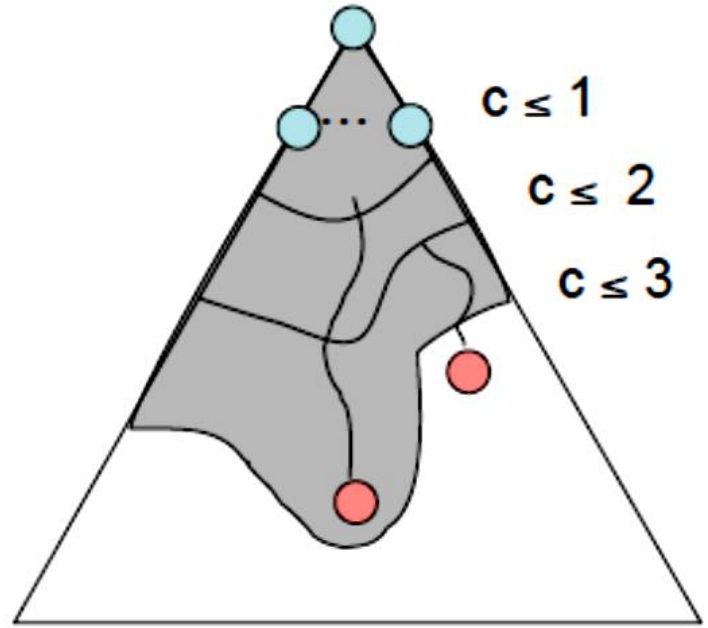
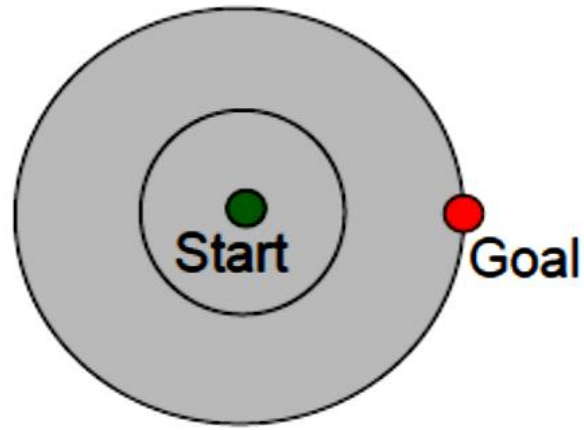
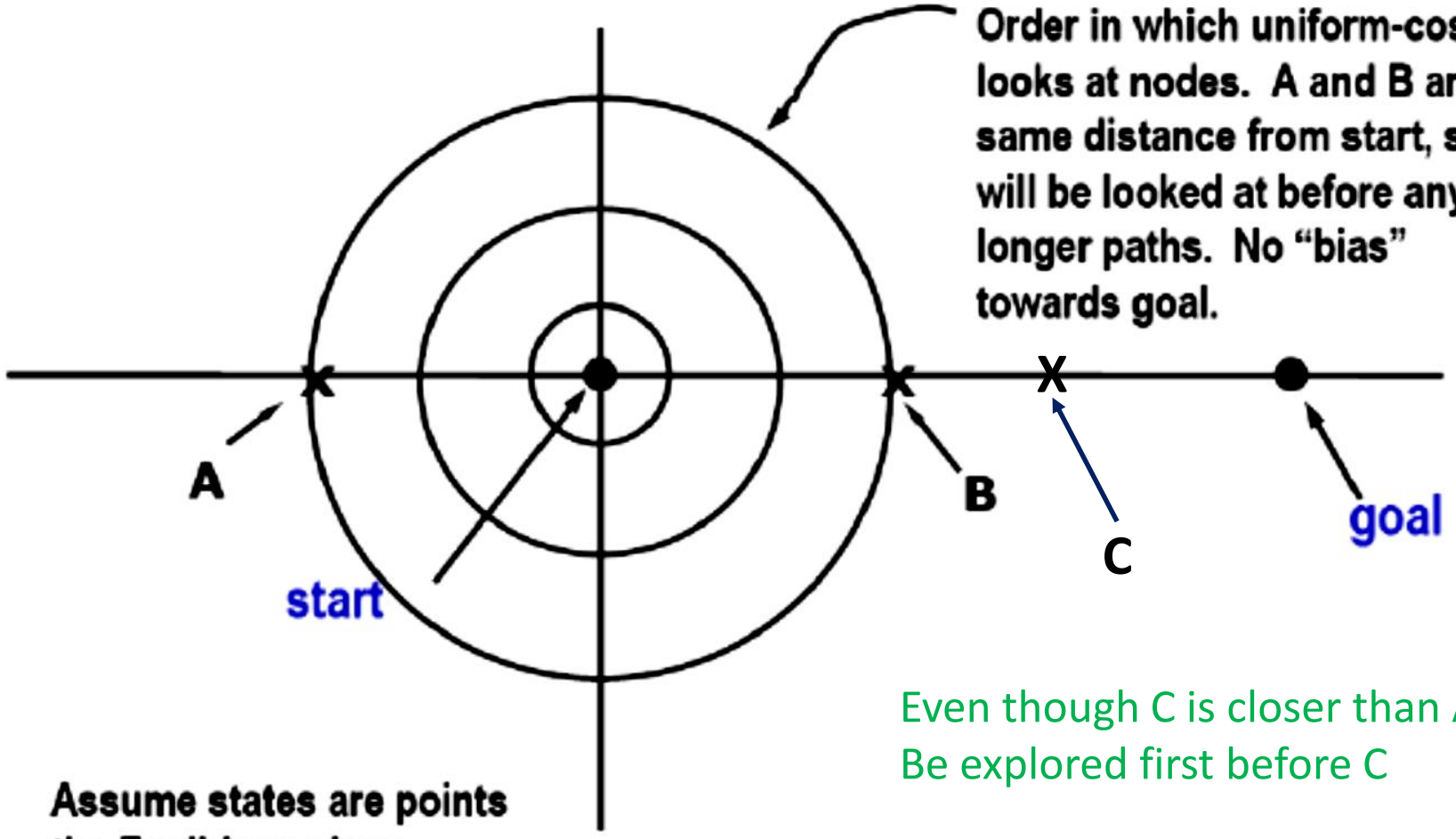
Uniform Cost Issues

- Remember: explores increasing cost contours
- The good: UCS is complete and optimal!
- The bad:
  - Explores options in every "direction"
  - No information about goal location



