Informed Search and Exploration

Chapter 4: Heuristic Search
(4.1-4.2)

Introduction

- Ch.3 searches – good building blocks for learning about search
- More examples (in class)
  - Romania, FSA
- But vastly inefficient
- Can we do better?

Overview

- Heuristic Search
  - Best-First Search Approach
  - Greedy
  - A*
  - Heuristic Functions

Informed Searching

- An *informed search* strategy uses knowledge beyond the definition of the problem
- The knowledge is embodied in an *evaluation function* f(n)

Best-First Search

- An algorithm in which a node is selected for expansion based on an evaluation function f(n)
  - Traditionally the node with the lowest evaluation function is selected
  - Fringe nodes ordered by f(n)
  - Not an accurate name…expanding the best node first would be a straight march to the goal.
  - Choose the node that *appears* to be the best

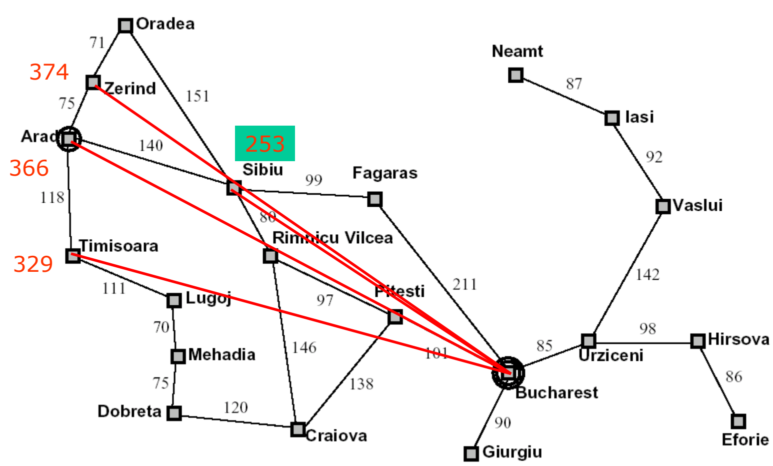
Best-First Search (cont.)

- Some BFS algorithms also include the notion of a heuristic function h(n)
- h(n) = estimated cost of the cheapest path from node n to a goal node
- Best way to include informed knowledge into a search
- Examples:
  - How far is it from point A to point B
  - How much time will it take to complete the rest of the task at current node to finish

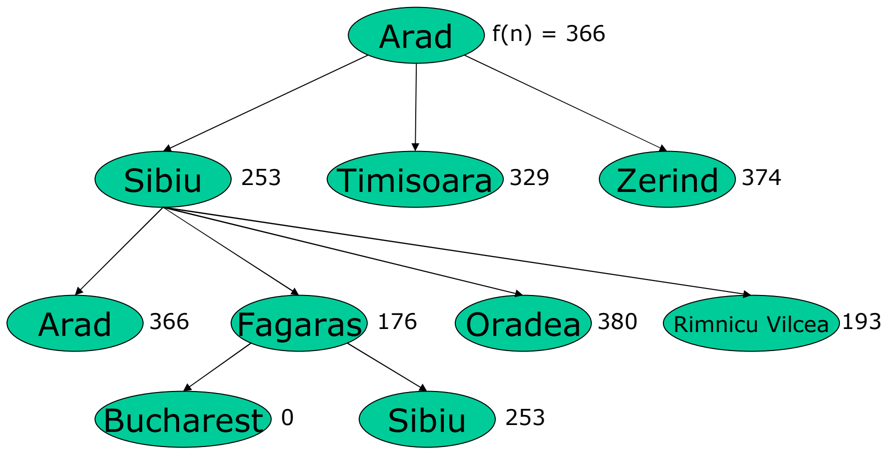
Greedy Best-First Search

- Expands node estimated to be closest to the goal
  - f(n) = h(n)
- Consider the route finding problem.
  - Can we use additional information to avoid costly paths that lead nowhere?
  - Consider using the straight line distance (SLD)

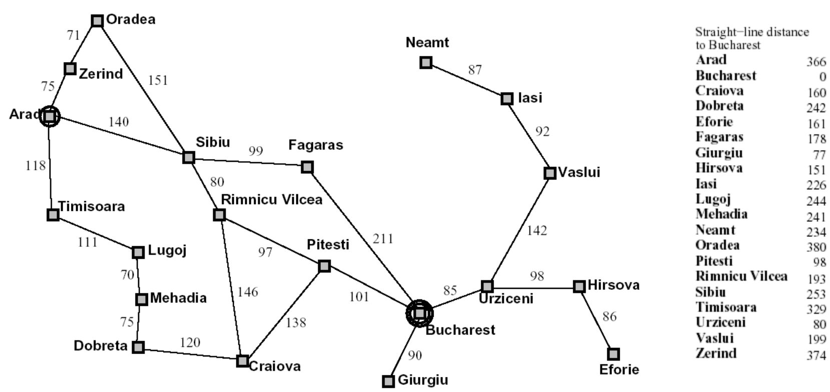
Route Finding



Route Finding



Exercise



So is Arad->Sibiu->Fagaras->Bucharest optimal?

