# Informed Search

## Use “heuristics” to guide the search

- What is a heuristic?
  - Guidelines/Intuitions that are based on information outside the problem formulation
- “Best” first
  - Greedy Search
  - A*
- Many variations of A* are used
  - Hill-climbing / Simulated Annealing

## Best-first search

- Idea:

  use an evaluation function for each node; estimate of ***"desirability"***

$\Rightarrow$ expand most desirable unexpanded node.

- Implementation:

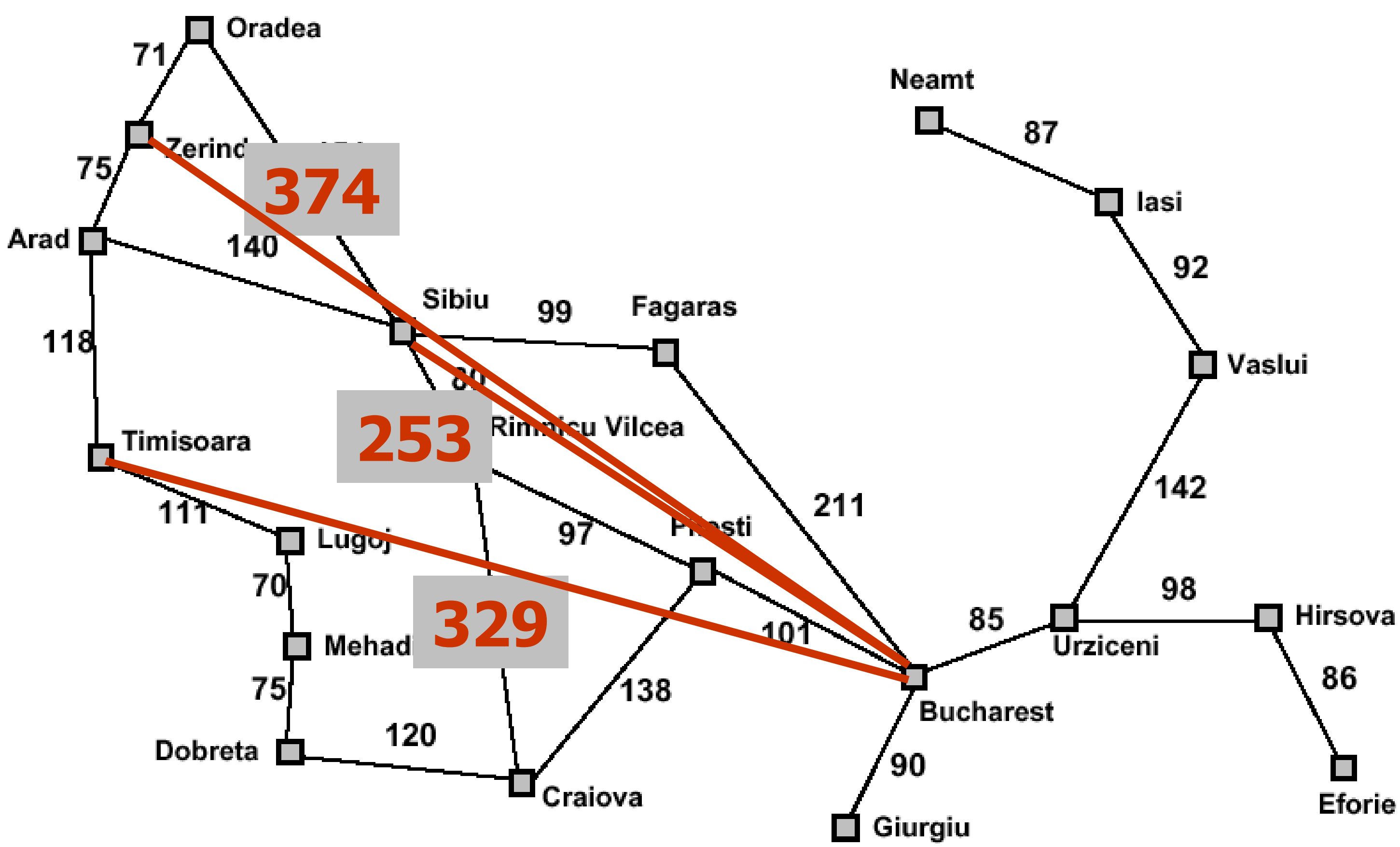
  **QueueingFn** = insert successors in decreasing order of **"desirability"**

- Examples of Best First search:

  Greedy search

  A* search

# Romania with step costs in km




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 |

## Greedy search

- Estimation function:

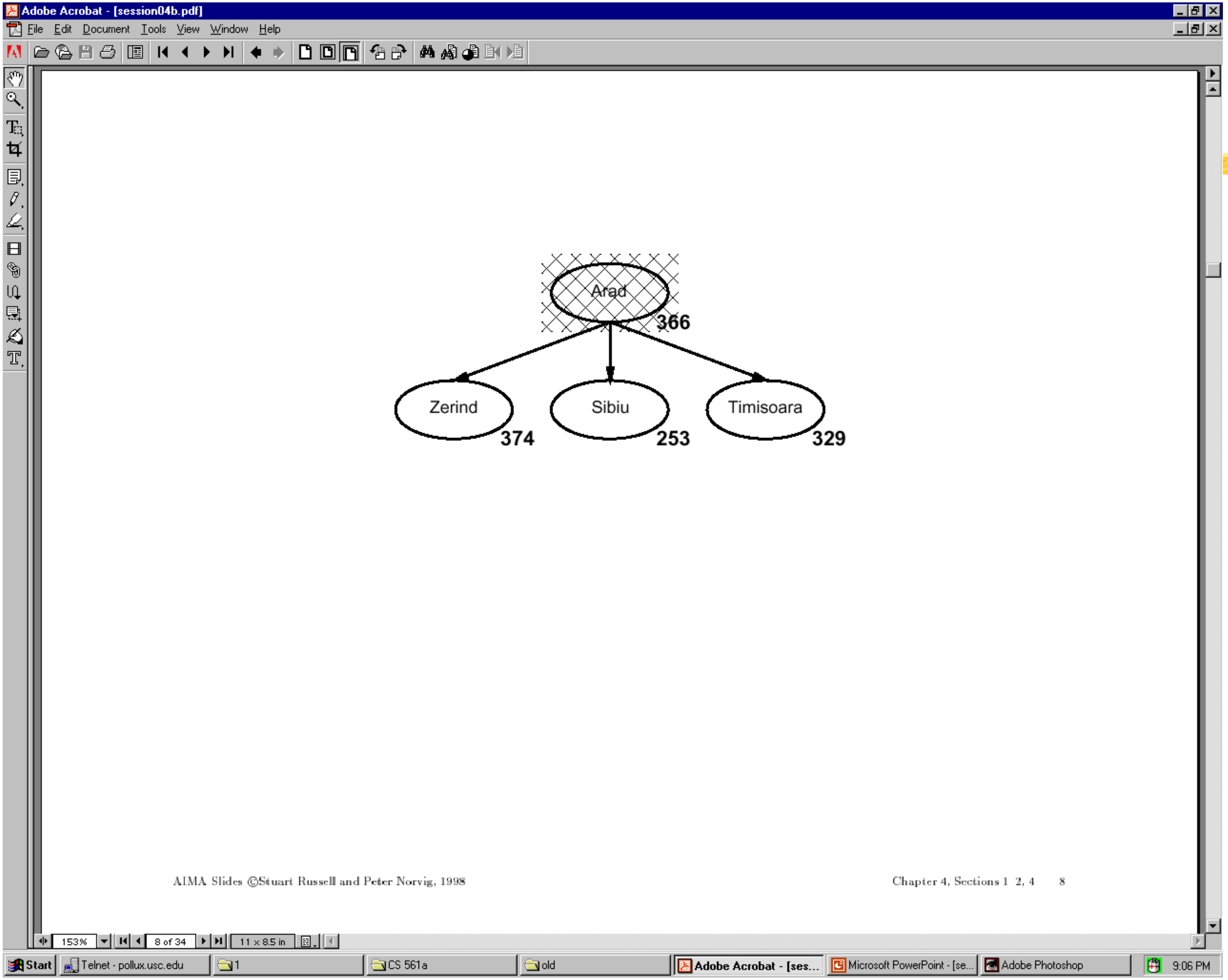
  $h(n)$ = estimate of cost from $n$ to goal (h ➔ heuristic)