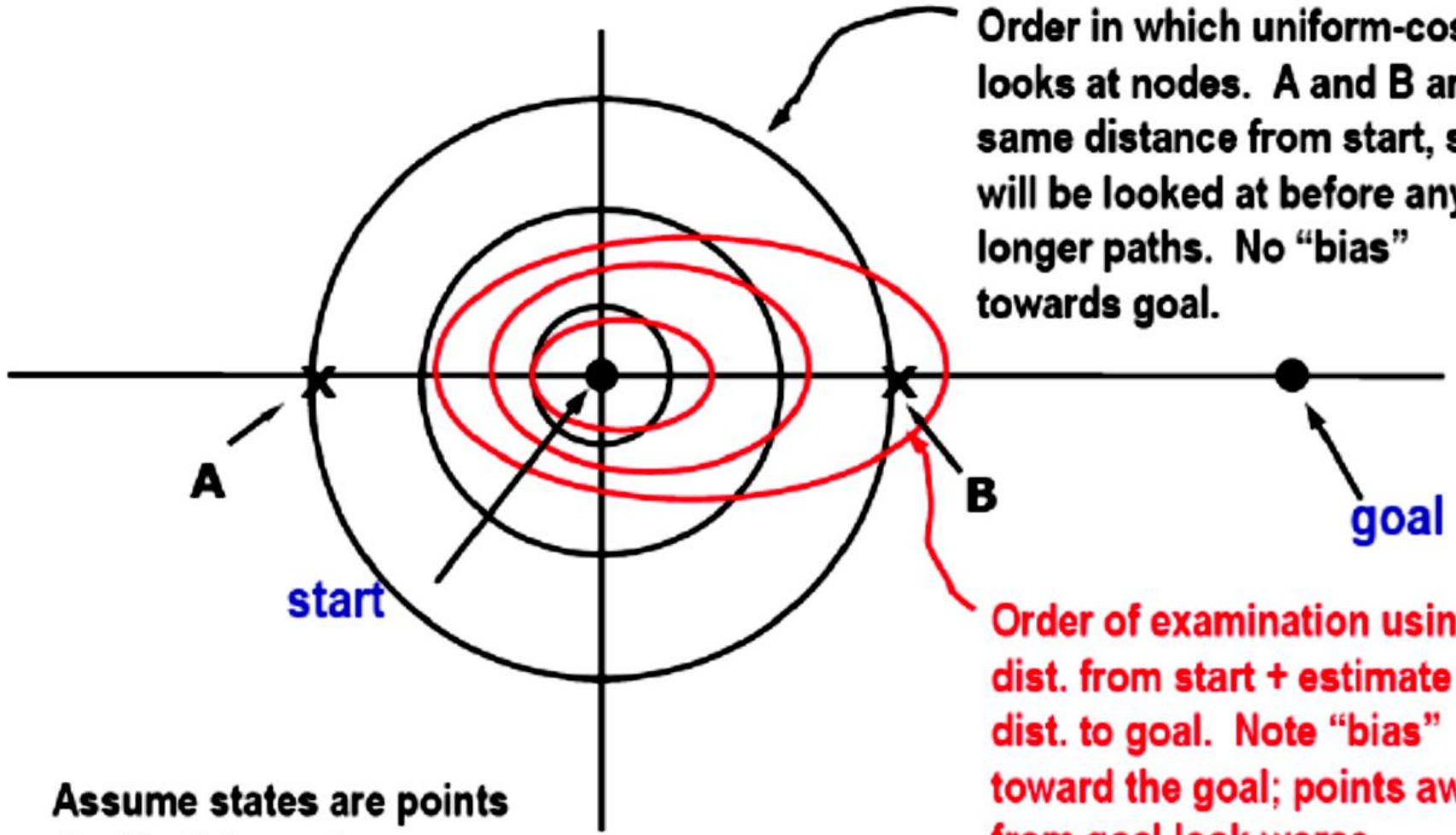


Uniform Cost Issues:



Even though C is closer than A to goal A will Be explored first before C

Why use estimate of goal distance?

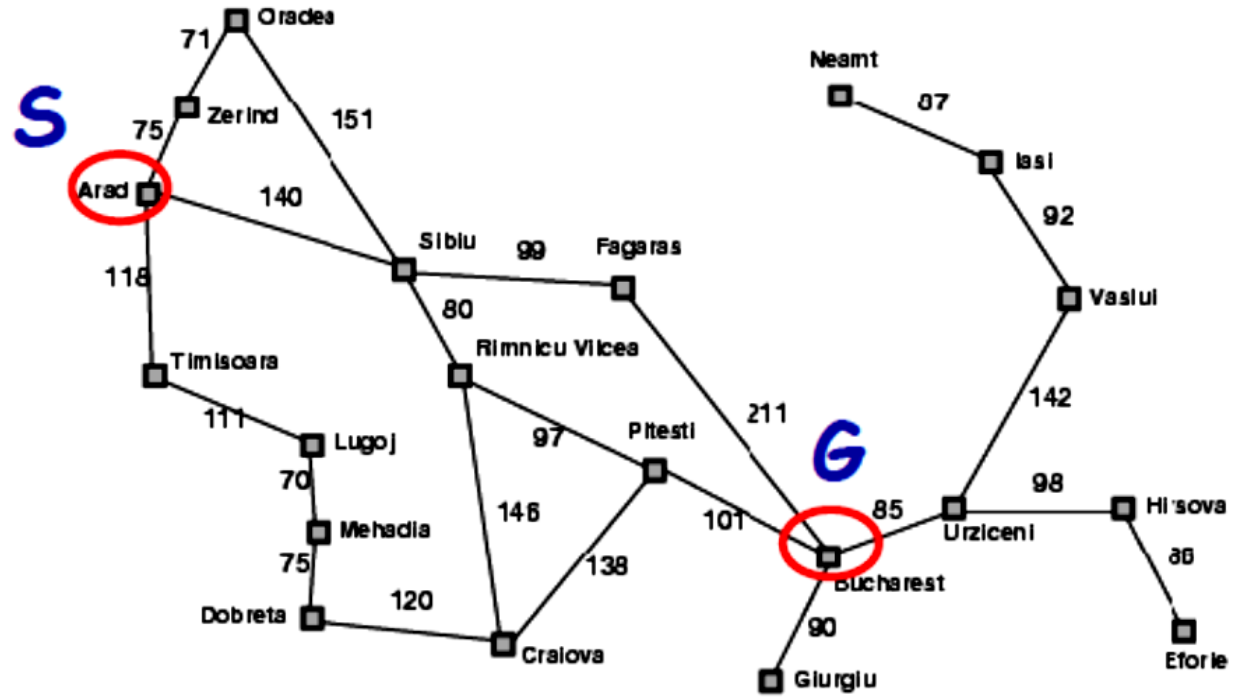


Uninformed vs. Informed Search

- Depth-first, breadth-first and uniform-cost searches are uninformed.
- In informed search there is an estimate available of the cost (distance) from each state (city) to the goal.
- This estimate (heuristic) can help you head in the right direction.
- Heuristic embodied in function h(n), estimate of remaining cost from search node n to the least cost goal.
- Graph being searched is a graph of states. Search algorithm defines a tree of search nodes. Two paths to the same state generate two different search nodes.
- Heuristic could be defined on underlying state; the path to a state does not affect estimate of distance to the goal.

Informed Search

- Sometimes called **Heuristic Search**
- General search problem: Actions have different costs

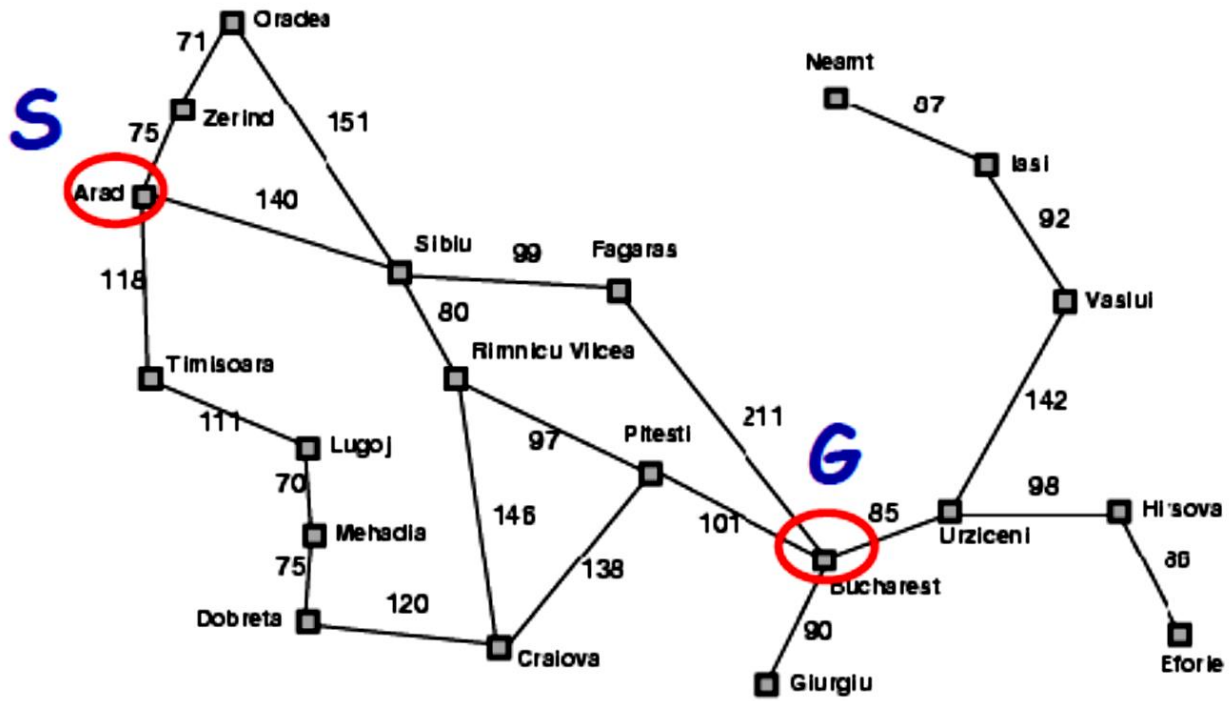


Find least cost solution from S to G

- Idea: Use problem-specific knowledge to guide search by using "heuristic function"

