CS 461
ARTIFICIAL INTELLIGENCE

Lecture # 05
March 24, 2021
SPRING 2021
FAST – NUCES, CFD Campus

Dr. Rabia Maqsood
rabia.maqsood@nu.edu.pk

Today's Topics

- Search strategies
  - *Informed search algorithms*
    - Heuristics
    - Best-First Search
    - A* algorithm
      - *Admissible heuristics*

INTRODUCTION TO HEURISTIC SEARCH

A.K.A Informed search strategies

Heuristic Search

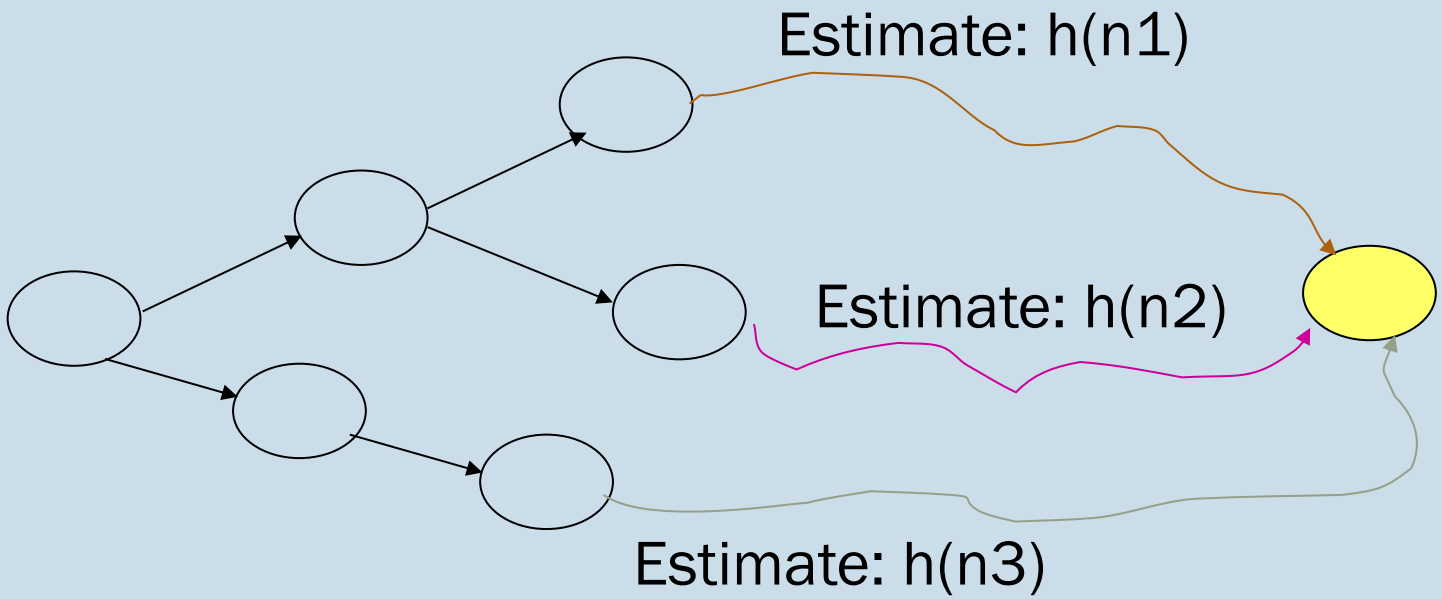
- Blind search algorithms do not consider the **goal** until they are at a goal node.

- Often there is extra knowledge that can be used to guide the search:
  - *an **estimate** of the distance/cost from node n to a goal node.*

- This estimate is called a search heuristic.

