# CS 1571 Introduction to AI
# Lecture 5

# Uninformed search methods II.

**Milos Hauskrecht**
milos@cs.pitt.edu
5329 Sennott Square

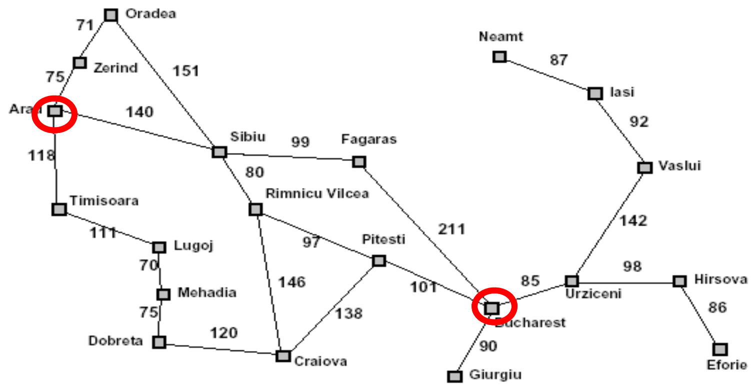


## Uninformed methods

- Uninformed search methods use only information available in the problem definition
  - **Breadth-first search (BFS)**
  - **Depth-first search (DFS)**
  - **Iterative deepening (IDA)**
  - **Bi-directional search**
- **For the minimum cost path problem:**
  - **Uniform cost search**

# Minimum cost path search

**Traveler example with distances [km]**




**Optimal path:** the shortest distance path between the initial and destination city



# Searching for the minimum cost path

- **General minimum cost path-search problem:**
  - **adds weights or costs** to operators (links)
- **Search strategy:**
  - "Intelligent" expansion of the search tree should be driven by the cost of the current (partially) built path
- **Implementation:**
  - **Path cost function for node *n* :** $g(n)$
    - length of the path represented by the search tree branch starting at the root of the tree (initial state) to *n*
  - **Search strategy:**
    - Expand the leaf node with the minimum $g(n)$ first
    - Can be implemented by the priority queue

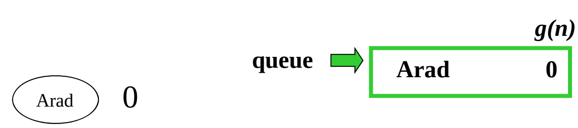
## Searching for the minimum cost path

- The basic algorithm for finding the minimum cost path:
  - **Dijkstra's shortest path**
- In AI, the strategy goes under the name
  - **Uniform cost search**
- **Note:**
  - When operator costs are all equal to 1 the uniform cost search is equivalent to the breadth first search BFS



## Uniform cost search

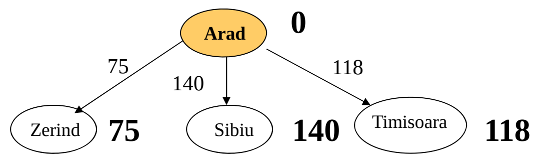
- Expand the node with the minimum path cost first
- **Implementation: a priority queue**



# Uniform cost search

**queue** ⇨

| | *g(n)* |
|---|---|
| **Zerind** | **75** |
| **Timisoara** | **118** |
| **Sibiu** | **140** |

