HW 1: Warmup
Missionaries and Cannibals

• Solve the Missionary-Cannibal Problem (with 3 missionaries and 3 cannibals) with a RECURSIVE DEPTH-FIRST SEARCH as follows:
  – You MUST use a recursive depth first search
  – No ancestor repeated states in a path
  – Keep counts of illegal states (cannibals eat missionaries), repeated states, total states searched
  – Use Python
  – Comment on each method and important code sections
  – Print all paths from start to goal
  – Print the final 3 counts.

• Due Jan 12 11:59pm. Late date Jan 14 11:59pm
• Your work must be YOUR OWN.
Informed (Heuristic) Search

Idea: be \textbf{smart} about what paths to try.
Blind Search vs. Informed Search

• What’s the difference?

• How do we formally specify this?

A node is selected for expansion based on an evaluation function that estimates cost to goal.
General Tree Search Paradigm

function tree-search(root-node)
  fringe ← successors(root-node)
  while ( notempty(fringe) )
    {node ← remove-first(fringe)
     state ← state(node)
     if goal-test(state) return solution(node)
     fringe ← insert-all(successors(node),fringe) }
  return failure
end tree-search

How do we order the successor list?
Best-First Search

- Use an evaluation function $f(n)$ for node $n$.
- Always choose the node from fringe that has the lowest $f$ value.

Diagram: A tree with nodes labeled 1, 3, 4, 5, 6. Node 1 is highlighted.
Heuristics

• What is a heuristic?

• What are some examples of heuristics we use?

• We’ll call the heuristic function $h(n)$. 
Greedy Best-First Search

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

- What does that mean?

- What is it ignoring?
Romanian Route Finding

• Problem
  – Initial State: Arad
  – Goal State: Bucharest
  – $c(s,a,s')$ is the length of the road from $s$ to $s'$

• Heuristic function: $h(s) = \text{the straight line distance from } s \text{ to Bucharest}$
What’s the real shortest path from Arad to Bucharest?
What’s the distance on that path?
Greedy Search in Romania
Greedy Best-First Search

• Is greedy search optimal?

• Is it complete?

No, can get into infinite loops in tree search. Graph search is complete for finite spaces.

• What is its worst-case complexity for a tree search with branching factor $b$ and maximum depth $m$?
  
  – time $O(b^m)$
  – space $O(b^m)$
Greedy Best-First Search

• When would we use greedy best-first search or greedy approaches in general?
A* Search

- Hart, Nilsson & Rafael 1968
  - Best-first search with \( f(n) = g(n) + h(n) \)
  
  where \( g(n) = \) sum of edge costs from start to \( n \)
  and \( h(n) = \) estimate of lowest cost path \( n \rightarrow \text{goal} \)
  
  - If \( h(n) \) is **admissible** then search will find optimal solution.

Never overestimates the true cost of any solution which can be reached from a node.

Space bound since the queue must be maintained.
Back to Romania

Straight-line distance to Bucharest

Arad: 366
Bucharest: 0
Craiova: 160
Dobrogea: 242
Eforie: 161
Fagaras: 178
Giurgiu: 77
Hirsova: 151
Iasi: 226
Lugoj: 244
Mehadia: 241
Neamț: 234
Oradea: 380
Pitești: 98
Rimnicu Vilcea: 193
Sibiu: 253
Timişoara: 329
Urziceni: 80
Vaslui: 199
Zerind: 374
A* for Romanian Shortest Path

Arad

366 = 0 + 366
\[ f(n) = g(n) + h(n) \]
8 Puzzle Example

• $f(n) = g(n) + h(n)$
• What is the usual $g(n)$?
• two well-known $h(n)$’s
  – $h_1 = \text{the number of misplaced tiles}$
  – $h_2 = \text{the sum of the distances of the tiles from their goal positions, using city block distance, which is the sum of the horizontal and vertical distances (Manhattan Distance)}$
8 Puzzle Using Number of Misplaced Tiles

1 2 3
8 4
7 6 5

goal

2 8 3
1 6 4
7 5

g=0
h=4
f=4
Exercise:
What are its children and their f, g, h?
Optimality of A* with Admissibility
(h never overestimates the cost to the goal)

Suppose a suboptimal goal G2 has been generated and is in the queue. Let n be an unexpanded node on the shortest path to an optimal goal G1.

\[ f(n) = g(n) + h(n) \]

\[ \leq g(G1) \quad \text{Why?} \]

\[ < g(G2) \quad G2 \text{ is suboptimal} \]

\[ = f(G2) \quad f(G2) = g(G2) \]

So \( f(n) < f(G2) \) and A* will never select G2 for expansion.
Optimality of A* with Consistency (stronger condition)

• $h(n)$ is consistent if
  – for every node $n$
  – for every successor $n'$ due to legal action $a$
  – $h(n) \leq c(n,a,n') + h(n')$

• Every consistent heuristic is also admissible.
Algorithms for A*

• Since Nillsson defined A* search, many different authors have suggested algorithms.

• Using Tree-Search, the optimality argument holds, but you search too many states.

• Using Graph-Search, it can break down, because an optimal path to a repeated state can be discarded if it is not the first one found.

• One way to solve the problem is that whenever you come to a repeated node, discard the longer path to it.
The Rich/Knight Implementation

• a node consists of
  – state
  – g, h, f values
  – list of successors
  – pointer to parent
• OPEN is the list of nodes that have been generated and had h applied, but not expanded and can be implemented as a priority queue.
• CLOSED is the list of nodes that have already been expanded.
1) /* Initialization */

OPEN <- start node

Initialize the start node

  g:
  h:
  f:

CLOSED <- empty list
Rich/Knight

2) repeat until goal (or time limit or space limit)

- if OPEN is empty, fail
- BESTNODE <- node on OPEN with lowest f
- if BESTNODE is a goal, exit and succeed
- remove BESTNODE from OPEN and add it to CLOSED
- generate successors of BESTNODE
for each successor \( s \) do

1. set its parent field
2. compute \( g(s) \)
3. if there is a node \( \text{OLD} \) on OPEN with the same state info as \( s \)
   
   \{
   \text{add } \text{OLD} \text{ to successors(BESTNODE)}
   \text{if } g(s) < g(\text{OLD}), \text{ update OLD and}
   \text{throw out } s
   \}
4. if (s is not on OPEN and there is a node OLD on CLOSED with the same state info as s

{ add OLD to successors(BESTNODE)
if g(s) < g(OLD), update OLD,
    remove it from CLOSED
    and put it on OPEN, throw out s

}
5. If $s$ was not on OPEN or CLOSED
   
   \{ add $s$ to OPEN
       
       add $s$ to successors(BESTNODE)
       
       calculate $g(s)$, $h(s)$, $f(s)$ \}

end of repeat loop