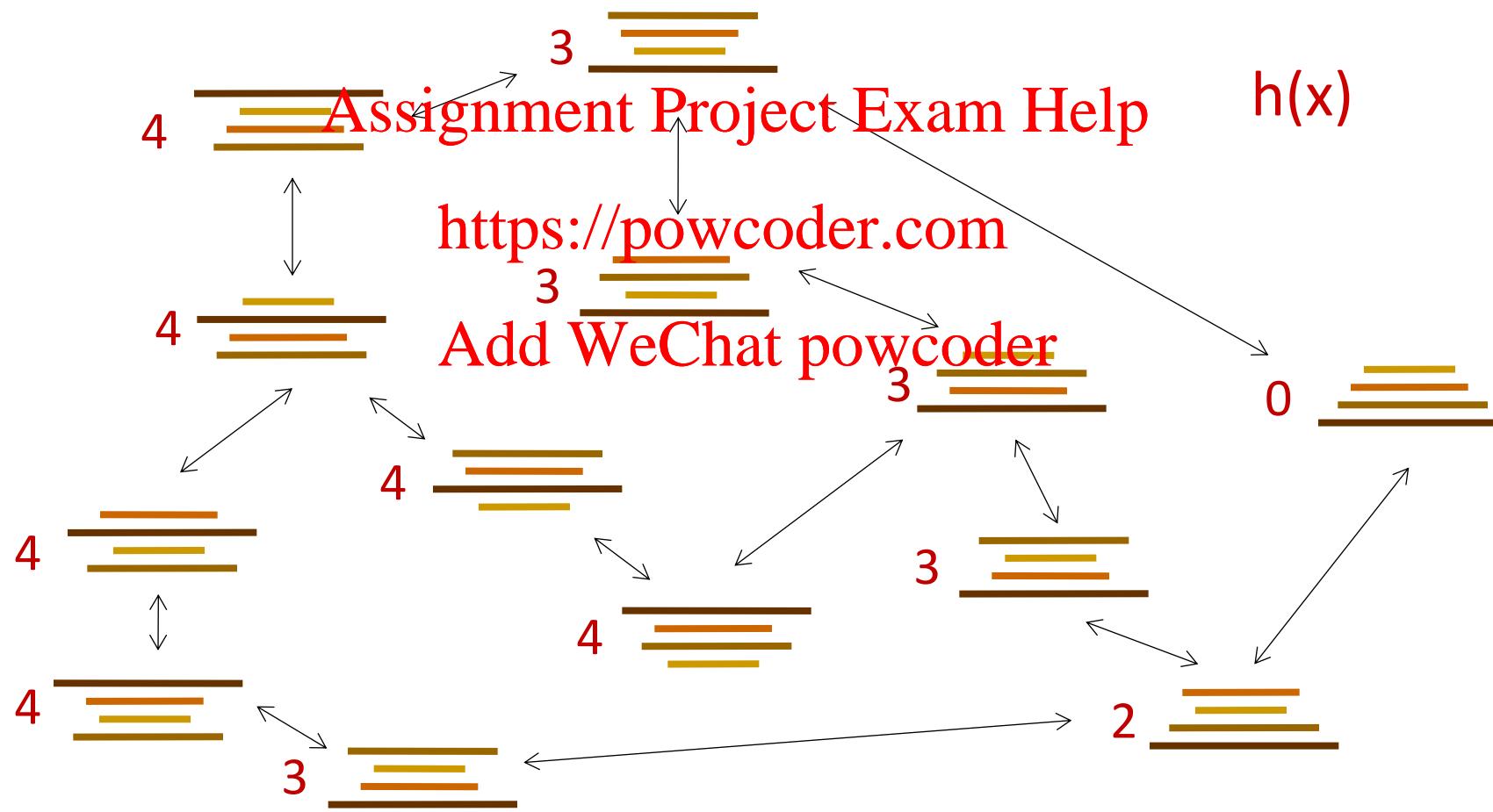
Example: Pancake Problem

State space graph with costs as weights

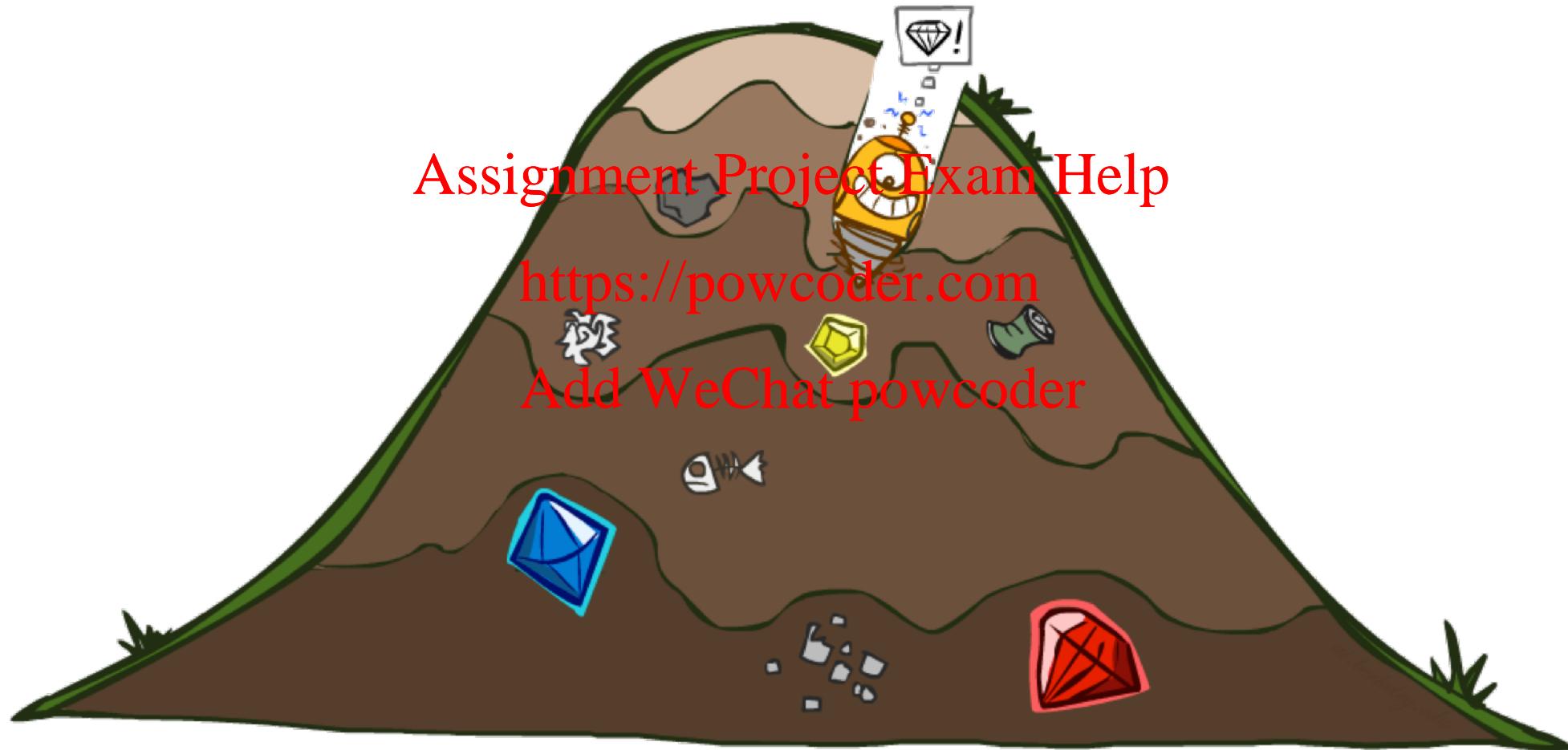


Example: Heuristic Function

Heuristic: the number of the largest pancake that is still out of place

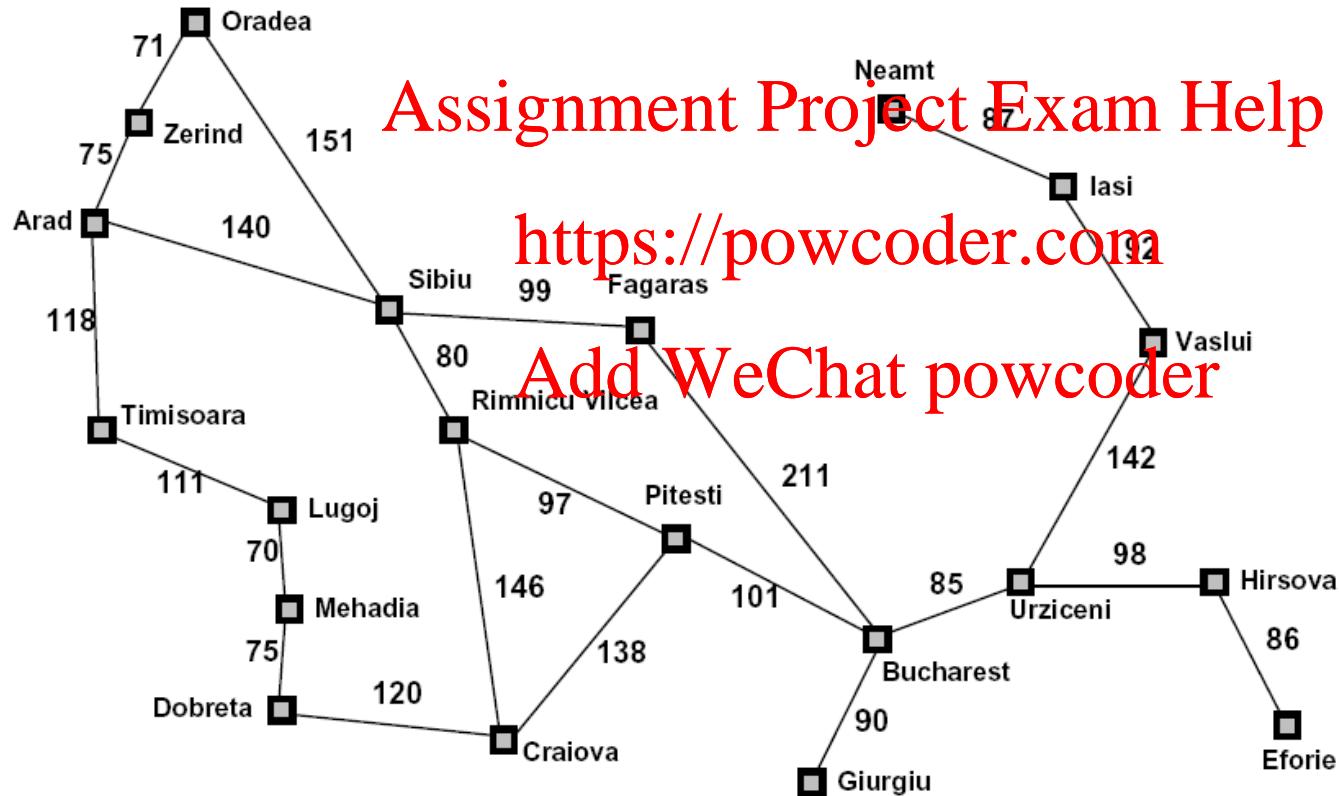


Greedy Search



Greedy Search

- Expand the node that seems closest...