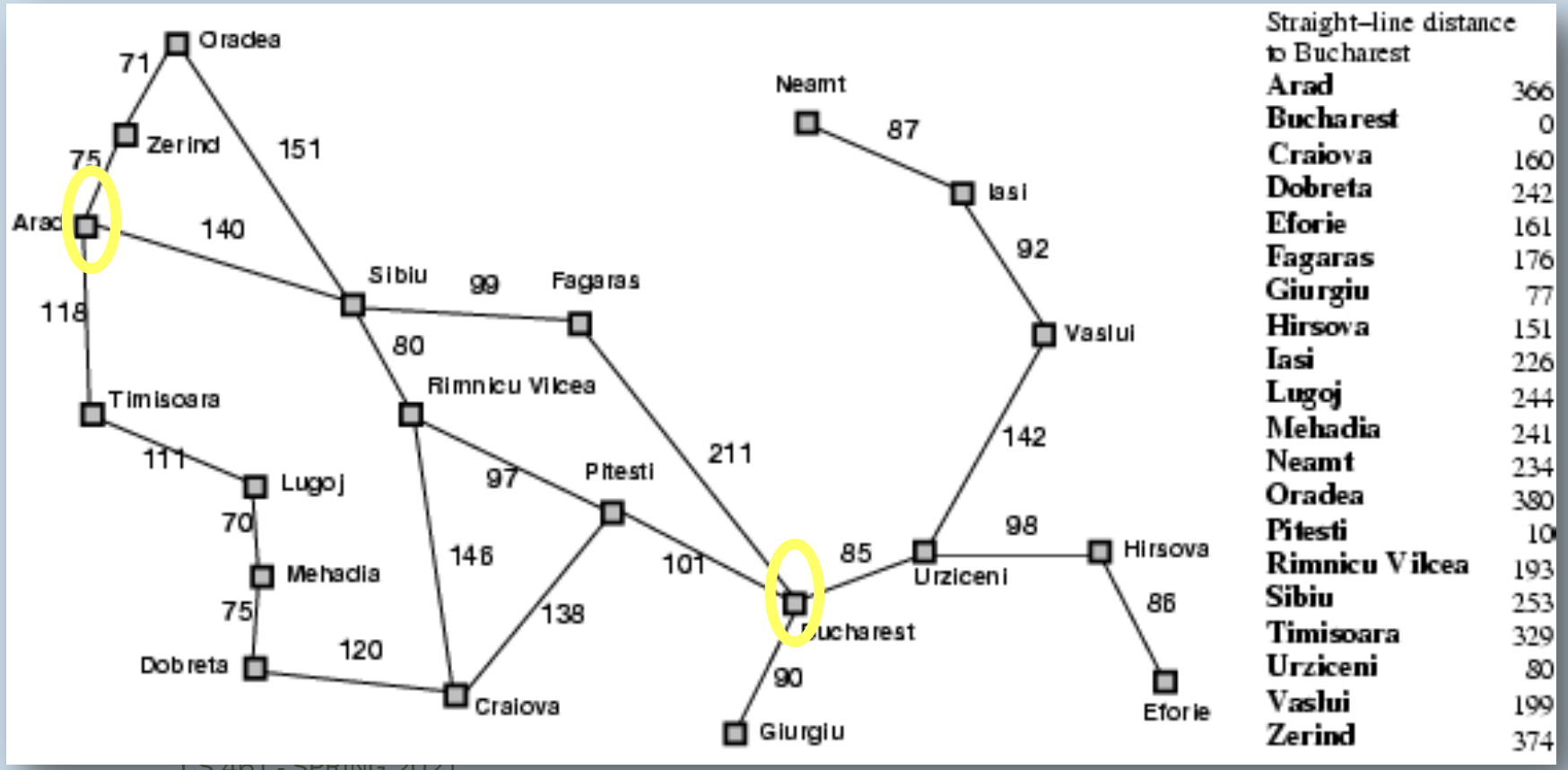
More formally

A search heuristic *h(n)* is an estimate of the cost of the optimal (cheapest) path from node *n* to a goal node.



Example: finding routes

- What could we use as h(n)? E.g., the straight-line (Euclidian) distance between source and goal node



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

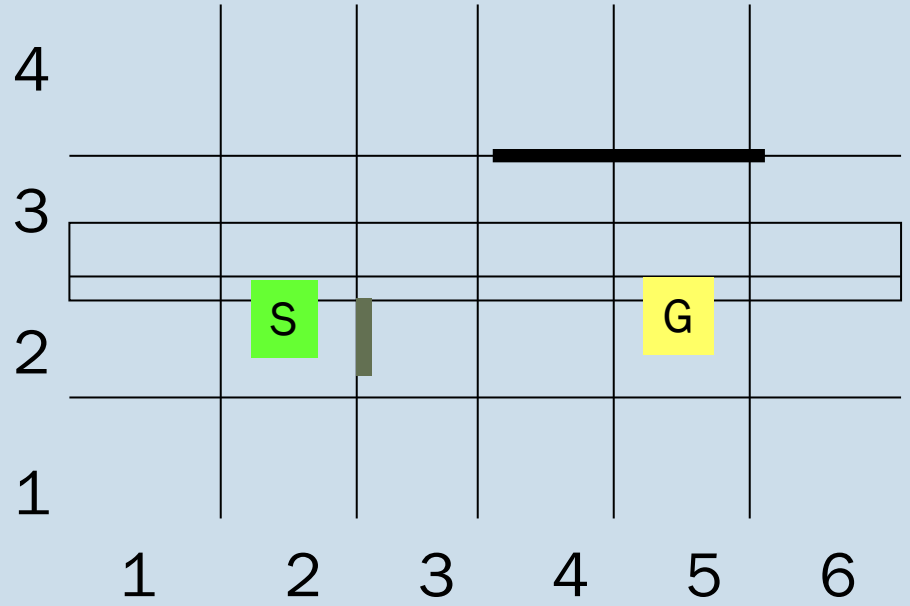
Example: robot navigation

Search problem: robot has to find a route from start to goal location on a grid with obstacles

Actions: move up, down, left, right from tile to tile

Cost : number of moves

Possible h(n)? Manhattan distance (L_1 distance) between two points: sum of the (absolute) difference of their coordinates