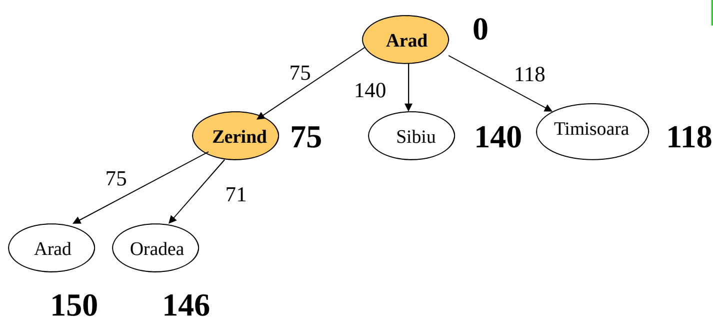





# Uniform cost search

**queue** ⇨

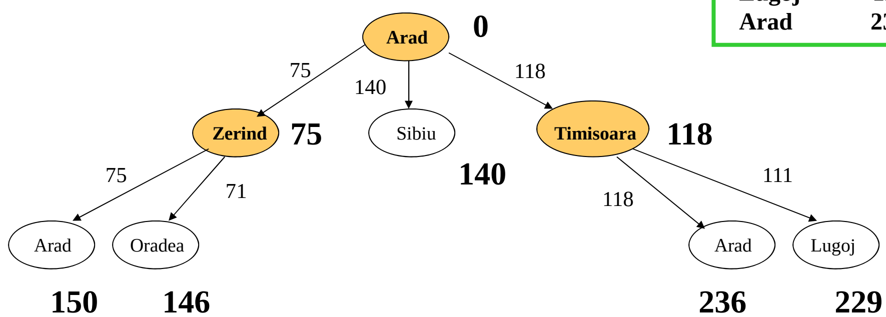
| | *g(n)* |
|---|---|
| Timisoara | 118 |
| Sibiu | 140 |
| **Oradea** | **146** |
| **Arad** | **150** |



## Uniform cost search

queue

| | g(n) |
|---|---|
| Sibiu | 140 |
| Oradea | 146 |
| Arad | 150 |
| **Lugoj** | **129** |
| **Arad** | **236** |






## Properties of the uniform cost search

- **Completeness:** **Yes**, assuming that operator costs are non-negative (the cost of path never decreases)

$$g(n) \leq g(\text{successor }(n))$$

- **Optimality:** **Yes.** Returns the least-cost path.
- **Time complexity:**
  **number of nodes with the cost *g(n)* smaller than the optimal cost**
- **Memory (space) complexity:**
  **number of nodes with the cost *g(n)* smaller than the optimal cost**

## Elimination of state repeats

**Idea:**

- A node is redundant and can be eliminated if there is another node with exactly the same state and a shorter path from the initial state

**Assuming positive costs:**

  – If the state was already expanded, is there a a shorter path to that node ?
  – **No !**

**Implementation:**

- Marking with the hash table



## Informed (heuristic) search methods

## Additional information to guide the search

- **Uninformed search methods**
  - use only the information from the problem definition; and
  - past explorations, e.g. cost of the path generated so far
- **Informed search methods**
  - incorporate additional measure of a potential of a specific state to reach the goal
  - a potential of a state (node) to reach a goal is measured by a **heuristic function**
- Heuristic function is denoted $h(n)$



## Evaluation-function driven search

- A search strategy can be defined in terms of **a node evaluation function**
  - Similarly to the path cost for the uniform cost search
- **Evaluation function**
  - Denoted $f(n)$
  - Defines the **desirability of a node to be expanded next**
- **Evaluation-function driven search:**
  - **expand the node (state) with the best evaluation-function value**
- **Implementation:**
  - **priority queue** with nodes in the decreasing order of their evaluation function value

## Uniform cost search

- **Uniform cost search (Dijkstra's shortest path):**
  - A special case of the evaluation-function driven search

$$f(n) = g(n)$$

- **Path cost function** $g(n)$ ;
  - path cost from the initial state to $n$

- **Uniform-cost search:**
  - Can handle general minimum cost path-search problem:
  - **weights or costs** associated with operators (links).

- **Note:** Uniform cost search relies on the problem definition only
  - It is an uninformed search method



## Best-first search

**Best-first search**

- incorporates a **heuristic function,** $h(n)$ , into the evaluation function $f(n)$ to guide the search.

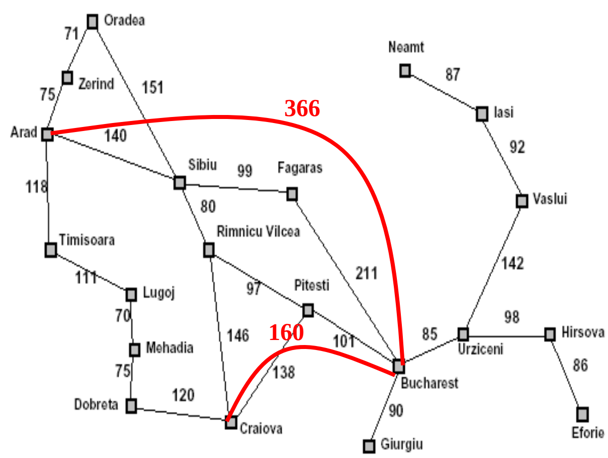
**Heuristic function:**

- Measures a potential of a state (node) to reach a goal
- Typically in terms of some distance to a goal estimate