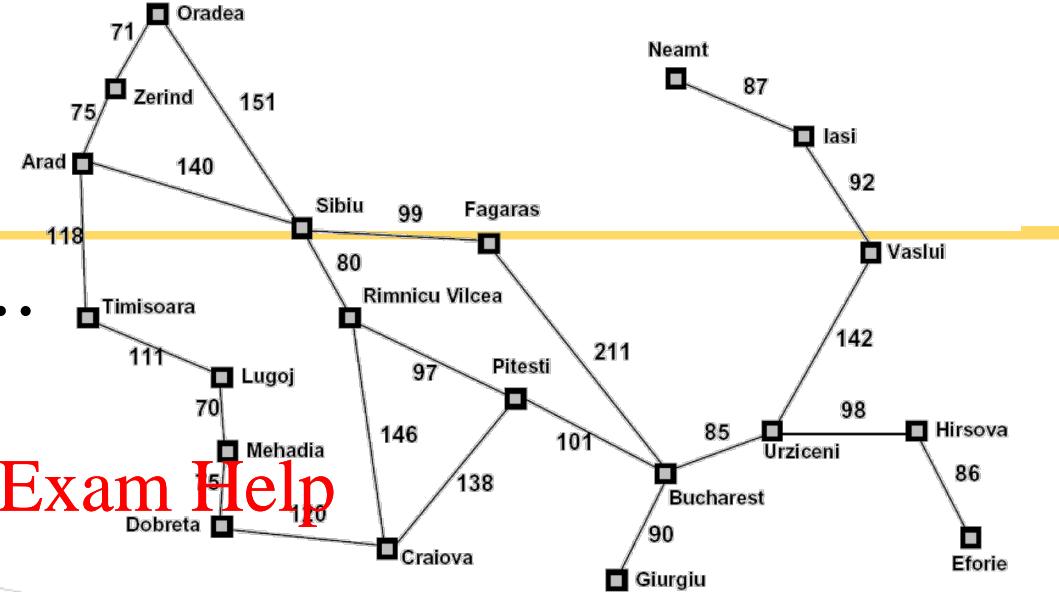


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(x)$

Greedy Search

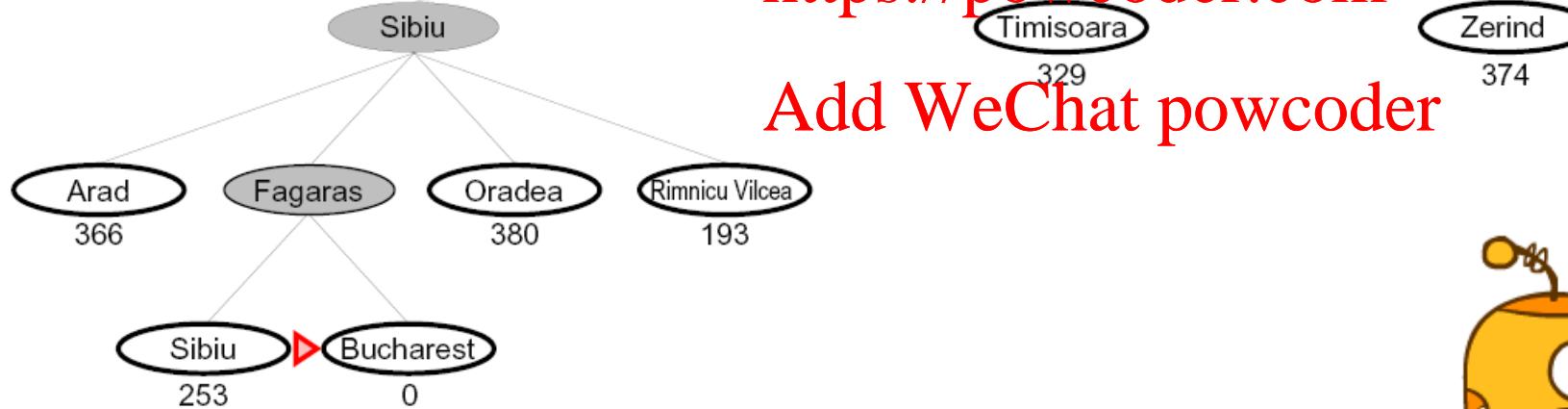
- Expand the node that seems closest...



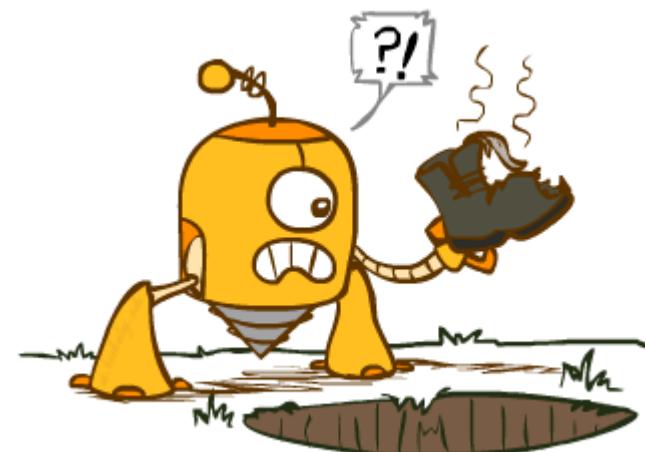
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

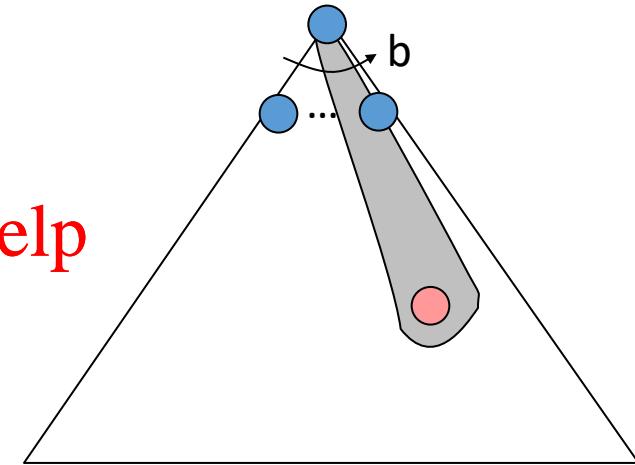


- What can go wrong?

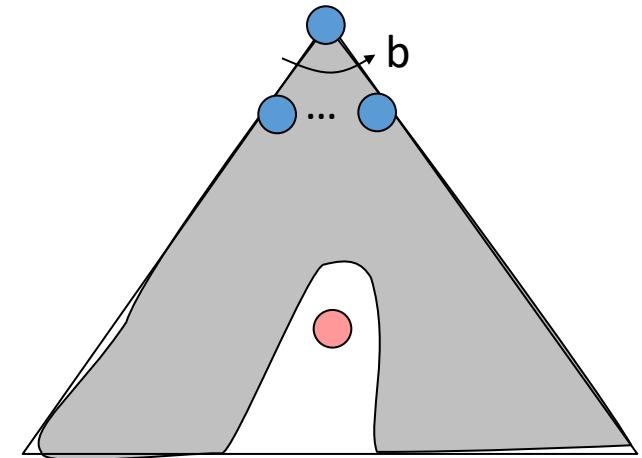


Greedy Search

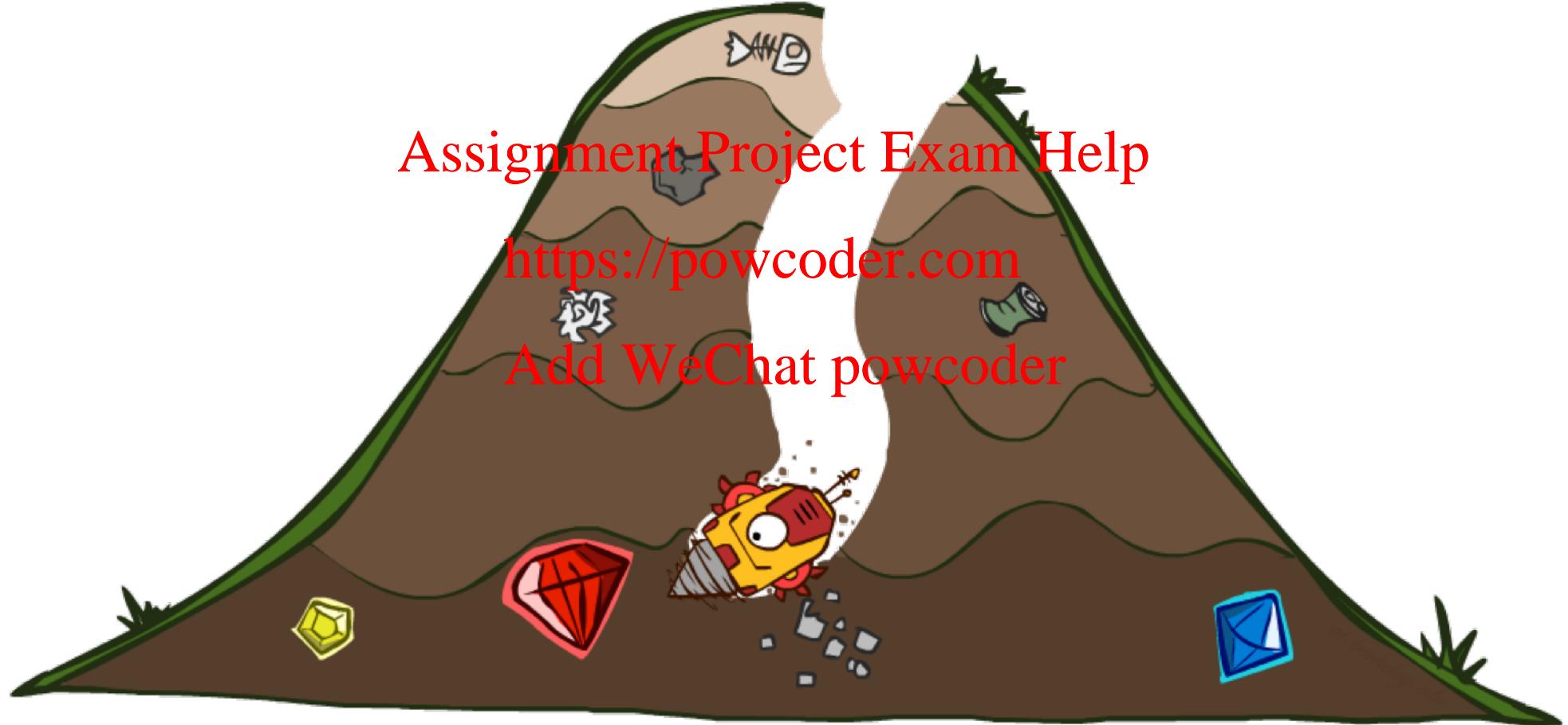
- A common case: **Assignment Project Exam Help**
 - Best-first takes you straight to the (wrong) goal
<https://powcoder.com>
- Worst-case: like a badly-guided DFS



