Lecture 10: Heuristic Search
Class Tests

• **Class Test 1 (Prolog):**
  – Tuesday 8\(^{th}\) November (Week 7)
  – 13:00-14:00
  – LIFS-LT2 and LIFS-LT3

• **Class Test 2 (Everything but Prolog)**
  – Tuesday 6\(^{th}\) December (Week 11)
  – 13:00-14:00
  – LIFS-LT2 and LIFS-LT3
Overview

• Last time
  – Depth-limited, iterative deepening and bi-directional search
  – Avoiding repeated states

• Today
  – Show how applying knowledge of the problem can help
  – Introduce uniform cost search: dependent on the cost of each node
  – Introduce heuristics: rules of thumb
  – Introduce heuristic search
    • Greedy search
    • A* search

• Learning outcome covered today:
  Identify, contrast and apply to simple examples the major search techniques that have been developed for problem-solving in AI
Real Life Problems

• Whatever search technique we use, we have exponential time complexity
• Tweaks to the algorithm will not reduce this to polynomial
• We need problem specific knowledge to guide the search
• Simplest form of problem specific knowledge is heuristics
• Usual implementation in search is via an evaluation function which indicates desirability of expanding a node
Path Cost Function

Recall: we have a path cost function, which gives the cost of each path. This is comprised of the step costs of each action on the path.
Finding The Best Paths

- Why not expand the *cheapest path first*?
- Intuition: cheapest is likely to be best!
- Performance is like breadth-first search but we use the minimum cost path rather than the shallowest to expand
- **Uniform cost search** orders the agenda as a priority queue using the lowest path cost of a node
Cheapest First
Cheapest First

Choose Zerind

Choose Zerind

75

140

118

Choose Zerind
Cheapest First

Choose Zerind

Choose Zerind

Choose Zerind

Choose Zerind
Cheapest First

Choose Zerind  Choose Timisoara

75  146  140  118
Cheapest First

Choose Zerind

Choose Timisoara
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea

Choose Sibiu
Choose Timisoara
Choose Oradea
Choose Rimnicu Vilcea
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj

Choose Sibiu
Choose Fagaras
Choose Sibiu
Choose Fagaras
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj

Choose Sibiu

Choose Oradea

Choose Rimnicu Vilcea
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras

Choose Sibiu
Choose Fagaras
Cheapest First

Choose Zerind  Choose Timisoara  Choose Sibiu  Choose Oradea
Choose Rimnicu Vilcea  Choose Lugoj  Choose Fagaras

Choose Sibiu

Choose Zerind

Choose Timisoara

Choose Sibiu

Choose Oradea

Choose Rimnicu Vilcea

Choose Lugoj

Choose Fagaras
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras
Choose Mehadia

Choose Arad
Choose Sibiu
Choose Fagaras
Choose Brașov
Choose București
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras
Choose Mehadia
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras
Choose Mehadia
Choose Pitesti

75
146
140
239
220
118
229
299
366
374
450
317
23
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras
Choose Mehadia
Choose Pitesti
Cheapest First

Choose Zerind
Choose Timisoara
Choose Sibiu
Choose Oradea
Choose Rimnicu Vilcea
Choose Lugoj
Choose Fagaras
Choose Mehadia
Choose Pitesti
General Algorithm for Uniform Cost Search

agenda = [initial state];
while agenda not empty do
    take node from agenda such that
    \[ g(\text{node}) = \min \{ g(n) \mid n \text{ in agenda} \} \]
    if node is goal state then
        return solution;
    new nodes = apply operations to node;
    add new nodes to the agenda;
Properties of Uniform Cost Search

- Uniform cost search **guaranteed** to find **cheapest** solution assuming path costs grow **monotonically**, i.e. the cost of a path never decreases as we move along it.
- In other words, adding another step to the solution makes it more costly, i.e. 
  \[ g(\text{successor}(n)) > g(n). \]
- If path costs don’t grow monotonically, then exhaustive search is required.
- Still requires many nodes to be examined.
Informed Strategies

