

CS7IS2: Artificial Intelligence

Lecture 2: Informed search

Ivana.Dusparic@scss.tcd.ie

Uninformed search: summary

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes	No	No	Yes*

Informed search

- › Estimate “desirability” of each node in the frontier and expand the most desirable unexpanded node
- › Heuristic search – estimate of the path cost from each node
 - $h(n)$ – heuristic function
 - Vs uninformed search = blind/exhaustive exploration of the full search space
- › Same as with uninformed searches – informed search algorithms only differ by what node do they choose to expand next

Best-first search

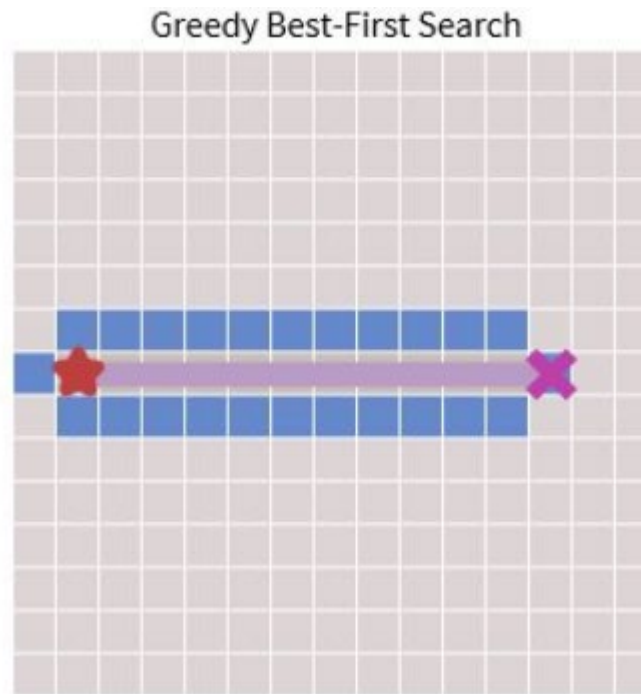
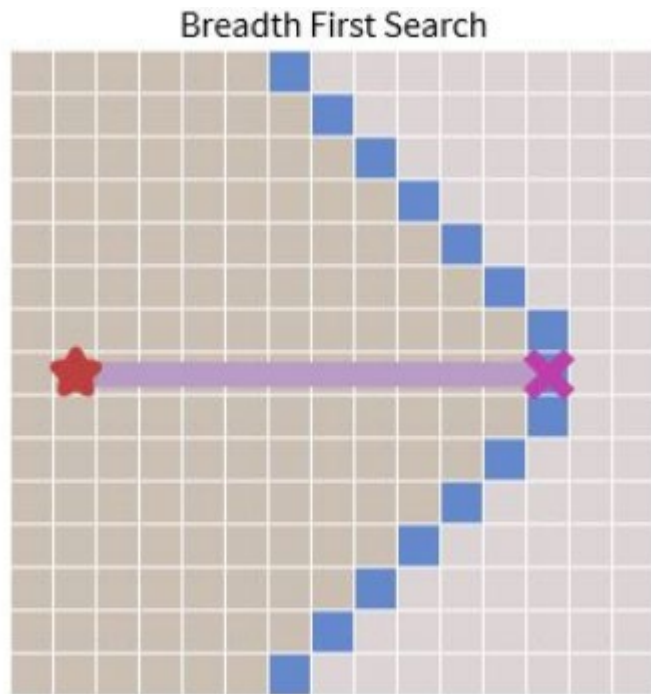
- › Node selected for expansion based on a function $f(n)$
- › $f(n)$ is a cost estimate, and the nodes with the lowest $f(n)$ are expanded first
- › Choice of $f(n)$ = search strategy
- › Special-cases: greedy search and A^* search

Greedy best-first search (GBFS)

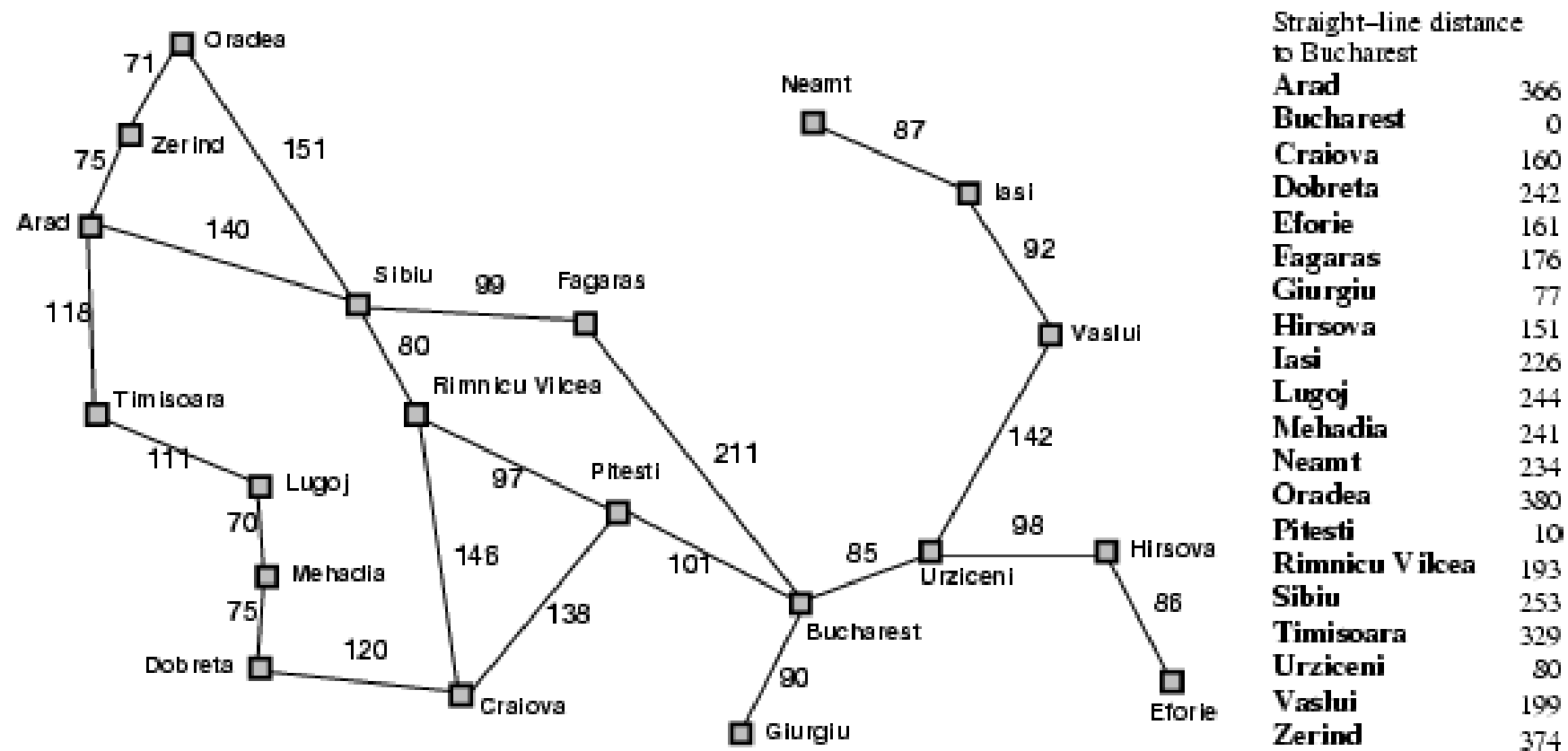
- › Expand the node **that appears to be closest** to the goal, under assumption that it is the closest to the goal
- › $f(n) = h(n)$

