

# Day 3

## Informed Search

(some slides based on Downy, Sood)

- Class business
  - Piazza/OH still best way to ask questions
  - Don't use comments in canvas (we can't easily see them)
  - Lab 0 grades posted on canvas
    - If you struggled with lab 0, I would be concerned about your ability to do well in this class.
    - Lab 0 first resubmission due April 10, 7pm
  - Lab 1 due April 14
- recap Uninformed search
- Informed search
  - Introduce lab 2

# Summary

- Search problems
- Search strategies
  - DFS, BFS, Depth limited, ID, Bi-directional
  - Issue: all these methods are slow (blind)
  - Can we fix this by adding guidance...
  - Next: Informed search

# BFS example

Exploration order: Up, Right, Down, Left

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<b>F</b>				
<b>G</b>	<b>H</b>	<b>I</b>		<b>J</b>
<b>K</b>				<b>L</b>
<b>M</b>	<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>

# DFS example

Exploration order: Up, Right, Down, Left

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>
<b>F</b>				
<b>G</b>	<b>H</b>	<b>I</b>		<b>J</b>
<b>K</b>				<b>L</b>
<b>M</b>	<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>

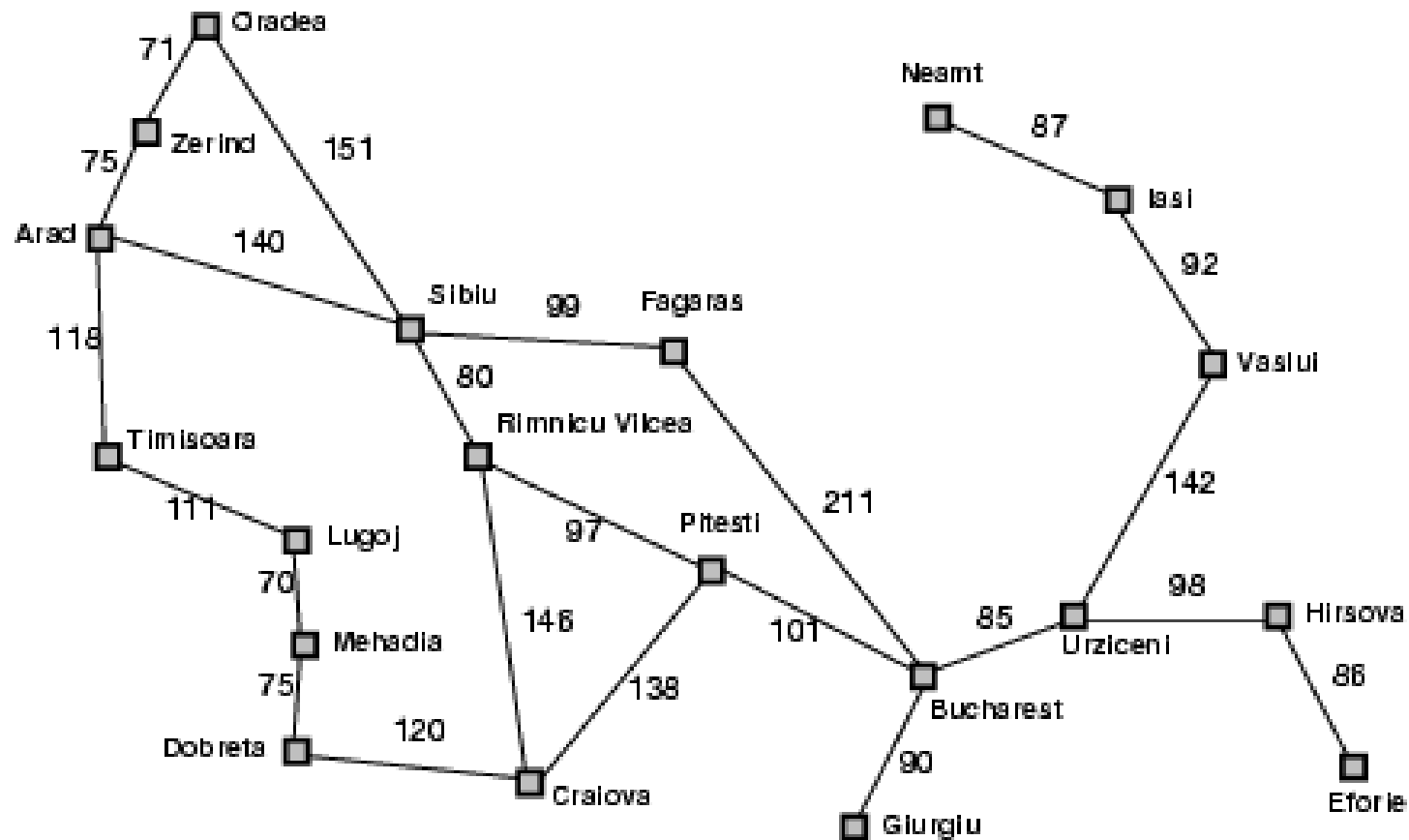
# Uninformed search

- DFS, Depth limited, ID
- BFS, Bi-directional
- Advantage of uninformed search?
- Disadvantages?
- Issue: all these methods are slow (blind)
- Can we fix this by adding guidance?

# Informed search

- A search strategy is defined by picking the order of node expansion
- Idea: use an evaluation function  $f(n)$  for each node
  - Estimate of “desirability” or quality
  - Expand most desirable nodes first
- Implementation: expand nodes on frontier in decreasing order of desirability

# example

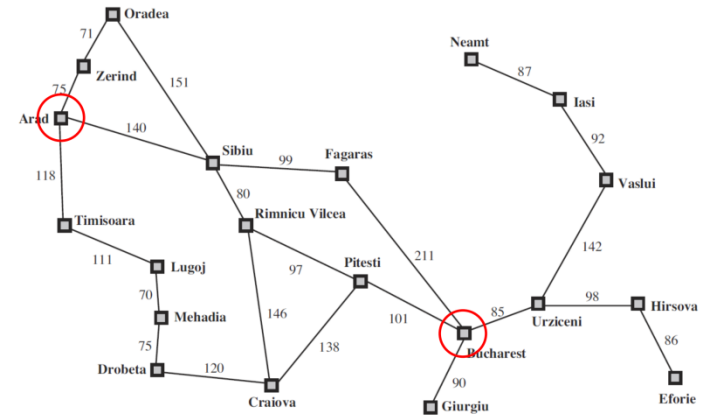


Straight-line distance  
to Bucharest

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

# Greedy best first

- Evaluation function  $f(n)=h(n)$  (heuristic)
  - Estimated cost from  $n$  to goal
    - Not computed from the problem itself, uses some expert knowledge
  - Example
    - $h_{sld}(n)$  = straight-line distance from  $n$  to goal
- Greedy best-first search expands the node that appears to be closest to goal



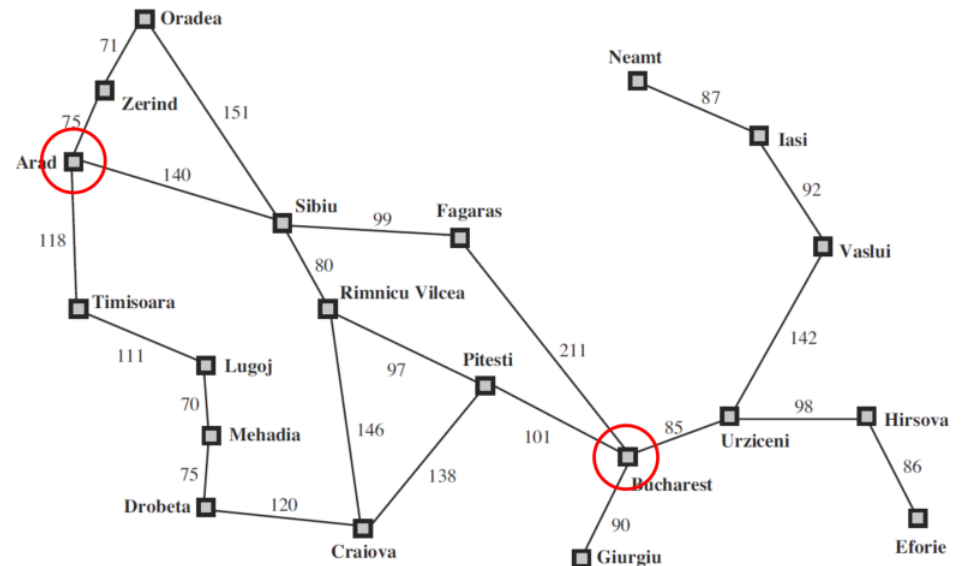


# tree search algorithm –keeping track of visited:

1. start with the initial node as curr
2. have expanded to curr before?
3. is curr the goal?
4. if neither, expand curr - add children/successors to frontier , add curr to expanded
5. choose a node in frontier according to the smallest  $f(n)=h(n)$  & go to step 2

# Greedy best-first search

- Evaluation function  $f(n) = h(n)$  (**h**euristic)  
= estimate of cost from  $n$  to *goal*
- e.g.,  $h_{SLD}(n)$  = straight-line distance from  $n$  to Bucharest