Add WeChat powcoder

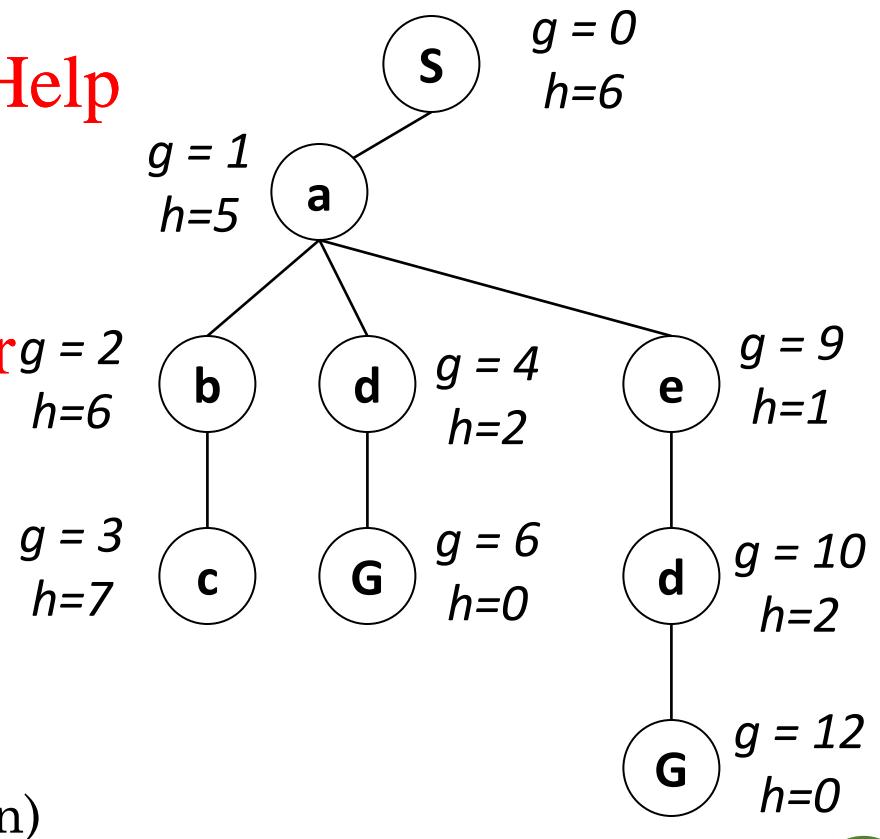
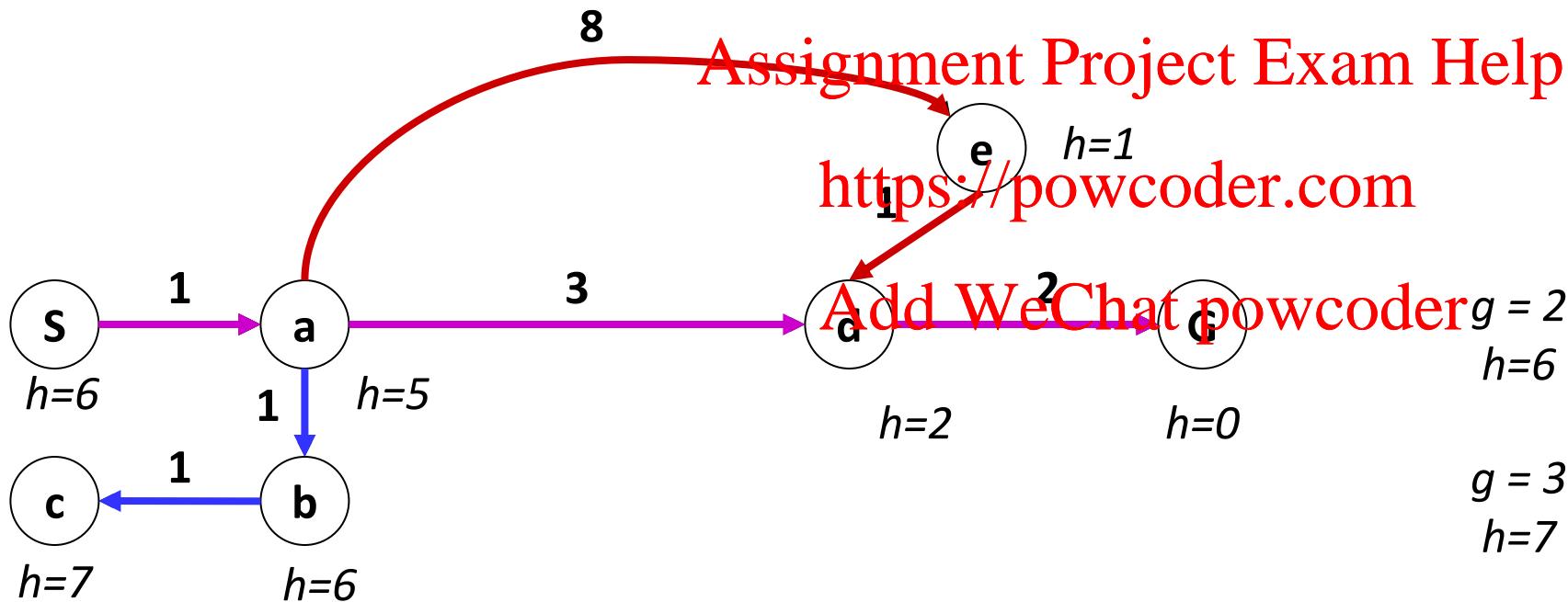


A* Search



Combining UCS and Greedy

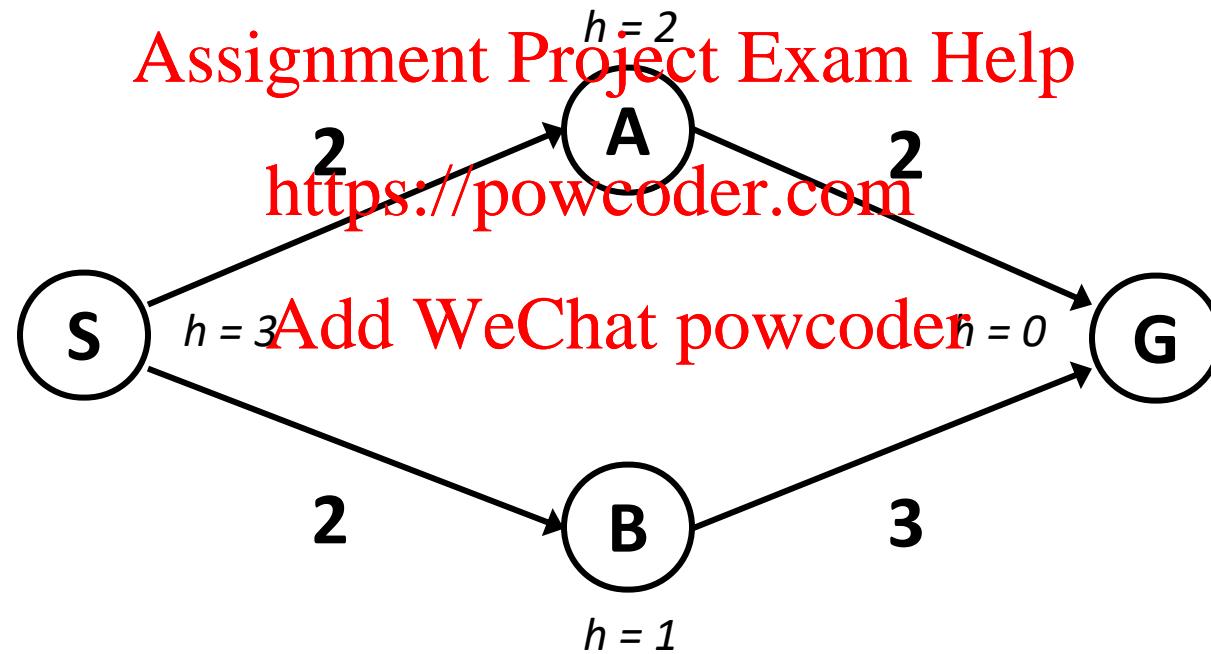
- Uniform-cost orders by path cost, or *backward cost* $g(n)$
- Greedy orders by goal proximity, or *forward cost* $h(n)$



- A* Search orders by the sum: $f(n) = g(n) + h(n)$

When should A* terminate?

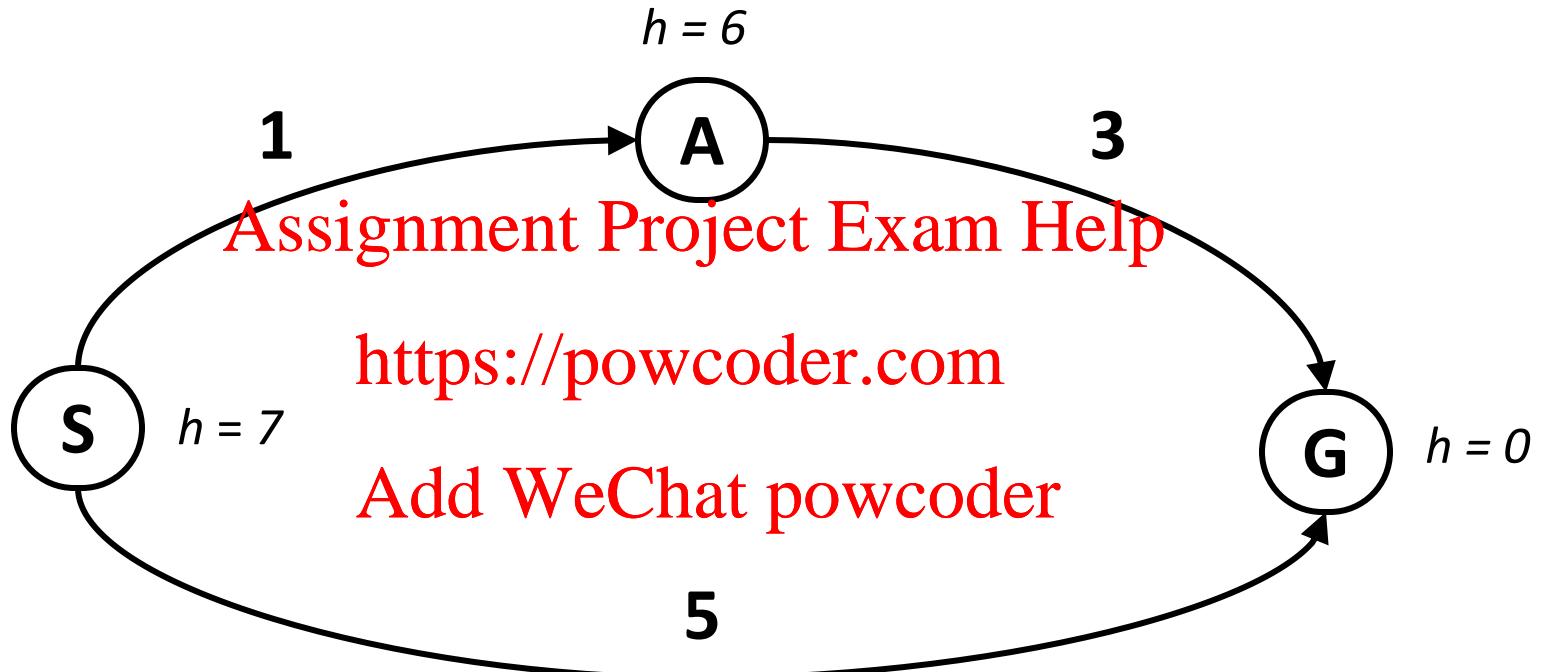
- Should we stop when we enqueue a goal?