Greedy best-first search

- Use a *heuristic* evaluation function $f(n) = h(n)$ = estimate of cost from n to *goal*



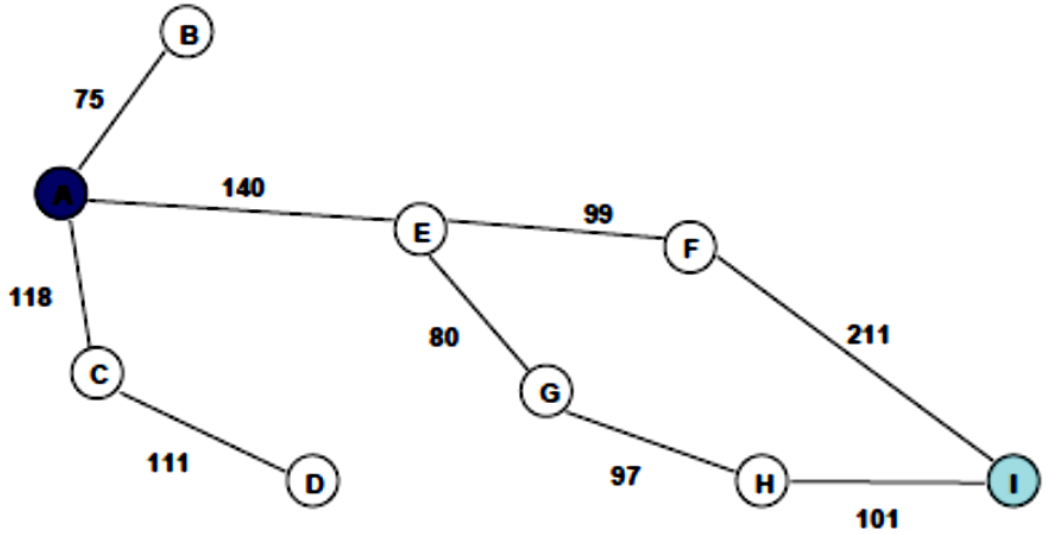
- E.g., $h_{SLD}(n)$ = straight-line distance from n to destination
- Greedy best-first search expands the node that appears to be closest to goal

Best-First Search

- **Greedy Search**

$$f(n) = h(n)$$

State	h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Current State	Available Successors	Selected Successor
A	B (374), C (329), E (253)	E
E	F (178), G (193)	F
F	I (0)	I

Optimal? A-E-F-I = 450 vs. A-E-G-H-I = 418
Complete? False start

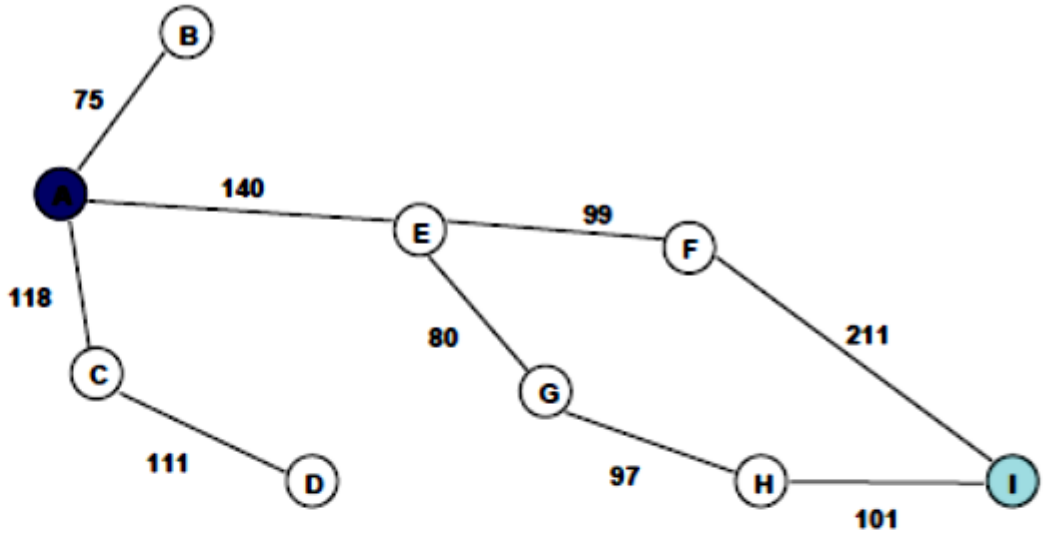
Combining UCS and Greedy

- Uniform-cost orders by path cost, or *backward cost* f(n)=g(n)
- Best-first orders by goal proximity, or *forward cost* f(n)=h(n)
- A* Search orders by the sum: f(n) = g(n) + h(n)

A* Search

- Let g(n) edge costs on path from start state to node n
- Let h(n) = best cost estimate from node n to goal state
- Cost function associated with node n is f(n) = g(n) + h(n)

State	h(n)
A	366
B	374
C	329
D	244
E	253
F	178
G	193
H	98
I	0



Queue

A(366) ←

A-B(449),A-C(447),A-E(393) ←

A-B(449),A-C(447),A-E-F(417),A-E-G(413) ←

A-B(449),A-C(447),A-E-F(417),A-E-G-H(415) ←

A-B(449),A-C(447),A-E-F(417),A-E-G-H-I(418)

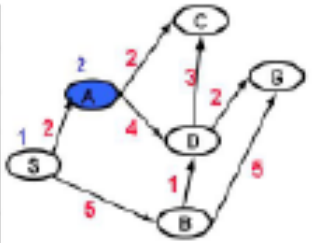
A-B(449),A-C(447),[illegible],A-E-G-H-I(418),A-E-F-I(450) ←

A* Search: Example 2

A*

Pick best (by path length+heuristic) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(4 A S) (8 B S)



Heuristic Values

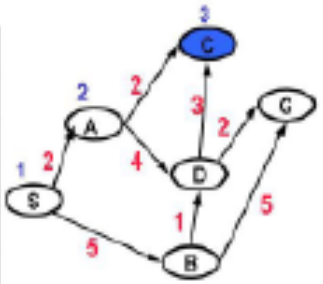
A=2	C=1	S=0
B=3	D=1	G=0

Added paths in blue; underlined paths are chosen for extension.
We show the paths in reversed order; the node's state is the first entry.

A*

Pick best (by path length+heuristic) element of Q; Add path extensions anywhere in Q

	Q
1	(0 S)
2	(4 A S) (8 B S)
3	(5 C A S) (7 D A S) (8 B S)

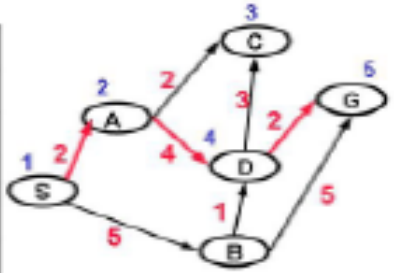


Heuristic Values

A=2	C=1	S=0
B=3	D=1	G=0

Pick best (by path length+heuristic) element of Q; Add path extensions anywhere in Q

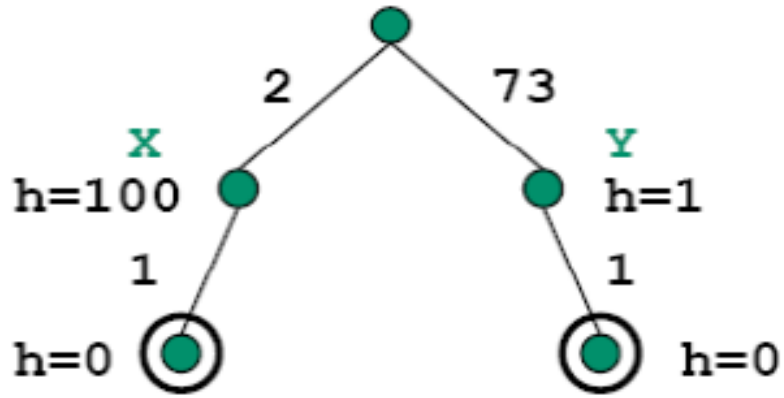
	Q
1	(0 S)
2	(4 A S) (8 B S)
3	(5 C A S) (7 D A S) (8 B S)
4	(7 D A S) (8 B S)
5	(8 G D A S) (10 C D A S) (8 B S)