**Example of a heuristic function:**

- Assume a shortest path problem with city distances on connections
- Straight-line distances between cities give additional information we can use to guide the search

## Example: traveler problem with straight-line distance information




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 |

- **Straight-line distances** give an estimate of the cost of the path between the two cities



## Best-first search

**Best-first search**

- incorporates a **heuristic function,** $h(n)$ , into the evaluation function $f(n)$ to guide the search.
- **heuristic function:** measures a potential of a state (node) to reach a goal

**Special cases** (differ in the design of evaluation function):

– **Greedy search**

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

– **A* algorithm**

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

\+ **iterative deepening** version of A* : **IDA***

## Greedy search method

- Evaluation function is equal to the heuristic function

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

- **Idea:** the node that seems to be the closest to the goal is expanded first



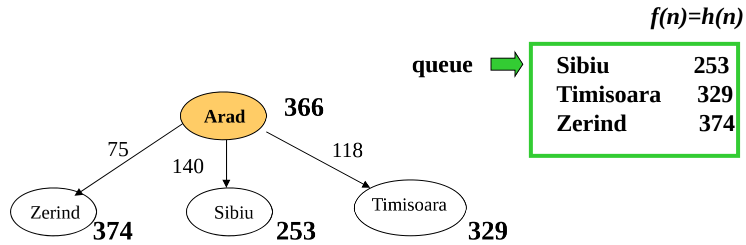
## Greedy search

***f(n)=h(n)***

**queue** → | **Arad** **366** |

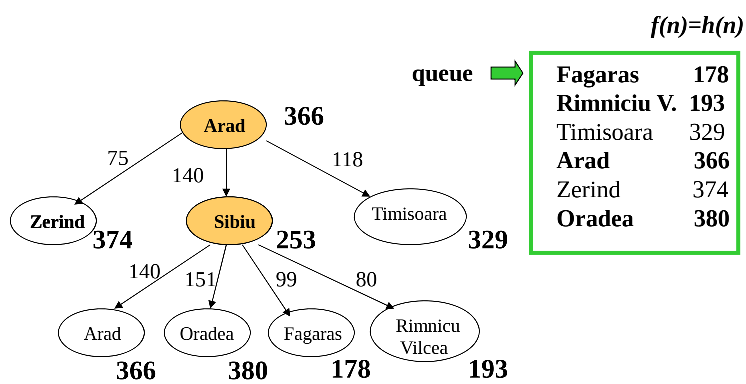
Arad **366**

# Greedy search






# Greedy search

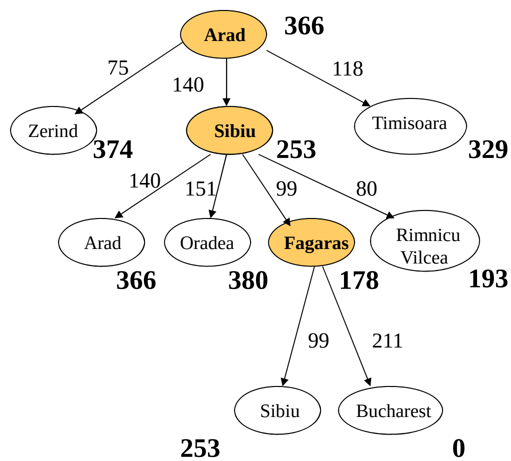


## Greedy search

*f(n)=h(n)*

**queue**

| | |
|---|---|
| **Bucharest** | **0** |
| Rimniciu V. | 193 |
| **Sibiu** | **253** |
| Timisoara | 329 |
| Arad | 366 |
| Zerind | 374 |
| Oradea | 380 |