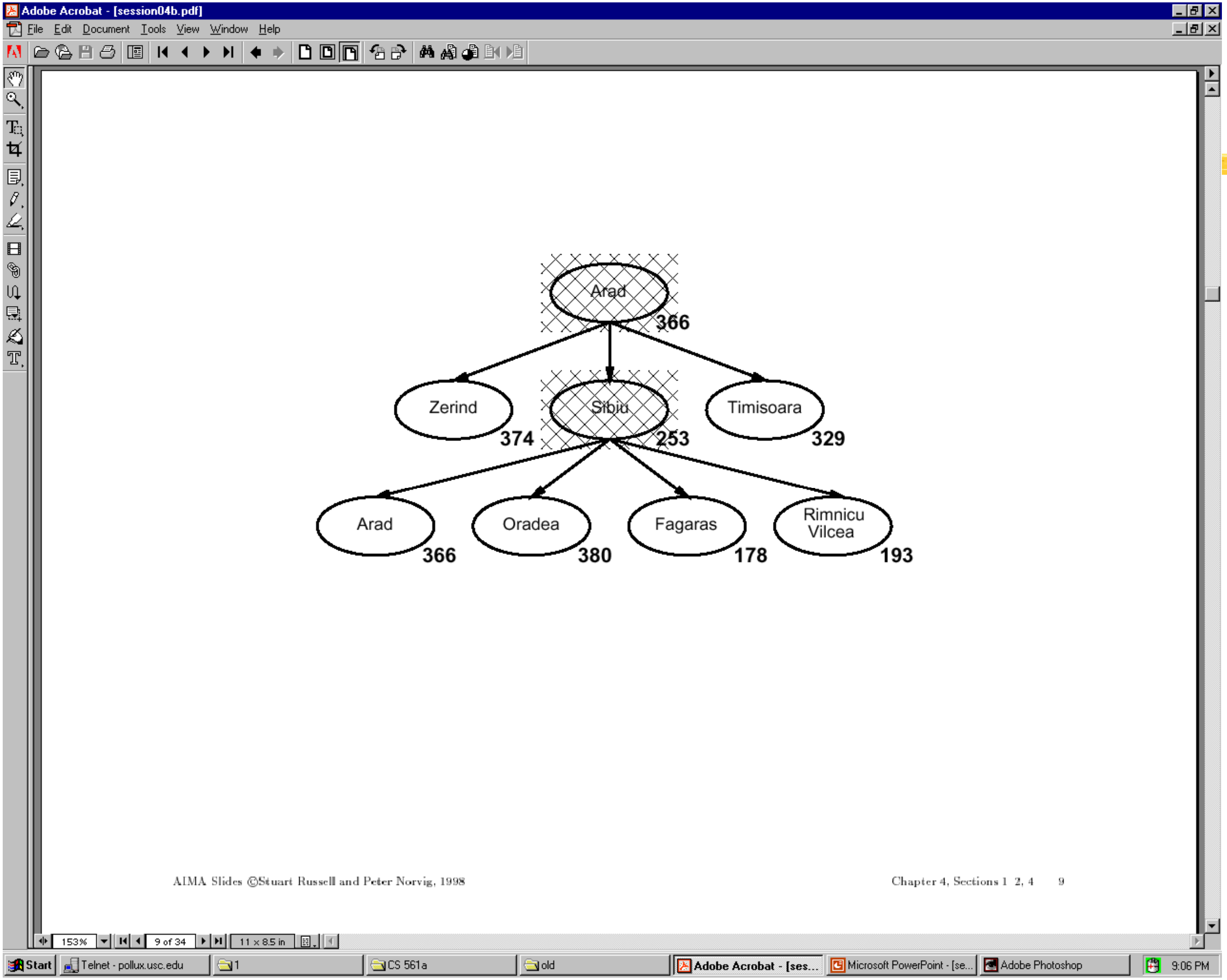
- For example:

  $h_{SLD}(n)$ = **straight-line distance** from $n$ to Bucharest

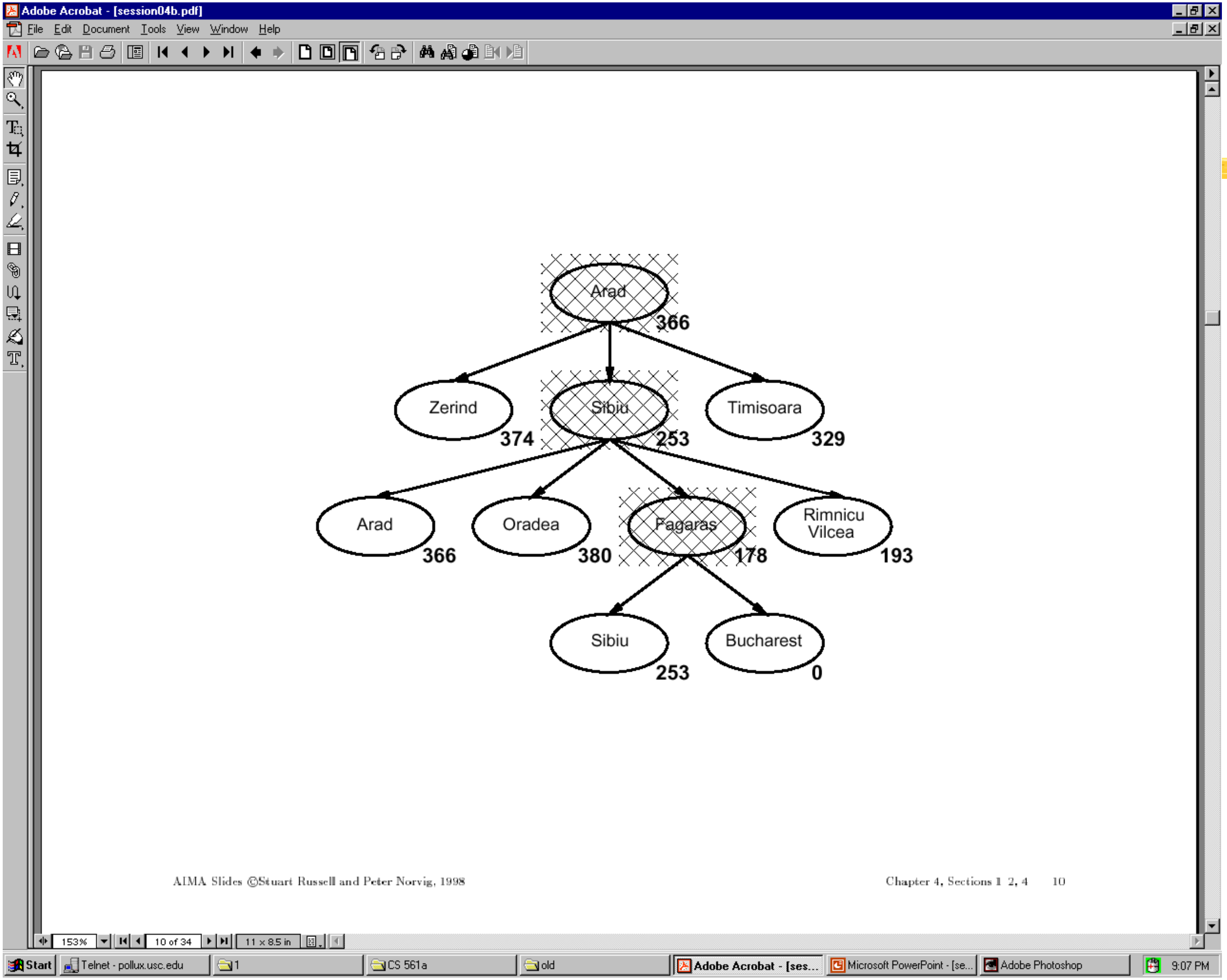
- Greedy search expands first the node that **appears promising** to be closest to the goal, according to $h(n)$. I.e., node expanded will be the one with the best $h(n)$

# Greedy search example


Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
Zerind
374
Sibiu
253
Timisoara
329
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4 8
153%
8 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:06 PM



Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
Zerind
374
Sibiu
253
Timisoara
329
Arad
366
Oradea
380
Fagaras
178
Rimnicu Vilcea
193
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4
9
153%
9 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:06 PM


Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
Zerind
374
Sibiu
253
Timisoara
329
Arad
366
Oradea
380
Fagaras
178
Rimnicu Vilcea
193
Sibiu
253
Bucharest
0
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4 10
153%
10 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:07 PM

# Properties of Greedy Search

- Complete?

- Time?

- Space?

- Optimal?

# Properties of Greedy Search

- Complete? No – can get stuck in loops
  e.g., Iasi > Neamt > Iasi > Neamt > …
  Complete in finite space with repeated-state checking.
- Time? O(b^m) but a "good" heuristic can provide dramatic improvement
- Space? O(b^m) – keeps all nodes in memory
- Optimal? No.