Heuristic Values

A=2	C=1	S=0
B=3	D=1	G=0

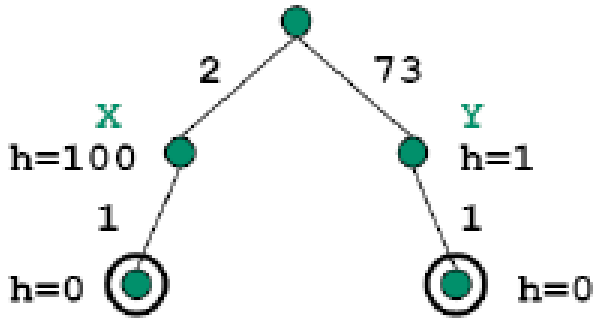
Issues in choosing h(n) values



Running A* on the example will not give optimal solution!

Admissible Heuristic

- What must be true about h for A* to find optimal path?
- A* finds optimal path if h is admissible; h is admissible when it never overestimates.
- In this example, h is not admissible.



g(X)+h(X) = 102

g(Y)+h(Y) = 74

Optimal path is not found!

Admissible Heuristics

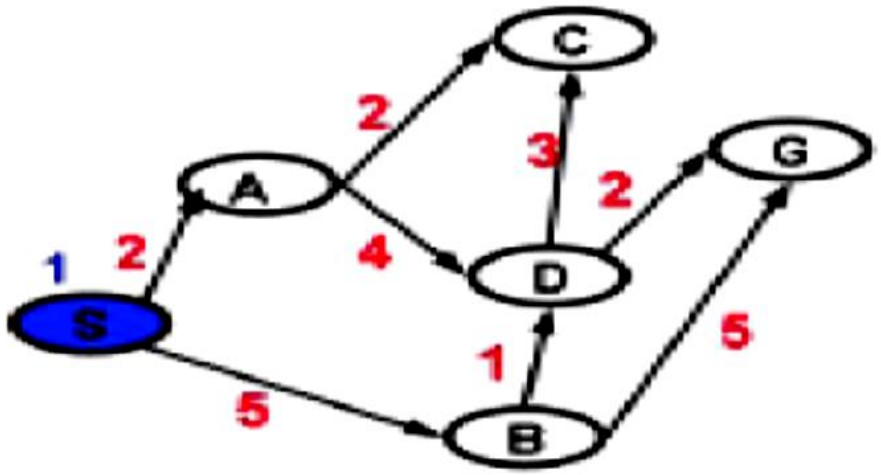
- A heuristic ***h(n)*** is **admissible** if for every node n,

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

  where $h^*(n)$ is the actual cost of the least cost path to reach the goal state from *n*.

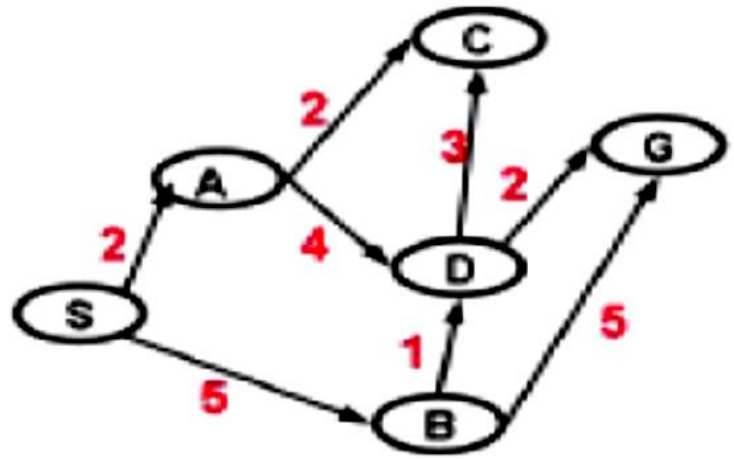
- An admissible heuristic never overestimates the cost to reach the goal, i.e., "it is very conservative and plays it safe"

Are these admissible?



Heuristic Values

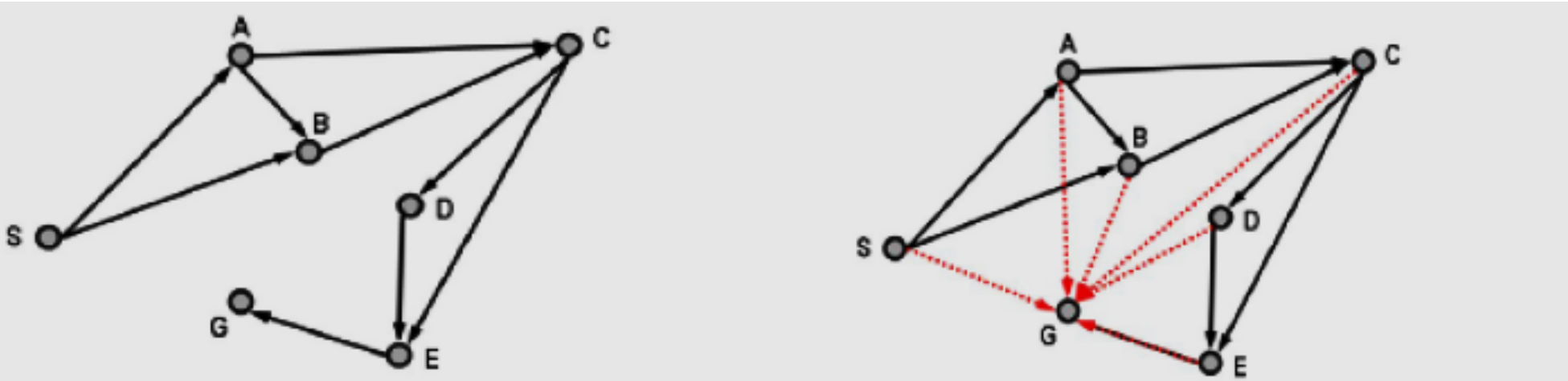
A=2	C=1	S=0
B=3	D=1	G=0



Heuristic Values

A=2	C=1	S=10
B=3	D=4	G=0

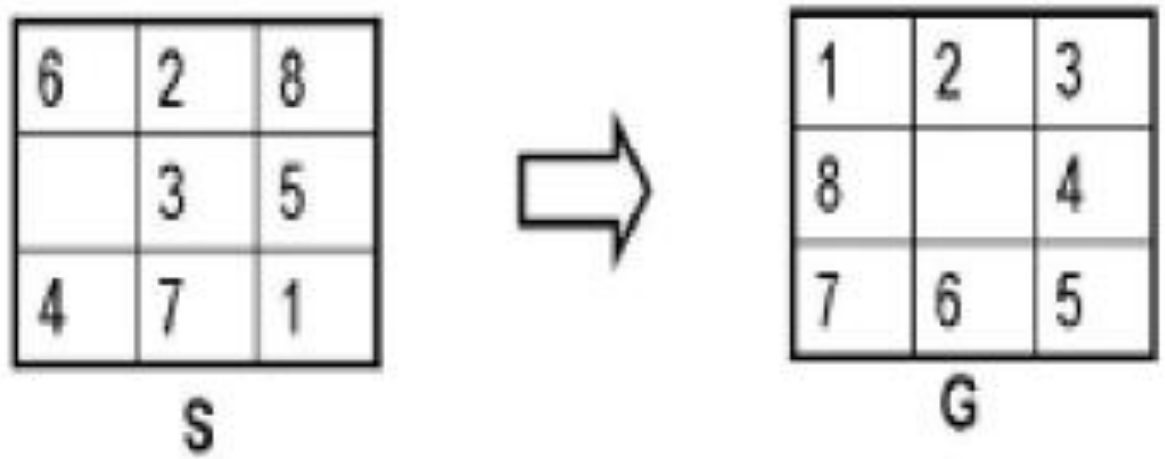
Admissible Heuristics: Straight-Line Estimate



We can use the straight-line distance (shown in red) as an underestimate of the actual driving distance between any city and the goal. The best possible distance between the cities cannot be better than the straight-line distance. But it can be worse.