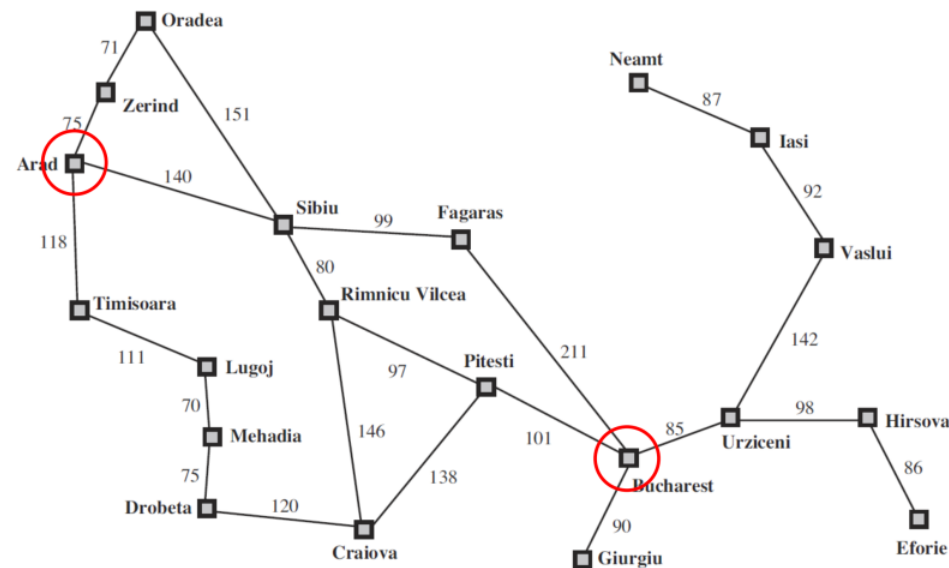


# Greedy best-first search example

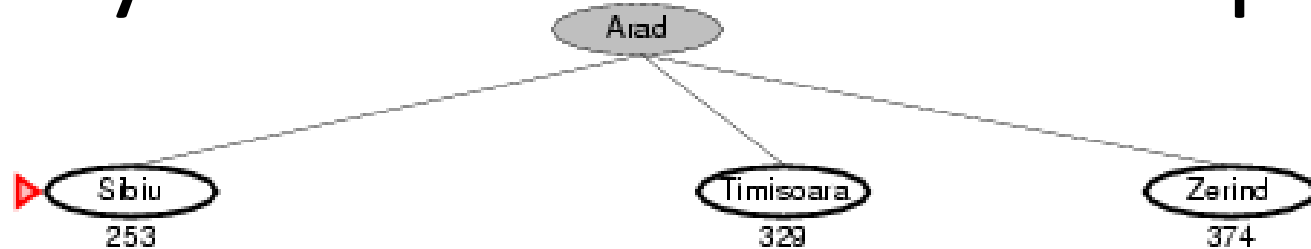


Straight-line distance  
to Bucharest

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

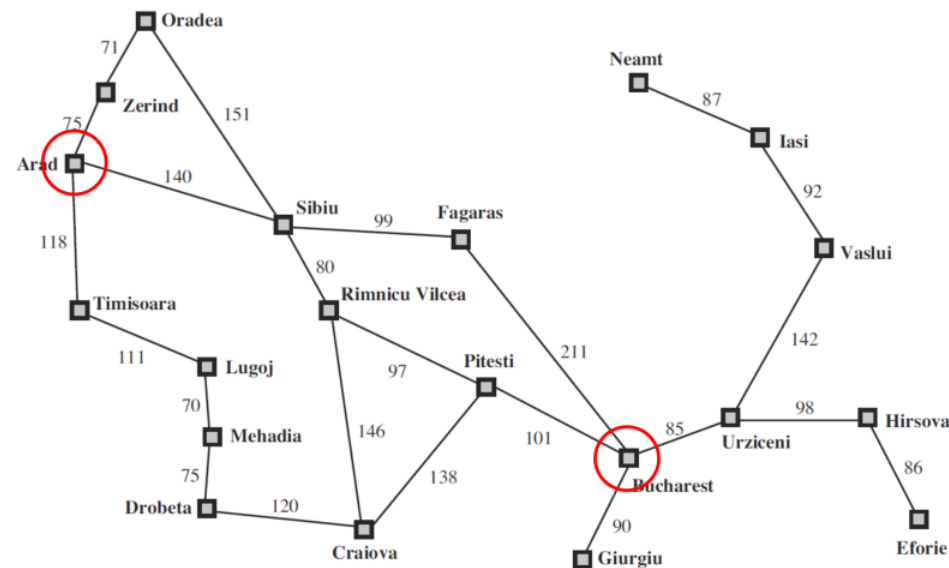


# Greedy best-first search example

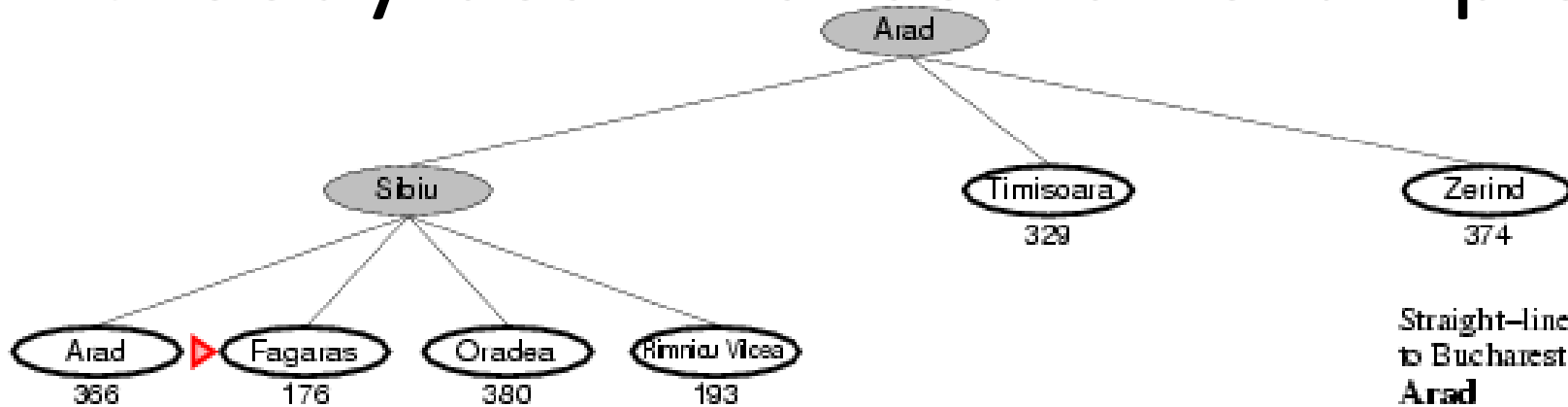


Straight-line distance  
to Bucharest

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

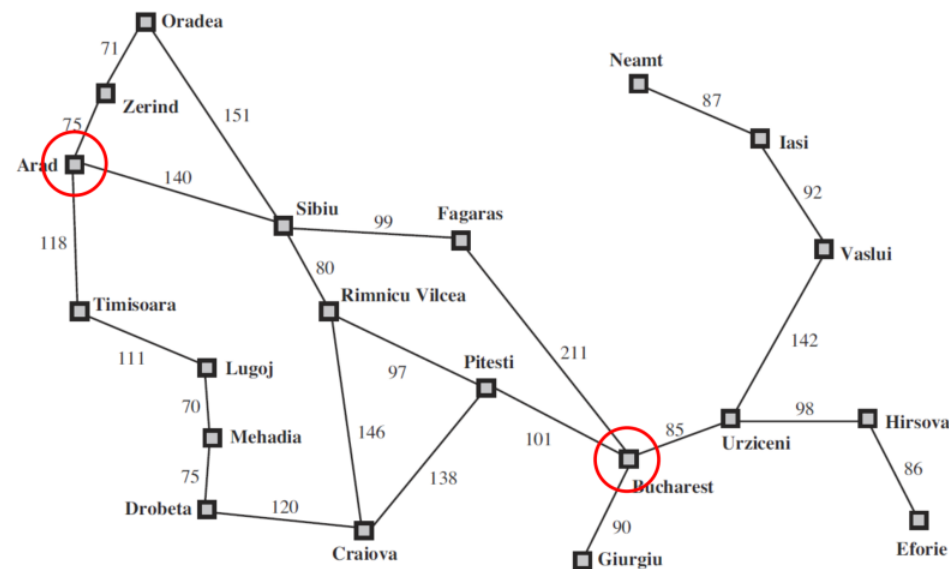


# Greedy best-first search example

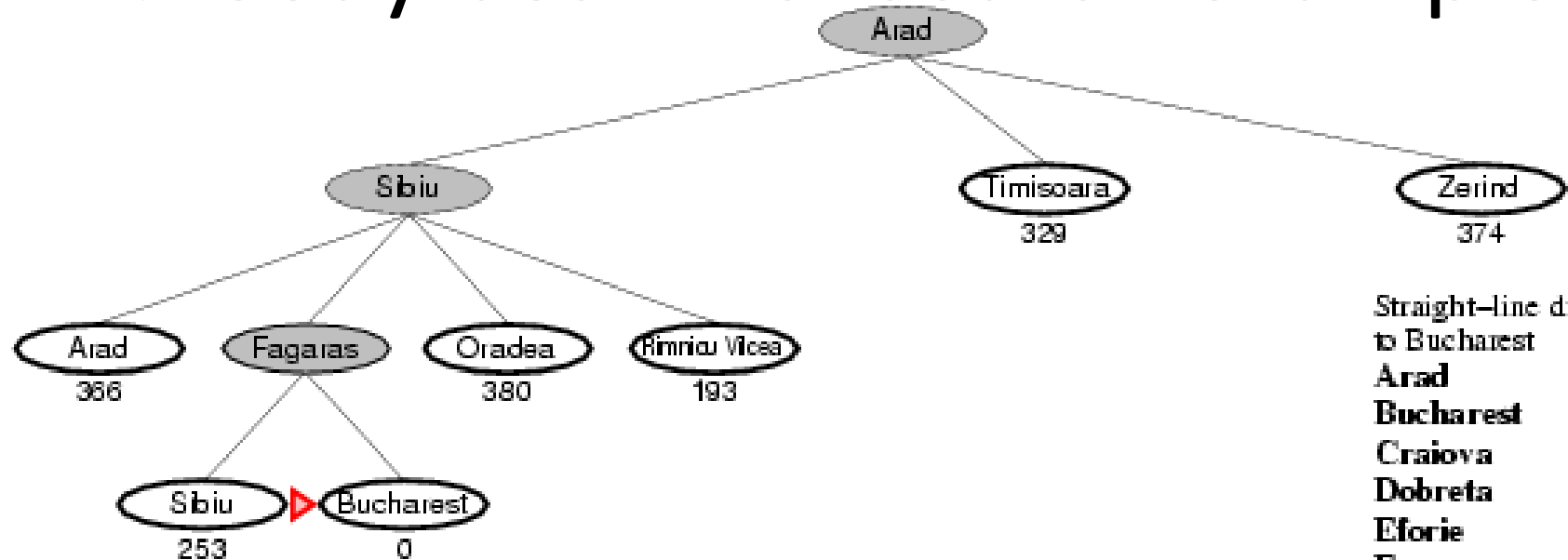


Straight-line distance  
to Bucharest

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

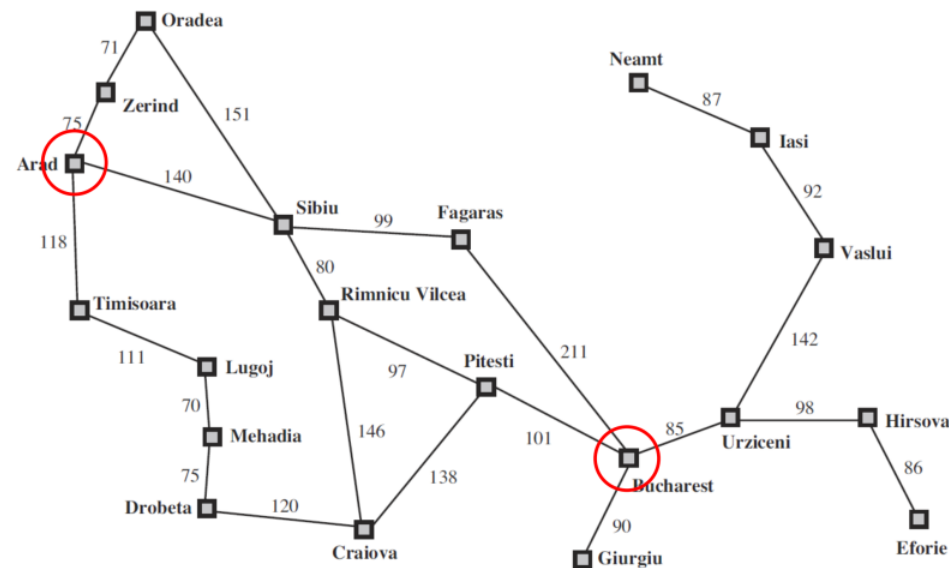


# Greedy best-first search example

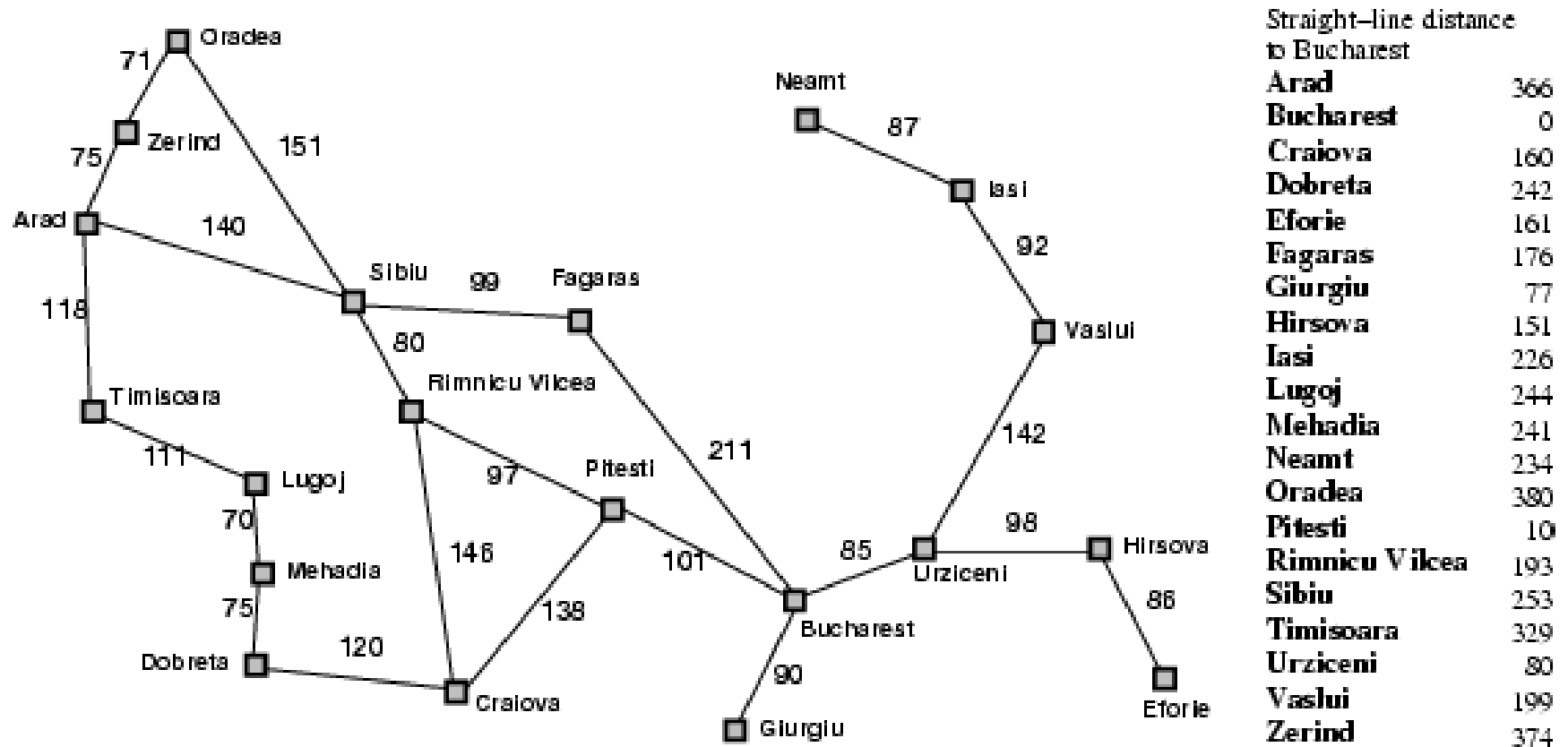


Straight-line distance  
to Bucharest

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



# Problem with greedy best-first search?

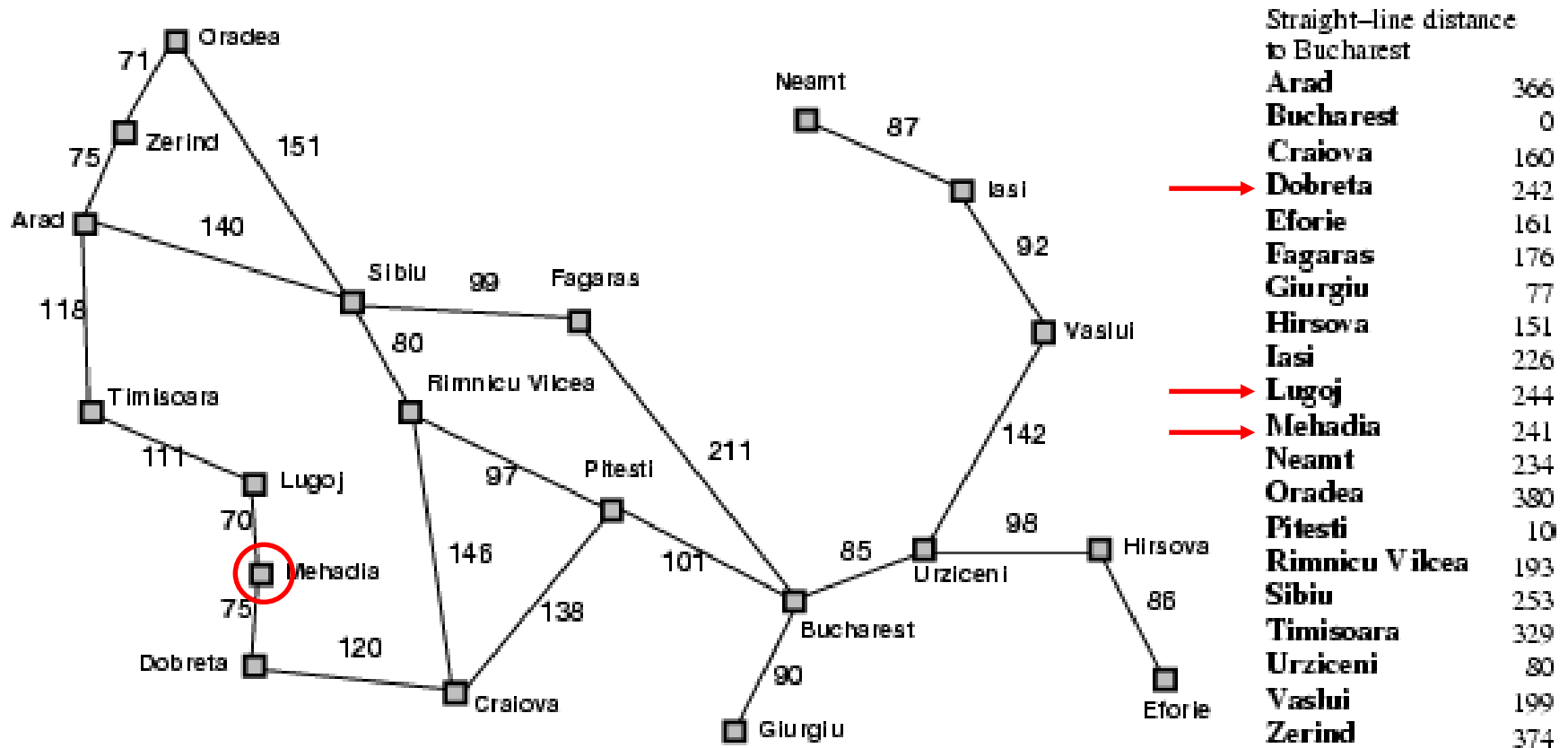