## A* search

- Basic Idea: avoid expanding paths that are **already** expensive

  evaluation function: $f(n) = g(n) + h(n)$ with:

  $g(n)$ – cost so far to reach $n$

  $h(n)$ – estimated cost to goal from $n$

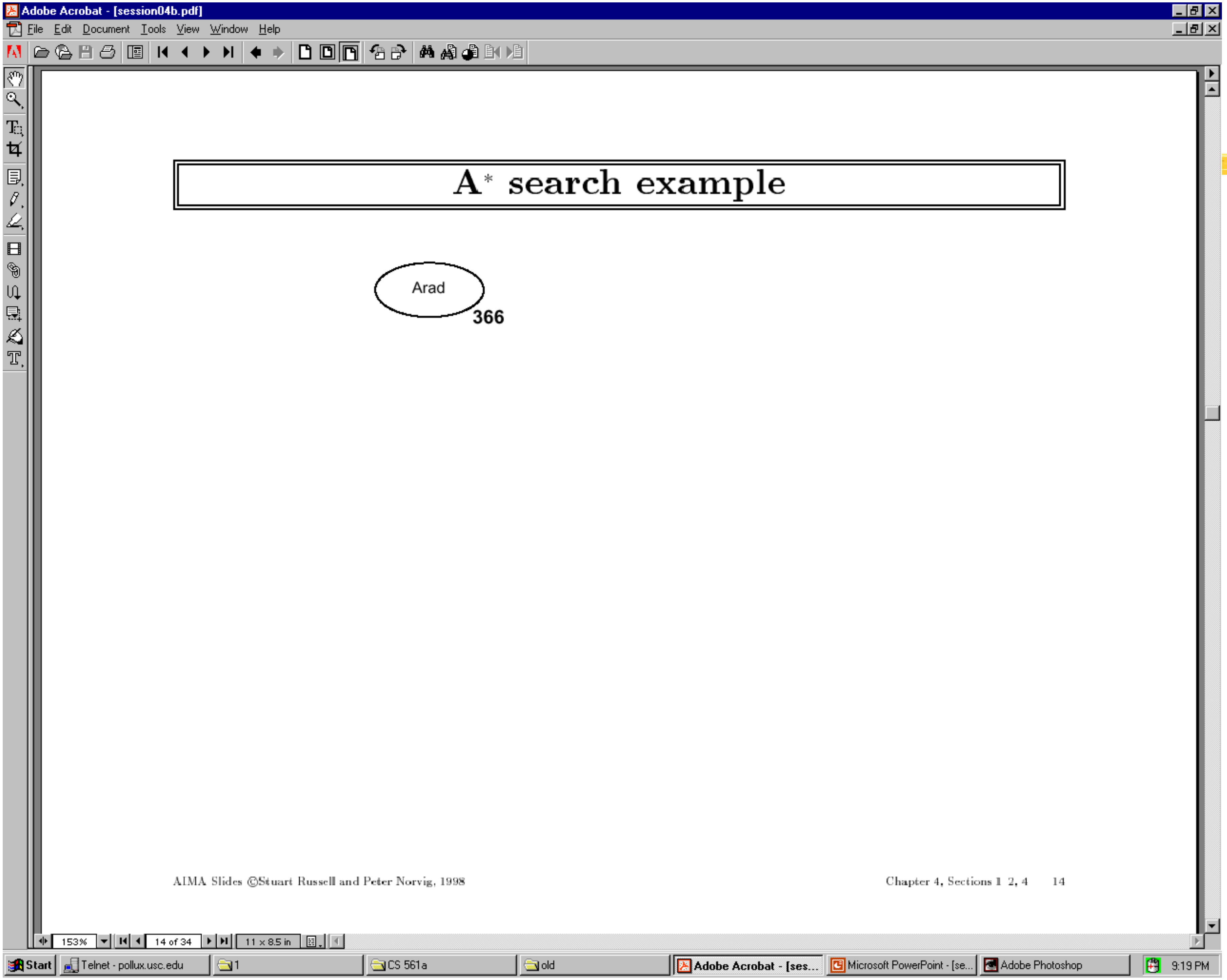
  $f(n)$ – estimated total cost of path through $n$ to goal

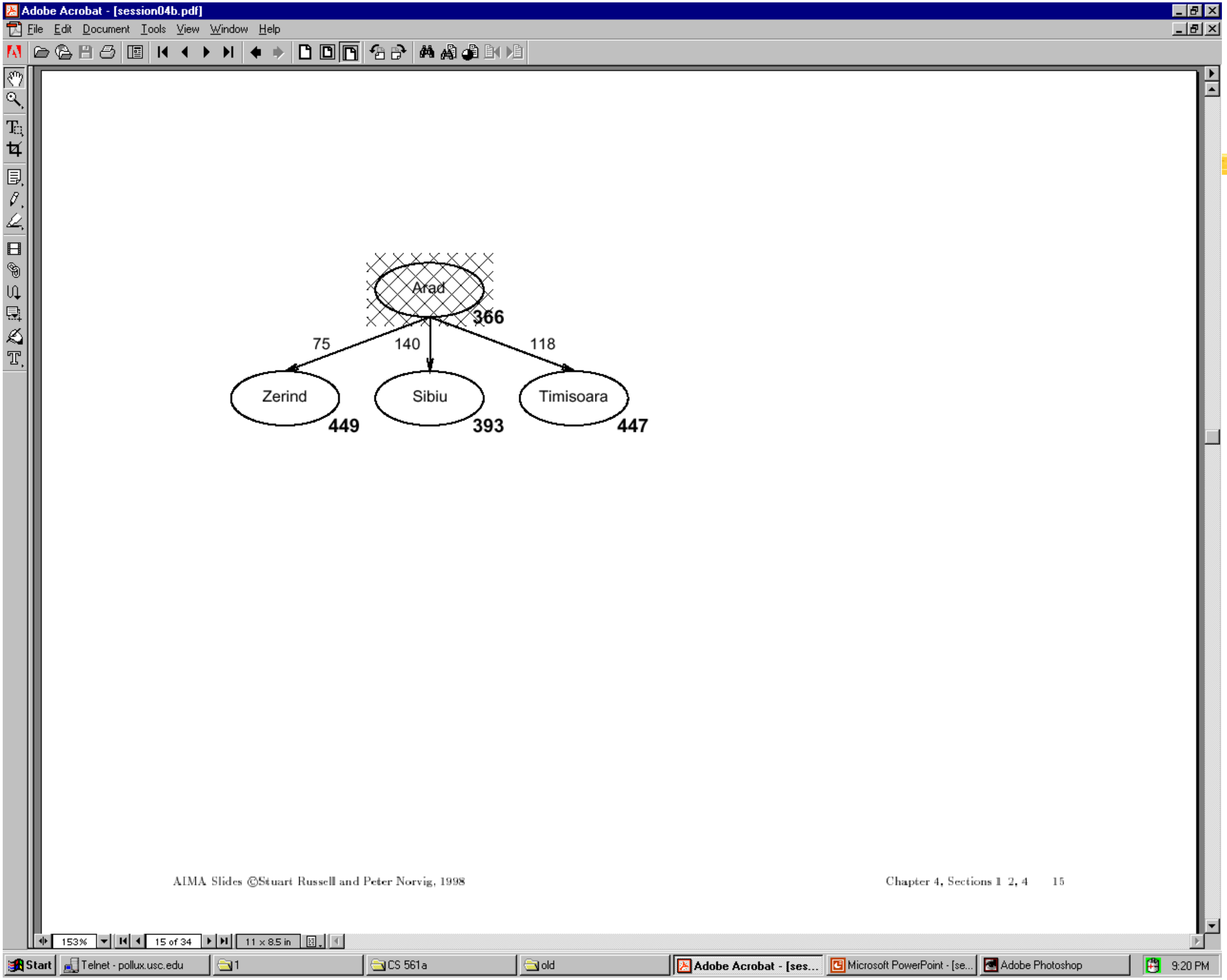
- A* search uses an **admissible** heuristic, that is,

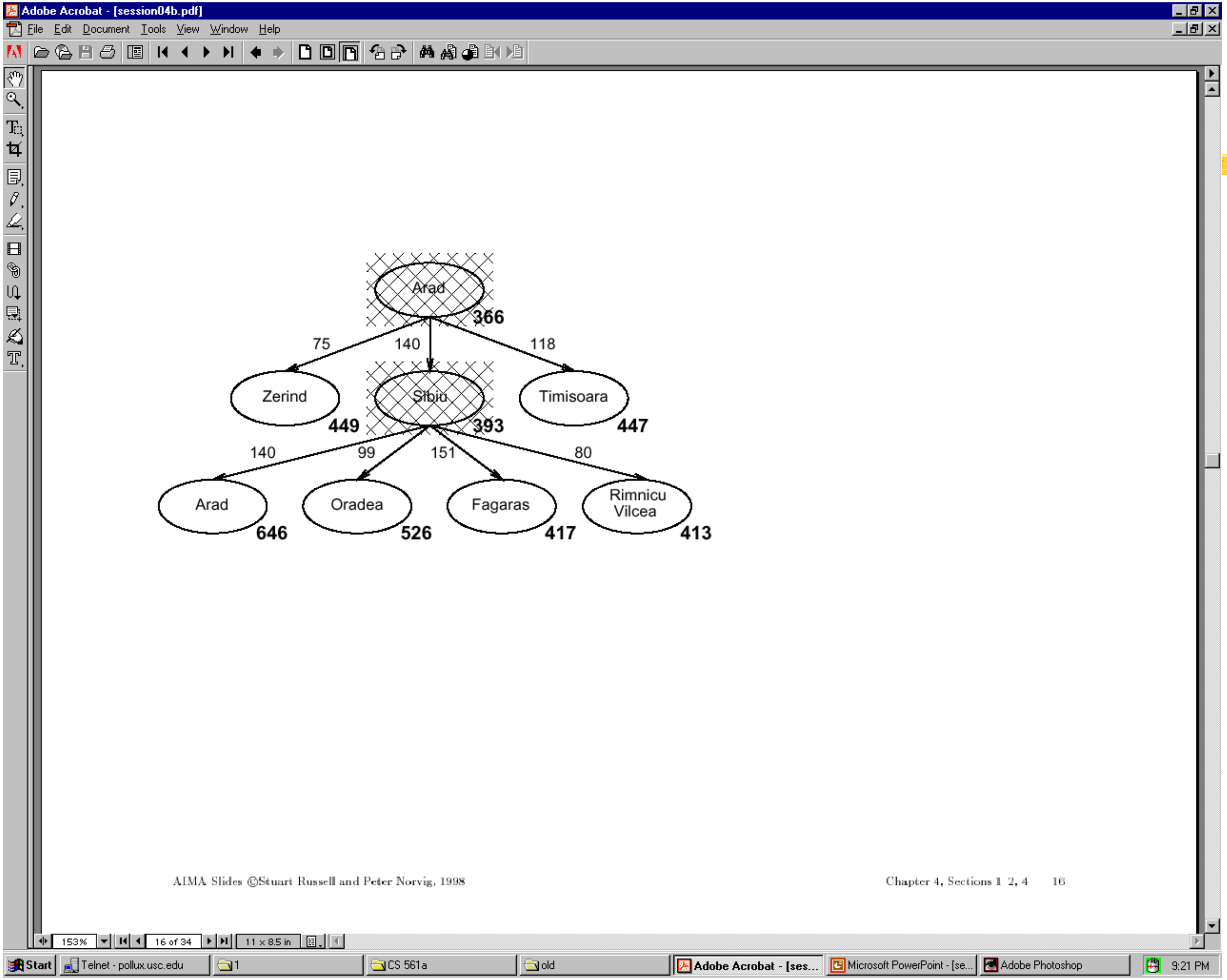
  $h(n) \le h^*(n)$ where $h^*(n)$ is the **true or actual** cost from $n$.