- No: only stop when we dequeue a goal



Is A* Optimal?



- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

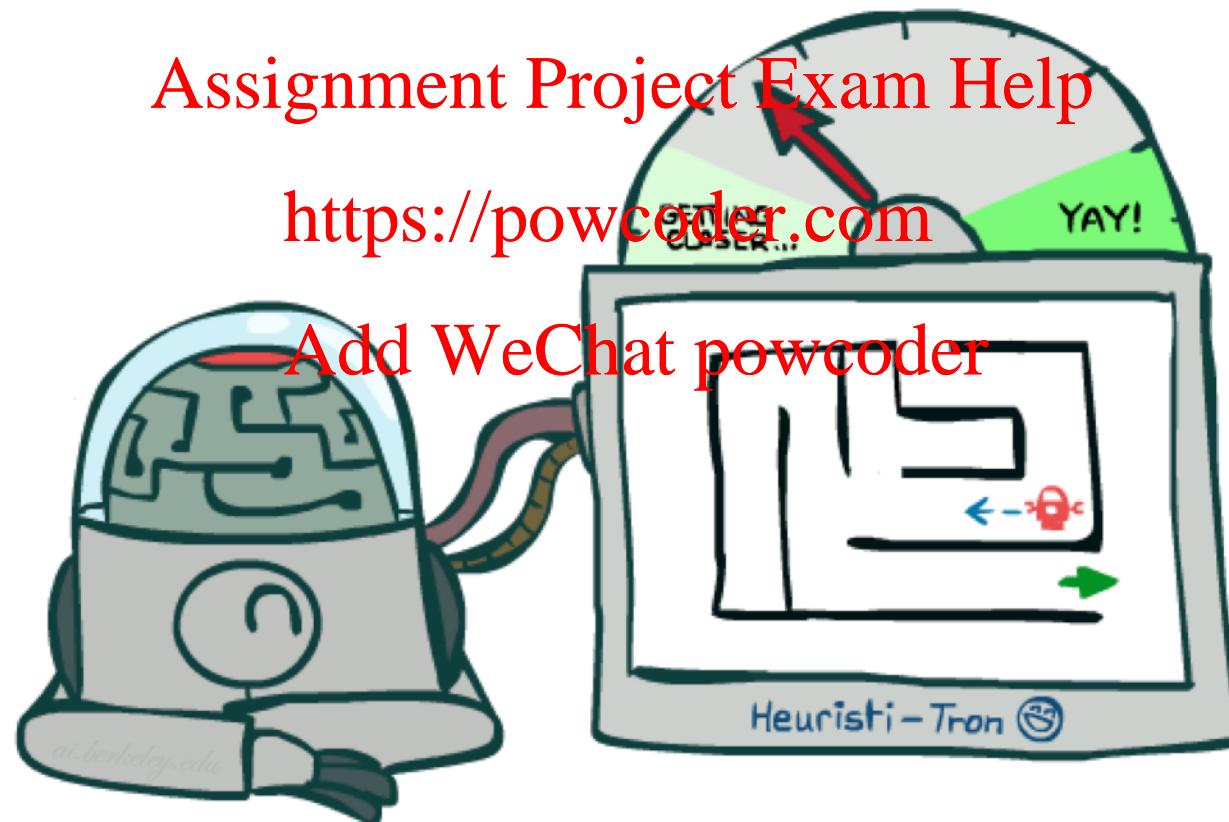


Admissible Heuristics

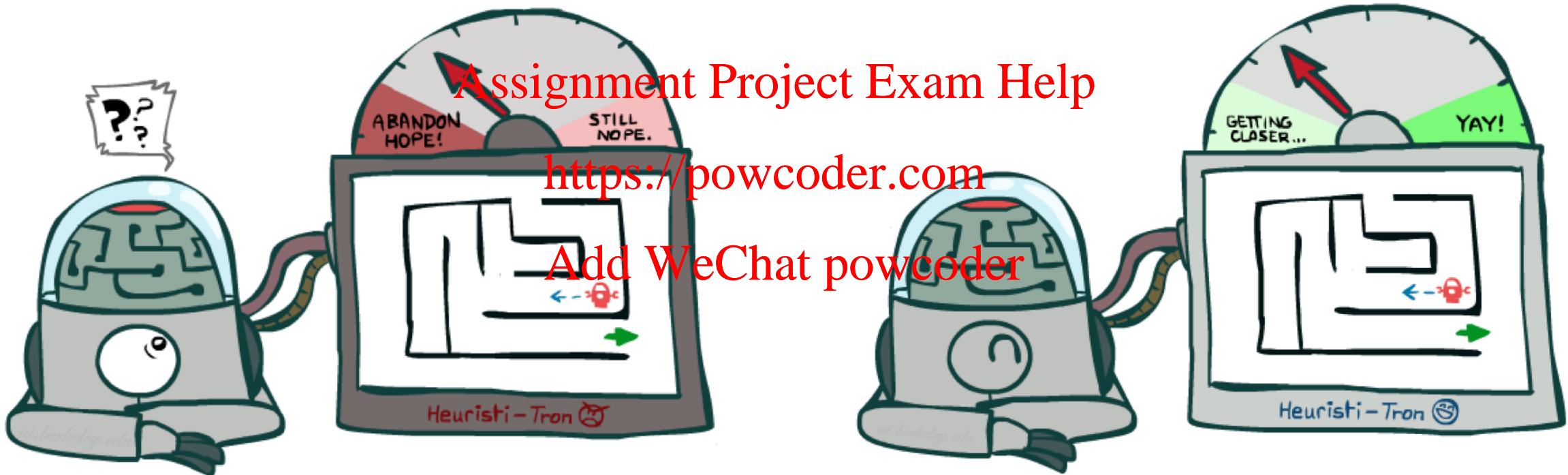
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Idea: Admissibility



Inadmissible (pessimistic) heuristics break optimality by trapping good plans on the fringe

Admissible (optimistic) heuristics never outweigh true costs



Admissible Heuristics

- A heuristic h is *admissible* (optimistic) if:

$$0 \leq h(n) \leq h^*(n)$$

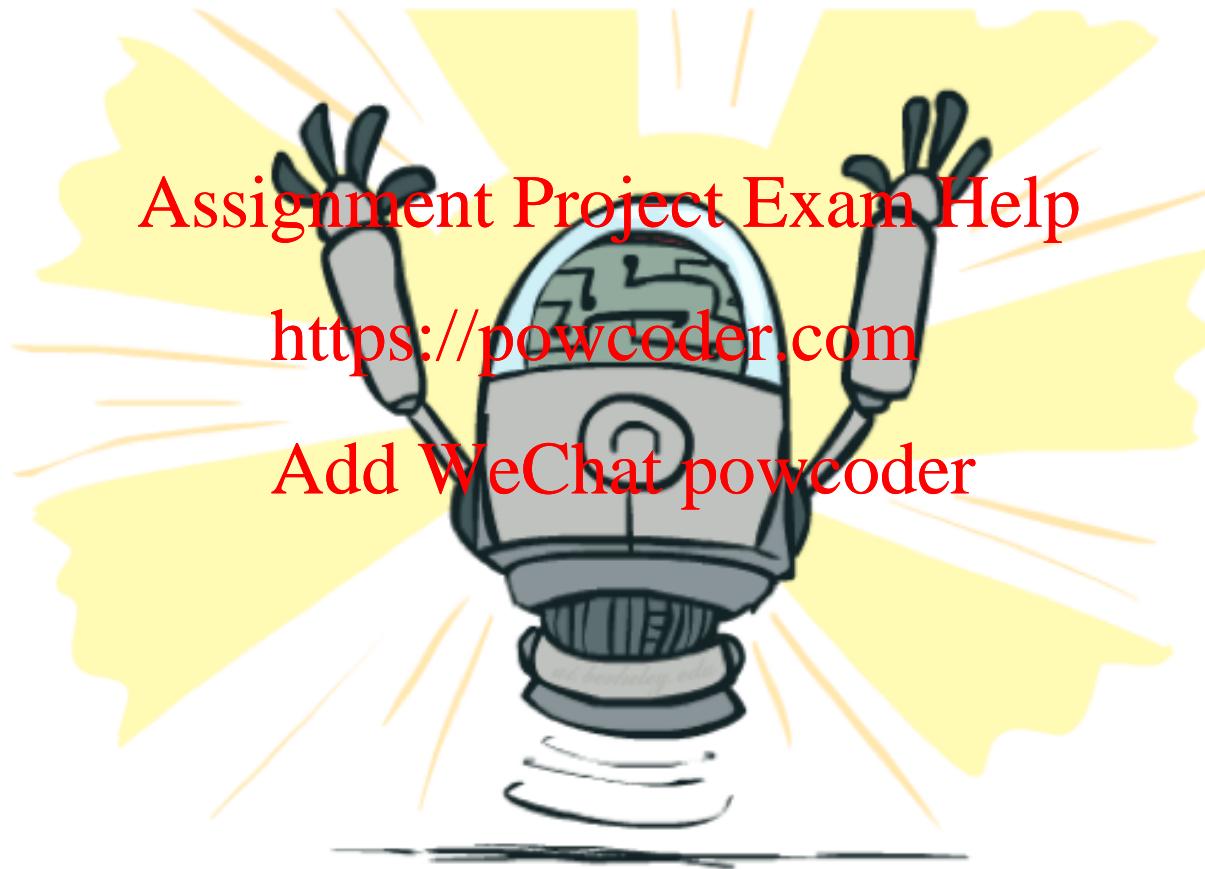
where $h^*(n)$ is the true cost to a nearest goal

- Examples:



- Coming up with admissible heuristics is most of what's involved in using A* in practice.