**Non-optimal (Arad to Bucharest)**

**Arad->Sibiu->Faragas->Bucharest is 23km longer than**

**Arad->Sibiu->R.V.->Pitesti->Bucharest**

# Problem with greedy best-first search?



**Non-optimal (Arad to Bucharest)**

**Arad->Sibiu->Faragas->Bucharest is 23km longer than**

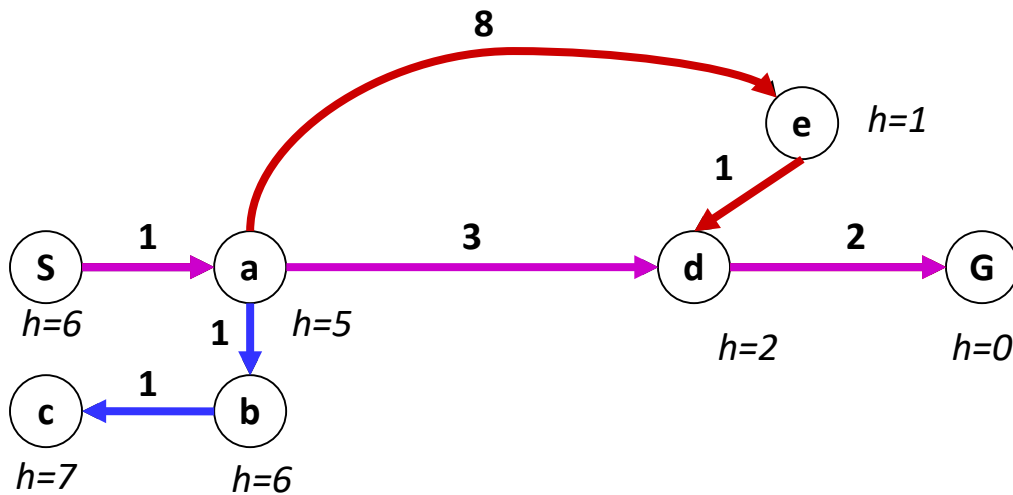
**Arad->Sibiu->R.V.->Pitesti->Bucharest**

Stuck in local minimum

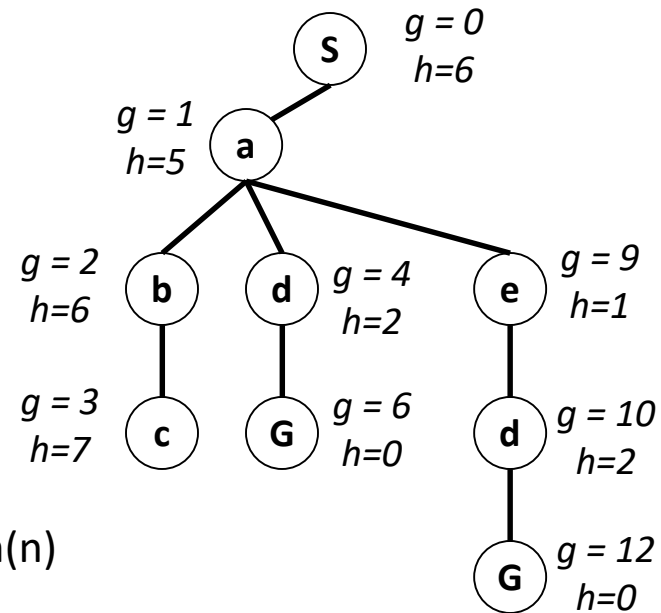


# Combining backward cost and Greedy

- backward-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\* search

- Idea: Estimate total path through nodes and avoid expanding paths that are already expensive
- Evaluation function  $f(n) = g(n) + h(n)$
- $g(n)$  = cost so far to reach  $n$
- $h(n)$  = estimated cost from  $n$  to goal
- $f(n)$  = estimated total cost of path through  $n$  to goal (the evaluation of the desirability of  $n$ )