  For example: $h_{SLD}(n)$ never overestimates actual road distance from n to the destination.

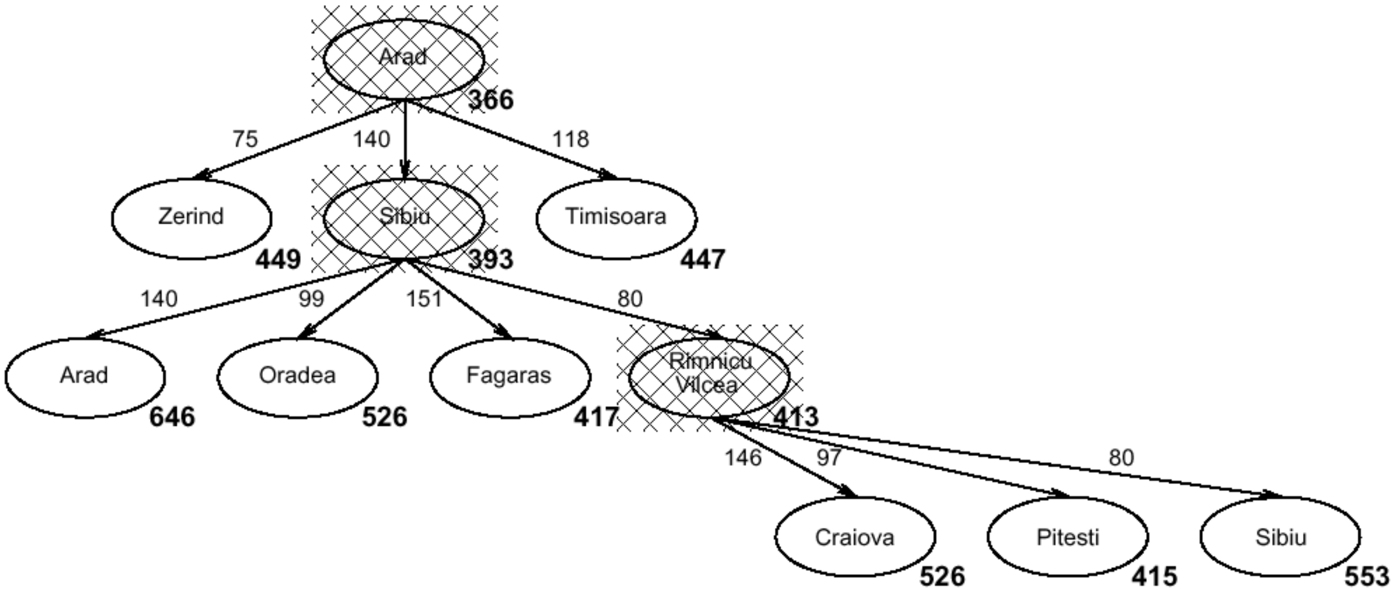
- Theorem: A* search is optimal

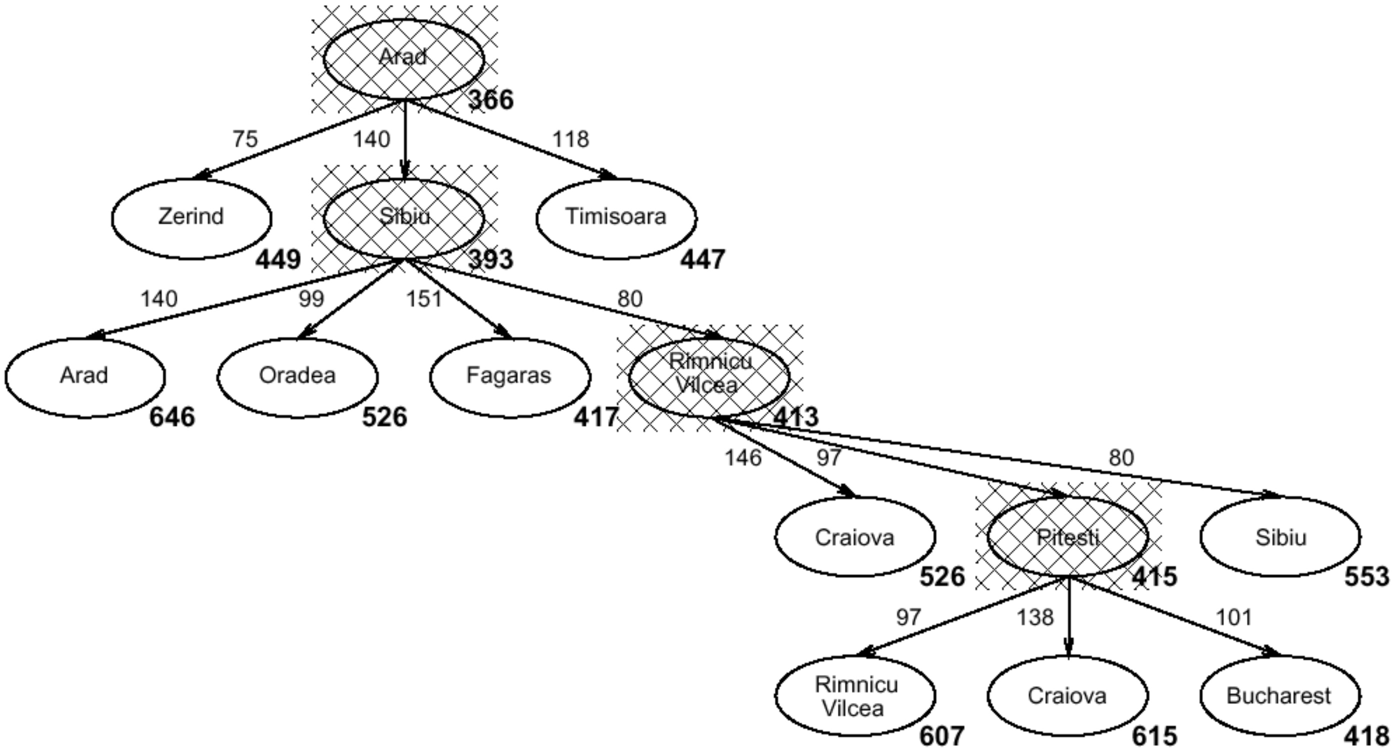

Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
A* search example
Arad
366
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4 14
153%
14 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:19 PM