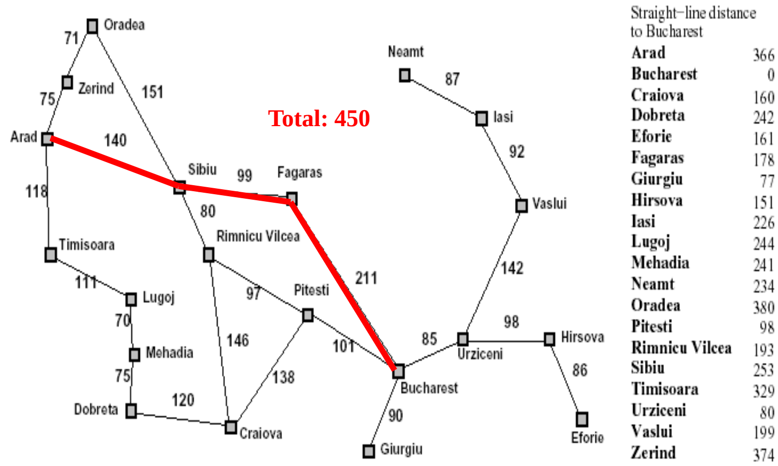

**Goal !!!**



## Properties of greedy search

- **Completeness: No.**
  We can loop forever. Nodes that seem to be the best choices can lead to cycles. Elimination of state repeats can solve the problem.
- **Optimality:** ?
- **Time complexity:**
- **Memory (space) complexity:**

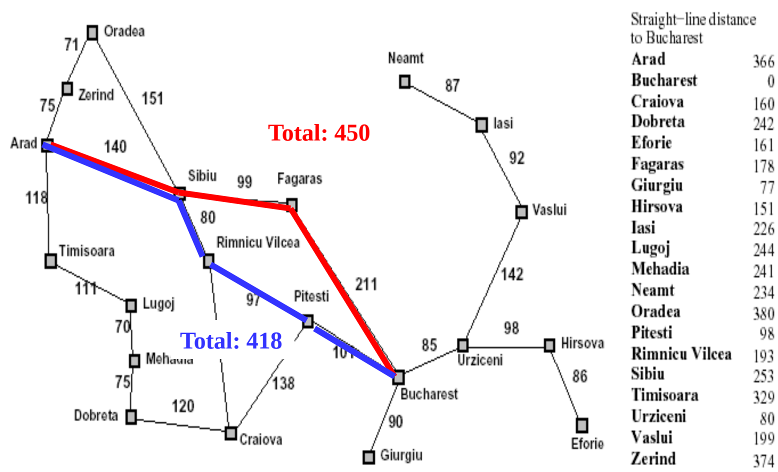
# Example: traveler problem with straight-line distance information




- **Greedy search result**



# Example: traveler problem with straight-line distance information




- **Greedy search and optimality**

## Properties of greedy search

- **Completeness: No.**
  We can loop forever. Nodes that seem to be the best choices can lead to cycles. Elimination of state repeats can solve the problem.
- **Optimality: No.**
  Even if we reach the goal, we may be biased by a bad heuristic estimate. **Evaluation function disregards the cost of the path built so far.**
- **Time complexity:** $O(b^m)$

  Worst case !!! But often better!
- **Memory (space) complexity:** $O(b^m)$

  Often better!



## A* search

- The problem with the **greedy search** is that it can keep expanding paths that are already very expensive.
- The problem with the **uniform-cost search** is that it uses only past exploration information (path cost), no additional information is utilized
- **A* search**