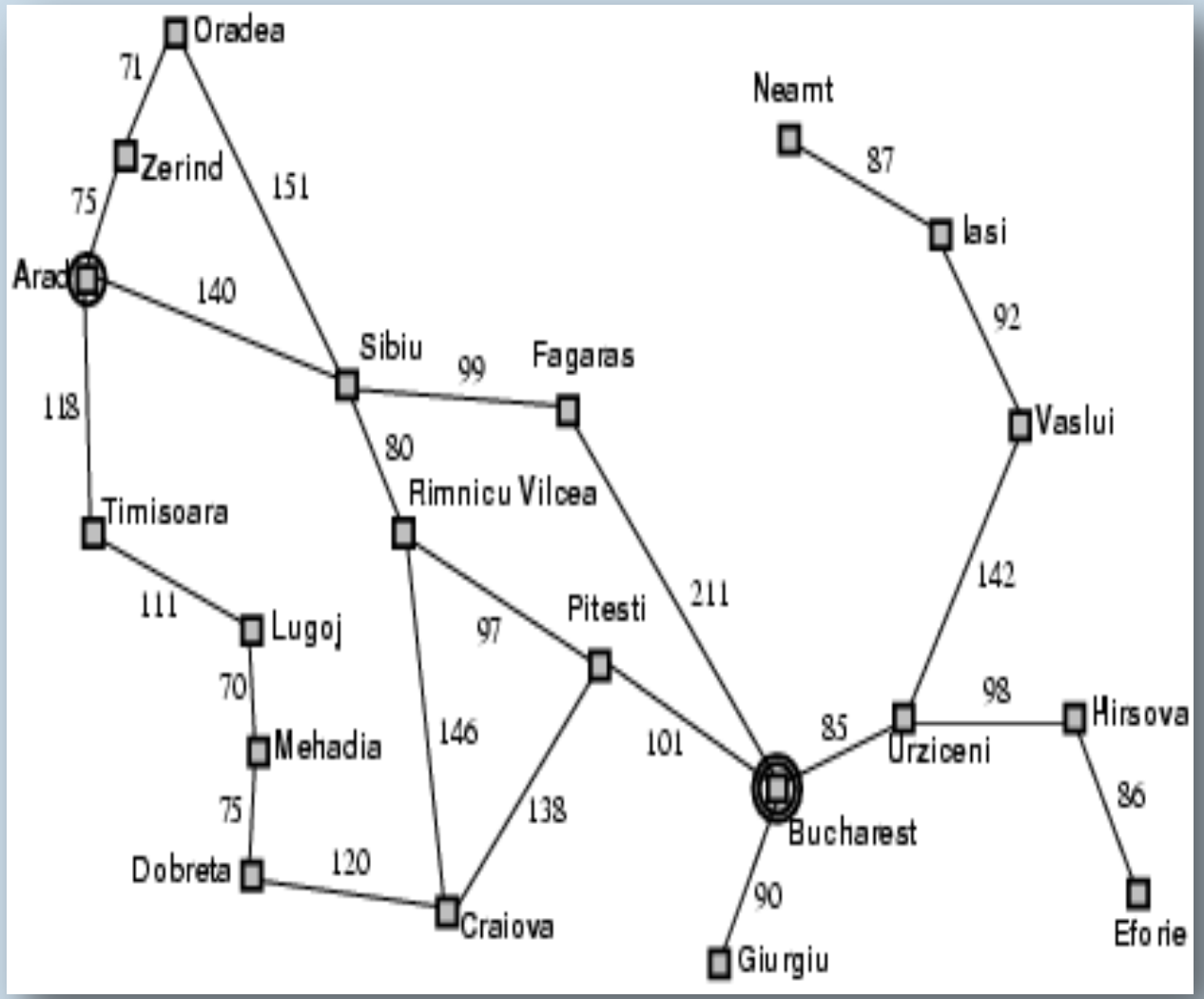
Best-First Search

- Evaluation function: h(n) – *heuristic*
  - *Estimated cost of the cheapest path from n to a goal node*
  - *E.g., $h_{SLD}(n)$ = straight-line distance from n to Bucharest*