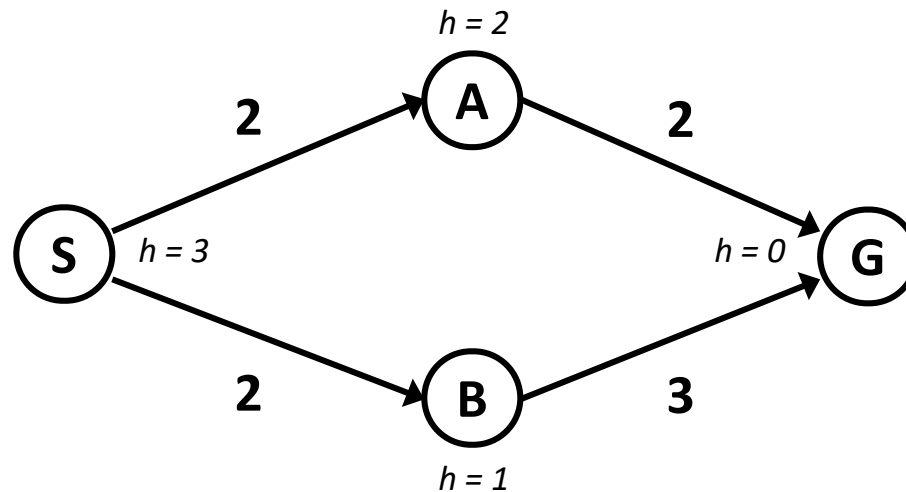
# A\* tree search algorithm –keeping track of visited nodes:

1. start with the initial node as curr
2. have I been to curr before? (is it in CLOSED)
3. is curr the goal?
4. if neither, expand curr - add children/successors to Frontier, add curr to CLOSED
5. choose a node curr from frontier according to the smallest  $f(n)$  & go to step 2

Note: This is basically\* lab3!

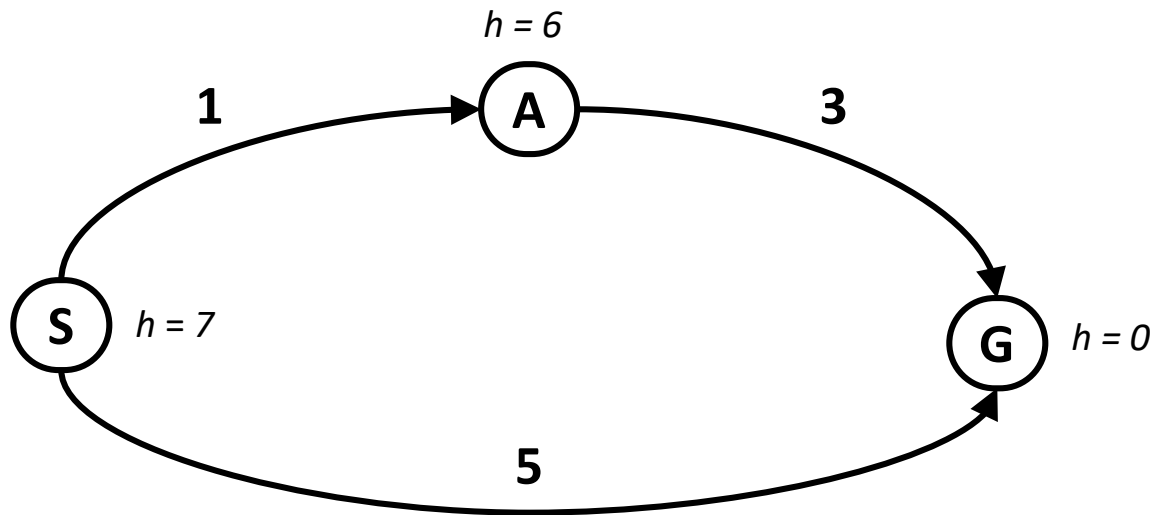
# When should A\* terminate?

- Should we stop when put a goal node in frontier?



- No: only stop when we expand a goal from frontier

# Is A\* Optimal?

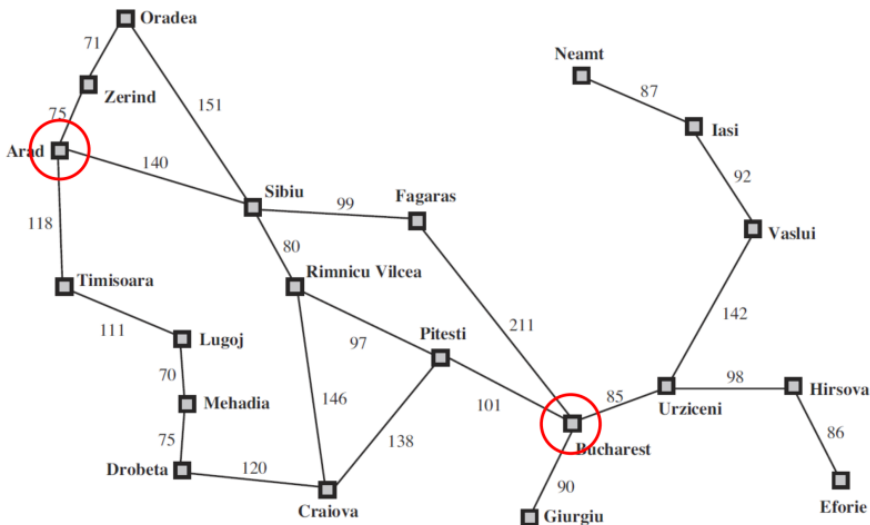


- What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!

# 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:  $h_{SLD}(n)$  (never overestimates the actual road distance)

# A\* search example



Straight-line distance  
to Bucharest

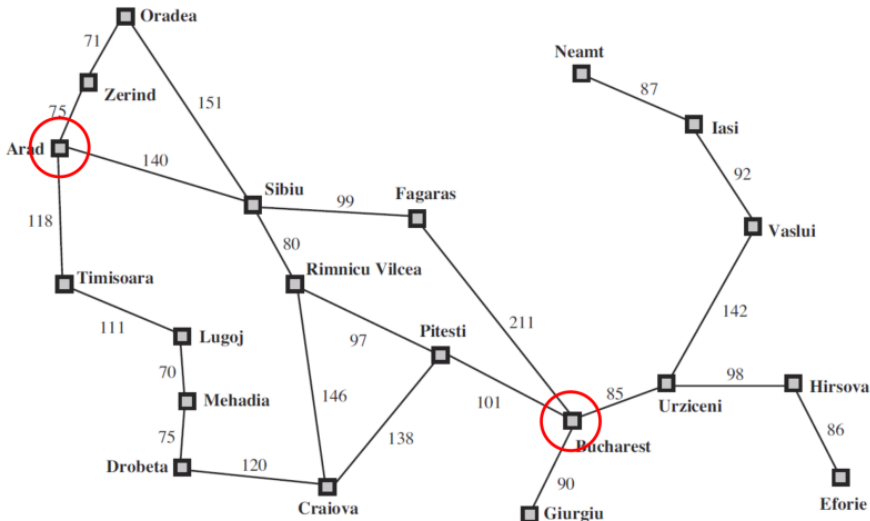
<b>Arad</b>	366
<b>Bucharest</b>	0
<b>Craiova</b>	160
<b>Dobreta</b>	242
<b>Eforie</b>	161
<b>Fagaras</b>	176
<b>Giurgiu</b>	77
<b>Hirsova</b>	151
<b>Iasi</b>	226
<b>Lugoj</b>	244
<b>Mehadia</b>	241
<b>Neamt</b>	234
<b>Oradea</b>	380
<b>Pitesti</b>	10
<b>Rimnicu Vilcea</b>	193
<b>Sibiu</b>	253
<b>Timisoara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