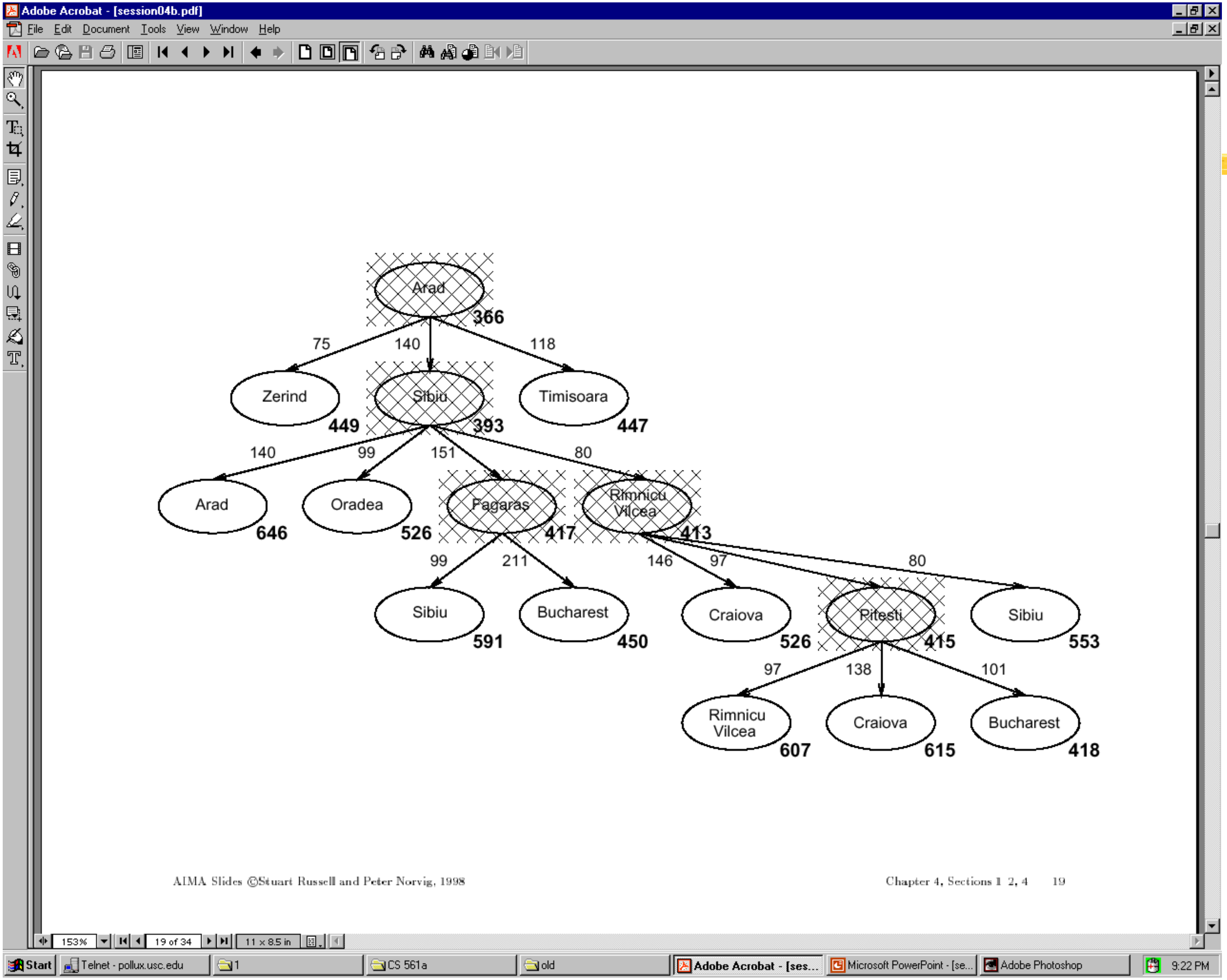

Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4 15
153%
15 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:20 PM


Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
99
151
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1 2, 4
16
153%
16 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:21 PM

Adobe Acrobat - [session04b.pdf]

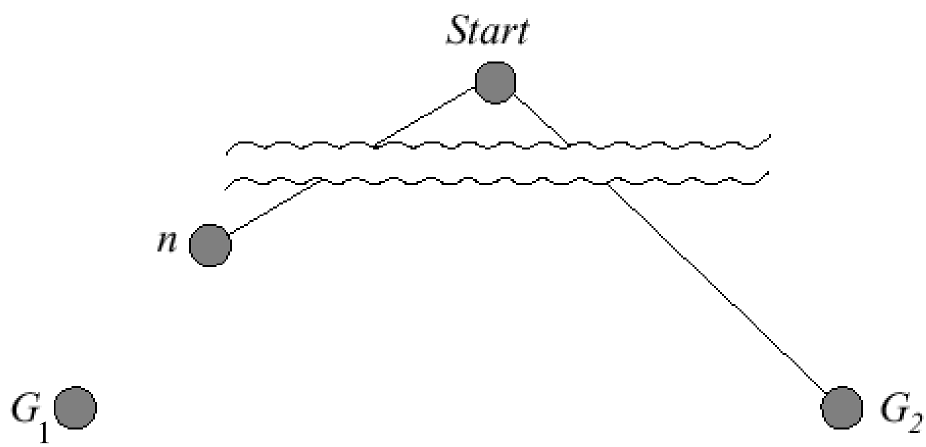



Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
99
151
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
146
97
80
Craiova
526
Pitesti
415
Sibiu
553
97
138
101
Rimnicu Vilcea
607
Craiova
615
Bucharest
418


Adobe Acrobat - [session04b.pdf]
File Edit Document Tools View Window Help
Arad
366
75
140
118
Zerind
449
Sibiu
393
Timisoara
447
140
99
151
80
Arad
646
Oradea
526
Fagaras
417
Rimnicu Vilcea
413
99
211
146
97
80
Sibiu
591
Bucharest
450
Craiova
526
Pitesti
415
Sibiu
553
97
138
101
Rimnicu Vilcea
607
Craiova
615
Bucharest
418
AIMA Slides ©Stuart Russell and Peter Norvig, 1998
Chapter 4, Sections 1–2, 4 19
153%
19 of 34
11 x 8.5 in
Start
Telnet - pollux.usc.edu
1
CS 561a
old
Adobe Acrobat - [ses...
Microsoft PowerPoint - [se...
Adobe Photoshop
9:22 PM

# Optimality of A* (an outline of standard proof)

Suppose some suboptimal goal $G_2$ has been generated and is in the queue. Let $n$ be an unexpanded node on a shortest path to an optimal goal $G_1$.



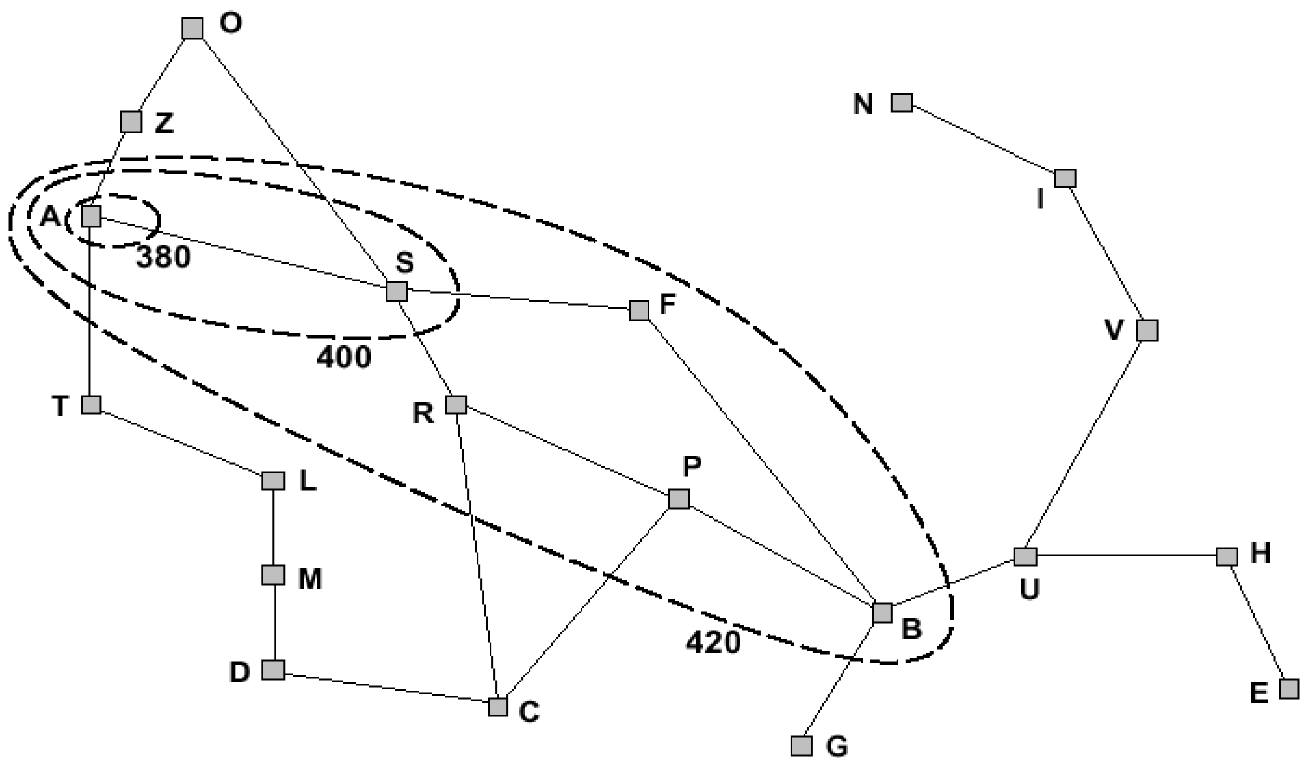

$$\begin{aligned} f(G_2) &= g(G_2) && \text{since } h(G_2) = 0 \\ &> g(G_1) && \text{since } G_2 \text{ is suboptimal} \\ &\geq f(n) && \text{since } h \text{ is admissible} \end{aligned}$$