• Use problem-specific knowledge
• More efficient than blind search
• The most *promising* path first!
• Rather than trying all possible search paths, you try to focus on paths that seem to be getting you nearer your target/goal state
Greedy Search

• Most heuristics *estimate cost of cheapest path from node to solution*
• We have a *heuristic function*,
  \[ h : \text{Nodes} \rightarrow \mathbb{R} \]
  which estimates the distance from the node to the goal
• \( h \) can be any function but should have \( h(n) = 0 \) if \( n \) is a goal
• Example: In route finding, heuristic might be straight line distance from node to destination
• Greedy search expands the node that *appears to be* closest to goal
Romania Example

<table>
<thead>
<tr>
<th>City</th>
<th>Straight-line distance to Bucharest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arad</td>
<td>366</td>
</tr>
<tr>
<td>Bucharest</td>
<td>0</td>
</tr>
<tr>
<td>Craiova</td>
<td>160</td>
</tr>
<tr>
<td>Dobreta</td>
<td>242</td>
</tr>
<tr>
<td>Eforie</td>
<td>161</td>
</tr>
<tr>
<td>Fagaras</td>
<td>176</td>
</tr>
<tr>
<td>Giurgiu</td>
<td>77</td>
</tr>
<tr>
<td>Hirsova</td>
<td>151</td>
</tr>
<tr>
<td>Iasi</td>
<td>226</td>
</tr>
<tr>
<td>Lugoj</td>
<td>244</td>
</tr>
<tr>
<td>Mehadia</td>
<td>241</td>
</tr>
<tr>
<td>Neamt</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>98</td>
</tr>
<tr>
<td>Rimnicu Vilcea</td>
<td>193</td>
</tr>
<tr>
<td>Sibiu</td>
<td>253</td>
</tr>
<tr>
<td>Timisoara</td>
<td>329</td>
</tr>
<tr>
<td>Urziceni</td>
<td>80</td>
</tr>
<tr>
<td>Vaslui</td>
<td>199</td>
</tr>
<tr>
<td>Zerind</td>
<td>374</td>
</tr>
</tbody>
</table>
Greedy Search Example
Greedy Search Example
Greedy Search Example

Choose Sibiu

Choose Sibiu

Choose Sibiu
Greedy Search Example

Choose Sibiu
Greedy Search Example

Choose Sibiu

Choose Fagaras

374

380

253

176

366

329

193
Greedy Search Example

Choose Sibiu
Choose Fagaras
Greedy Search Example

Choose Sibiu
Choose Fagaras

NOT the cheapest path
BUT far less effort
Search Tree
Exercise

• Suppose that in the Romania example the initial state is Iasi and the goal state is Fagaras. How do you think a greedy search, using the straight line distance as a heuristic, might proceed?
General Algorithm for Greedy Search

agenda = [initial state];
while agenda not empty do
    take node from agenda such that
        h(node) = min \{ h(n) \mid n \text{ in agenda} \}
    if node is goal state then
        return solution;
    new nodes = apply operations to node;
    add new nodes to the agenda;
Properties of Greedy Search

• Greedy search finds solutions quickly
• Doesn’t always find the best
• May not find a solution if there is one (incomplete)
• Susceptible to false starts
• Only looking at current node. Ignores past!
• Short sighted
A* Search

• A* is a very efficient search strategy
• Basic idea is to combine uniform cost search and greedy search
• We look at the cost so far and the estimated cost to goal
• Gives heuristic $f$:
  $$f(n) = g(n) + h(n)$$
• where
  $g(n)$ is path cost of $n$
  $h(n)$ is expected cost of cheapest solution from $n$
• Aims to minimise overall cost
A* Search Example
A* Search Example
A* Search Example