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

$g(n)$ - cost of reaching the state

$h(n)$ - estimate of the cost from the current state to a goal

$f(n)$ - estimate of the path length

- **Additional A*condition**: admissible heuristic

$$h(n) \leq h^*(n) \quad \text{for all } n$$

# A* search example

*f(n)*

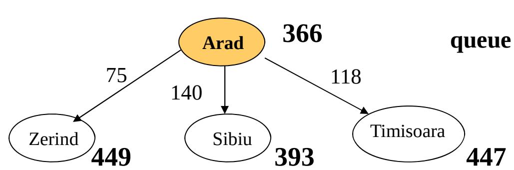

queue ⇨

| | |
|---|---|
| Arad | 366 |



# A* search example

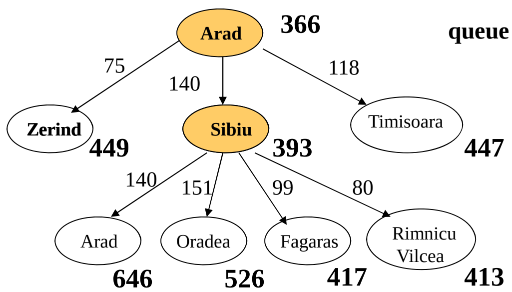
*f(n)*




queue ⇨

| | |
|---|---|
| Sibiu | 393 |
| Timisoara | 447 |
| Zerind | 449 |

# A* search example

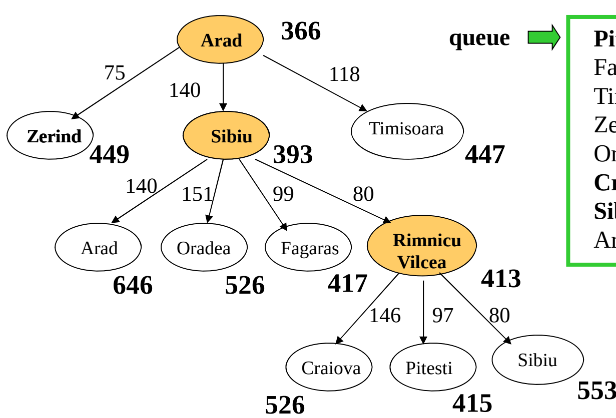



queue ⇨

| | *f(n)* |
|---|---|
| **Rimnicu V.** | **413** |
| **Fagaras** | **417** |
| Timisoara | 447 |
| Zerind | 449 |
| **Oradea** | **526** |
| **Arad** | **646** |



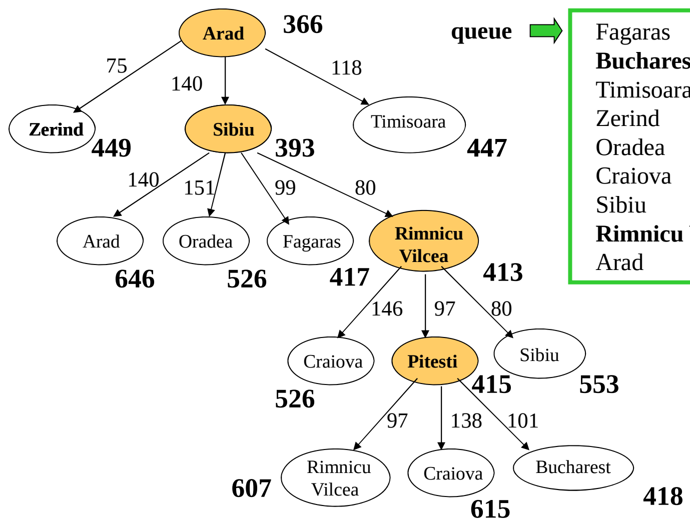
# A* search example




queue ⇨

| | *f(n)* |
|---|---|
| **Pitesti** | **415** |
| Fagaras | 417 |
| Timisoara | 447 |
| Zerind | 449 |
| Oradea | 526 |
| **Craiova** | **526** |
| **Sibiu** | **553** |
| Arad | 646 |