Since $f(G_2) > f(n)$, A* will never select $G_2$ for expansion

# Optimality of A* (Another, perhaps more useful proof)

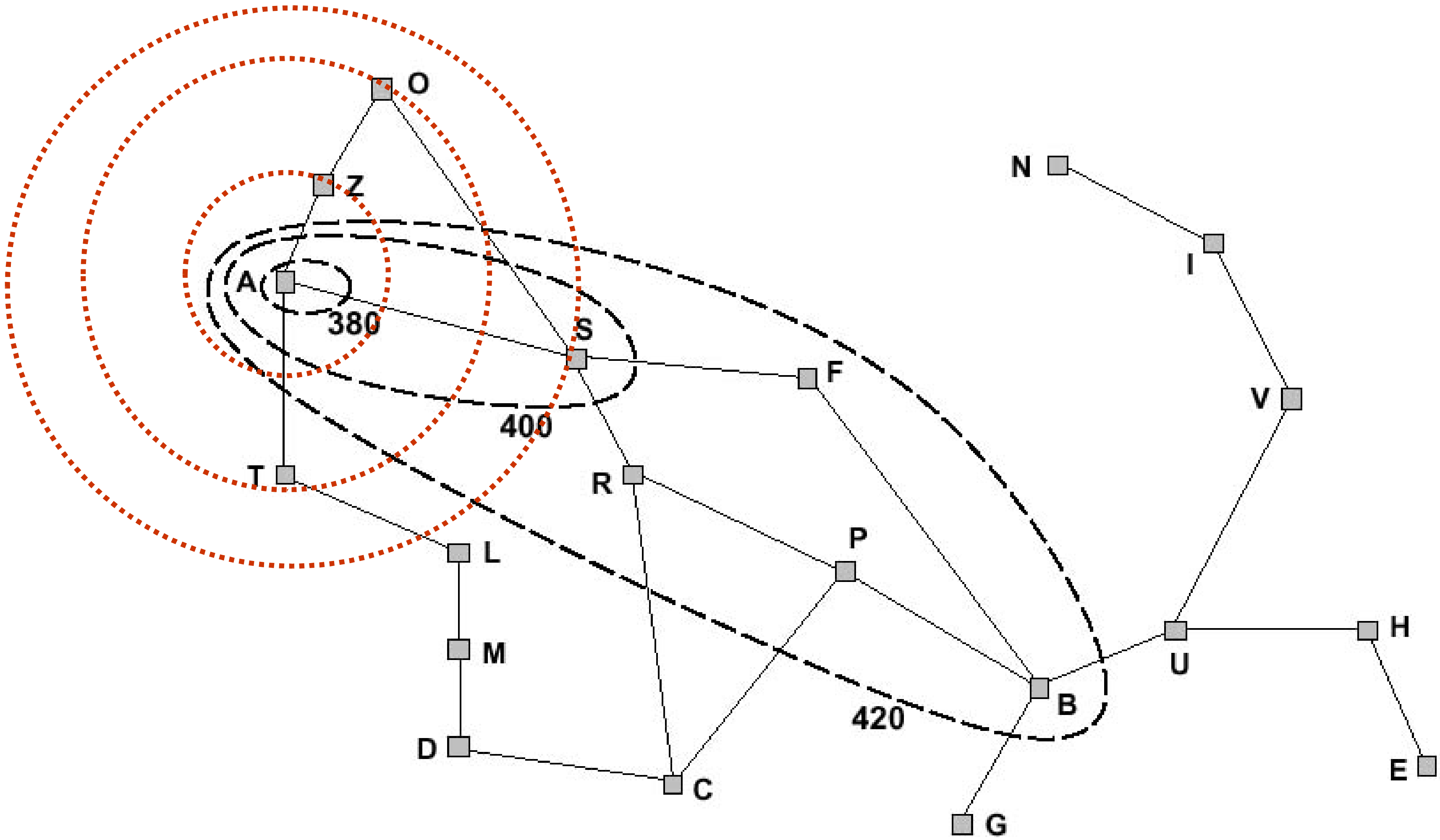
Lemma: A* expands nodes in order of increasing $f$ value

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



# f-contours

How do the contours look like when h(n) =0?



## Properties of A*

- Complete?
- Time?
- Space?
- Optimal?

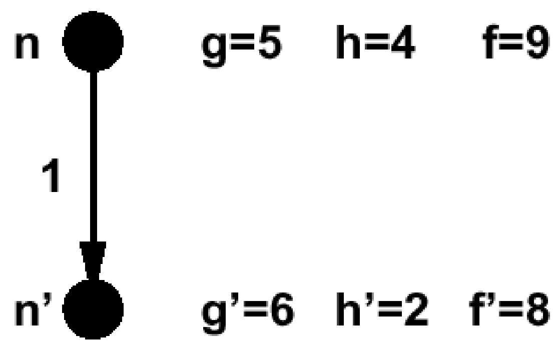
# Properties of A*

- Complete? Yes, unless infinitely many nodes with $f \leq f(G)$
- Time? Exponential in [(relative error in h) x (length of solution)]
- Space? Keeps all nodes in memory
- Optimal? Yes – cannot expand $f_{i+1}$ until $f_i$ is finished

## Proof of lemma: pathmax

For some admissible heuristics, $f$ may *decrease* along a path

E.g., suppose $n'$ is a successor of $n$

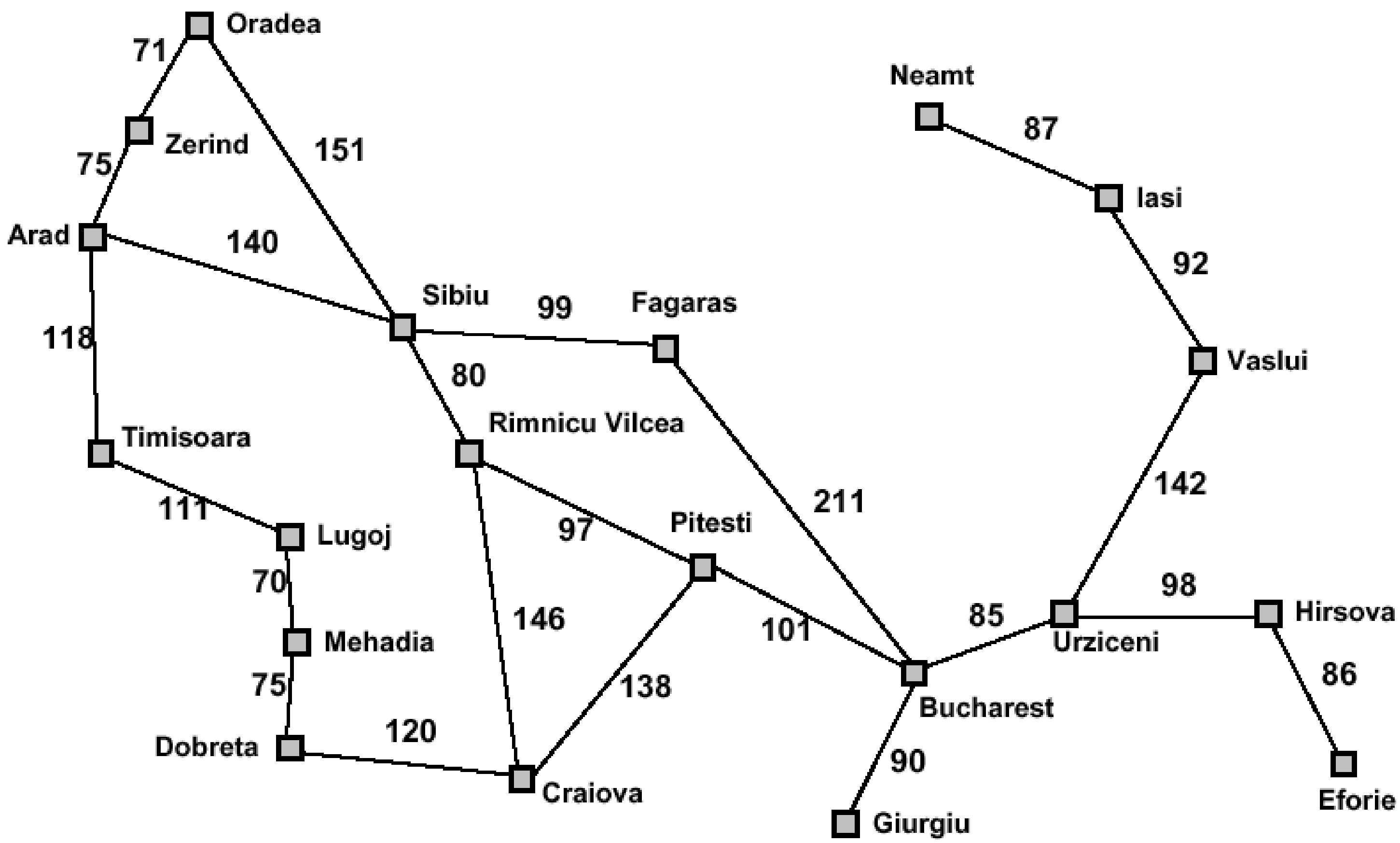



But this throws away information!
$f(n) = 9 \Rightarrow$ true cost of a path through $n$ is $\geq 9$
Hence true cost of a path through $n'$ is $\geq 9$ also

Pathmax modification to A$^*$:
Instead of $f(n') = g(n') + h(n')$, use $f(n') = max(g(n') + h(n'), f(n))$

With pathmax, $f$ is always nondecreasing along any path

# Romania with step costs in km – SLD heuristic




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 |

# Exercise: Search Algorithms

The following figure shows a portion of a partially expanded search tree. Each arc between nodes is labeled with the cost of the corresponding operator, and the leaves are labeled with the value of the heuristic function, *h*.