Admissible Heuristics

8 Puzzle: Move tiles to reach goal. Think of a move as moving "empty" tile.



Alternative underestimates of "distance" (number of moves) to goal:

1. Number of misplaced tiles (7 in example above)

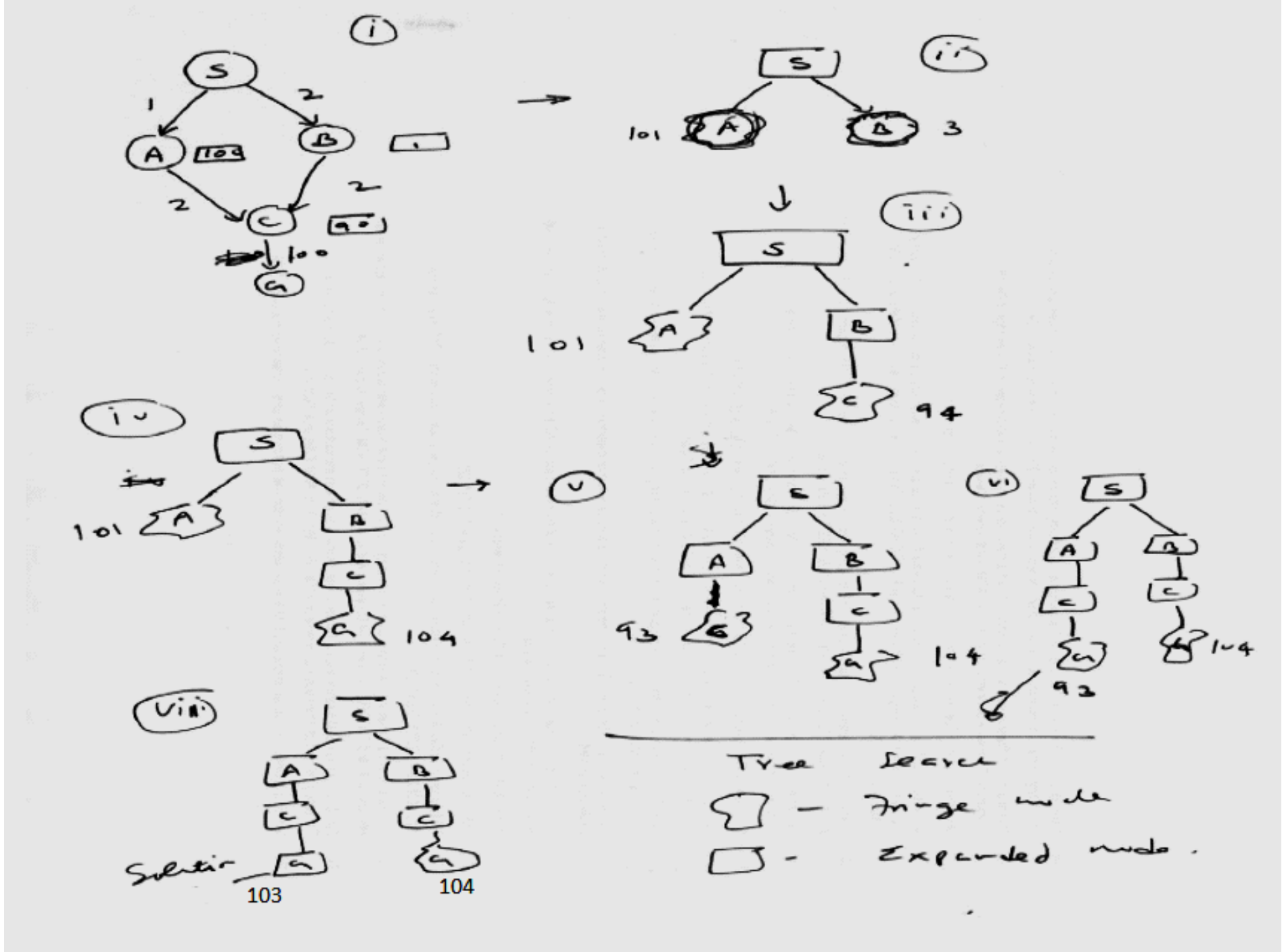
Admissible Heuristics :

8 Puzzle: Move tiles to reach goal. Think of a move as moving "empty" tile.

Alternative underestimates of "distance" (number of moves) to goal:

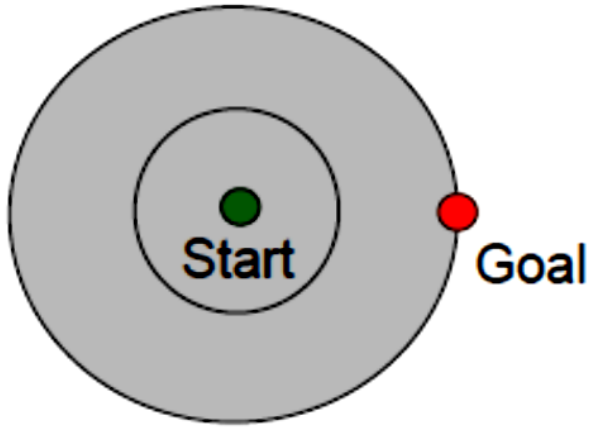
1. **Number of misplaced tiles (7 in example above)**
2. **Sum of Manhattan distance of tile to its goal location (17 in example above). Manhattan distance between (x_1,y_1) and (x_2,y_2) is $|x_1-x_2|+|y_1-y_2|$ Each move can only decrease the distance of exactly one tile.**

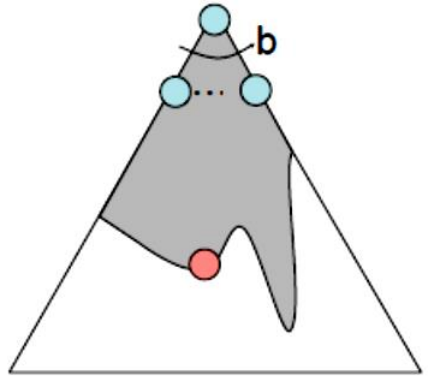
The second of these is much better at predicting actual number of moves.


i
S
1
2
A
104
B
1
2
2
C
90
100
G
ii
S
101
A
B
3
iii
S
101
A
B
C
94
iv
S
101
A
B
C
G
104
v
C
A
B
93
G
C
G
vi
S
A
G
C
C
104
93
Solution
103
104
Tree Search
- Fringe node
- Expanded node.

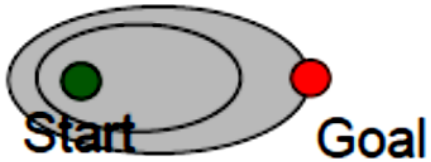
UCS vs A* Contours

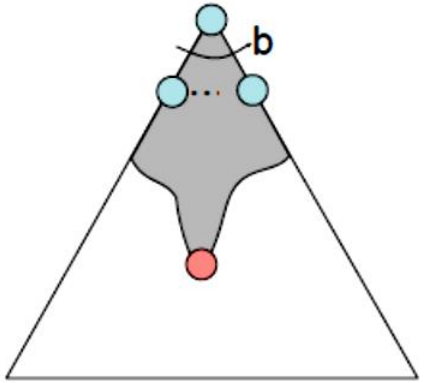
- Uniform-cost expanded in all directions