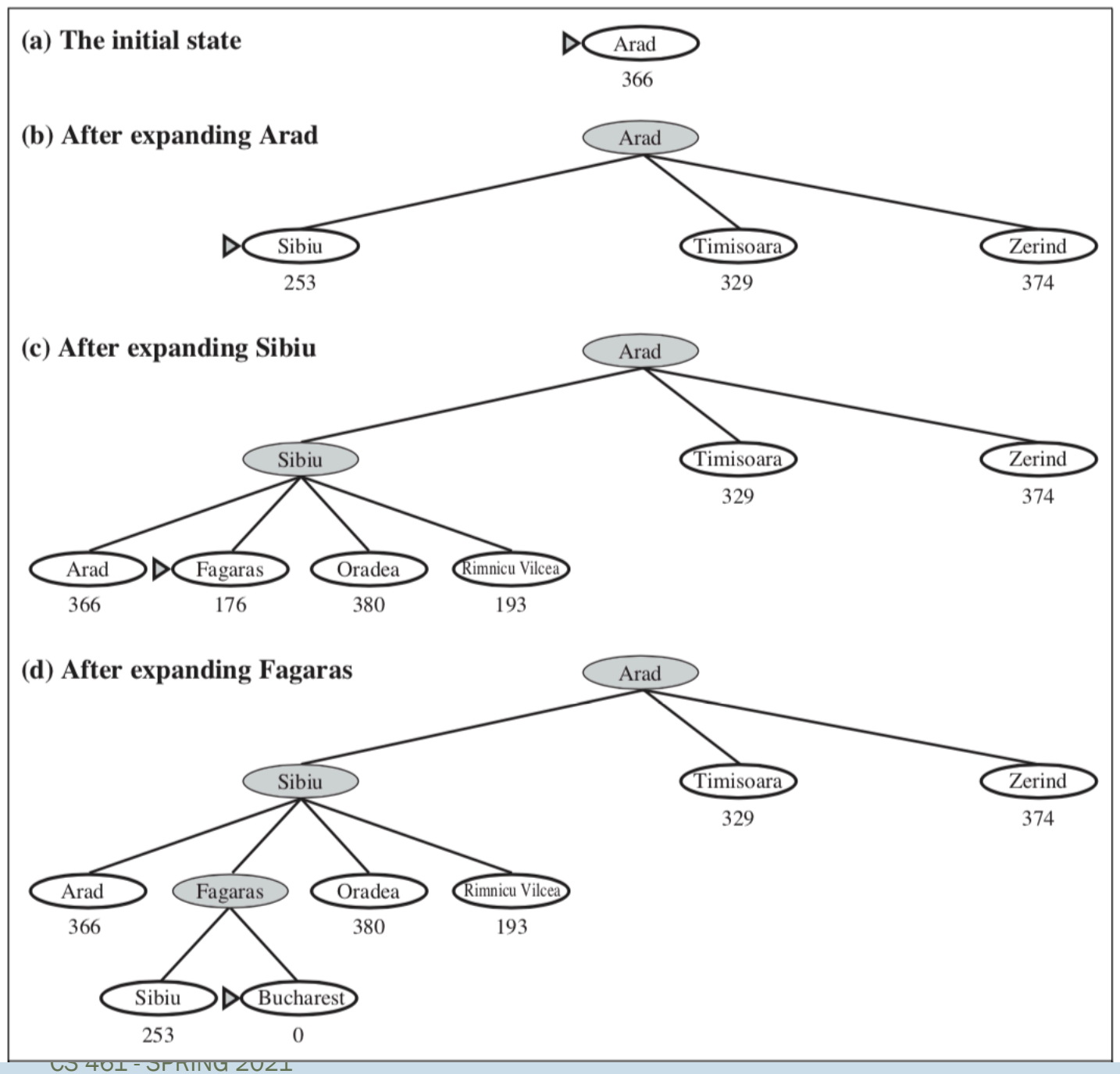
- Based on **greedy** approach

- Greedy search expands the node that appears to be closest to goal

Practice Work: Apply Best-First Search



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


(a) The initial state
Arad
366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
253
329
374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
329
374
Arad
Fagaras
Oradea
Rimnicu Vilcea
366
176
380
193
(d) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
329
374
Arad
Fagaras
Oradea
Rimnicu Vilcea
366
380
193
Sibiu
Bucharest
253
0

Properties of BestFS

- Complete? No – can get stuck in loops
  - *Complete in finite space with repeated state checking*
- Optimal? No
- Time? $O(b^m)$, but a good heuristic can give dramatic improvement
- Space? $O(b^m)$, keeps all nodes in memory

Why would one ever use BestFS then?

If the heuristic is good, it can find solution very fast!

What we have learned so far?

- Having estimates of the distance to the goal can speed things up a lot, but by itself it can also mislead the search (i.e. Best First Search)

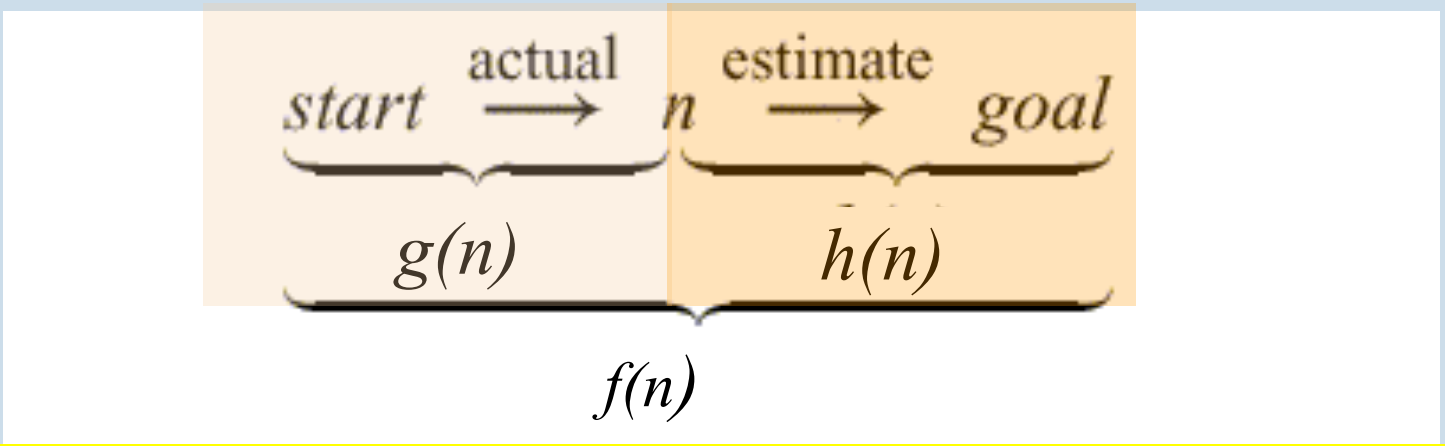
- On the other hand, taking only path costs into account allows UCS to find the optimal solution, but the search process is still uniformed as far as distance to the goal goes.