Choose Sibiu

75+374
140+253
0+366
118+329
A* Search Example

Choose Sibiu

Choose Sibiu

Choose Sibiu

Choose Sibiu
A* Search Example

Choose Sibiu

Choose Rimnicu Vilcea
413 < 415

75+374

291+380

140+253

239+176

0+366

118+329

220+193
A* Search Example

Choose Sibiu

Choose Rimnicu Vilcea

413 < 415

75 + 374

291 + 380

140 + 253

239 + 176

317 + 100

455 + 160

118 + 329

0 + 366

118 + 329

220 + 193
A* Search Example

Choose Sibiu
Choose Rimnicu Vilcea
Choose Fagaras

413 < 415
415 < 417

75 + 374
0 + 366
118 + 329

291 + 380
140 + 253
220 + 193

118 + 329

239 + 176

317 + 100

455 + 160
A* Search Example

Choose Sibiu

Choose Rimnicu Vilcea

413 < 415

Choose Fagaras

415 < 417

75+374

291+380

140+253

239+176

0+366

118+329

140+253

220+193

317+100

450+0

239+176

455+160
A* Search Example

Choose Sibiu
Choose Rimnicu Vilcea
Choose Fagaras
Choose Pitesti

75+374
413 < 415
291+380
415 < 417
140+253
417 < 450
239+176
220+193
118+329
317+100
450+0
455+160
A* Search Example

Choose Sibiu
75 + 374

Choose Rimnicu Vilcea
291 + 380

Choose Fagaras
140 + 253

Choose Pitesti
239 + 176

Choose Rimnicu Vilcea
413 < 415

Choose Fagaras
415 < 417

Choose Pitesti
417 < 450
A* Search Example

Choose Sibiu
75 + 374
Choose Rimnicu Vilcea
291 + 380
Choose Fagaras
140 + 253
Choose Pitesti
239 + 176
Cheapest route found – all open nodes > 418

Choose Sibiu

413 < 415
Choose Rimnicu Vilcea
415 < 417
Choose Fagaras
417 < 450
Choose Pitesti

0 + 366
118 + 329
140 + 253
220 + 193
239 + 176
455 + 160

418 + 0
317 + 100
239 + 176
450 + 0
Search Tree
General Algorithm for A* Search

agenda = [initial state];
while agenda not empty do
  take node from agenda such that
    \( f(\text{node}) = \min \{ f(n) \mid n \text{ in agenda} \} \)
    where \( f(n) = g(n) + h(n) \)
  if node is goal state then
    return solution;
  new nodes = apply operations to node;
  add new nodes to the agenda;
Properties of A* Search

• Complete, provided
  – only finitely many nodes with $f < f(G)$
  – an admissible heuristic is used
    • Never overestimates the distance
    • i.e., $h(n) < true(n)$
      where $true(n)$ is the true cost from $n$
    • Also require $h(n) \geq 0$, so $h(G) = 0$ for any goal $G$

• Exponential time
• Keeps all nodes in memory
• Optimal
Admissible Heuristics

• Example - for the 8-puzzle:
  
  $h_1(n) = \text{number of misplaced tiles}$
  
  $h_2(n) = \text{total Manhattan distance}$

• (i.e., no. of squares from desired location of each tile)
Exercise

• Calculate $h_1$ and $h_2$ for the 8 puzzle on the previous slide.
• Are these heuristics admissible?
Importance of the Heuristic Choice

• Typical search costs:
  – $d = 14$
    • IDS = 3,473,941 nodes
    • $A^*(h_1) = 539$ nodes
    • $A^*(h_2) = 113$ nodes
  – $d = 24$
    • IDS ≈ 54,000,000,000 nodes
    • $A^*(h_1) = 39,135$ nodes
    • $A^*(h_2) = 1,641$ nodes
Summary

• Heuristic functions estimate costs of shortest paths
• Good heuristics can dramatically reduce search cost
• Greedy best-first search expands lowest $h$
  – incomplete and not always optimal
• A* search expands lowest $g + h$
  – complete and optimal
  – also optimally efficient

• Next time
  – Search in complex environments (partial observation) and in game playing