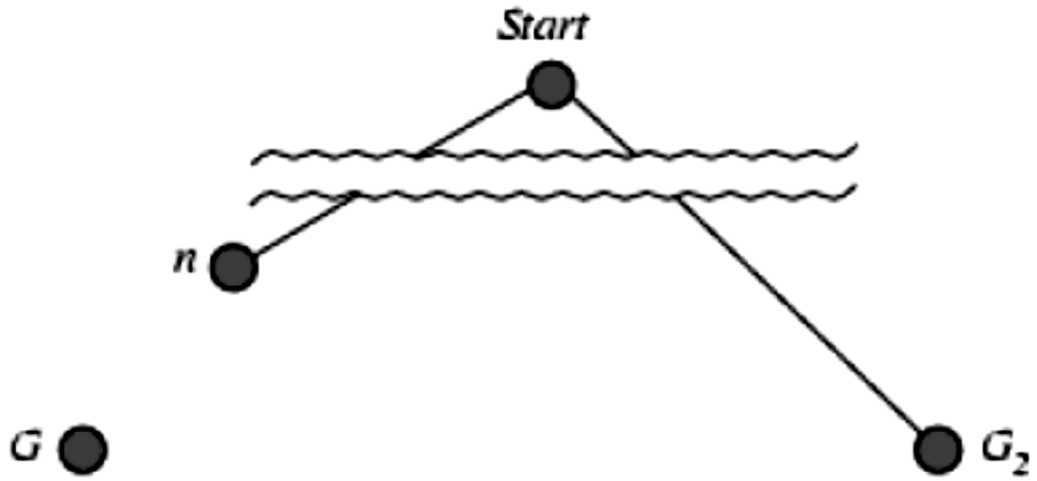
- A* expands mainly toward the goal, but does hedge its bets to ensure optimality





Optimality Proof of A* Search

Suppose some suboptimal goal G2 has been generated and is in the frontier. Let n be an unexpanded node in the frontier such that n is on a shortest path to an optimal goal G.



$f(G2) = g(G2)$ since $h(G2) = 0$

$> g(G)$ since G2 is suboptimal

$f(G) = g(G)$ since $h(G) = 0$

$f(G2) > f(G)$ from above

$f(G) < f(G2)$

$h(n) \leq h^*(n)$ since h is admissible

$g(n) + h(n) \leq g(n) + h^*(n) = f(G)$

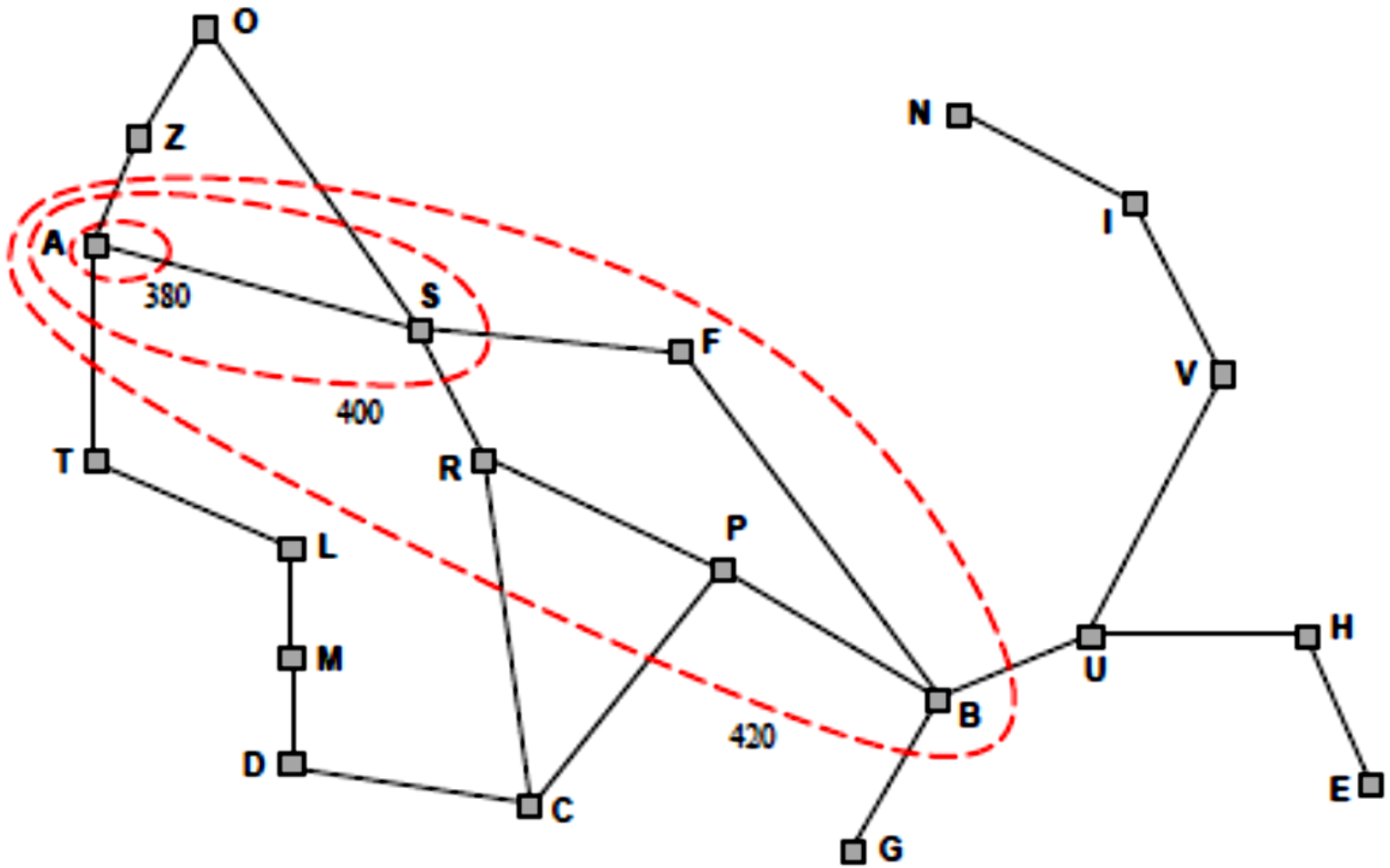
$f(n) \leq f(G) < f(G2)$

Hence $f(n) < f(G2)$

Therefore A* will never select G2 for expansion.

Lemma: A* expands nodes in order of increasing f value*

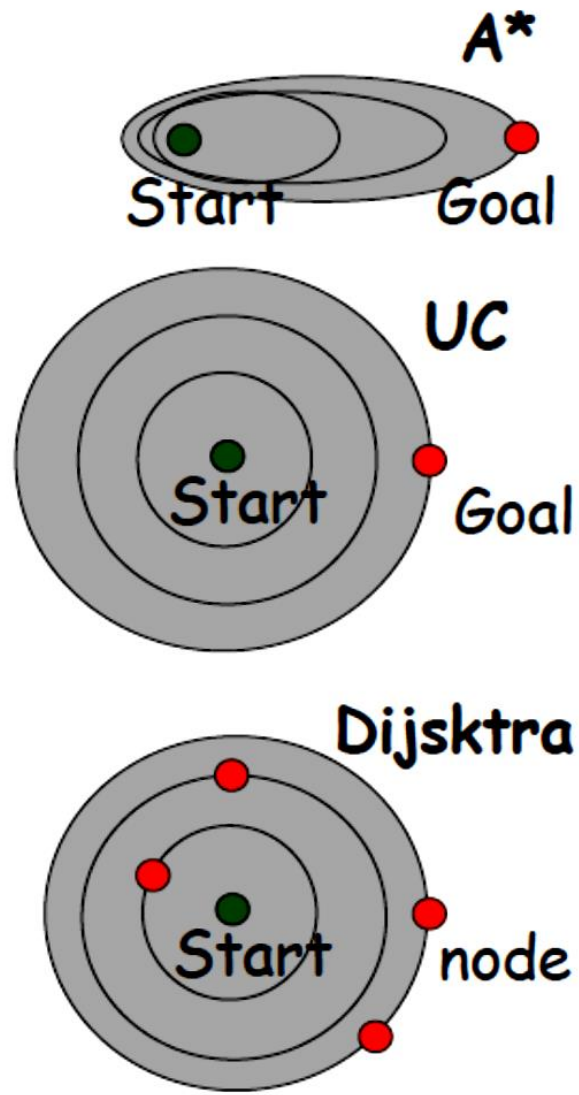
Gradually adds "f-contours" of nodes (cf. breadth-first adds layers)
Contour i has all nodes with $f = f_i$, where $f_i < f_{i+1}$