What we have learned so far?

- UCS orders by path cost or backward cost, ***g(n)***
- BestFS orders by goal proximity or forward cost, ***h(n)***

A* Search

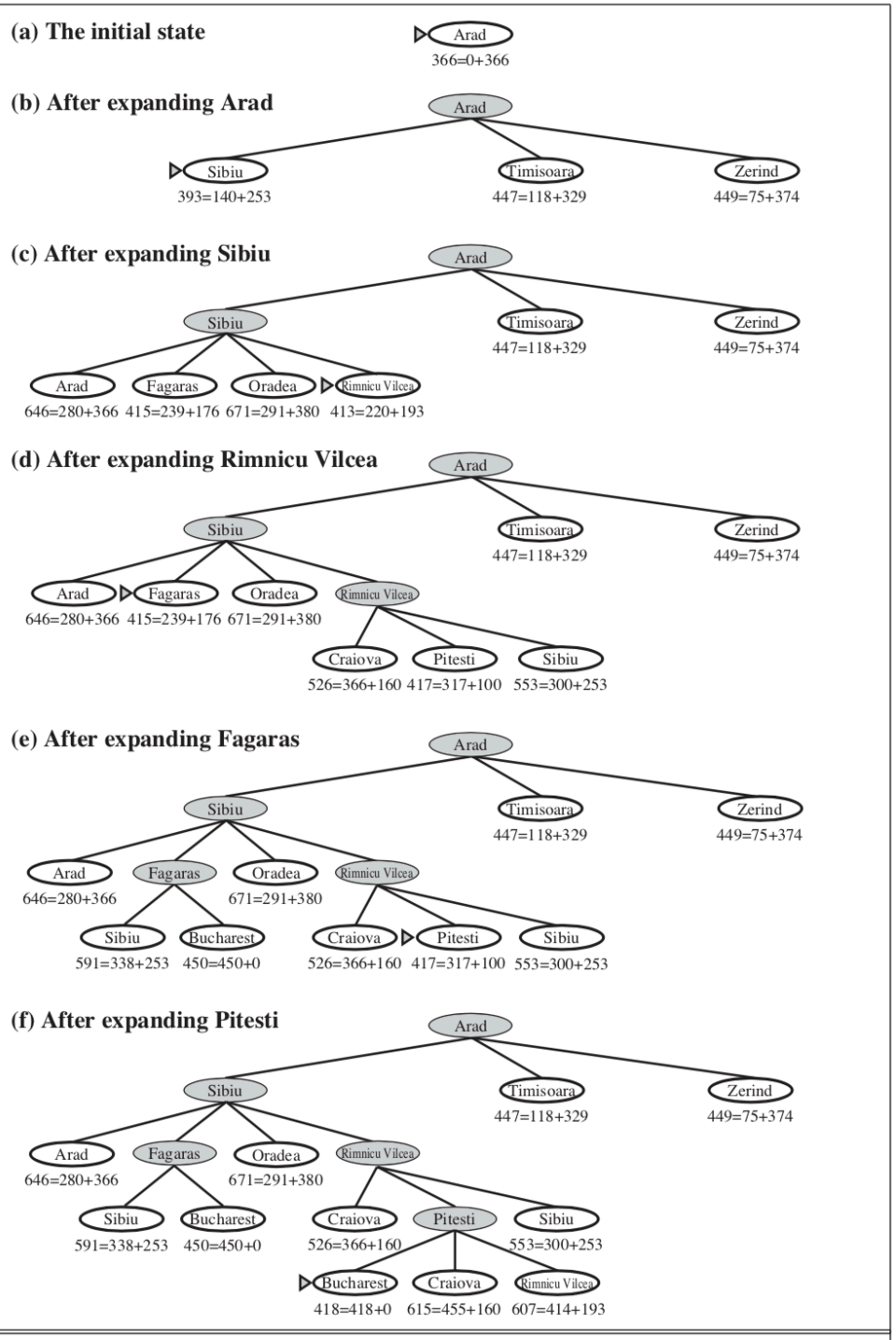
- A* search takes into account both
  - *the cost of the path to a node g(n)*
  - *the heuristic value of that path h(n)*
- Let *f(n) = g(n) + h(n)*



A* always chooses the path on the frontier with the lowest estimated distance from the start to a goal node constrained to go via that path.