Optimality of A* Tree Search



Optimality of A* Tree Search

- Heuristic function h is admissible
- Claim: A* tree search is optimal

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Optimality of A* Tree Search

Assume:

- A is an optimal goal node
- B is a suboptimal goal node
- h is admissible

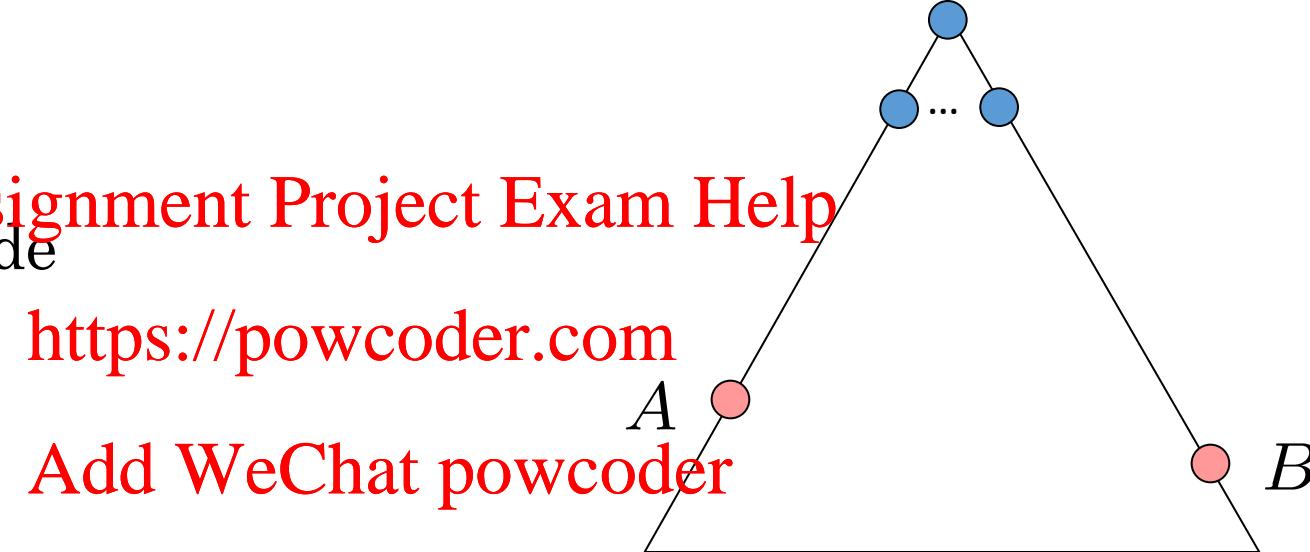
Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Claim:

- A will exit the fringe before B

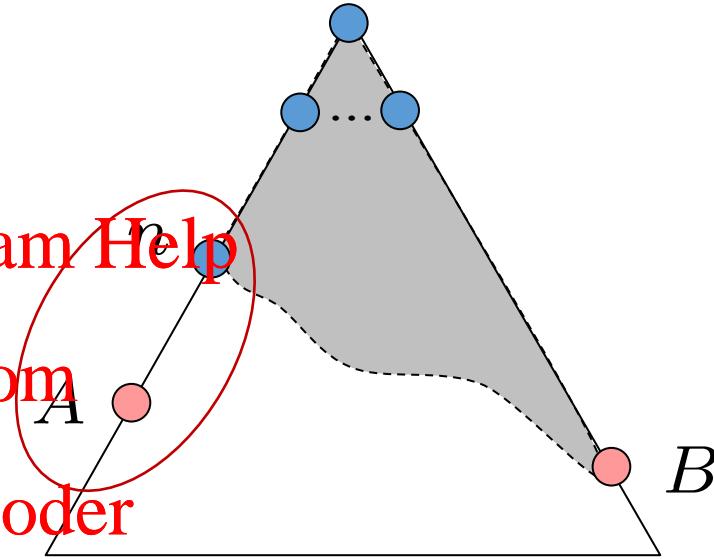


Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B
 - 1. $f(n)$ is less or equal to $f(A)$

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder



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

$$f(n) \leq g(A)$$

$$g(A) = f(A)$$

Definition of f-cost

Admissibility of h

$h = 0$ at a goal

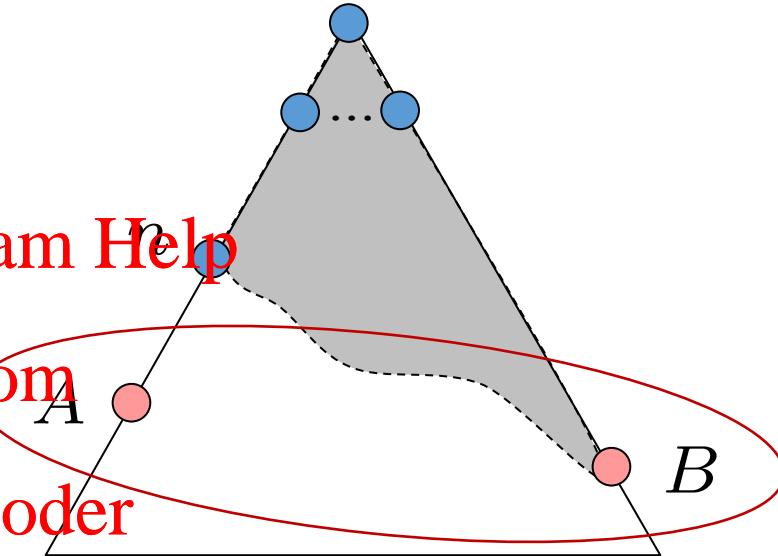
Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B
 1. $f(n)$ is less or equal to $f(A)$
 2. $f(A) < f(B)$

Assignment Project Exam Help

<https://powcoder.com>
Add WeChat powcoder



$$g(A) < g(B)$$

$$f(A) < f(B)$$

B is suboptimal

$h = 0$ at a goal

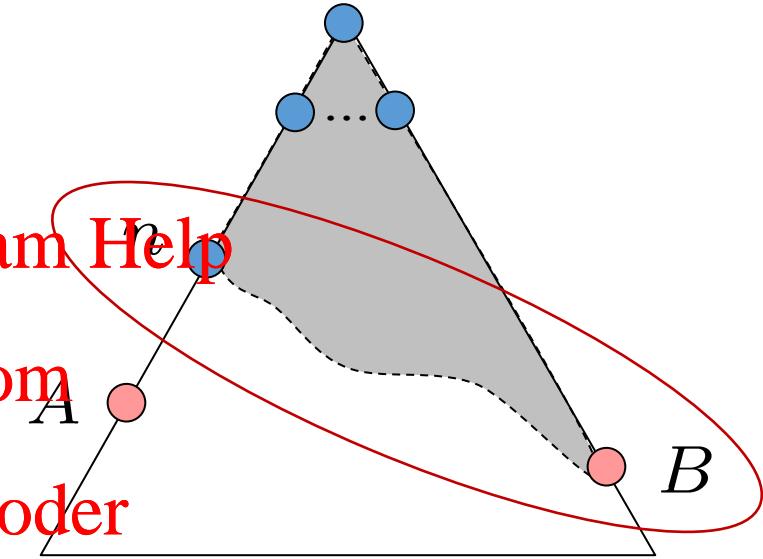


Optimality of A* Tree Search: Blocking

Proof:

- Imagine B is on the fringe
- Some ancestor n of A is on the fringe, too (maybe A!)
- Claim: n will be expanded before B
 1. $f(n)$ is less or equal to $f(A)$
 2. $f(A)$ is less than $f(B)$
 3. n expands before B
- All ancestors of A expand before B
- A expands before B
- A* search is optimal

Assignment Project Exam Help
<https://powcoder.com>
Add WeChat powcoder

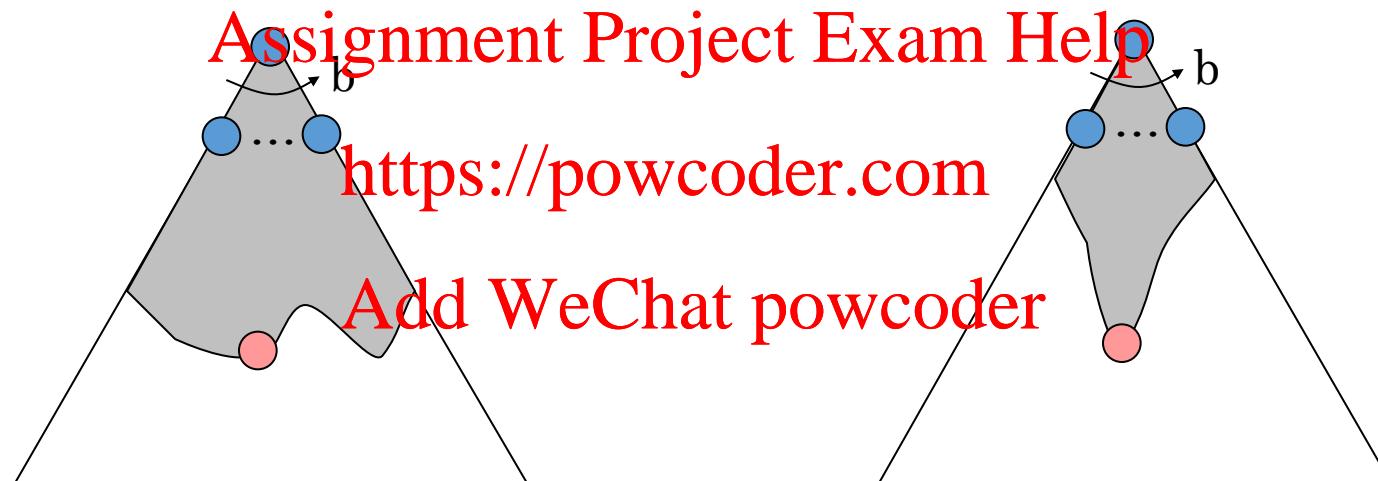


$$f(n) \leq f(A) < f(B)$$



Properties of A*

Uniform-Cost



UCS vs A* Contours

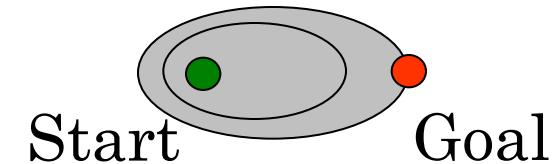
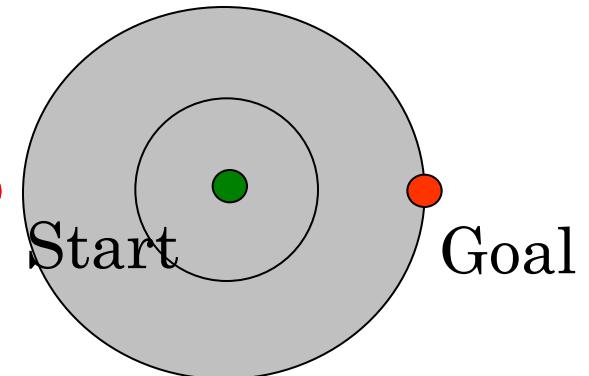
- Uniform-cost expands equally in all “directions”

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

- A* expands mainly toward the goal, but does hedge its bets to ensure optimality



A* Applications

- Video games
 - Pathing / routing pro
Assig
 - Resource planning p
h
 - Robot motion plannin
A
 - Language analysis
 - Machine translation
 - Speech recognition
 - ...