Greedy best-first search

- › Each step moving closer to the target, e.g.,



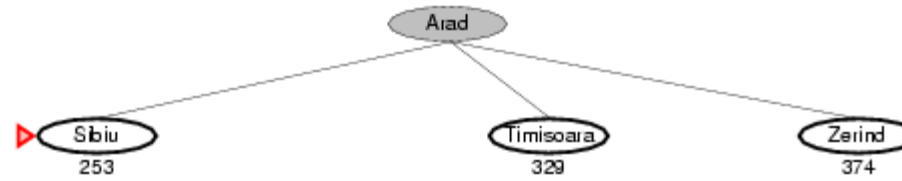
Greedy best-first search – Romania example



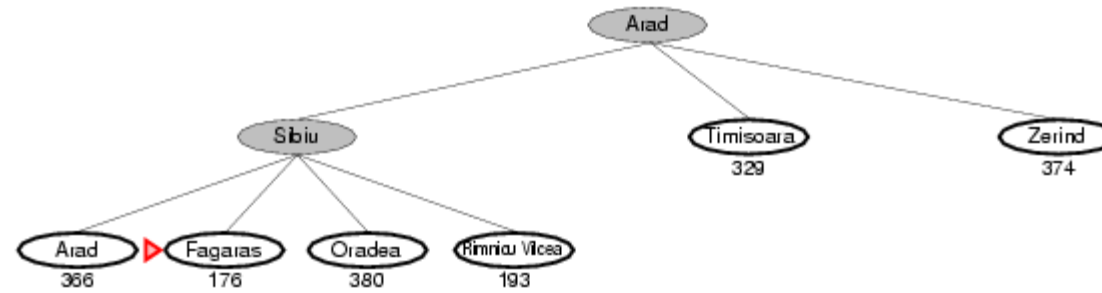
Greedy best-first search example



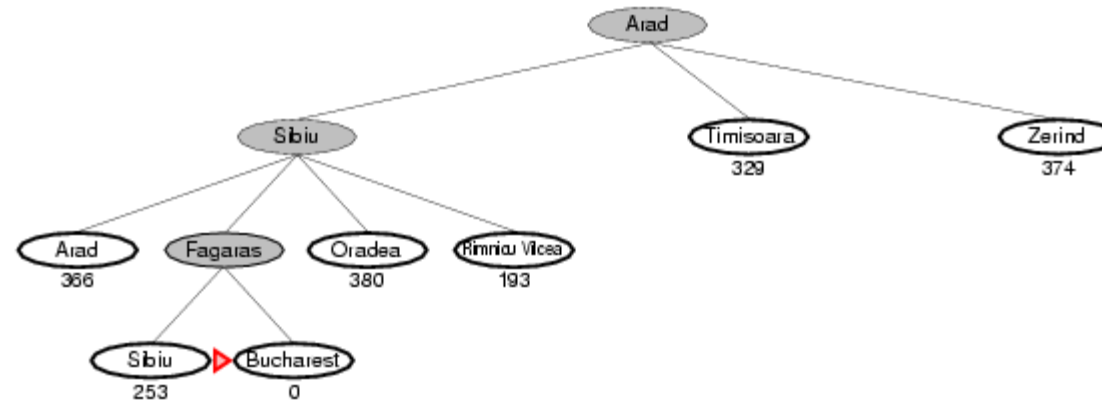
Greedy best-first search example



Greedy best-first search example



Greedy best-first search example



Properties of greedy best-first search

Complete?

- No, can get stuck in loops, e.g., lasi \rightarrow Neamt \rightarrow lasi \rightarrow Neamt \rightarrow

Complete in finite space with repeated-state checking

Time?

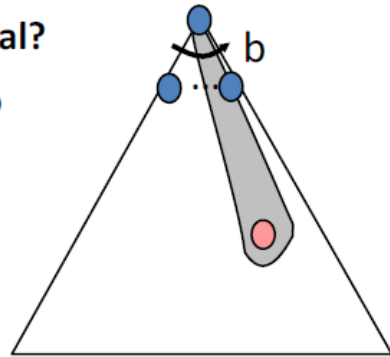
- $O(b^m)$, but a good heuristic can give dramatic improvement

Space?