A* Search

- Avoid expanding paths that are already expensive
- Evaluation function:
  - $f(n) = g(n) + h(n)$
    - g(n) = exact cost so far to reach n
    - h(n) = estimated cost to goal from n
    - f(n) = estimated total cost of cheapest path from start to goal through n
  - *Also, $h(n) \geq 0$ and $h(G)=0$ for any goal G*


(a) The initial state
Arad
366=0+366
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
393=140+253
447=118+329
449=75+374
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
413=220+193
(d) After expanding Rimnicu Vilcea
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
415=239+176
671=291+380
Craiova
Pitesti
Sibiu
526=366+160
417=317+100
553=300+253
(e) After expanding Fagaras
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
417=317+100
553=300+253
(f) After expanding Pitesti
Arad
Sibiu
Timisoara
Zerind
447=118+329
449=75+374
Arad
Fagaras
Oradea
Rimnicu Vilcea
646=280+366
671=291+380
Sibiu
Bucharest
Craiova
Pitesti
Sibiu
591=338+253
450=450+0
526=366+160
553=300+253
Bucharest
Craiova
Rimnicu Vilcea
418=418+0
615=455+160
607=414+193

Optimality of A*

- A* is complete (finds a solution, if one exists)
- And is optimal (finds the optimal path to a goal) if:
  - *the branching factor is finite*
  - *arc costs are > 0*
  - *h(n) is admissible*

Admissibility of a heuristic

- A heuristic is admissible if it *never overestimates* the cost to reach the goal

- Let c(n) denotes the optimal path from node n to any goal node. A search heuristic h(n) is called admissible if $h(n) \leq c(n)$ for all nodes n, i.e. if for all nodes it is an underestimate of the cost to any goal.

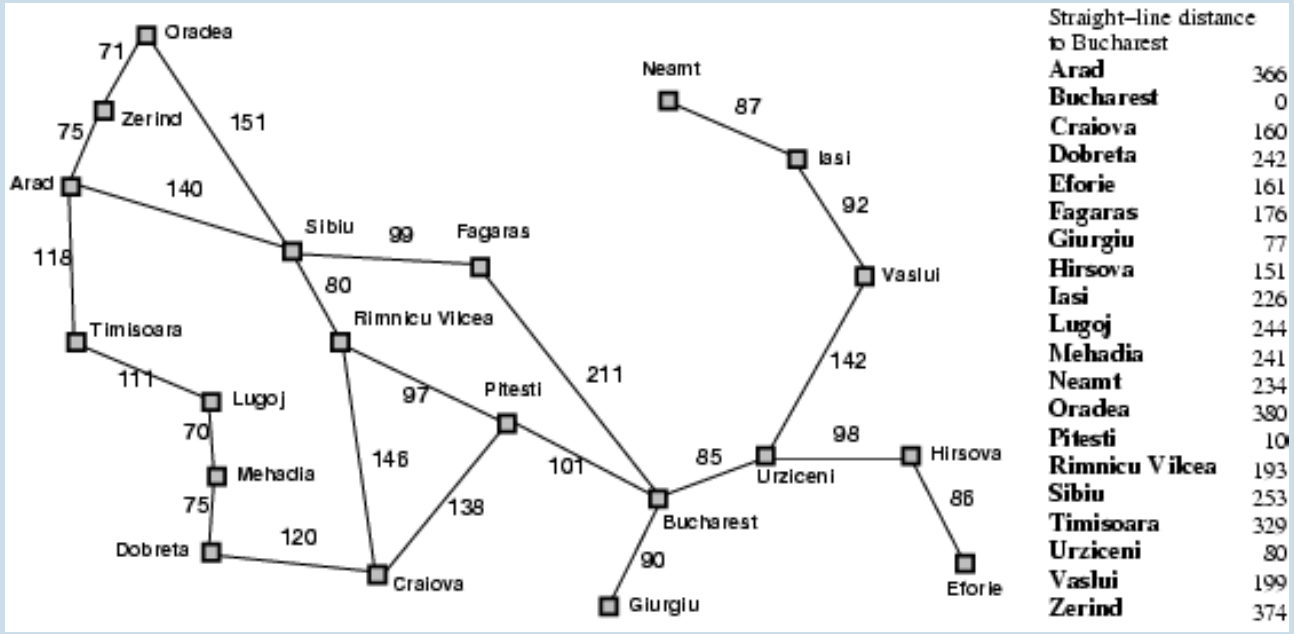
Admissibility of a heuristic

Let c(n) denote the cost of the optimal path from node n to any goal node. A search heuristic h(n) is called
admissible if h(n) ≤ c(n) for all nodes n, i.e. if for all nodes it is an underestimate of the cost to any goal.

- Example: is the straight-line distance (SLD) admissible?

YES

The shortest distance between two points is a straight line.

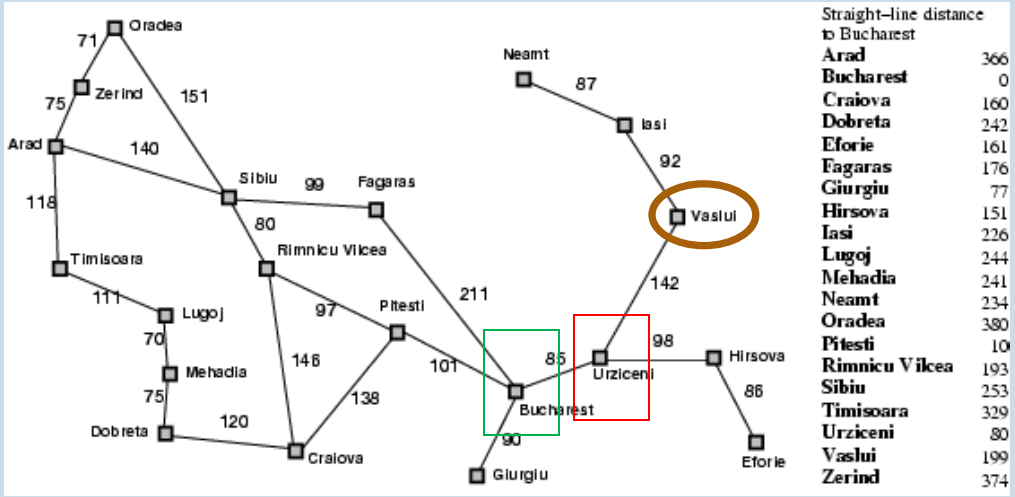


Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Admissibility of a heuristic