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Example: 8 Puzzle

7	2	4
5		6
8	3	1

Start State

3	7	1
2	4	5
8		6

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Actions

	1	2
3	4	5
6	7	8

Goal State

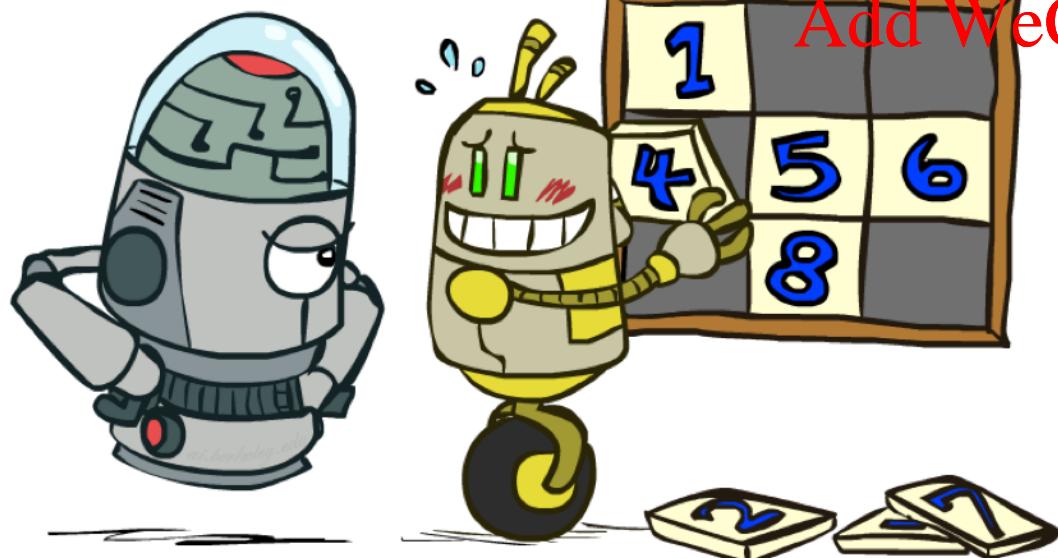
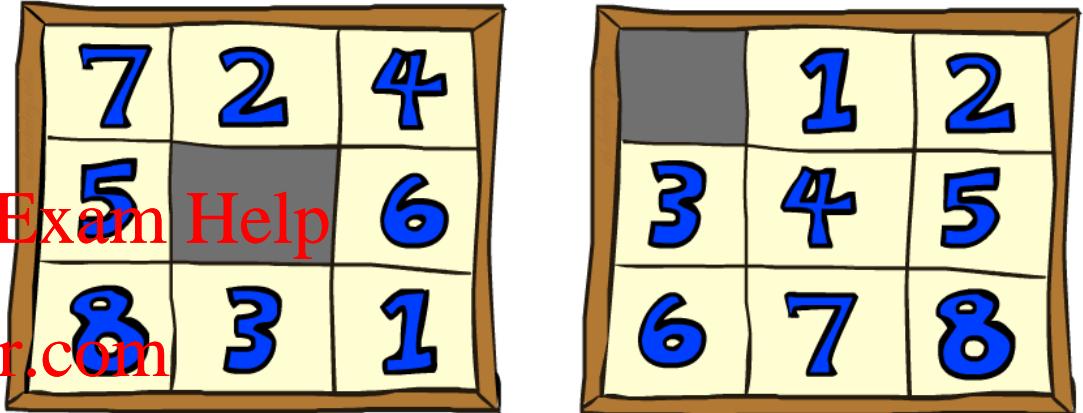
- What are the states?
- How many states?
- What are the actions?
- How many successors from the start state?
- What should the costs be?



8 Puzzle I

- Heuristic: Number of tiles misplaced
- Why is it admissible?
- $h(\text{start}) = 8$
- This is a *relaxed-problem* <https://powcoder.com/heuristic>

Assignment Project Exam Help



Add WeChat powcoder

Start State

Goal State

Average nodes expanded when
the optimal path has...

...4 steps	...8 steps	...12 steps
UCS	112	6,300
TILES	13	39

Statistics from Andrew Moore

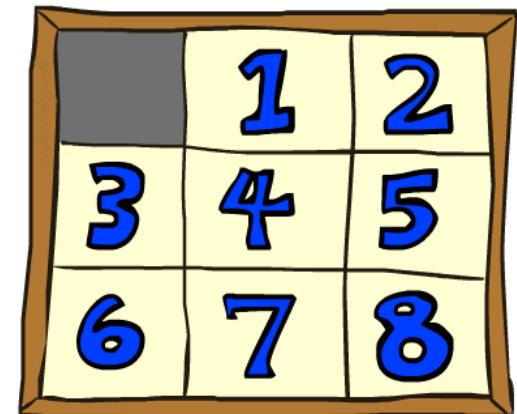
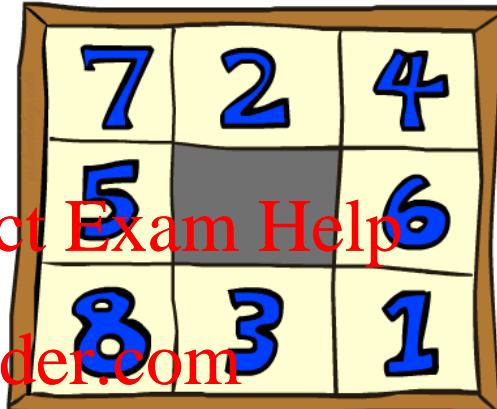
8 Puzzle II

- What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Start State

Goal State

- Why is it admissible?

- $h(\text{start}) =$

$$3 + 1 + 2 + \dots = 18$$

Average nodes expanded when
the optimal path has...

TILES	...4 steps	...8 steps	...12 steps
MANHATTAN	13	39	227
	12	25	73

8 Puzzle III

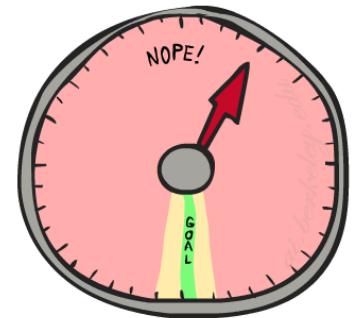
- How about using the *actual cost* as a heuristic?
 - Would it be admissible?
 - Would we save on nodes expanded?
 - What's wrong with it?

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

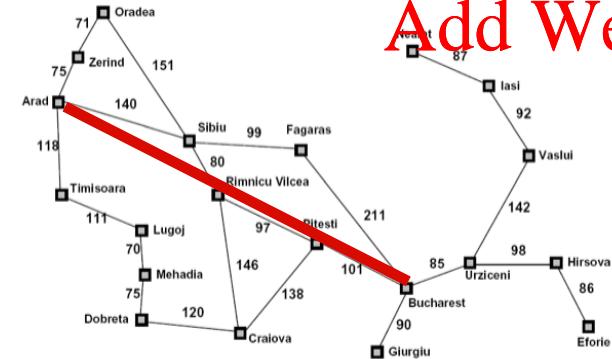
- With A*: a trade-off between quality of estimate and work per node
 - As heuristics get closer to the true cost, you will expand fewer nodes but usually do more work per node to compute the heuristic itself



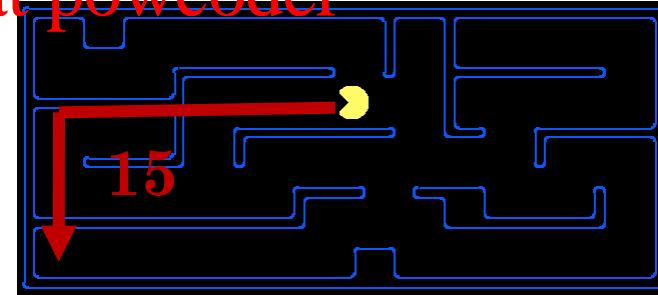
Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics
- Often, admissible heuristics are solutions to *relaxed problems*, where new actions are available
<https://powcoder.com>

366



Add WeChat powcoder



- Inadmissible heuristics are often useful too (why?)



Trivial Heuristics, Dominance

- Dominance: $h_a \geq h_c$ if

$$\forall n : h_a(n) \geq h_c(n)$$

Assignment Project Exam Help

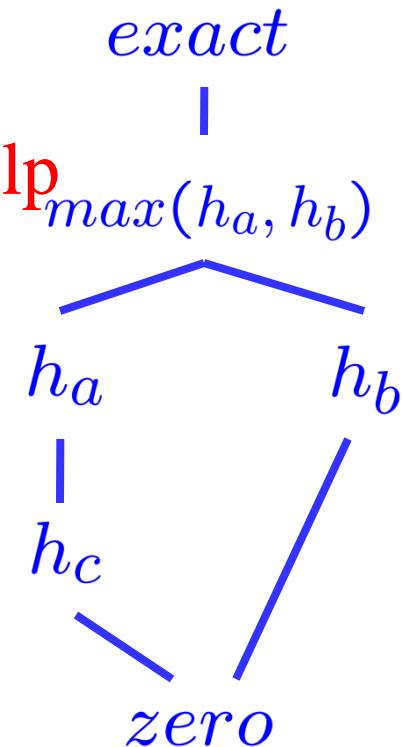
- Heuristics form a semi-lattice:
<https://powcoder.com>

- Max of admissible heuristics is admissible

$$h(n) = \max(h_a(n), h_b(n))$$

Add WeChat powcoder

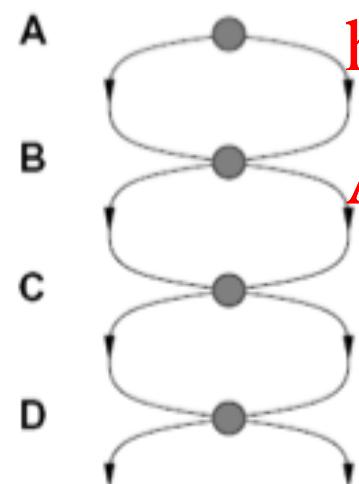
- Trivial heuristics
 - Bottom of lattice is the zero heuristic (what does this give us?)
 - Top of lattice is the exact heuristic



Tree Search: Extra Work!