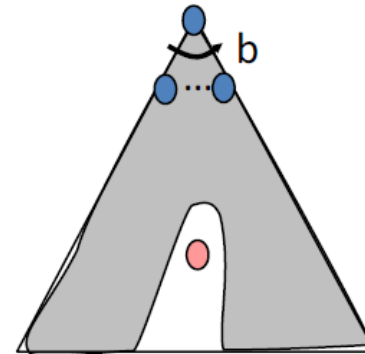
- $O(b^m)$, keeps all nodes in memory

Optimal?

- No



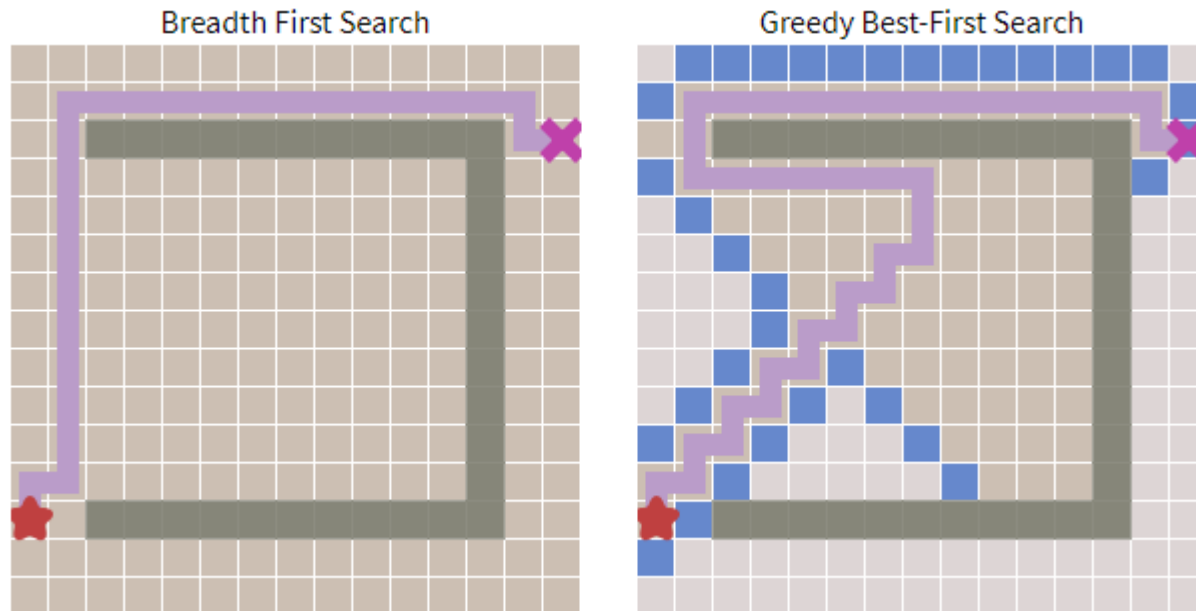
Common case: takes you straight to the (wrong) goal



Worst case: badly-guided DFS

Optimality

- › Demo of BFS vs Greedy best-first search
- › <https://cs.stanford.edu/people/abisee/tutorial/greedy.html>



Uniform-cost search (UCS): revise

Expand least-cost unexpanded node

- $g(n)$ cost function for a given path to the node n

Implementation:

- **Fringe** = **priority queue** ordered by path cost, lowest first

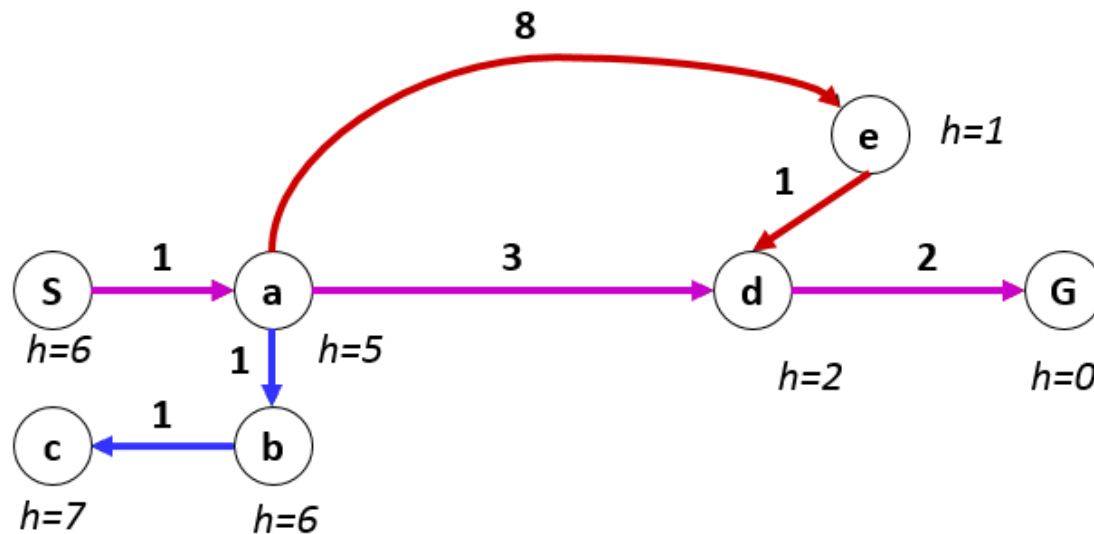
Goal test is applied to a node when it is selected for expansion

A test is added in case a better path is found to a node currently on the frontier