# A* search example

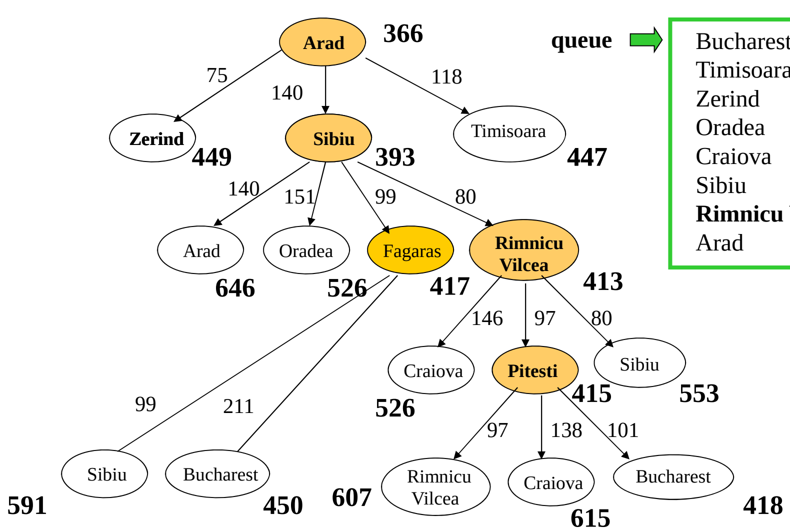



queue

| | *f(n)* |
|---|---|
| Fagaras | 417 |
| **Bucharest** | **418** |
| Timisoara | 447 |
| Zerind | 449 |
| Oradea | 526 |
| Craiova | 526 |
| Sibiu | 553 |
| **Rimnicu V.** | **607** |
| Arad | 646 |



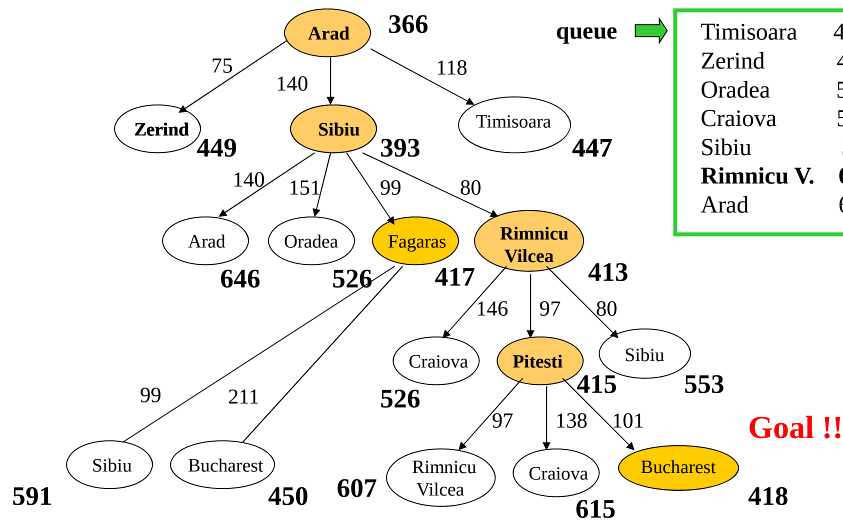
# A* search example




queue

| | *f(n)* |
|---|---|
| Bucharest | 418 |
| Timisoara | 447 |
| Zerind | 449 |
| Oradea | 526 |
| Craiova | 526 |
| Sibiu | 553 |
| **Rimnicu V.** | **607** |
| Arad | 646 |

# A* search example




queue

| | *f(n)* |
|---|---|
| Timisoara | 447 |
| Zerind | 449 |
| Oradea | 526 |
| Craiova | 526 |
| Sibiu | 553 |
| **Rimnicu V.** | **607** |
| Arad | 646 |



# Properties of A* search

- **Completeness: Yes.**
- **Optimality: ?**
- **Time complexity:**
  - **?**
- **Memory (space) complexity:**
  - **?**

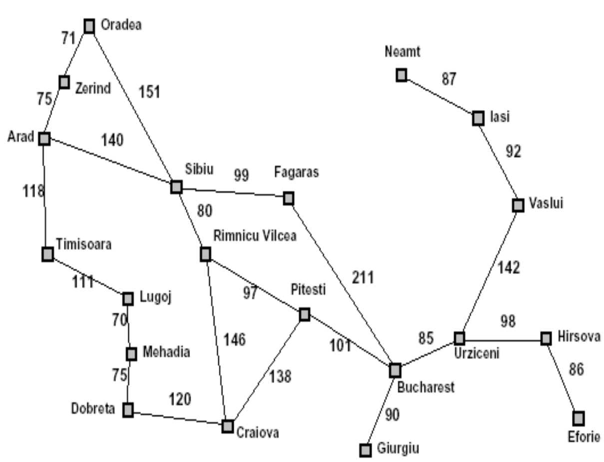
## Optimality of A*

- In general, a heuristic function $h(n)$ :

  It can overestimate, be equal or underestimate the true distance of a node to the goal $h^*(n)$
- Is the A* optimal for an arbitrary heuristic function?