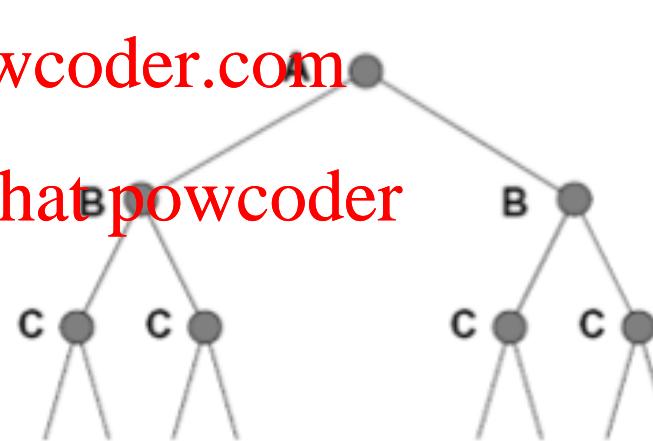
- Failure to detect repeated states can cause exponentially more work. Why?

Assignment Project Exam Help



<https://powcoder.com>

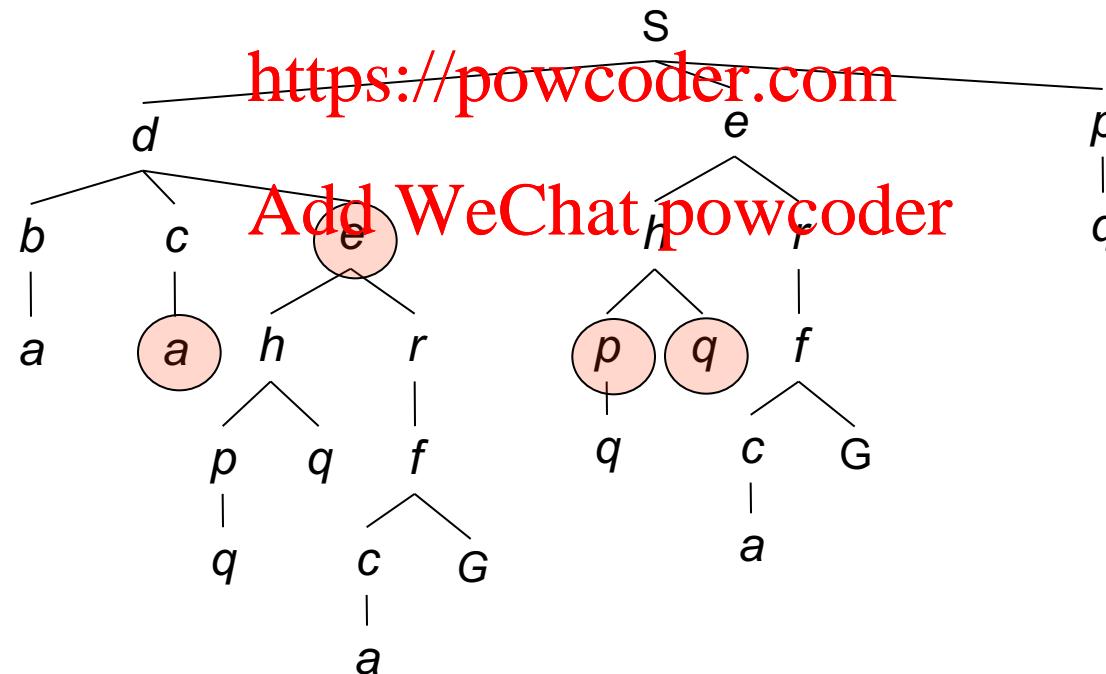
Add WeChat powcoder



Graph Search

- In BFS, for example, we shouldn't bother expanding some nodes (which, and why?)