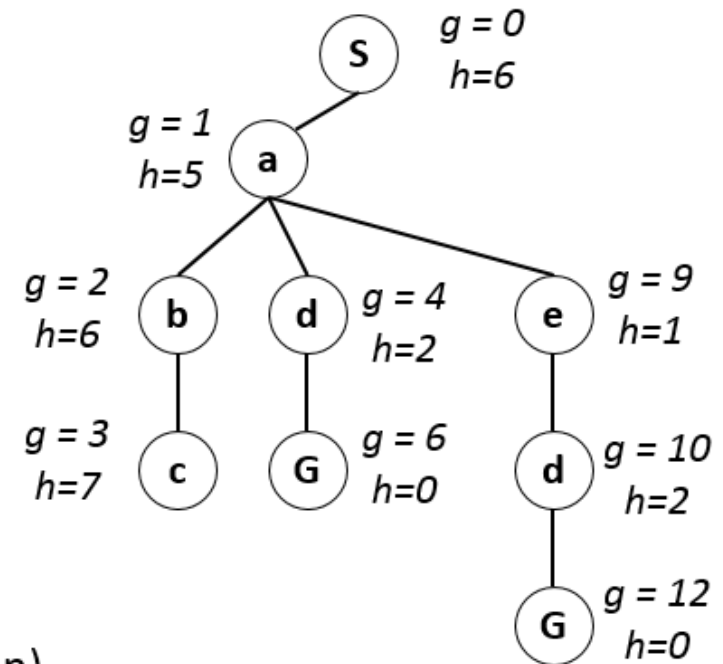
Equivalent to breadth-first if step costs all equal

A* - Combine UCS with GBFS

- **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: $f(n) = g(n) + h(n)$



Example: Teg Grenager

A^*

› actual cost of path so far + estimated cost to goal

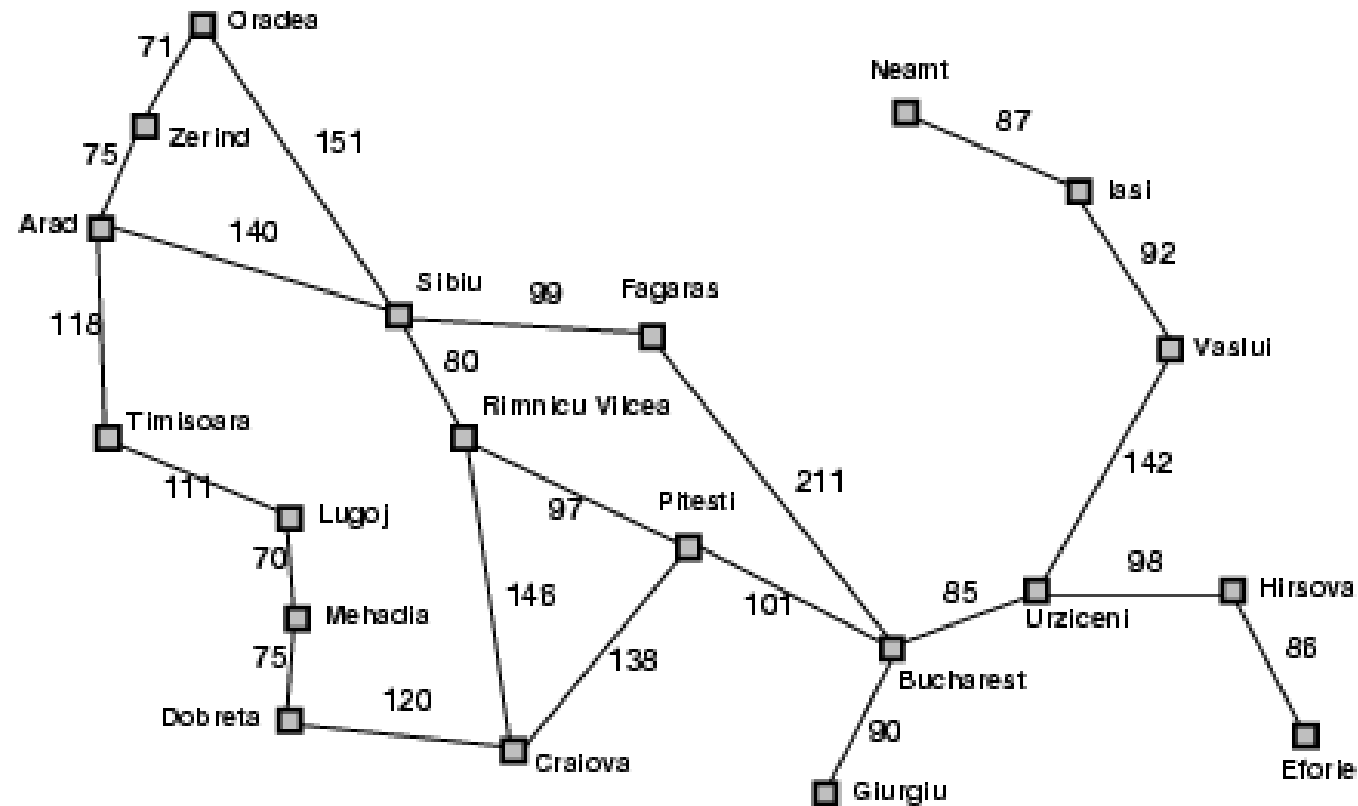
↓

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

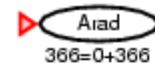


- › This helps avoid local maxima traps
- › May be slower than Greedy Best-First
 - But guarantees shortest path
- › Main idea: avoid paths that have already been expensive