# A\* search example



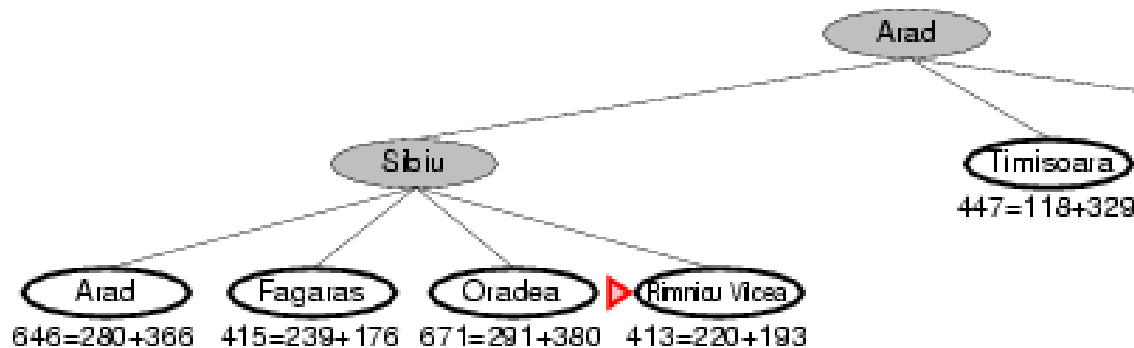
Straight-line distance  
to Bucharest

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



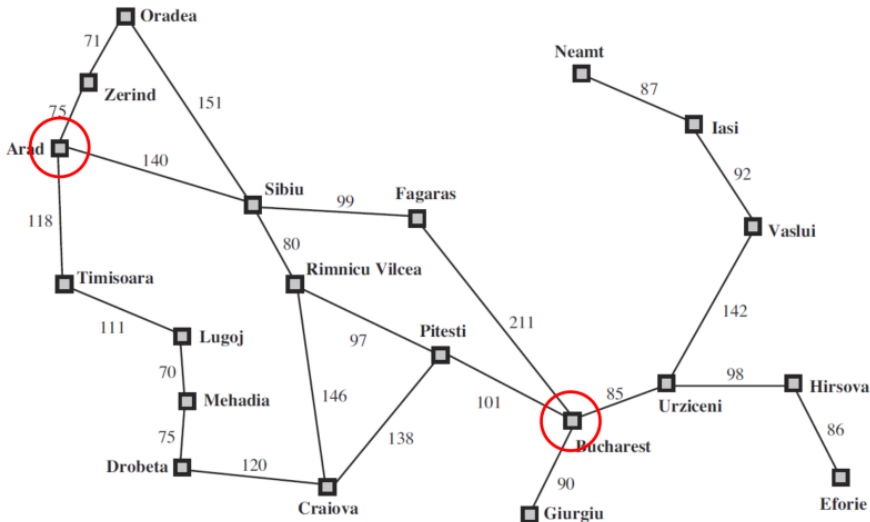


# A\* search example

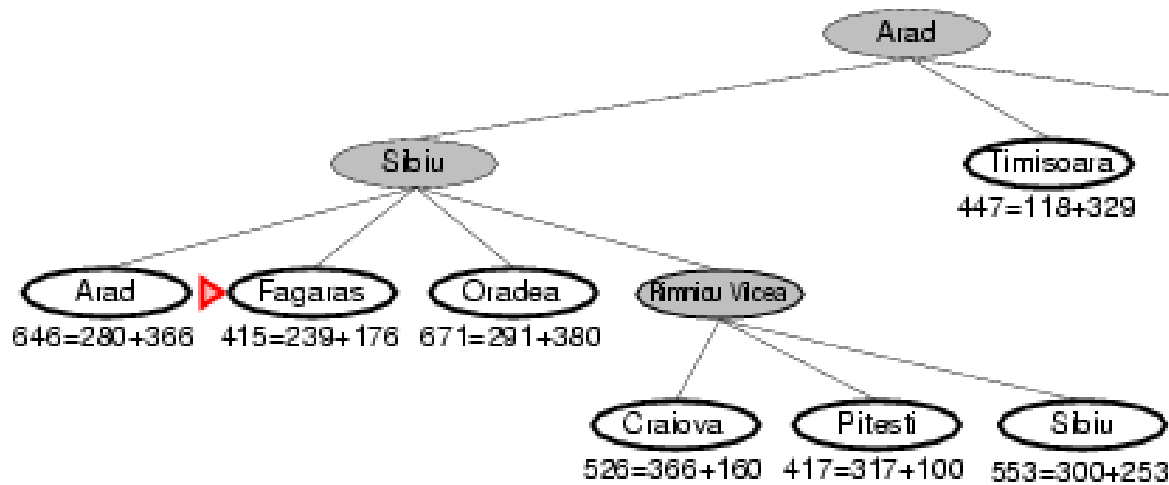


Straight-line distance  
to Bucharest

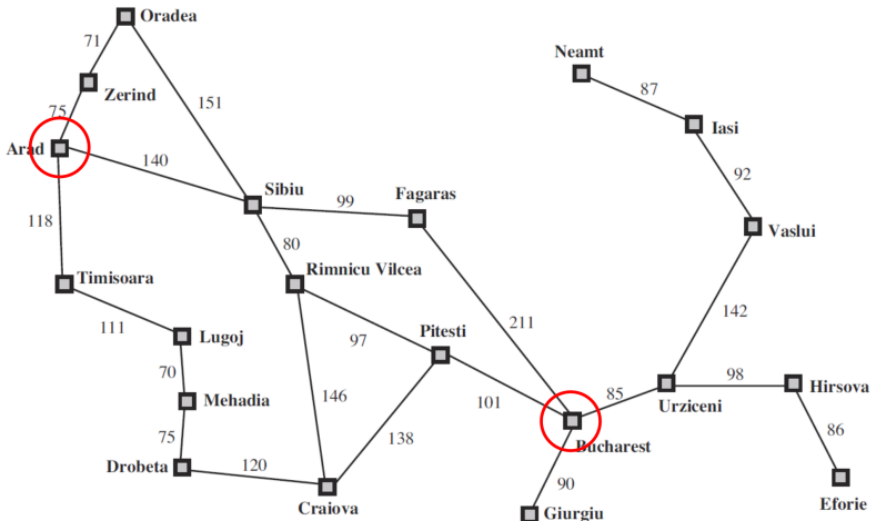
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374