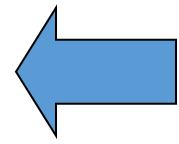
Assignment Project Exam Help



Graph Search

- Very simple fix: never expand a state type twice

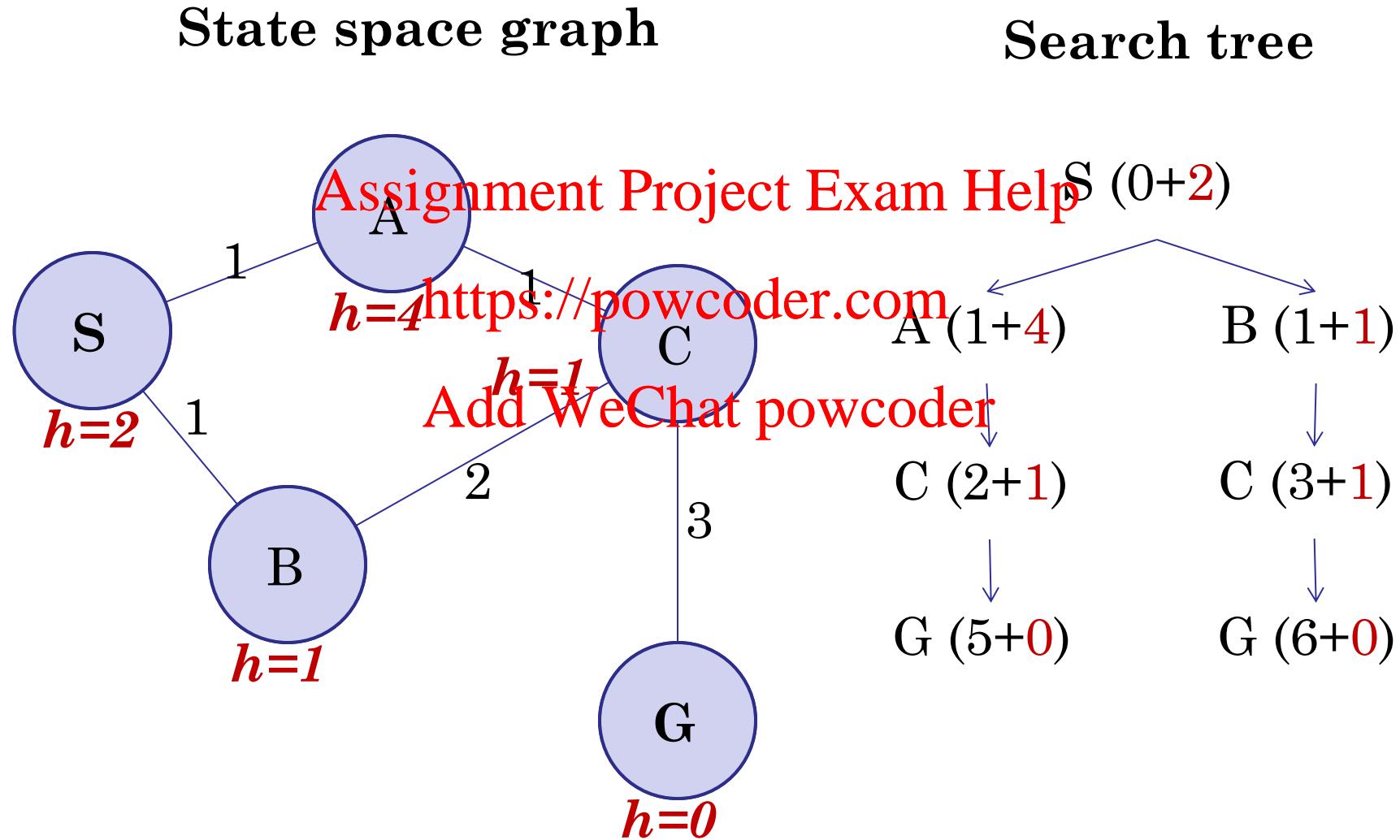
```
function GRAPH-SEARCH(problem, fringe) returns a solution, or failure
  closed  $\leftarrow$  an empty set
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FIRST(fringe)
    if GOAL-TEST(problem,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



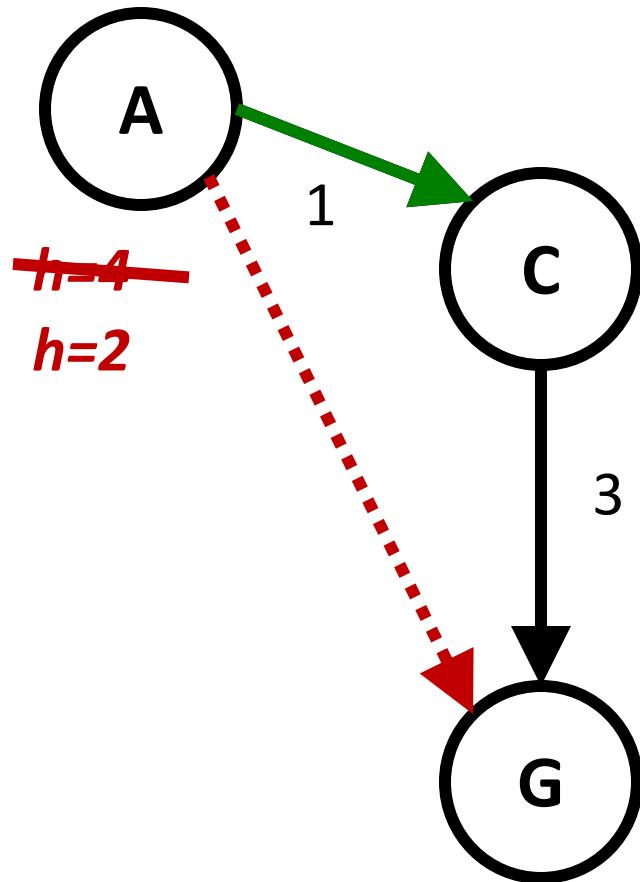
- Can this wreck completeness? Why or why not?
- How about optimality? Why or why not?



A* Graph Search Gone Wrong



Consistency of Heuristics



- Main idea: estimated heuristic costs \leq actual costs

- Admissibility: heuristic cost \leq actual cost to goal

Assignment Project Exam Help
 $h(A) \leq$ actual cost from A to G

<https://powcoder.com>
Consistency: heuristic “arc” cost \leq actual cost for each arc

Add WeChat powcoder
 $h^{(A)} - h^{(C)} \leq \text{cost}(A \text{ to } C)$

- Consequences of consistency:

- The f value along a path never decreases

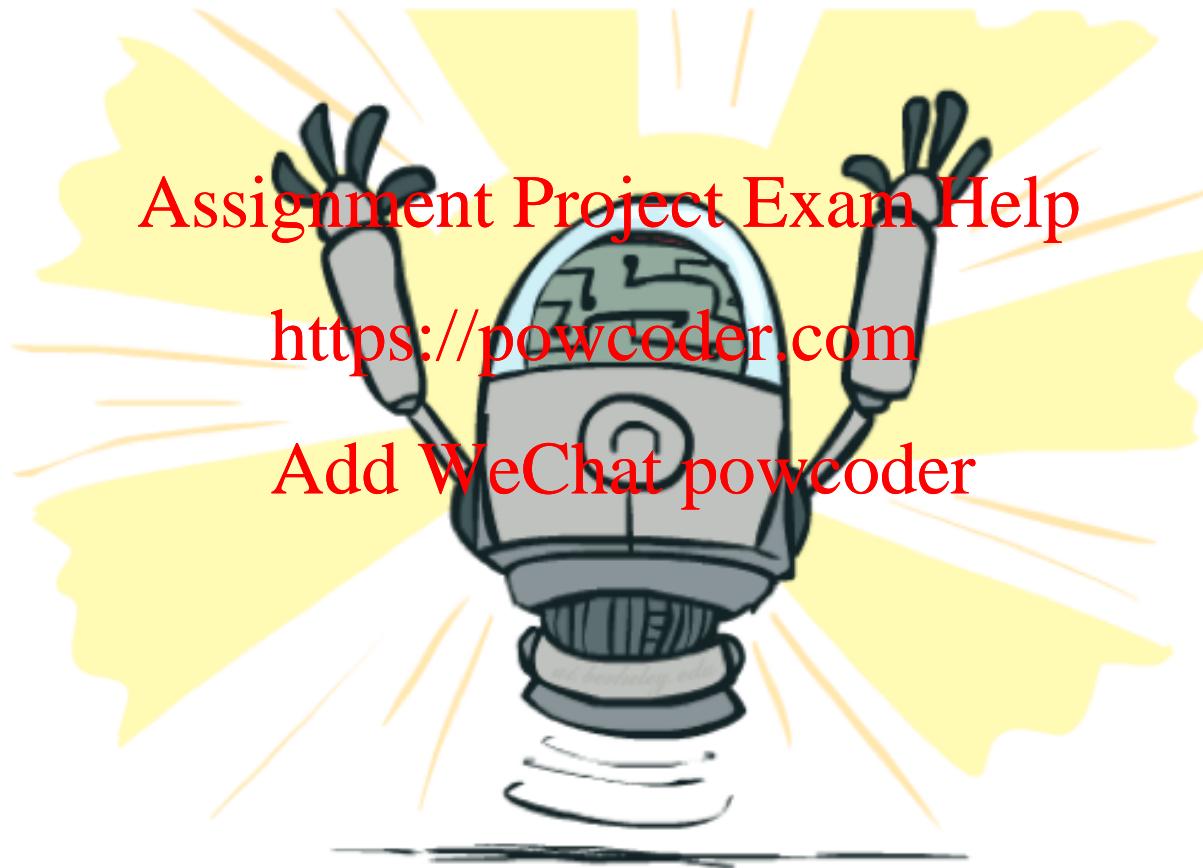
$$h(A) \leq \text{cost}(A \text{ to } C) + h(C)$$

$$f(A) = g(A) + h(A) \leq g(A) + \text{cost}(A \text{ to } C) + h(C) \leq f(C)$$

- A* graph search is optimal



Optimality of A* Graph Search



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



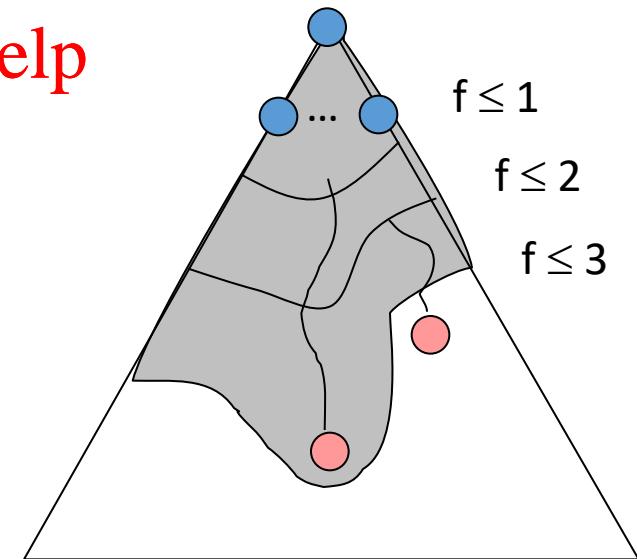
Optimality of A* Graph Search

- Sketch: consider what A* does with a consistent heuristic:

- Fact 1: In tree search, A* expands nodes in increasing total f value (f -contours)
Assignment Project Exam Help
<https://powcoder.com>

- Fact 2: For every state s , nodes that reach s optimally are expanded before nodes that reach s suboptimally
Add WeChat powcoder

- Result: A* graph search is optimal



Optimality

- Tree search:
 - A* is optimal if heuristic is admissible
 - UCS is a special case ($h = 0$)

Assignment Project Exam Help

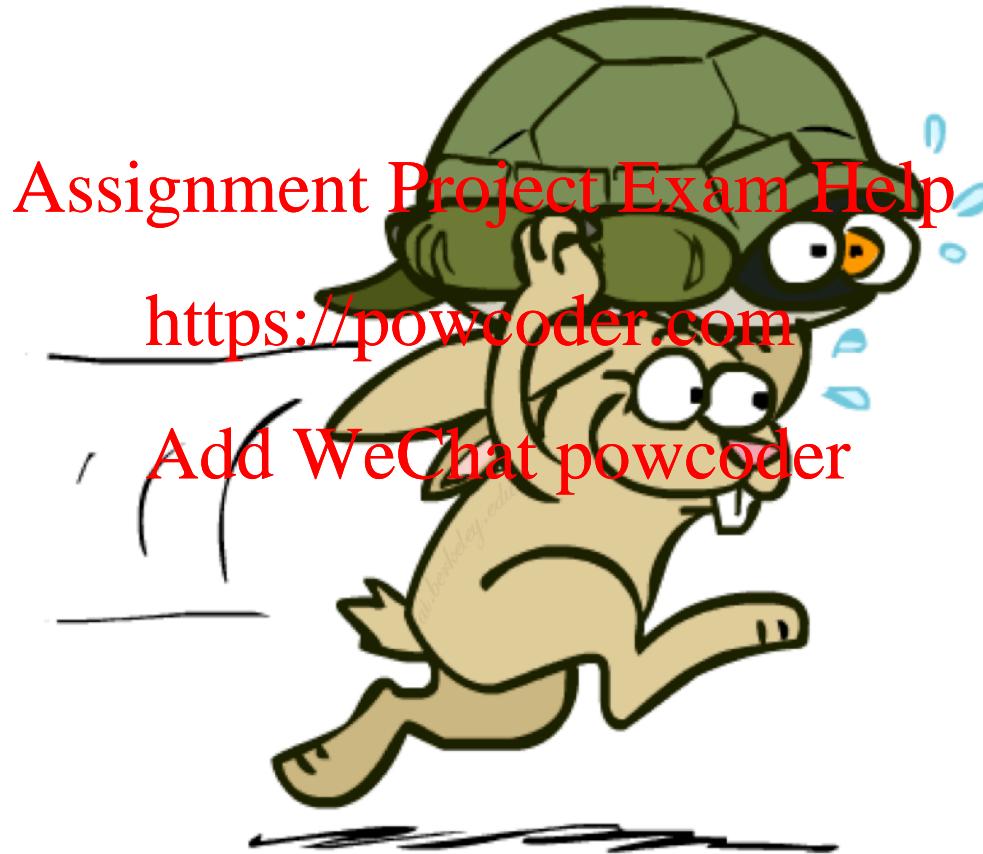
<https://powcoder.com>

Add WeChat powcoder

- Graph search:
 - A* optimal if heuristic is consistent
 - UCS optimal ($h = 0$ is consistent)
- Consistency implies admissibility
- In general, most natural admissible heuristics tend to be consistent, especially if from relaxed problems



A*: Summary



A*: Summary

- A* uses both backward costs and (estimates of) forward costs

Assignment Project Exam Help

- A* is optimal with admissible / consistent heuristics
<https://powcoder.com>

- Heuristic design is key: often use relaxed problems



Tree Search Pseudo-Code

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  REMOVE-FRONT(fringe)
    if GOAL-TEST(problem, STATE[node]) then return node
    for child-node in EXPAND(STATE[node], problem) do
      fringe  $\leftarrow$  INSERT(child-node, fringe)
  end
end
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder



Graph Search Pseudo-Code

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
  closed  $\leftarrow$  an empty set
  fringe  $\leftarrow$  INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
  loop do
    if fringe is empty then return failure
    node  $\leftarrow$  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
      for child-node in EXPAND(STATE[node], problem) do
        fringe  $\leftarrow$  INSERT(child-node, fringe)
    end
  end
```

Assignment Project Exam Help

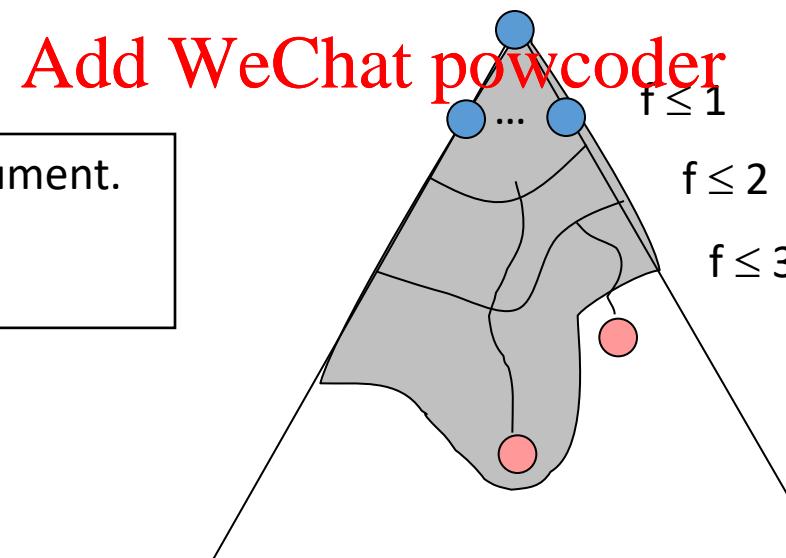
<https://powcoder.com>

Add WeChat powcoder



Optimality of A* Graph Search

- Consider what A* does:
 - Expands nodes in increasing total f value (f-contours)
Reminder: $f(n) = g(n) + h(n)$ = cost to n + heuristic
 - Proof idea: the optimal goal(s) have the lowest f value, so it must get expanded first



Optimality of A* Graph Search

Proof:

- New possible problem: some n on path to G^* isn't in queue when we need it, because some worse n' for the same state dequeued and expanded first (disaster!)
- Take the highest such n in tree
- Let p be the ancestor of n that was on the queue when n' was popped
- $f(p) < f(n)$ because of consistency
- $f(n) < f(n')$ because n' is suboptimal
- p would have been expanded before n'
- Contradiction!