A* – Romania example



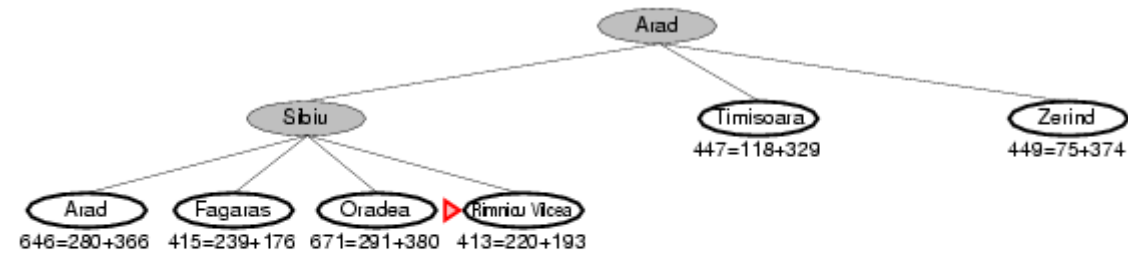
A* search example



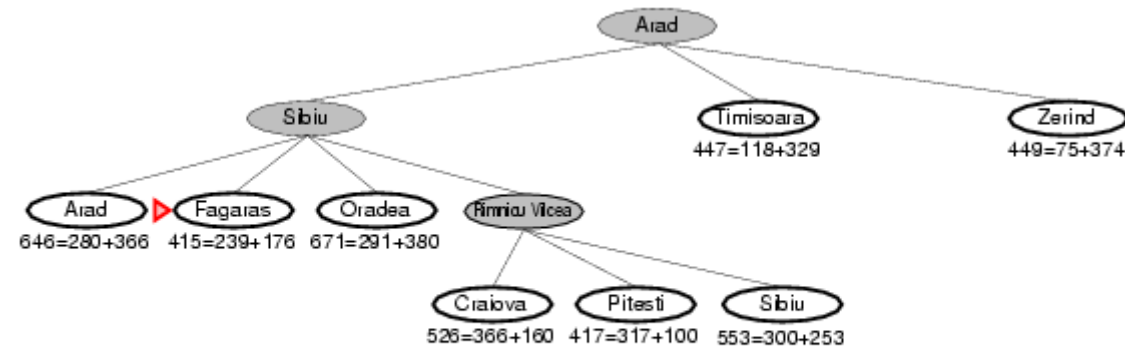
A* search example



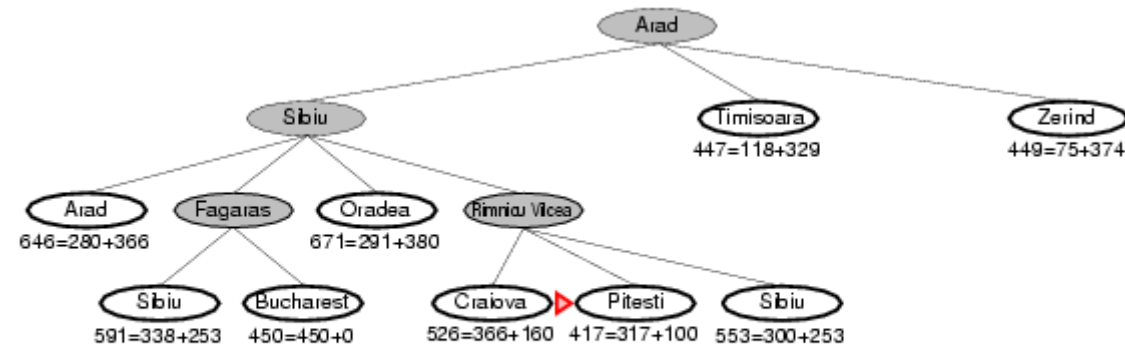
A* search example



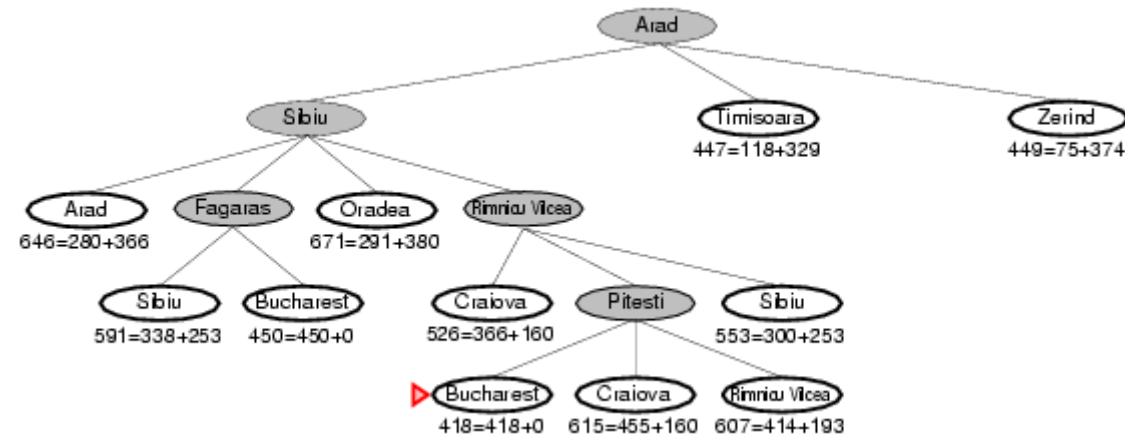
A* search example



A* search example

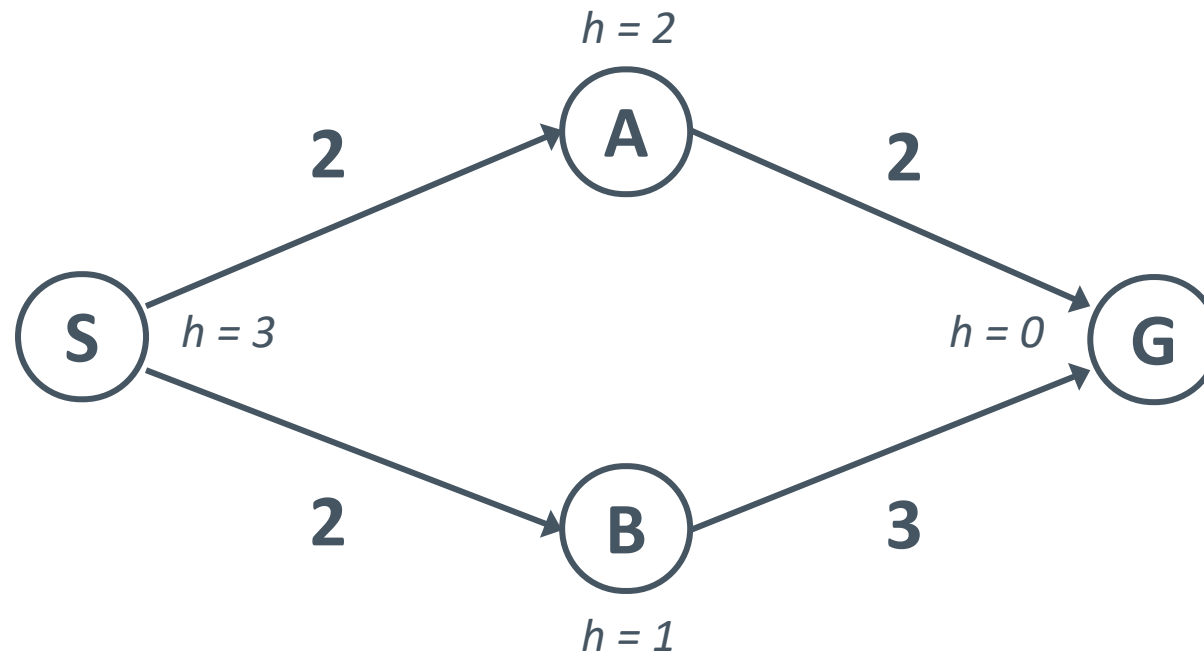


A* search example



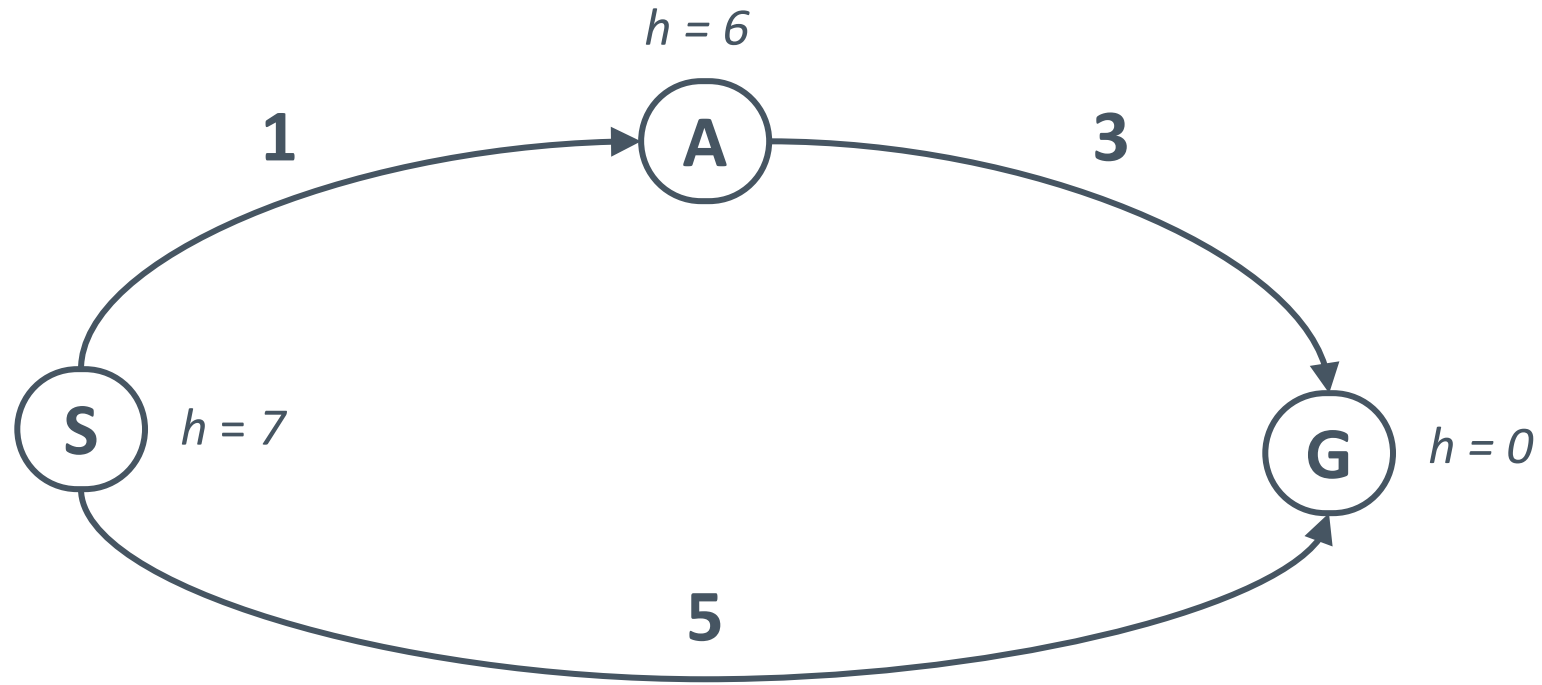
When should A* terminate?

- › 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 in order for A* to be optimal

Admissible heuristics

- › A heuristic $h(n)$ is **admissible** if for every node n ,
 $h(n) \leq h^*(n)$, where $h^*(n)$ is the **true** cost to reach the goal state from n .
- › An admissible heuristic **never overestimates** the cost to reach the goal, i.e., it is **optimistic**
- › Example: *Romania example straight line distance* - never overestimates the actual road distance
- › **Theorem**: If $h(n)$ is admissible, A^* using TREE-SEARCH is optimal

Admissible heuristics

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

Properties of A^*

Complete?

- Yes, unless there are infinitely many nodes with $f \leq f(G)$

Time?

- Exponential in [relative error in h length of soln.]

Space?

- Keeps all nodes in memory

Optimal?