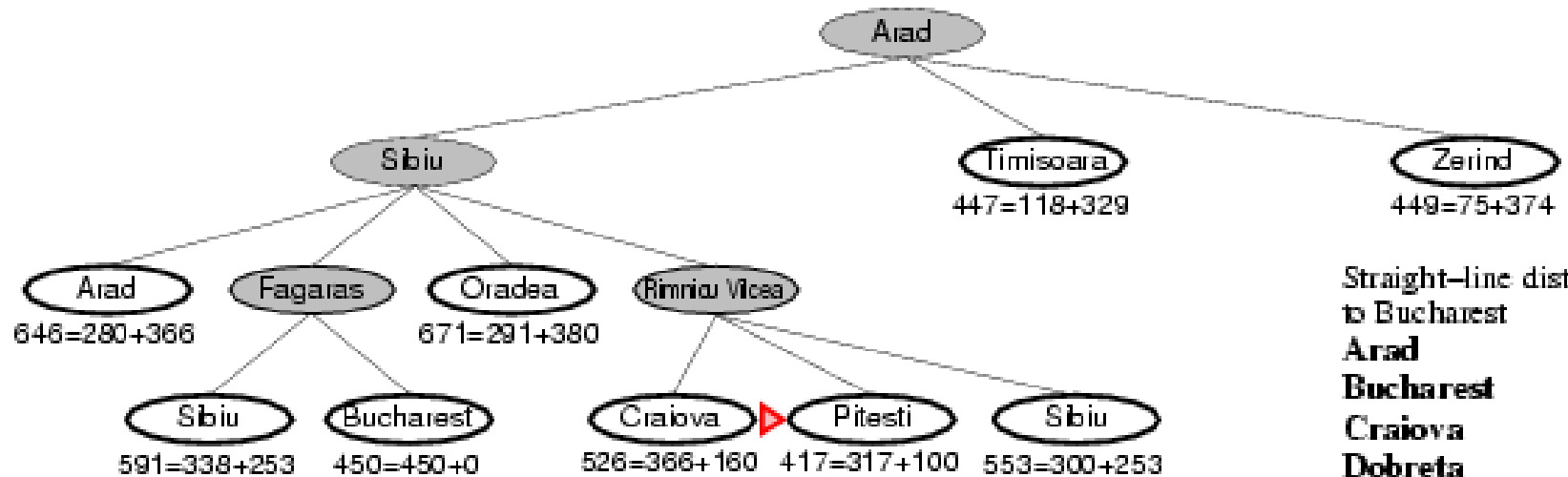
# A\* search example

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

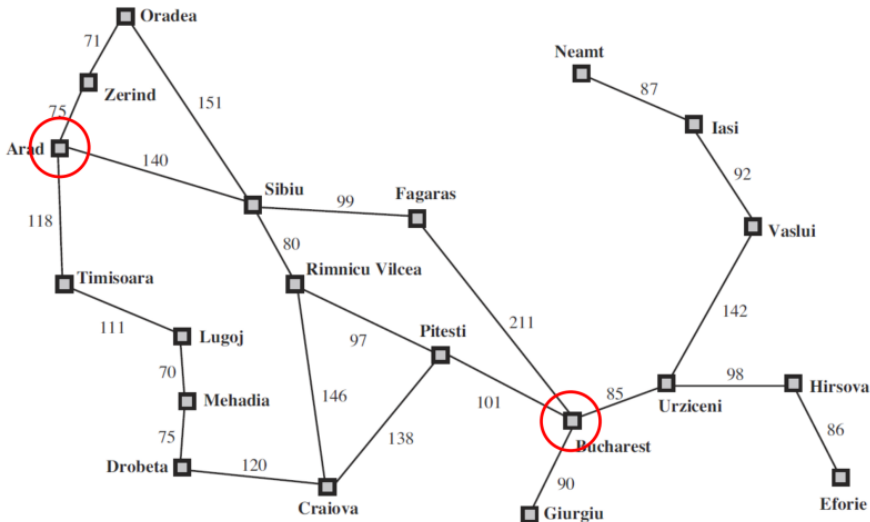


# A\* search example

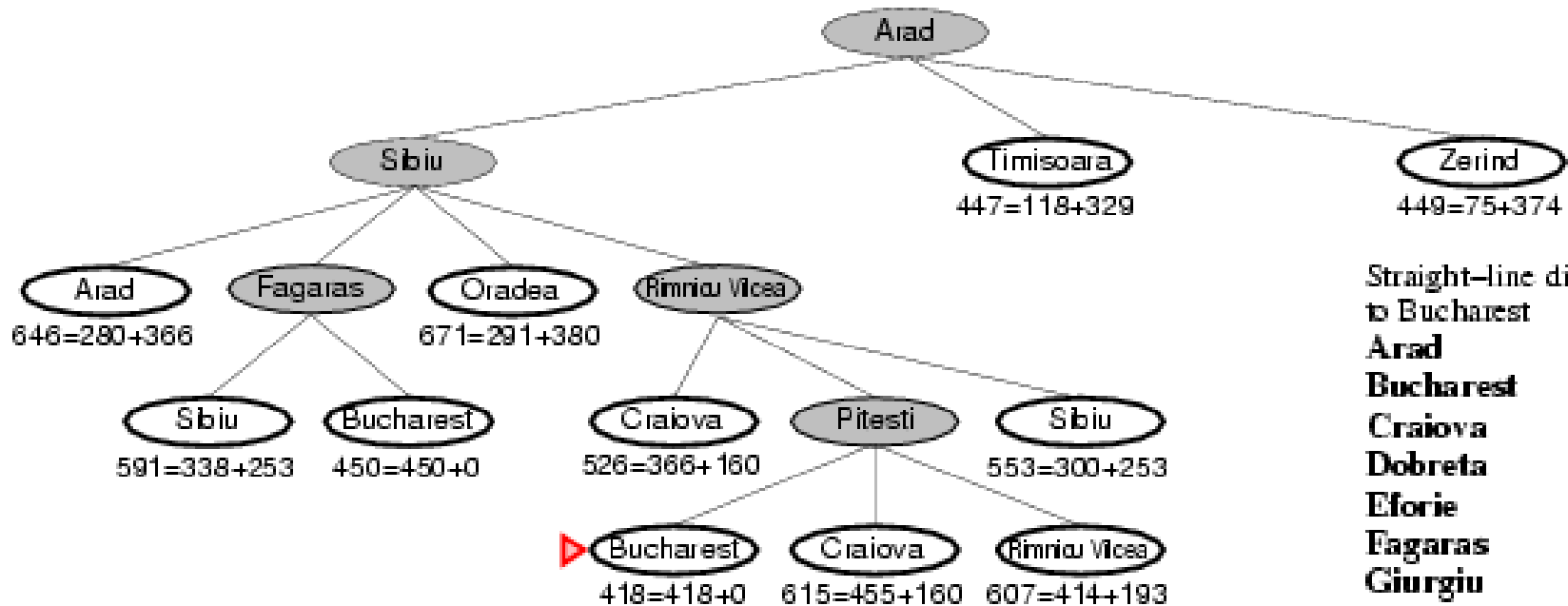


Straight-line distance  
to Bucharest

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

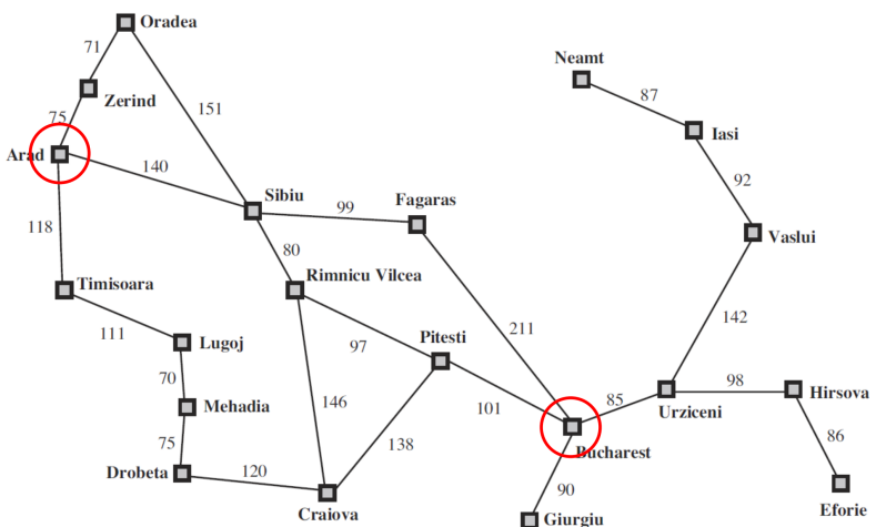


# A\* search example



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



# Properties of A\*

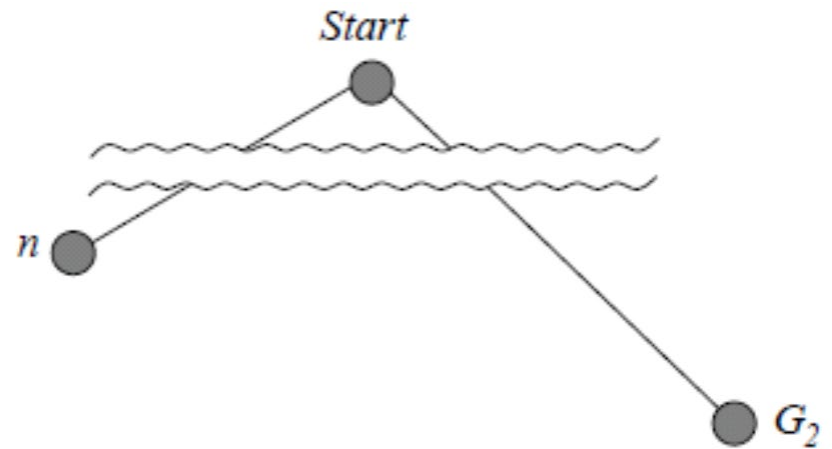
- If we use an admissible heuristic
  - Complete
    - If there are finite number of nodes with  $f(n) < f(\text{Goal})$
  - Optimal -yes!
    - Proof

# Optimal A\*

By contradiction:

- Suppose  $G_2$  is a suboptimal goal and is in the frontier. Let  $n$  be an unexpanded node on the shortest path to optimal goal  $G_1$
- $F(G_2) = g(G_2)$ 
  - $> g(G_1)$  (since  $G_2$  is suboptimal)
  - $\geq f(n)$  since  $h$  is admissible

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



# Properties of A\*

- If we use an admissible heuristic
  - Complete
    - If there are finite number of nodes with  $f() < f(\text{Goal})$
  - Optimal -yes!
    - Proof
  - Optimally efficient
    - No algorithm that is optimal is guaranteed to expand fewer nodes than A\*

# Admissible heuristics

Relax constraints of problem:

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

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



# Admissible heuristics

Relax constraints of problem:

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

- $h_1(S) = 8$
- $h_2(S) = 3+1+2+2+2+3+3+2=18$
- Which one is closer to the correct value?

# Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$  = number of misplaced tiles
- $h_2(n)$  = total Manhattan distance

7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State

Average nodes expanded

	Depth 4	Depth 12
BFS	112	$3.6 \times 10^6$
H1	13	227
H2	12	73

# Heuristic functions

- Sub-problem database

*	2	4
*		*
*	3	1

Start State

	1	2
3	4	*
*	*	*

Goal State

- Admissible?
- 15 puzzle reduction of 10,000 nodes vs Manhattan distance
- 24 puzzle, roughly 1 million reduction

# Combining heuristics

- $0 \leq h() \leq h_{\text{exact}}()$ 
  - Closer to exact, the better
  - What happens if  $h() = h_{\text{exact}}()$  ?
- $h(n) = \max(h_a(n), h_b(n), \dots)$ 
  - $h(n)$  is  $\geq$  any of the component heuristics
  - $h(n)$  is still admissible

# A\* example

From past midterm:

A robot starts in location “P” and its goal is “B” in the map below. The robot can only move up/down/left/right (no diagonals). Give the order in which it expands nodes given it is using A\* and the “manhattan distance” heuristic.

Write order of node expansion

<b>A</b>		<b>B</b>	<b>C</b>	<b>D</b>
<b>E</b>				<b>F</b>
<b>G</b>		<b>H</b>		<b>I</b>
<b>J</b>		<b>K</b>		<b>L</b>
<b>M</b>	<b>N</b>	<b>O</b>	<b>P</b>	<b>Q</b>

# Local search algorithms

- Many optimization problems the path to the goal is irrelevant, only finding the goal is important.
- State space= set of valid configurations
- Find configuration satisfying constraints
- In such cases, we can use local search algorithms
- For example:
  - Keep a single “current” state, and try to improve it

# Greedy Hill climbing

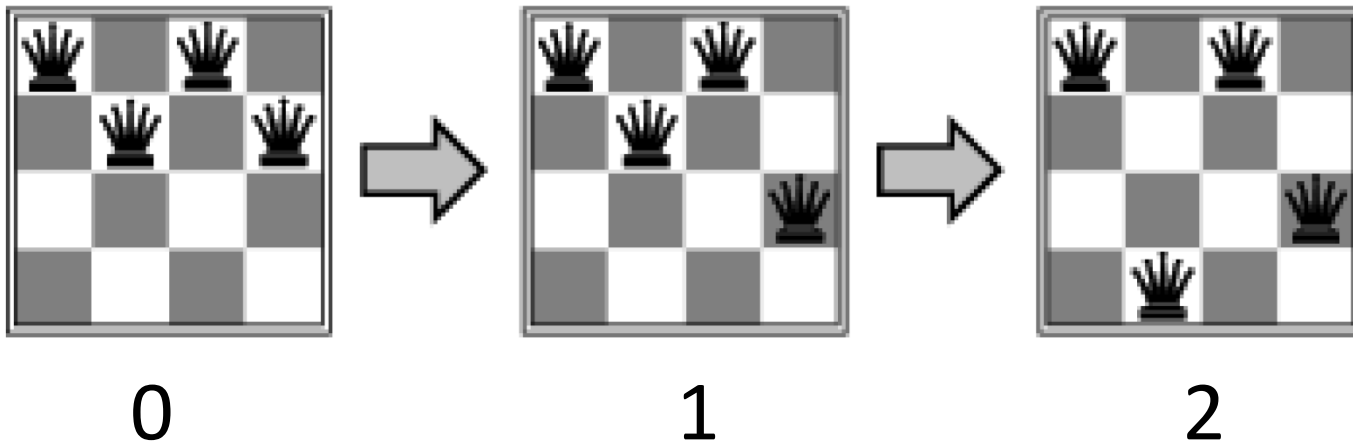
- “like climbing Everest in thick fog with amnesia” \*Lab 2\*