Let c(n) denote the cost of the optimal path from node n to any goal node. A search heuristic h(n) is called
admissible if h(n) ≤ c(n) for all nodes n, i.e. if for all nodes it is an underestimate of the cost to any goal.

Example:
the goal is Urzizeni (red box), but all we know is the straight-line distances to Bucharest (green box)



- Possible h(n) = sld(n, Bucharest) + cost(Bucharest, Urzineni)
- Admissible?

NO

Cost of going from **Vastul** to **Urzineni** is shorter than this estimate

Example: robot navigation

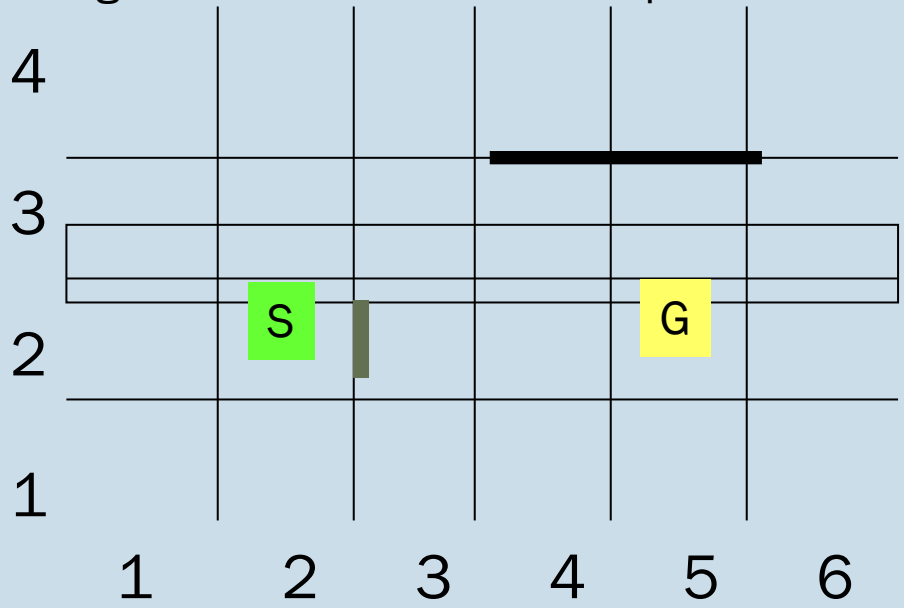
Search problem: robot has to find a route from start to goal location on a grid with obstacles

Actions: move up, down, left, right from tile to tile

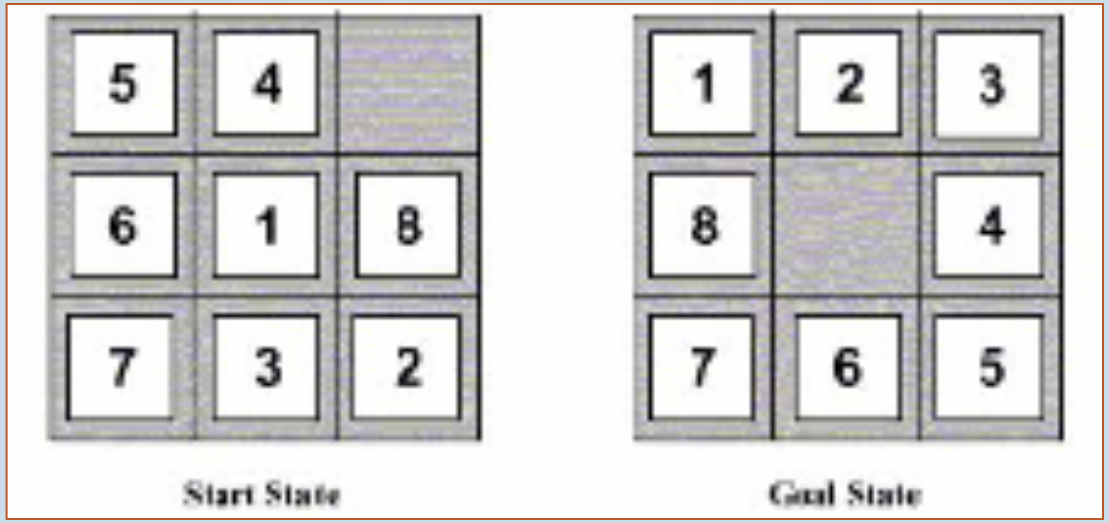
Cost : number of moves

Possible h(n)? Manhattan distance (L_1 distance) between two points: sum of the (absolute) difference of their coordinates

Admissible? Yes. Manhattan distance is the shortest path between any two tiles of the grid given the actions available and no walls. Including the walls will force the agent to take some extra steps to avoid them



Heuristic Function for 8-puzzle



An admissible heuristics for the 8-puzzle is?