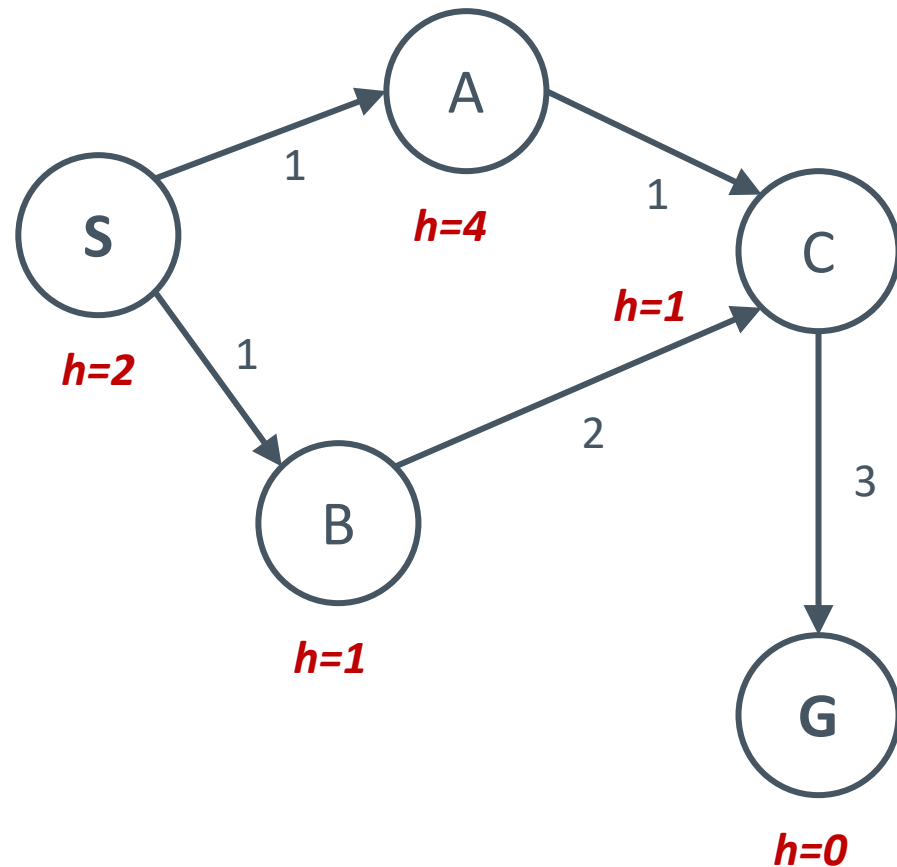
- Yes, cannot expand f_{i+1} until f_i is finished

A* applications

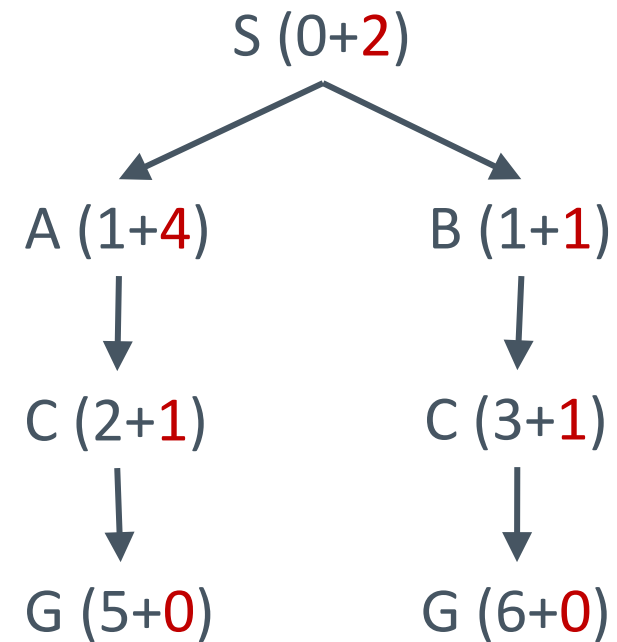
- › Video games
- › Pathing / routing problems
- › Resource planning problems
- › Robot motion planning
- › Language analysis
- › Machine translation
- › Speech recognition

A* Graph Search Gone Wrong?

State space graph

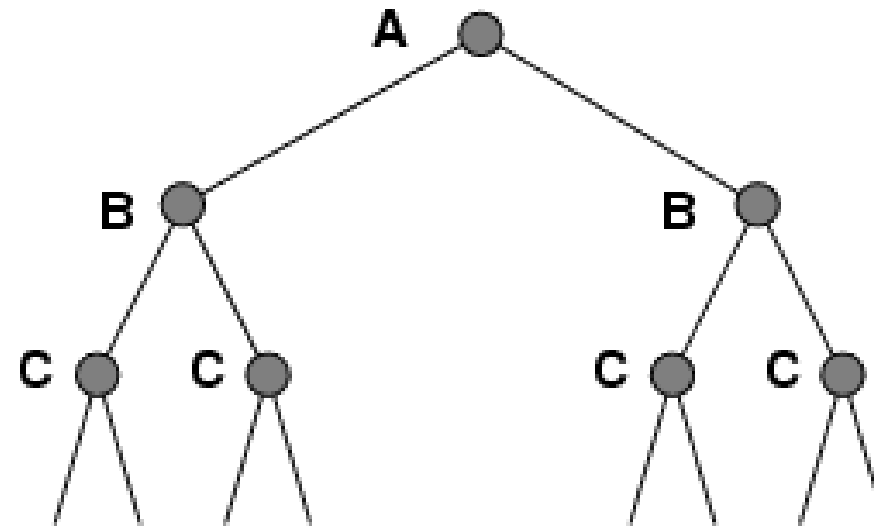
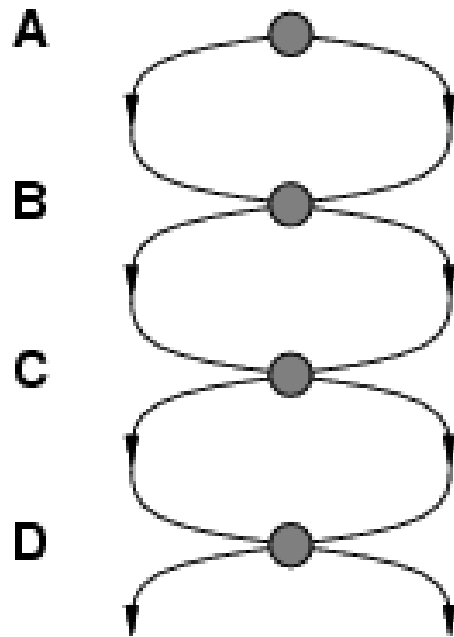