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  inputs: problem, a problem
  local variables: current, a node
                  neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
  loop do
    neighbor ← a highest-valued successor of current
    if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
    current ← neighbor
```

# Example: n-queens









Put  $n$  queens on a  $n \times n$  board with no queens able to attack each other





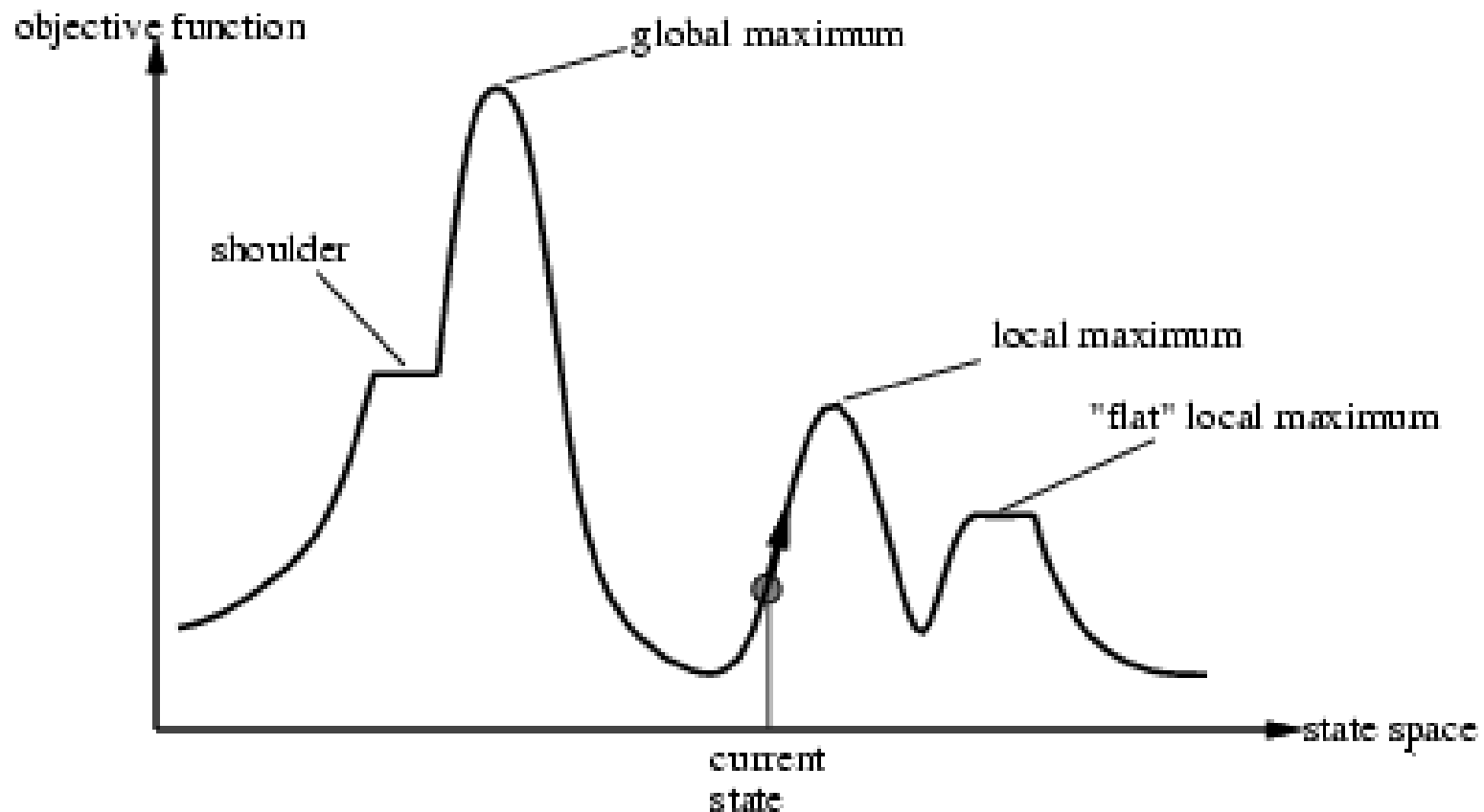
# Greedy Hill-climbing search: 8 queens

- Number of queens attacking for moving within a column.
- Greedy will solve 8queens %14 of the time
  - On average taking 4 steps
- Not bad considering 8queens has 17 million states

18	12	14	13	13	12	14	14
14	16	13	15	12	14	12	16
14	12	18	13	15	12	14	14
15	14	14		13	16	13	16
	14	17	15		14	16	16
17		16	18	15		15	
18	14		15	15	14		16
14	14	13	17	12	14	12	18

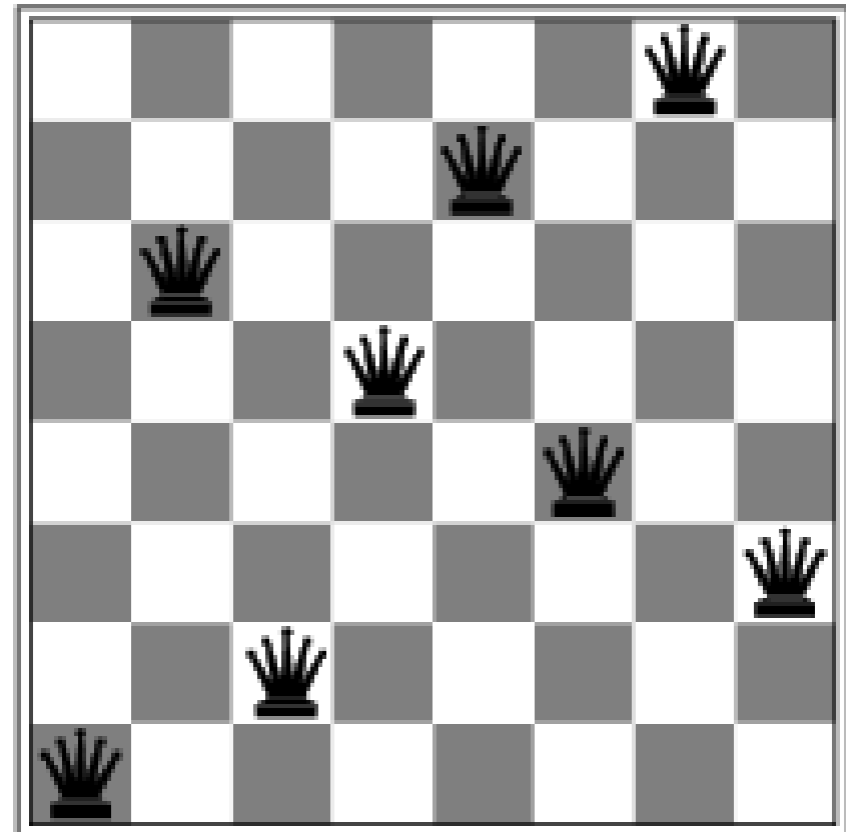
# Greedy Hill climbing

- Problem: depending on initial state, can get stuck in local maxima



# Greedy Hill-climbing search: 8 queens

- Local minimum with cost = 1
- No move (in column) can reduce cost (so in local minima)



# Greedy Hill climbing random restart

- If stuck at non-goal, choose a new starting state, re-run hill climbing
  - If probability of solving is  $P$ 
    - Expected restarts till goal found is  $1/P$
- Can solve the million queens problem in under a min

# Local beam search

- Keep track of  $K$  states rather than 1
- Start with  $K$  randomly generated states
- At each iteration, generate all successors for all  $K$  states
- If any one is the goal state, stop;
  - Else select the  $K$  best successors and repeat
- Stochastic beam search
  - Choose successors at random, with higher likelihood to successors with higher value.

# Simulated annealing

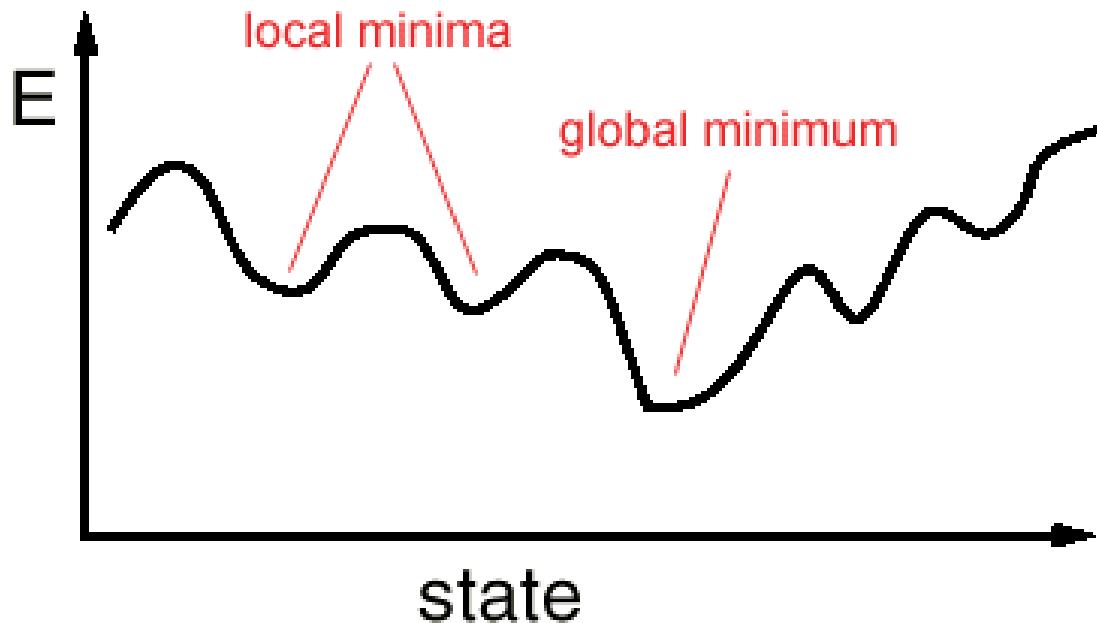
- Idea: escape local maxima by allowing some “bad” moves, but gradually decrease how frequently you allow this

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
           schedule, a mapping from time to “temperature”
  local variables: current, a node
                    next, a node
                    T, a “temperature” controlling prob. of downward steps

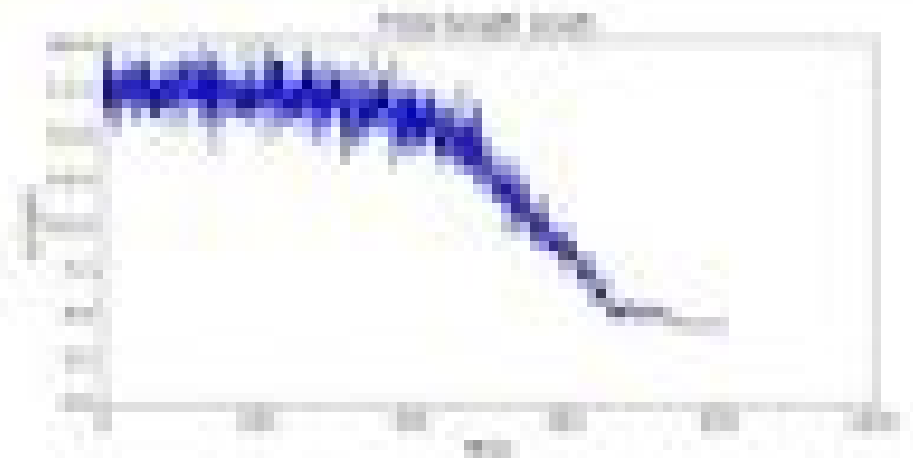
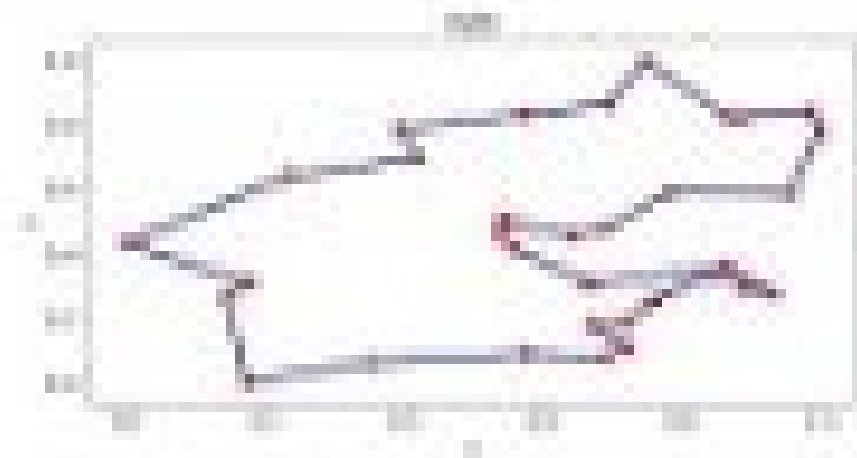
  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
     $\Delta E \leftarrow \text{VALUE}[\textit{next}] - \text{VALUE}[\textit{current}]$ 
    if  $\Delta E > 0$  then current ← next
    else current ← next only with probability  $e^{\Delta E/T}$ 
```

# Properties of simulated annealing search

- One can prove: if  $T$  decreases slowly enough, then simulated annealing will find a global optimum with probability approaching 1



# Simulated annealing example



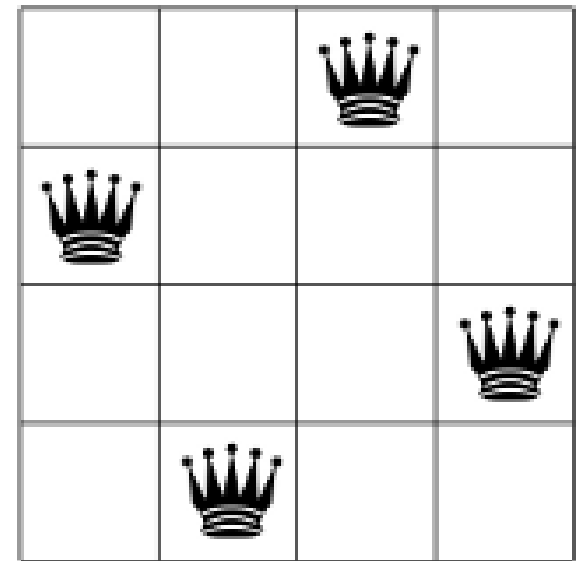


# Genetic algorithms

1. Start with  $K$  randomly generated states (population)
2. Evaluate the fitness for all states in population using a fitness function
3. Generate next generation of population through reserving, selection, crossover, and mutation.
4. Goto step 2.