A* expands all nodes with $f(n) < C^*$
A* expands some nodes with $f(n) = C^*$
A* expands no nodes with $f(n) > C^*$

A* vs. Uniform Cost Search vs. Dijkstra

- A* expands mainly toward the goal with the help of the heuristic function
- Uniform-cost and Dijkstra expand in all directions
- A* can be more efficient if the heuristic is good



Strategy for creating Admissible Heuristics

Relaxed Problem:

- A problem with fewer restrictions on the actions is called a relaxed problem
- The cost of an optimal solution to a relaxed problem is an admissible heuristic for the original problem

Relaxed Problems

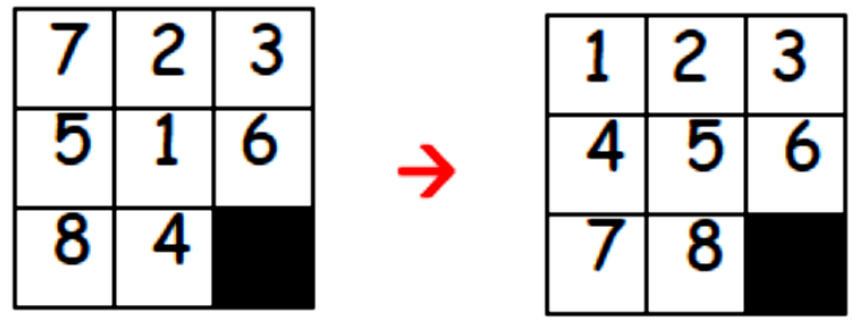
- Derive admissible heuristic from **exact** cost of a solution to a **relaxed** version of problem

For route planning, what is a relaxed problem?

Relax requirement that car has to stay on road

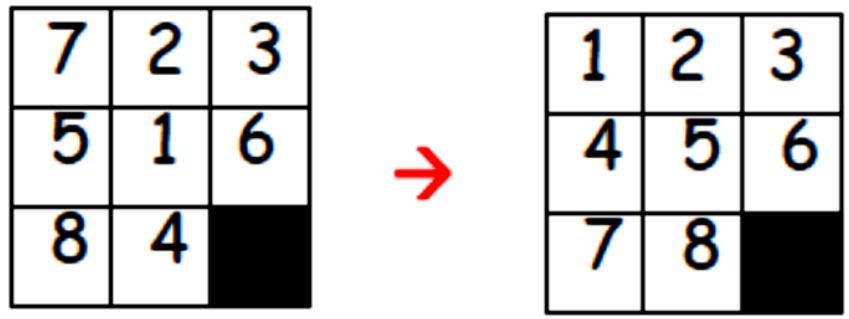
→ Straight Line Distance becomes optimal cost

Heuristics for eight puzzle



- What can we relax?

Heuristics for eight puzzle



Original: Tile can move from location A to B if A is horizontally or vertically next to B *and* B is blank

Relaxed 1: Tile can move from any loc A to any loc B

Cost = h_1 = number of misplaced tiles

Relaxed 2: Tile can move from loc A to loc B if A is horizontally or vertically next to B

Cost = h_2 = total Manhattan distance

State space graph of relaxed problem is a supergraph of the original state space graph – Why?

Removal of restrictions causes additional edges to be added to the original state space graph

Optimal solution to the original problem is also optimal solution for the relaxed version of the problem.

But the relaxed version may provide better solutions if the added edges provide shortcuts.

Hence cost of optimal solution to relaxed version is an admissible heuristic for the original problem.

Key: Solutions to these relaxed problems can be computed without search and therefore provide a heuristic that is easy/fast to compute.

Comparing Heuristics

- We have 3 heuristics for the 8-Puzzle Problem
  - h_0 is 0
  - h_1 is Number of Displaced Tiles
  - h_2 is Manhattan Distance

  Which one should we pick?

Dominance

- If $h_2(n) \geq h_1(n)$ for all n (both admissible) then h_2 dominates h_1
- h_2 is better for search (why?)

  Getting closer to the actual cost to goal

- Does one dominate the other for:

  $h_1(n)$ = number of misplaced tiles

  $h_2(n)$ = total Manhattan distance

Implications of a Better or More Informed Heuristic

- **Theorem:**

If A_1^* uses h_1 and A_2^* uses h_2, then every node expanded by A_2^* is also expanded by A_1^*

Recall:

A* expands all nodes with $f(n) < C^*$
A* expands some nodes with $f(n) = C^*$
A* expands no nodes with $f(n) > C^*$

A* expands all nodes $h_1(n)+g(n) < C^*$ - (1)
A* expands all nodes $h_2(n)+g(n) < C^*$ - (2)

But $h_1(n) \leq h_2(n)$
So if a node satisfies (2) it also satisfies (1)
Hence all nodes expanded by A* with h_2 will also be expanded by A* with h_1

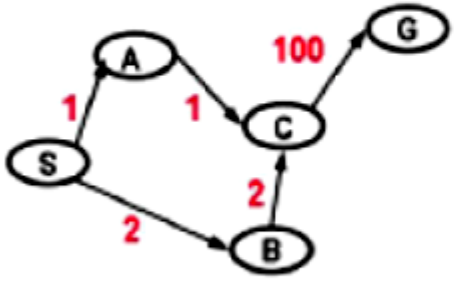
Dominance

- For 8-puzzle heuristics h_1 and h_2, typical search costs (average number of nodes expanded for solution depth d):

- *d=12* IDS = 3,644,035 nodes
  $A^*(h_1)$ = 227 nodes
  $A^*(h_2)$ = 73 nodes
- *d=24* IDS = too many nodes to fit in memory
  $A^*(h_1)$ = 39,135 nodes
  $A^*(h_2)$ = 1,641 nodes

Implications on A* Graph Search

Example Illustrating Suboptimal solution for graph search (with Closed/Expanded List)

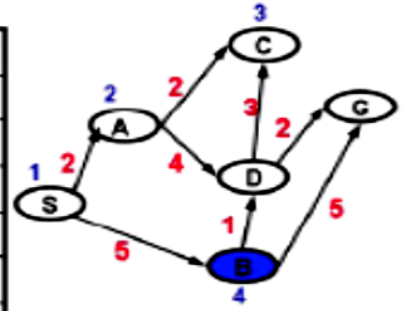


	Q	Expanded
1	(0 S)	
2	(3 B S) (101 A S)	S
3	(94 C B S) (101 A S)	B, S
4	(101 A S) (104 G C B S)	C, B, S
5	(104 G C B S)	A, C, B, S

Why: Because f(n) does not monotonically increase on a path

A* - Graph Search

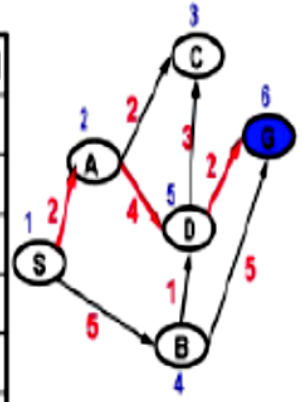
	Q	Expanded
1	(0 S)	
2	(2 A S) (5 B S)	S
3	(4 C A S) (6 D A S) (5 B S)	S,A
4	(6 D A S) (5 B S)	S,A,C

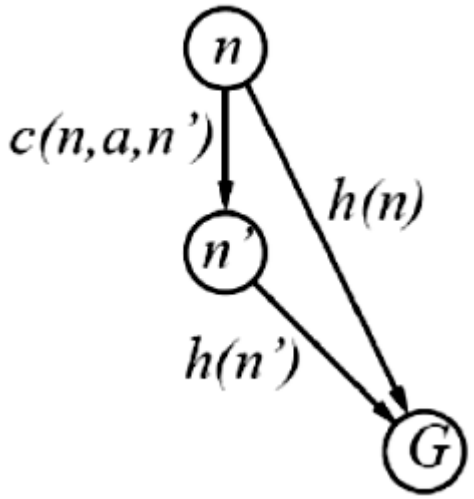


In contrast f(n) which is always g(n) for uniform cost search only grows monotonically. Why?