Search tree

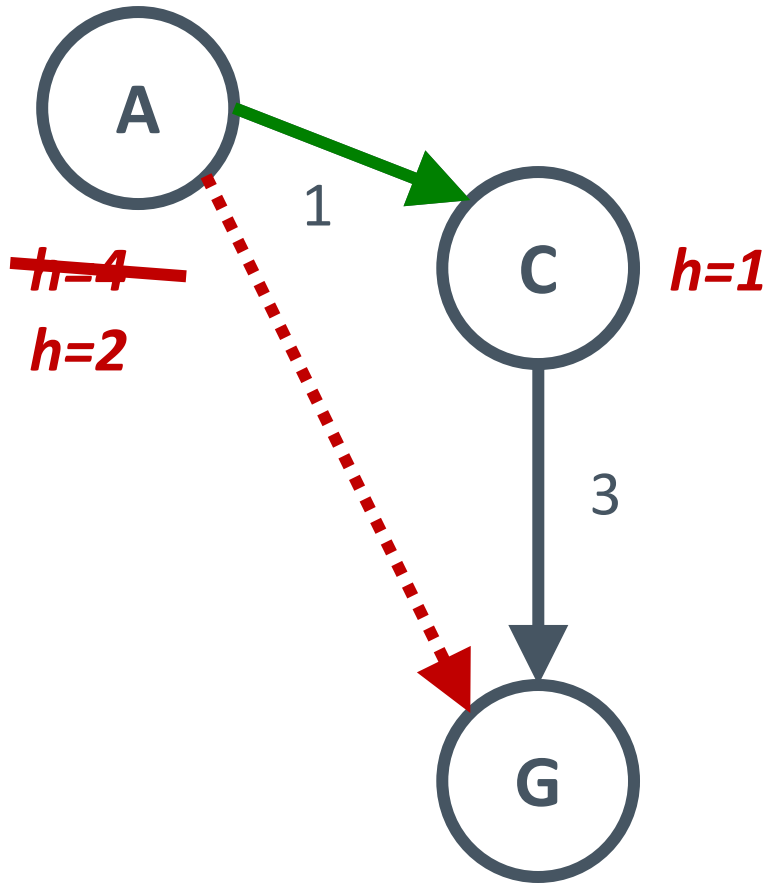


Repeated states

- › Reminder: graph search keeps track of closed/repeated states



Consistency of Heuristics



- › Main idea: estimated heuristic costs \leq actual costs
 - Admissibility: heuristic cost \leq actual cost to goal
$$h(A) \leq \text{actual cost from A to G}$$
 - Consistency: heuristic “arc” cost \leq actual cost for each arc
$$h(A) - h(C) \leq \text{cost(A to C)}$$
- › Consequences of consistency:
 - The f value along a path never decreases
$$h(A) \leq \text{cost(A to C)} + h(C)$$
 - A* graph search is optimal

Optimality

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

Generating heuristics?

- › Effect of heuristic accuracy on performance
- › How to generate admissible heuristics?
 - From relaxed problems
 - From sub-problems
 - From experience

Heuristic accuracy vs performance

- › Effective branching factor b^* - if a total number of nodes generated by A^* is N , and the solution depth is d , then b^* is the branching factor that a uniform tree of depth d would have to have in order to contain $N+1$ nodes
- › $N+1 = 1 + b^* + (b^*)^2 + \dots + (b^*)^d$
- › Well-designed heuristic would have b^* closer to 1 -> allows large problems to be solved at a reasonable computational cost
- › Evaluate usefulness of a heuristic experimentally on a small set of problems

Heuristic dominance

› Ideas for heuristics?

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Heuristic dominance

› How to compare 2 heuristics?

- E.g., for the 8-puzzle:

$h_1(n)$ = number of misplaced tiles

$h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Heuristic dominance

› How to compare 2 heuristics?

Calculate h_1 and h_2 given the start state in the picture?

- $h_1(S) = 8$

- $h_2(S) = 3+1+2+2+2+3+3+2 = 18$

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Heuristic dominance

- › Which heuristic to choose?

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

	Average nodes expanded when the optimal path has...		
	...4 steps	...8 steps	...12 steps
TILES	13	39	227
MANHATTAN	12	25	73

Heuristic dominance

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

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

Heuristic 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

Generating heuristics from relaxed problems

- › Admissible heuristics can be derived from the exact solution cost of a **relaxed version** of the problem
- › If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- › If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution
- › **Key point:** the optimal solution cost of a relaxed problem is no greater than the optimal solution cost of the real problem

Generating heuristic

- › From sub-problems:
 - Eg 8-puzzle problem
 - Subproblem: getting tiles 1-4 into correct position (instead of all 8)
 - The cost of optimal solution is clearly a lower bound on the cost of the complete problem
- › Learning from experience:
 - Solving a lot of 8-puzzle problems – given experience of a number of states and costs to the path, predict the cost from remaining states

Summary of search strategies

Strategy	What to select from frontier	complete/ halts?	space
DFS	First node added	No	Linear
BFS	Last node added	Yes	Exponential
Best first	Globally minimal $h(p)$	No	Exponential
Lowest Cost first	Minimal cost (p)	Yes	Exponential
A^*	Minimal cost $(p) + h(p)$	Yes	Exponential

Heuristics

- › Ideas for a heuristic to find a solution in:
 - a maze
 - Connect 4 game
 - Chess