Number of misplaced tiles: One needs at least that many moves to get the board in the goal state

What is the value of *h* for given start state?

How to Construct an Admissible Heuristic

- Identify relaxed version of the problem:
  - *where one or more constraints have been dropped*
  - *problem with fewer restrictions on the actions*
- Robot navigation: the agent can move through walls
- Route finding: the agent can move straight
- 8 puzzle:
  - *"number of misplaced tiles":*

    tiles can move everywhere and occupy same spot as others
  - *"sum of moves between current and goal position":*
    tiles can occupy same spot as others

Why does this lead to an admissible heuristic?

- The problem only gets easier!
- Less costly to solve it

How to Construct an Admissible Heuristic

- You should identify constraints which, when dropped, make the problem easy to solve
  - *important because heuristics are not useful if they're as hard to solve as the original problem!*

This was the case in our examples

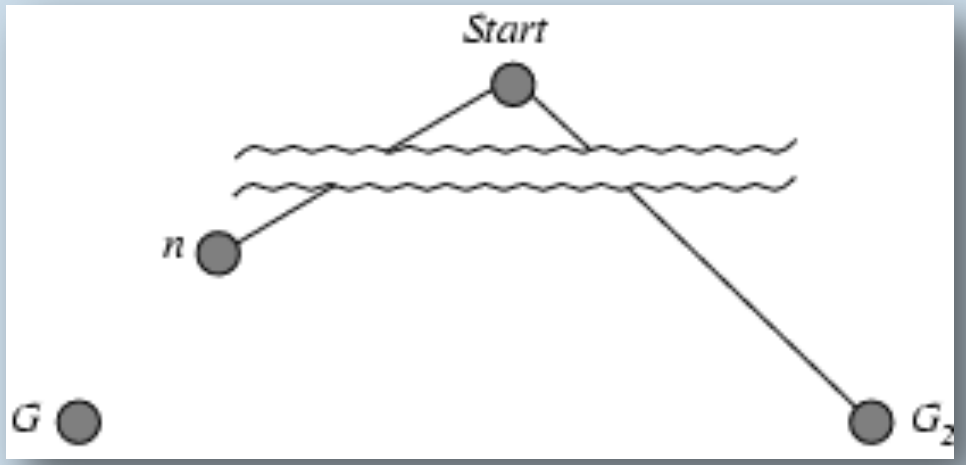
Robot: allowing the agent to move through walls. Optimal solution to this relaxed problem is Manhattan distance

Driver: allowing the agent to move straight. Optimal solution to this relaxed problem is straight-line distance

8puzzle: tiles can move anywhere. Optimal solution to this relaxed problem is number of misplaced tiles

Optimality of A* (proof)

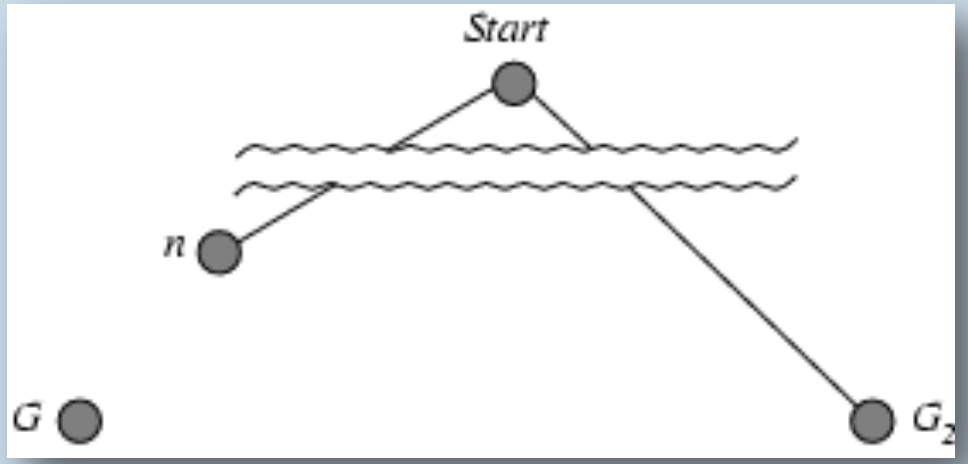
- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n* is on a shortest path to an optimal goal *G*.



- We are going to show that any sub-path *n* of G on the frontier will be expanded before G2. Therefore, A* will find G before G2.

Optimality of A* (proof)

- Suppose some suboptimal goal G_2 has been generated and is in the fringe. Let *n* be an unexpanded node in the fringe such that *n* is on a shortest path to an optimal goal *G*.



$f(G_2) = g(G_2)$	since $h(G_2) = 0$
$g(G_2) > g(G)$	since G_2 is suboptimal
$f(G) = g(G)$	since $h(G) = 0$
$f(G_2) > f(G)$	from above

Hence $f(G_2) > f(n)$, and A* will never select G_2 for expansion and thus A* is optimal

Reading Material

- Russell & Norvig: Chapter # 3
- David Poole: Chapter # 3