Greedy Best-First Search

- Not optimal.
- Not complete.
  - Could go down a path and never return to try another.
  - e.g., Iasi → Neamt → Iasi → Neamt → …
- Time Complexity
  - $O(b^m)$ (but a good heuristic can give a dramatic improvement)
- Space Complexity
  - $O(b^m)$ – keeps all nodes in memory

Heuristic Functions

- Example: 8-Puzzle
  - Average solution cost for a random puzzle is 22 moves
  - Branching factor is about 3
    - Empty tile in the middle -> four moves
    - Empty tile on the edge -> three moves
    - Empty tile in corner -> two moves
  - 3^{22} is approx 3.1e10
    - Get rid of repeated states
    - 181440 distinct states

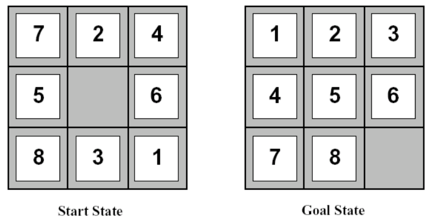
7	2	4
5		6
8	3	1

Start State

1	2	3
4	5	6
7	8	

Goal State

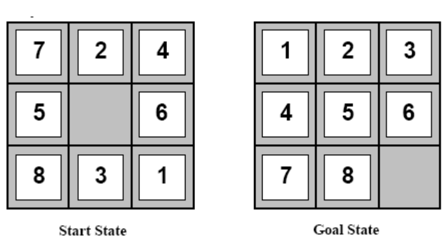
Heuristic Functions



- h1 = number of misplaced tiles
- h2 = sum of distances of tiles to goal position.

Heuristic Functions

- h1 = ?
- h2 = ?



Heuristic Functions

- A heuristic function h(n) is *admissible* if it never overestimates the cost to reach the goal from n
  - Is h1 admissible?
  - Is h2 admissible?
- Another property of heuristic functions (relevant for graph search) is *consistency*

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
- Typical search costs (average number of nodes expanded):
- *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 Functions

- Heuristics are often obtained from *relaxed problem*
  - Simplify the original problem by removing constraints
  - The cost of an optimal solution to a relaxed problem is an admissible heuristic.

8-Puzzle

- Original
  - A tile can move from *A* to *B* if *A* is horizontally or vertically adjacent to *B* and *B* is blank.
- Relaxations
  - Move from *A* to *B* if *A* is adjacent to *B*
    - *Is this h1, h2 or ?*
  - Move from *A* to *B* if *B* is blank
    - *Is this h1, h2 or ?*
  - Move from *A* to *B*
    - *Is this h1, h2 or ?*

8-Puzzle

- What heuristic is obtained from *'Move from A to B if B is blank'*?
  - Gaschnig's heuristic (1979)
  - To compute, repeat the following until the goal state is reached: Let B be the location of the blank. If B is occupied by tile X (not the blank) in the goal state, then move X to B. Otherwise, move any misplaced tile to B.
- Example: permute 2 adjacent tiles in the goal state
  - h1 =?
  - h2 = ?
  - Gaschnig's heuristic = ?

How to Obtain Heuristics?

- Ask the domain expert (if there is one)
- Solve example problems and generalize your experience on which operators are helpful in which situation (particularly important for state space search)
- Try to develop sophisticated evaluation functions that measure the closeness of a state to a goal state (particularly important for state space search)
- Run your search algorithm with different parameter settings trying to determine which parameter settings of the chosen search algorithm are "good" to solve a particular class of problems.
- Write a program that selects "good parameter" settings based on problem characteristics (frequently very difficult) relying on machine learning