Which node (use the node's letter) will be expanded next by each of the following search algorithms?

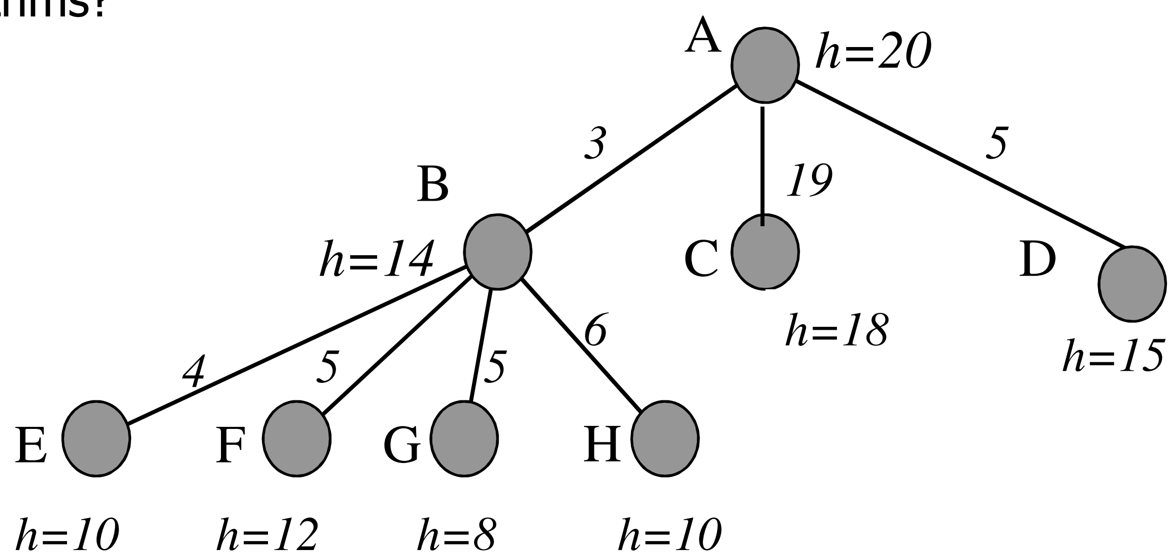
(a) Depth-first search

(b) Breadth-first search

(c) Uniform-cost search

(d) Greedy search

(e) A* search



# Depth-first search

Node queue: initialization

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |

# Depth-first search

Node queue: add successors to queue front; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |
| 1 | A | 0 | 0 | -- |

## Depth-first search

Node queue: add successors to queue front; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |
| 1 | A | 0 | 0 | -- |

## Depth-first search

Node queue: add successors to queue front; empty queue from top

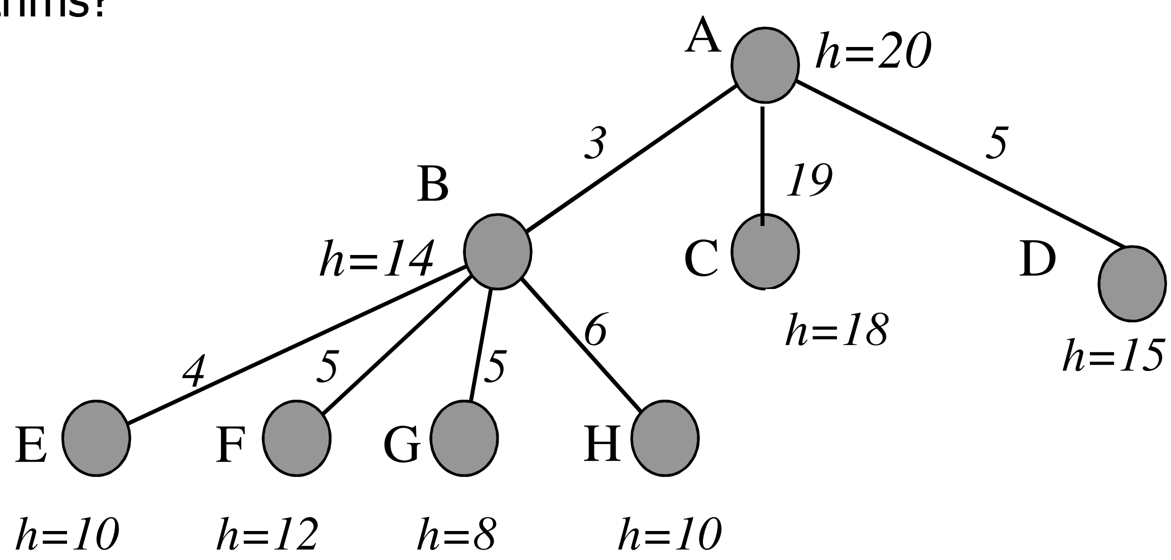
| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |
| 1 | A | 0 | 0 | -- |

# Exercise: Search Algorithms

The following figure shows a portion of a partially expanded search tree. Each arc between nodes is labeled with the cost of the corresponding operator, and the leaves are labeled with the value of the heuristic function, *h*.

Which node (use the node's letter) will be expanded next by each of the following search algorithms?

(a) Depth-first search
(b) Breadth-first search
(c) Uniform-cost search
(d) Greedy search
(e) A* search



# Breadth-first search

Node queue: initialization

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |

# Breadth-first search

Node queue: add successors to queue end; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |

# Breadth-first search

Node queue: add successors to queue end; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |

# Breadth-first search

Node queue: add successors to queue end; empty queue from top

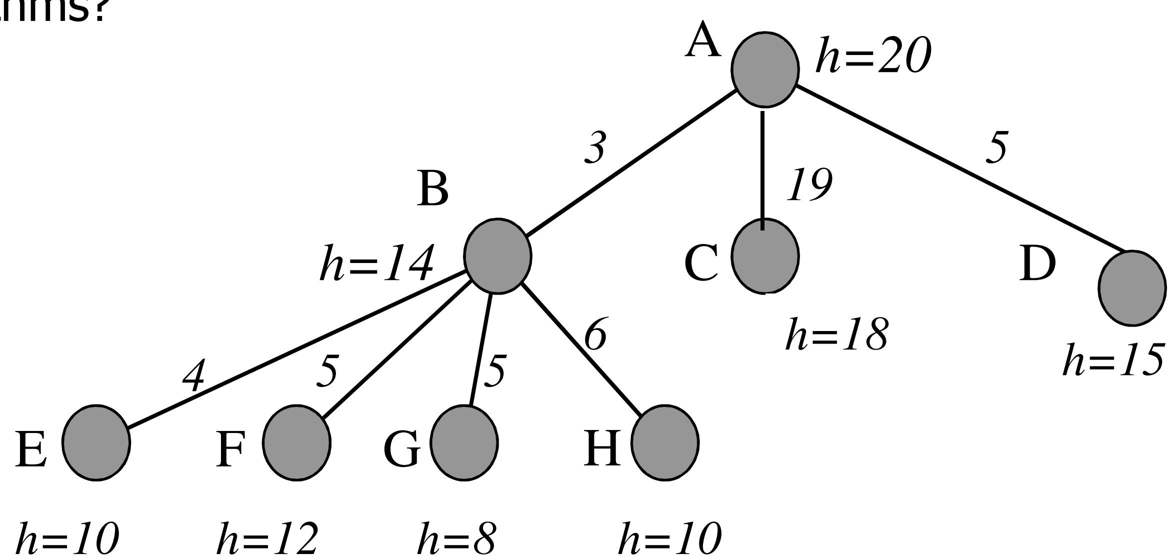
| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | C | 1 | 19 | 1 |
| 4 | D | 1 | 5 | 1 |
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |

# Exercise: Search Algorithms

The following figure shows a portion of a partially expanded search tree. Each arc between nodes is labeled with the cost of the corresponding operator, and the leaves are labeled with the value of the heuristic function, *h*.

Which node (use the node's letter) will be expanded next by each of the following search algorithms?