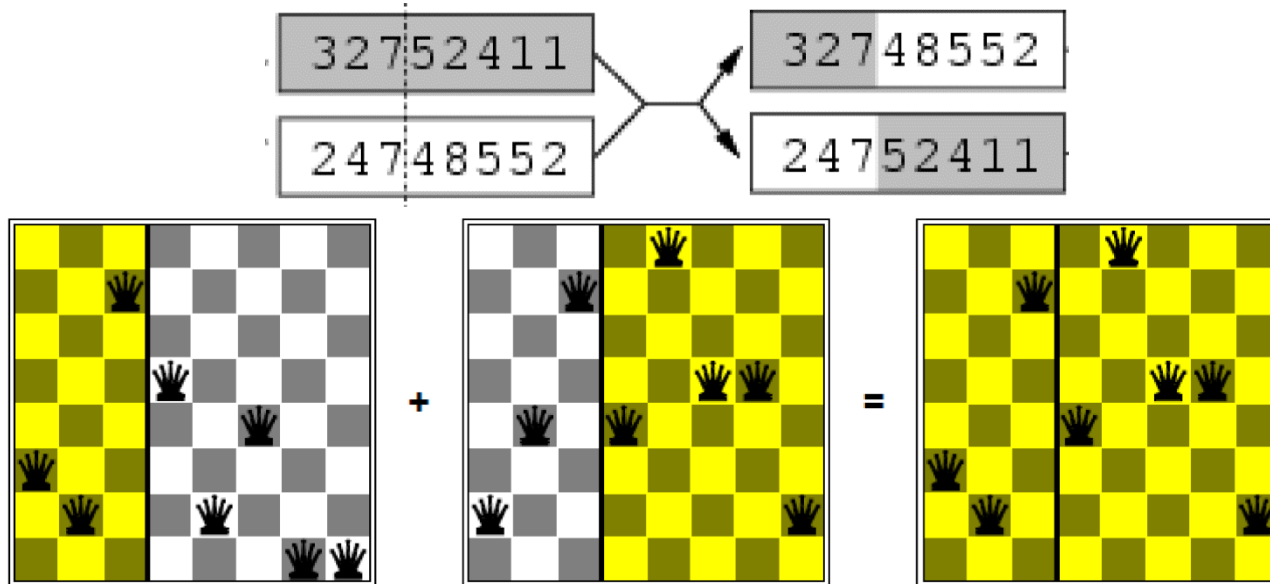
# Individuals

- Each individual consists of an encoded description of its state
- The encoded description is evaluated by the fitness function
- Example:N-queens
  - description
    - 2,4,1,3
  - Fitness function
    - Number of queens attacking



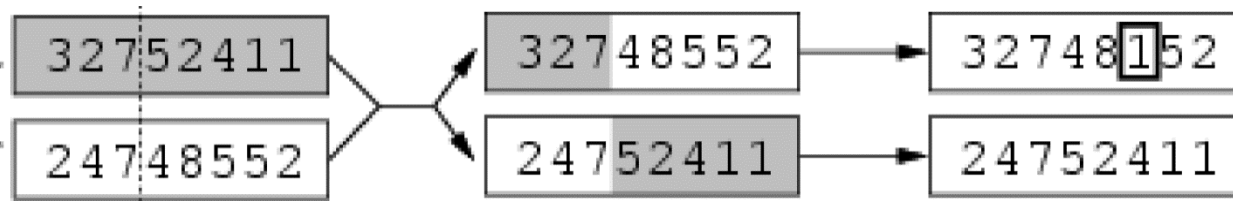
# Generate new population

1. Select 2 individuals based on fitness
  - Higher the fitness, more likely the selection (survival of the fittest)
2. Create “offspring” using crossover



# Generate new population

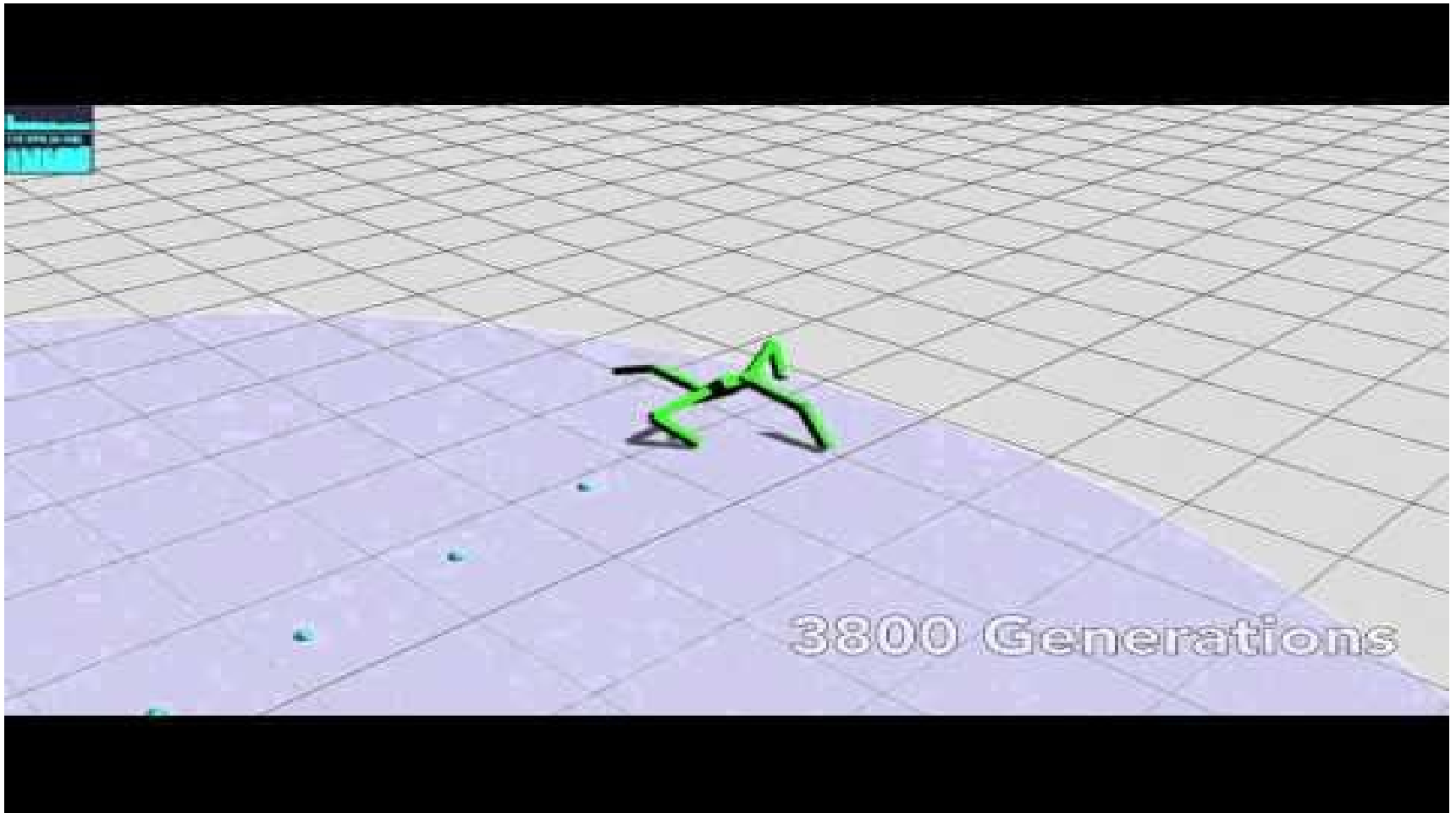
1. Select 2 individuals based on fitness
  - Higher the fitness, more likely the selection (survival of the fittest)
2. Create “offspring” using crossover
3. Modify offspring using mutation



# Generate new population

1. Select 2 individuals based on fitness
  - Higher the fitness, more likely the selection (survival of the fittest)
2. Create “offspring” using crossover
3. Modify offspring using mutation
4. Add offspring to next population
5. While (population not full), goto 1

# Genetic algorithm example



# Genetic algorithm example

- Karl Sims 1994



# Summary

---

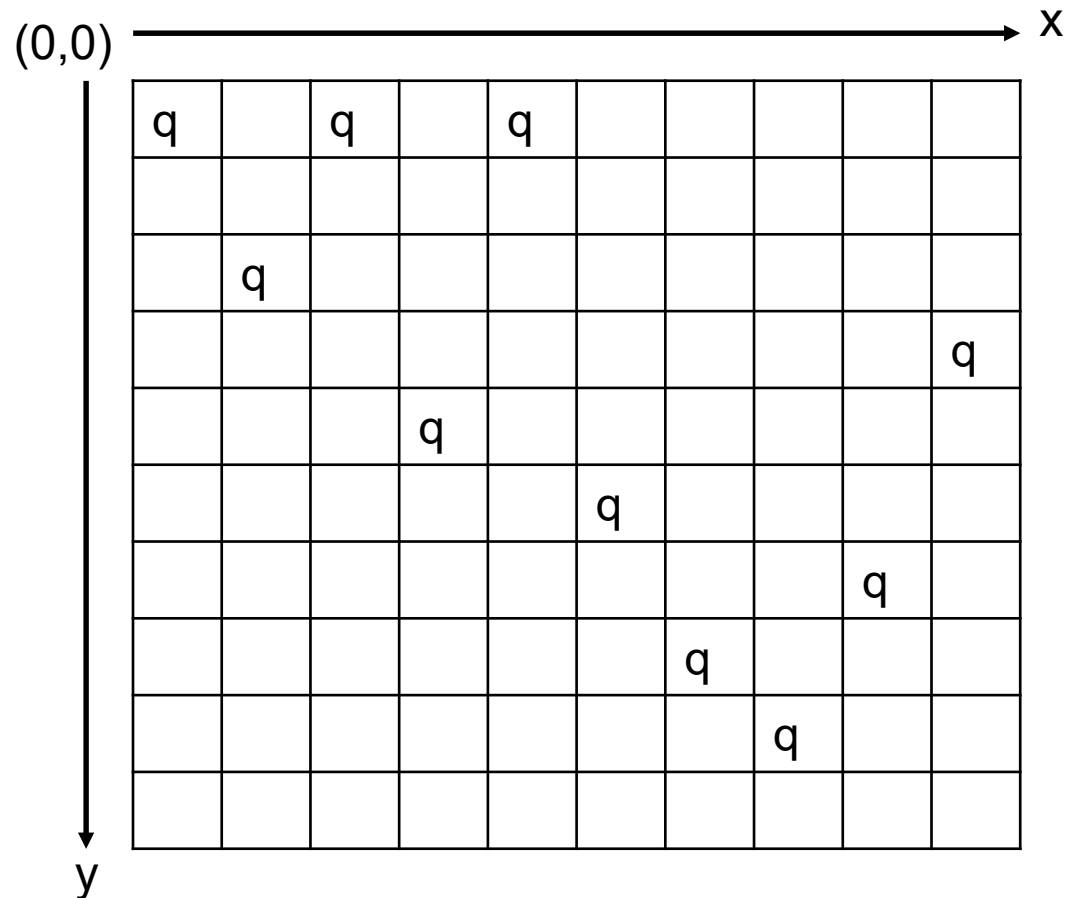
- Informed search
  - Greedy best first
  - $A^*$
- Local search
  - Greedy hill climb
  - Simulated annealing
  - Local beam
  - Genetic algorithm
- Next class: adversarial search



# Lab 2, Due April 21, 7pm

---

- 10-queens problem
  - Queens constrained to column



# Lab 2

---

- 10-queens problem
  - Queens constrained to column
  - Find local minima using greedy search
    - Each step is motion of one queen in her column
      - 81 possible next moves
    - Follow tie breaking rules