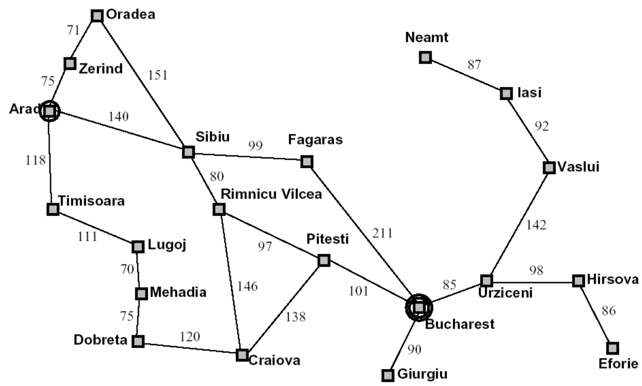
A* Search

- The greedy best-first search does not consider how costly it was to get to a node.
- Idea: avoid expanding paths that are already expensive
- Combine g(n), the cost to reach node n, with h(n)
  - f(n) = g(n) + h(n)
  - estimated cost of cheapest solution through n

A* Search

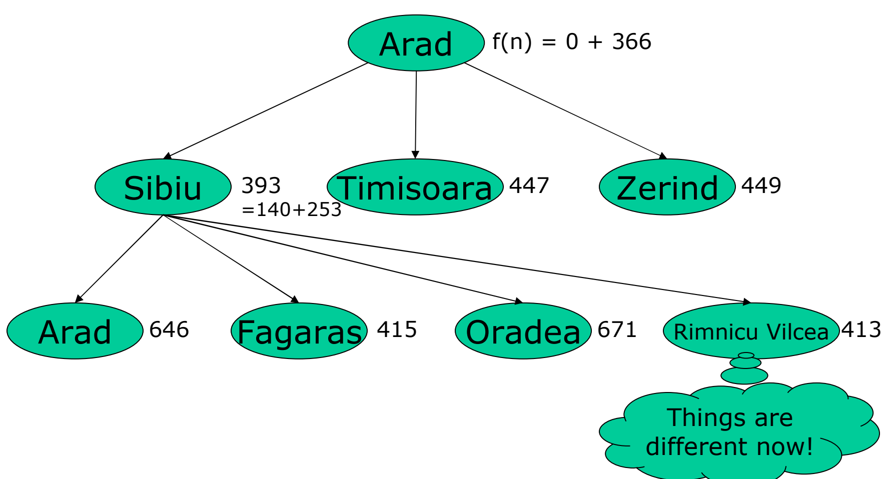
- When h(n) = actual cost to goal
  - Only nodes in the correct path are expanded
  - Optimal solution is found
- When h(n) < actual cost to goal
  - Additional nodes are expanded
  - Optimal solution is found
- When h(n) > actual cost to goal
  - Optimal solution can be overlooked

Route Finding

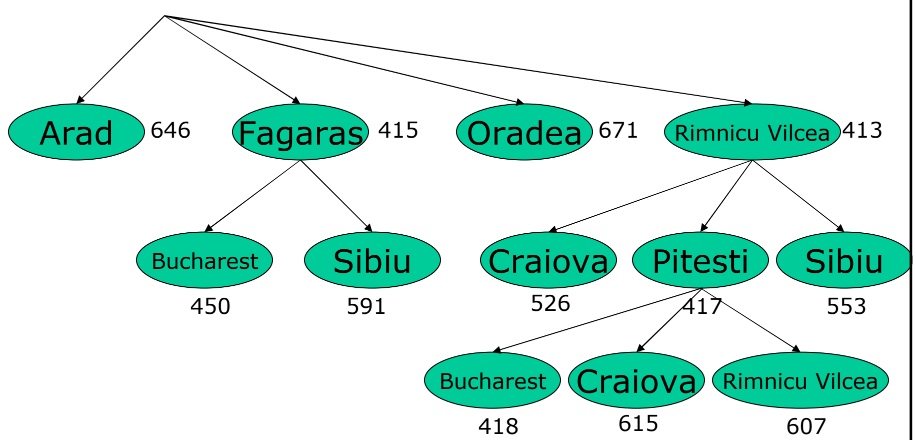


Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* Search



A* Search Continued



A* Search

- Complete
  - Yes, unless there are infinitely many nodes with f <= f(G)
- Time
  - Exponential in [relative error of h x length of soln]
  - The better the heuristic, the better the time
    - Best case h is perfect, O(d)
    - Worst case h = 0, $O(b^d)$ same as BFS
- Space
  - Keeps all nodes in memory and save in case of repetition
  - This is $O(b^d)$ or worse
  - A* usually runs out of space before it runs out of time
- Optimal
  - Yes, cannot expand f_{i+1} unless f_i is finished

Memory Bounded Heuristic Search

- Ways of getting around memory issues of A*:
- IDA* (iterative deepening algorithm)
  - Cutoff = f(n) instead of depth
- Recursive Best First Search