Uniform cost search

	Q	Expanded
1	(0 S)	
2	(2 A S) (5 B S)	S
3	(4 C A S) (6 D A S) (5 B S)	S,A
4	(6 D A S) (5 B S)	S,A,C
5	~~(6 D B S)~~ (10 G B S) (6 D A S)	S,A,C,B
6	(8 G D A S) ~~(9 C D A S)~~ ~~(10 G B S)~~	S,A,C,B,D





We have:

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

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

Now $g(n') = c(n,a,n') + g(n)$ - (a is an action)

We want:

$f(n') \geq f(n)$

i.e. $c(n,a,n') + g(n) + h(n') \geq g(n) + h(n)$

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

Consistency Condition

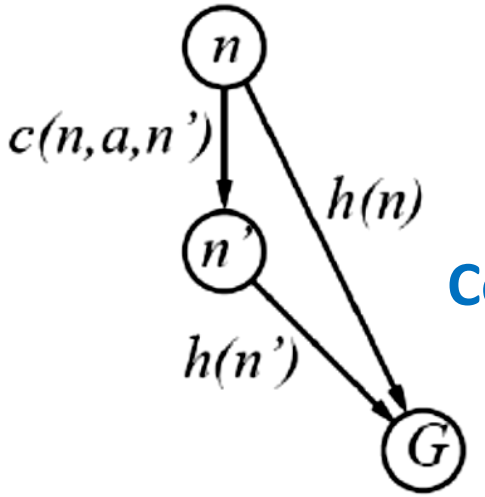
A heuristic is consistent if

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

If h is consistent, we have

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= g(n) + c(n, a, n') + h(n') \\ &\geq g(n) + h(n) \\ &= f(n) \end{aligned}$$

I.e., $f(n)$ is nondecreasing along any path.



Consistency Condition: Triangle Inequality

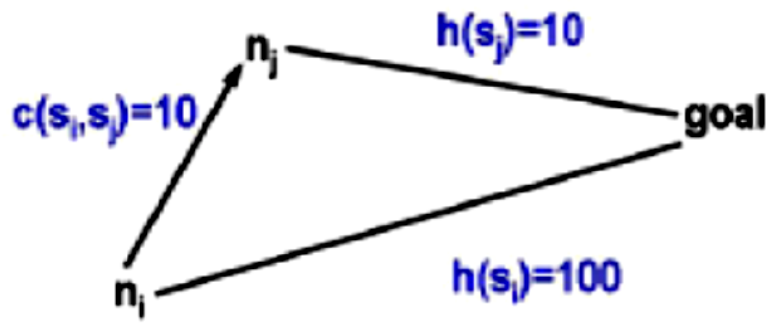
In addition to Admissible we have to impose Consistency to guarantee Optimal solution for A* graph search.

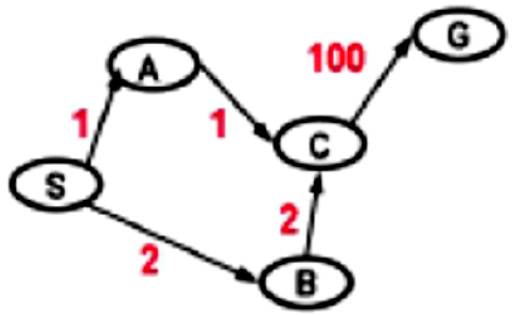
Note they are sufficient conditions

Consistent Heuristic

Consistency Violation

- A simple example of a violation of consistency.
- $h(s_i) - h(s_j) \cdot c(s_i,s_j)$
- In example, 100-10 > 10
- If you believe goal is 100 units from n_i, then moving 10 units to n_j should not bring you to a distance of 10 units from the goal.

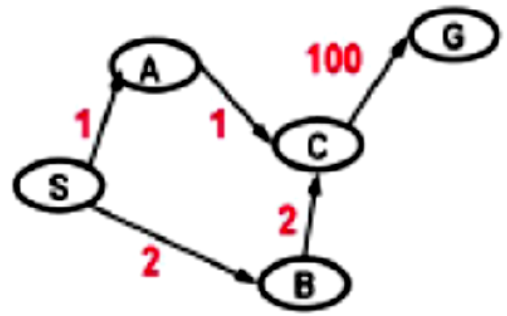




Heuristic Values

A=100	C=90	S=0
B=1		G=0

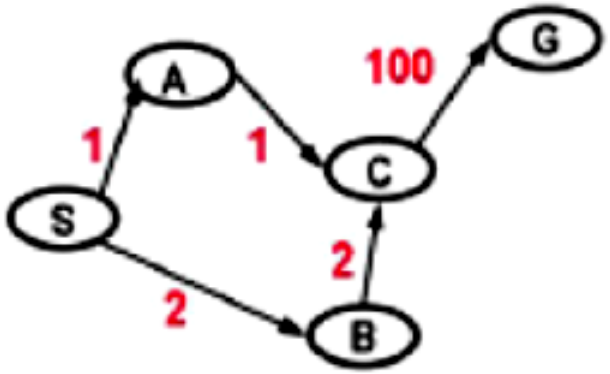
Not Consistent



Heuristic Values

A=100	C=100	S=90
B=88		G=0

Consistent

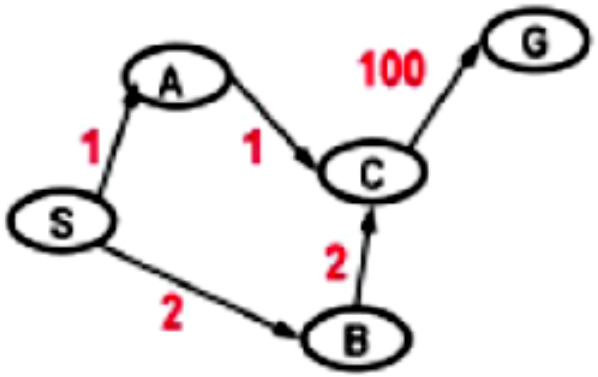


Heuristic Values

A=100 C=90 S=0

B=1 G=0

Not Consistent

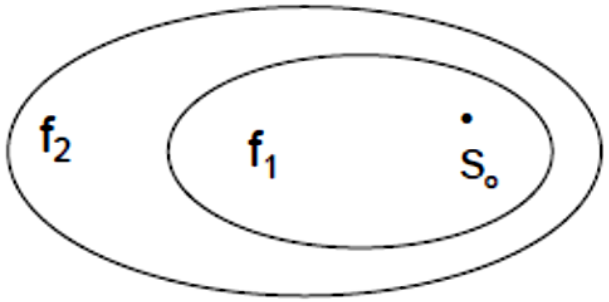


Heuristic Values

A=100 C=100 S=90

B=88 G=0

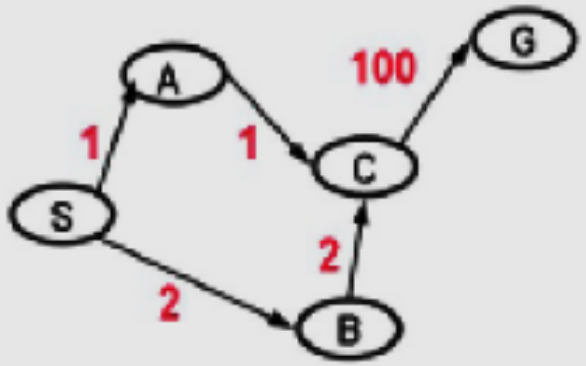
Consistent



f_1 values < f_2 values

During A* search: **Nodes expanded in the order of increasing f values**

	Q	Expanded
1	(90 S)	
2	(90 B S) (101 A S)	S
3	(101 A S) (104 C B S)	B,S
4	(102 C A S) (~~104 C B S~~)	AB,S
5	(102 G C A S)	C,A,B,S



Heuristic Values

A=100 C=100 S=90

B=88 G=0

Example of A* Graph Search with Admissible+Consistent Heuristic

Summary

Uninformed search:

(1) Breadth-first search

(2) (2) Uniform-cost search

(3) Depth-first search

(4) Depth-limited search

(5) Iterative deepening search

Informed search:

(1) Greedy Best-First

(2) A*

Lots of variants of A*. Research in A* has increased dramatically since A* is the key algorithm used by map engines.

Also used in path planning algorithms (autonomous vehicles), and general (robotics) planning, and many other applications