CS 1571 Intro to AI M. Hauskrecht

## Example: traveler problem with straight-line distance information




Straight-line distance to Bucharest

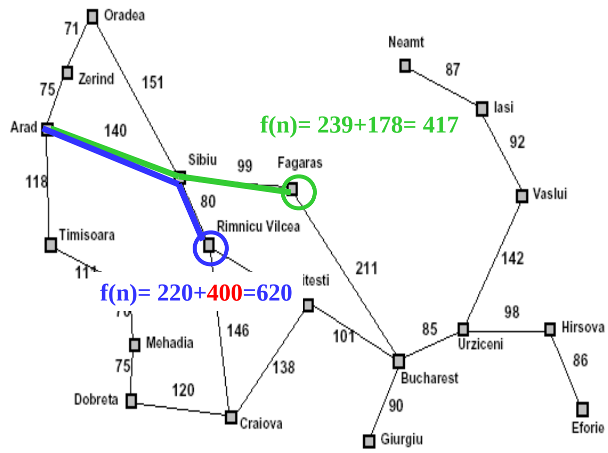
| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 178 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 98 |
| Rimnicu Vilcea | 400 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

- **Admissible heuristics**

overestimate

CS 1571 Intro to AI M. Hauskrecht

## Example: traveler problem with straight-line distance information



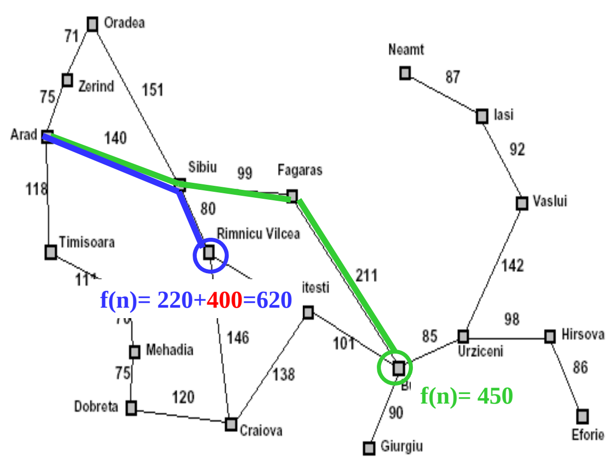

Straight−line distance to Bucharest

| | |
|---|---|
| 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 | 400 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

- **Admissible heuristics**



## Example: traveler problem with straight-line distance information




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 | 400 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

- **Admissible heuristics**

**Total path: 450 is suboptimal**

## Optimality of A*

- In general, a heuristic function $h(n)$ :

  Can overestimate, be equal or underestimate the true distance of a node to the goal $h^*(n)$
- Is the A* optimal for an arbitrary heuristic function?
- **No!**

CS 1571 Intro to AI M. Hauskrecht

## Optimality of A*

- In general, a heuristic function $h(n)$ :

  Can overestimate, be equal or underestimate the true distance of a node to the goal $h^*(n)$
- **Admissible heuristic condition**
  - **Never overestimate the distance to the goal !!!**

$$h(n) \le h^*(n) \quad \text{for all } n$$

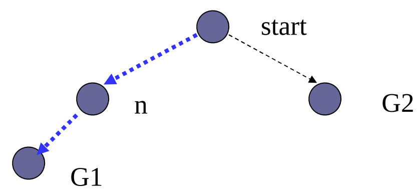
**Example:** the straight-line distance in the travel problem never overestimates the actual distance

**Is A* search with an admissible heuristic is optimal ??**

CS 1571 Intro to AI M. Hauskrecht

# Optimality of A* (proof)

- Let G1 be the optimal goal (with the minimum path distance). Assume that we have a sub-optimal goal G2. Let *n* be a node that is on the optimal path and is in the queue together with G2




**Then:** $f(G2) = g(G2)$ since $h(G2) = 0$

$> g(G1)$ since G2 is suboptimal

$\geq f(n)$ since $h$ is admissible

**And** thus **A* never selects G2 before *n***



# Properties of A* search

- **Completeness: Yes.**
- **Optimality: Yes (with the admissible heuristic)**
- **Time complexity:**
  - **Order roughly the number of nodes with *f(n)* smaller than the cost of the optimal path *g****
- **Memory (space) complexity:**
  - **Same as time complexity (all nodes in the memory)**