(a) Depth-first search
(b) Breadth-first search
(c) Uniform-cost search
(d) Greedy search
(e) A* search



## Uniform-cost search

Node queue: initialization

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |

# Uniform-cost search

Node queue: add successors to queue so that entire queue is sorted by path cost so far; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | D | 1 | 5 | 1 |
| 4 | C | 1 | 19 | 1 |

# Uniform-cost search

Node queue: add successors to queue so that entire queue is sorted by path cost so far; empty queue from top

| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | D | 1 | 5 | 1 |
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |
| 4 | C | 1 | 19 | 1 |

# Uniform-cost search

Node queue: add successors to queue so that entire queue is sorted by path cost so far; empty queue from top

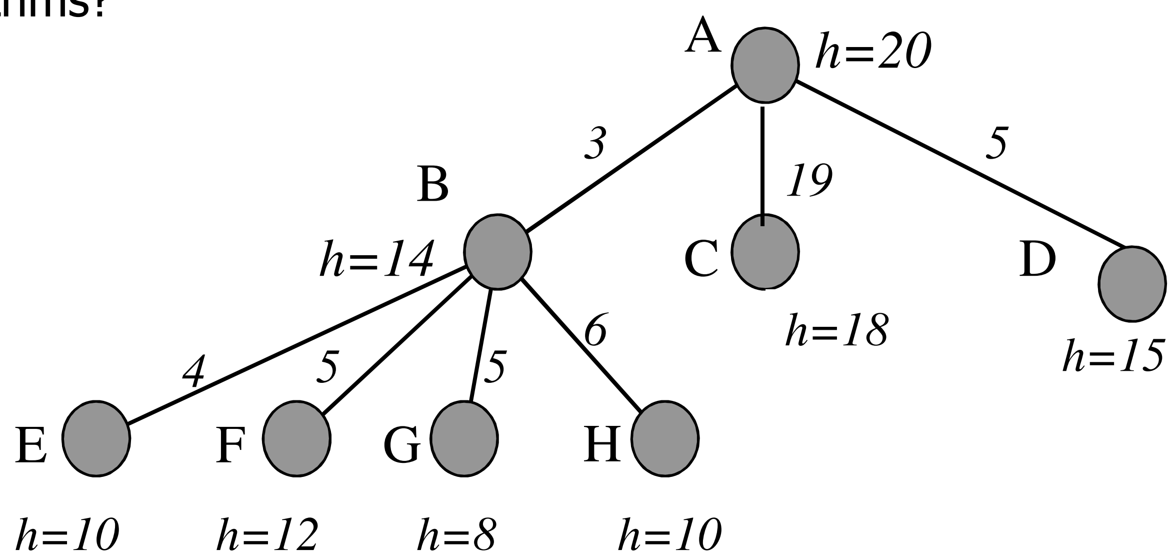
| # | state | depth | path cost | parent # |
|---|---|---|---|---|
| 1 | A | 0 | 0 | -- |
| 2 | B | 1 | 3 | 1 |
| 3 | D | 1 | 5 | 1 |
| 5 | E | 2 | 7 | 2 |
| 6 | F | 2 | 8 | 2 |
| 7 | G | 2 | 8 | 2 |
| 8 | H | 2 | 9 | 2 |
| 4 | C | 1 | 19 | 1 |

## Exercise: Search Algorithms

The following figure shows a portion of a partially expanded search tree. Each arc between nodes is labeled with the cost of the corresponding operator, and the leaves are labeled with the value of the heuristic function, *h*.

Which node (use the node's letter) will be <u>expanded</u> next by each of the following search algorithms?

(a) Depth-first search
(b) Breadth-first search
(c) Uniform-cost search
(d) Greedy search
(e) A* search



# Greedy search

Node queue: initialization

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |

# Greedy search

Node queue: Add successors to queue, sorted by cost to goal.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

Sort key

# Greedy search

Node queue: Add successors to queue, sorted by cost to goal.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 5 | G | 2 | 8 | 8 | 16 | 2 |
| 7 | E | 2 | 7 | 10 | 17 | 2 |
| 6 | H | 2 | 9 | 10 | 19 | 2 |
| 8 | F | 2 | 8 | 12 | 20 | 2 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

## Greedy search

Node queue: Add successors to queue, sorted by cost to goal.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 5 | G | 2 | 8 | 8 | 16 | 2 |
| 7 | E | 2 | 7 | 10 | 17 | 2 |
| 6 | H | 2 | 9 | 10 | 19 | 2 |
| 8 | F | 2 | 8 | 12 | 20 | 2 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

# Exercise: Search Algorithms

The following figure shows a portion of a partially expanded search tree. Each arc between nodes is labeled with the cost of the corresponding operator, and the leaves are labeled with the value of the heuristic function, *h*.

Which node (use the node's letter) will be expanded next by each of the following search algorithms?

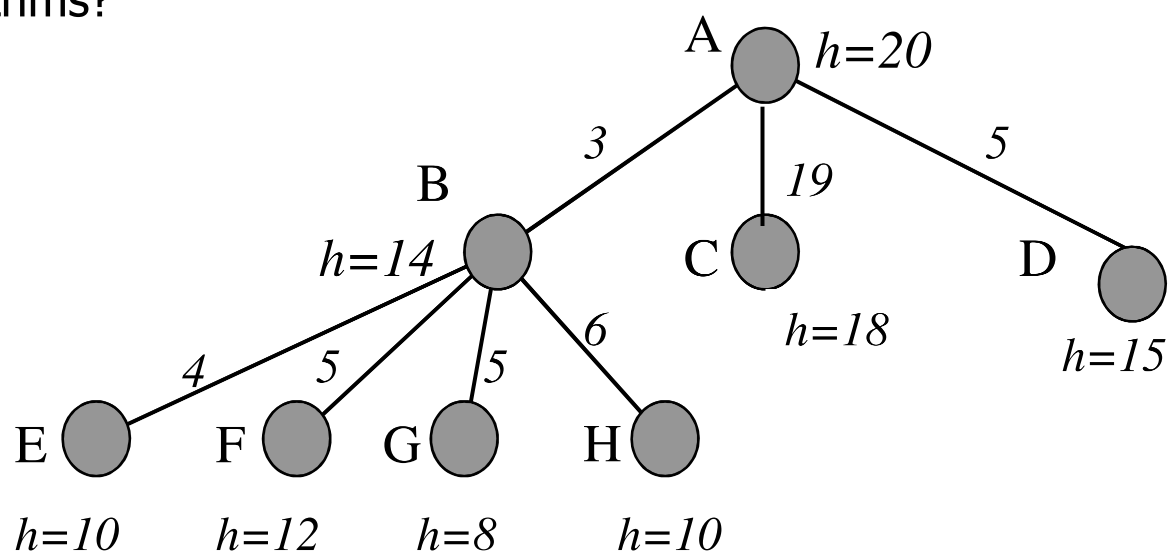
(a) Depth-first search

(b) Breadth-first search

(c) Uniform-cost search

(d) Greedy search

(e) A* search



# A* search

Node queue: initialization

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |

# A* search

Node queue: Add successors to queue, sorted by total cost.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

Sort key

# A* search

Node queue: Add successors to queue front, sorted by total cost.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 5 | G | 2 | 8 | 8 | 16 | 2 |
| 6 | E | 2 | 7 | 10 | 17 | 2 |
| 7 | H | 2 | 9 | 10 | 19 | 2 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 8 | F | 2 | 8 | 12 | 20 | 2 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

# A* search

Node queue: Add successors to queue front, sorted by total cost.

| # | state | depth | path cost | cost to goal | total cost | parent # |
|---|---|---|---|---|---|---|
| 1 | A | 0 | 0 | 20 | 20 | -- |
| 2 | B | 1 | 3 | 14 | 17 | 1 |
| 5 | G | 2 | 8 | 8 | 16 | 2 |
| 6 | E | 2 | 7 | 10 | 17 | 2 |
| 7 | H | 2 | 9 | 10 | 19 | 2 |
| 3 | D | 1 | 5 | 15 | 20 | 1 |
| 8 | F | 2 | 8 | 12 | 20 | 2 |
| 4 | C | 1 | 19 | 18 | 37 | 1 |

## Admissible 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)

| | | |
|---|---|---|
| 5 | 4 | |
| 6 | 1 | 8 |
| 7 | 3 | 2 |

**Start State**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 8 | | 4 |
| 7 | 6 | 5 |

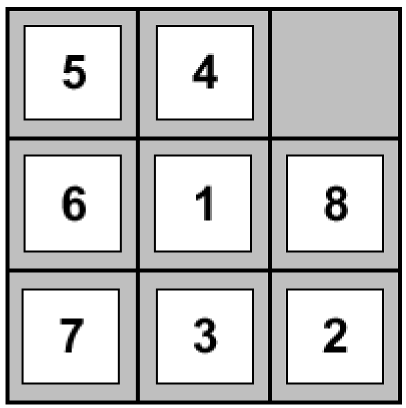
**Goal State**

$h_1(S)$ =??
$h_2(S)$ =??

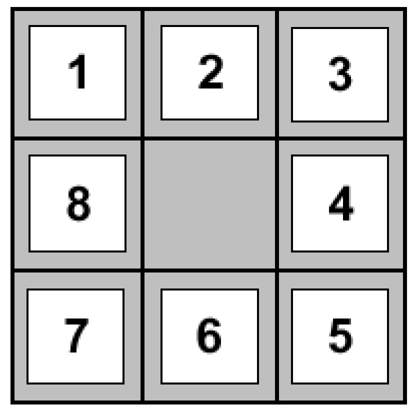
## Admissible heuristics

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)




Start State




Goal State

$h_1(S) = ??$ 7
$h_2(S) = ??$ 2+3+3+2+4+2+0+2 = 18

## Admissibility By Defining a “Relaxed” Problem

- 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.