Uninformed Search II
Informed Search I
Outline for today’s lecture

Uninformed Search

- **Iterative deepening search** *(AIMA 3.4.4-3.4.5)*
  - *Strange Subroutine: Depth-limited search*
  - *Depth-limited search + iteration = WIN!!*

- Briefly: Bidirectional Search

- “Uniform Cost” Search (UCS)

Informed Search

- Introduction to Informed search
  - Heuristics

- 1\text{st} attempt: Greedy Best-first search
Search Conundrum

- **Breadth-first**
  - ✓ Complete,
  - ✓ Optimal
  - ✗ *but* uses $O(b^d)$ space

- **Depth-first**
  - ✗ Not complete *unless* $m$ is bounded
  - ✗ Not optimal
  - ✗ Uses $O(b^m)$ time; terrible if $m \gg d$
  - ✓ *but* only uses $O(b \times m)$ space

**How can we get the best of both?**
Depth-limited search: A building block

• **Depth-First search** *but with depth limit* \( \ell \).
  • i.e. nodes at depth \( \ell \) *have no successors.*
  • No infinite-path problem!

• If \( \ell = d \) (by luck!), then optimal
  • But:
    — If \( \ell < d \) then incomplete 😞
    — If \( \ell > d \) then not optimal 😞

• **Time complexity:** \( O(b^\ell) \)
• **Space complexity:** \( O(bl) \) 😊
Iterative deepening search

- A general strategy to find best depth limit \( l \).
  - Key idea: use \textit{Depth-limited search} as subroutine, with increasing \( l \).

\[
\text{For } l = 0 \text{ to } \infty \text{ do}
\]
\[
\begin{align*}
\text{depth-limited-search to level } l \\
\text{if it succeeds then return solution}
\end{align*}
\]

- \textit{Complete & optimal}: Goal is always found at depth \( d \), the depth of the shallowest goal-node.

Could this possibly be efficient?
Nodes constructed at each deepening

- Depth 0: 0 (Given the node, doesn’t construct it.)

- Depth 1: $b^1$ nodes

- Depth 2: $b$ nodes + $b^2$ nodes

- Depth 3: $b$ nodes + $b^2$ nodes + $b^3$ nodes

... 

Suppose the first solution is the last node at depth 3:
Total nodes constructed:
$3^*b$ nodes + $2^*b^2$ nodes + $1^*b^3$ nodes
ID search, Evaluation II: Time Complexity

• More generally, the time complexity is

\[(d)b + (d-1)b^2 + \ldots + (1)b^d = O(b^d)\]

• As efficient in terms of \(O(\ldots)\)

  as Breadth First Search:

\[b + b^2 + \ldots + b^d = O(b^d)\]
ID search, Evaluation III

- Complete: YES (no infinite paths) 😊

- Time complexity: \( O(b^d) \)

- Space complexity: \( O(bd) \) 😊

- Optimal: YES if step cost is 1. 😊
## Summary of algorithms

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Depth-First</th>
<th>Depth-limited</th>
<th>Iterative deepening</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Time</td>
<td>$b^d$</td>
<td>$b^m$</td>
<td>$b^l$</td>
<td>$b^d$</td>
</tr>
<tr>
<td>Space</td>
<td>$b^d$</td>
<td>$b^m$</td>
<td>$b^l$</td>
<td>$bd$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
Outline for today’s lecture

Uninformed Search

- Iterative deepening search
  - Strange Subroutine: Depth-limited search
  - Depth-limited search + iteration = WIN!!
- Briefly: Bidirectional Search (AIMA 3.4.6)
- “Uniform Cost” Search (UCS)

Informed Search

- Introduction to Informed search
  - Heuristics
- 1st attempt: Greedy Best-first search
Very briefly: Bidirectional search

- Two simultaneous searches from start an goal.
  - Motivation: \( b^{d/2} + b^{d/2} < b^d \)
- Check whether the node belongs to the other frontier before expansion.
- Complete and optimal if both searches are Breadth-First.
- Most significant weakness: Space complexity
How to search backwards?

- The *predecessor* of each node must be efficiently computable.
  - Works well when actions are easily reversible.
Outline for today’s lecture

**Uninformed Search**
- Iterative deepening search
  - Strange Subroutine: Depth-limited search
  - Depth-limited search + iteration = WIN!!
- Briefly: Bidirectional Search
- “Uniform Cost” Search (UCS) (AIMA 3.4.2)

**Informed Search**
- Introduction to Informed search
  - Heuristics
- 1st attempt: Greedy Best-first search
“Uniform Cost” Search

“In computer science, uniform-cost search (UCS) is a tree search algorithm used for traversing or searching a weighted tree, tree structure, or graph.” - Wikipedia
Motivation: Romanian Map Problem

- All our search methods so far assume \( \text{step-cost} = 1 \)
- This is only true for some problems
**g(N): the path cost function**

- **Our assumption so far:** All moves equal in cost
  - Cost = # of nodes in path - 1
  - $g(N) = depth(N)$ in the search tree
  - *Equivalent to what we’ve been assuming so far*

- **More general:** Assigning a (potentially) unique cost to each step
  - $N_0, N_1, N_2, N_3 =$ nodes visited on path $p$ from $N_0$ to $N_3$
  - $C(i,j)$: Cost of going from $N_i$ to $N_j$
  - If $N_0$ the root of the search tree,
    \[ g(N3) = C(0,1) + C(1,2) + C(2,3) \]
Uniform-cost search (UCS)

- Extension of BF-search:
  - Expand node with *lowest path cost*

- Implementation:
  - $frontier = \text{priority queue ordered by } g(n)$

- Subtle but significant difference from BFS:
  - Tests if a node is a goal state when it is selected for expansion, *not when it is added to the frontier*.
  - Updates a node on the frontier if a better path to the same state is found.
  - So always enqueues a node *before checking whether it is a goal*.

**WHY???>**
Uniform Cost Search

Expand cheapest node first:

Frontier is a priority queue

No longer ply at a time, but follows cost contours

Therefore: Must be optimal

Slide from Stanford CS 221 (from slide by Dan Klein (UCB) and many others)
Complexity of UCS

- Complete!
- Optimal!
  - if the cost of each step exceeds some positive bound $\varepsilon$.
- Time complexity: $O(b^{C^*/\varepsilon} + 1)$
- Space complexity: $O(b^{C^*/\varepsilon} + 1)$

where $C^*$ is the cost of the optimal solution
(if all step costs are equal, this becomes $O(b^{d+1})$

NOTE: Dijkstra’s algorithm just UCS without goal
### Summary of algorithms (for notes)

<table>
<thead>
<tr>
<th>Criterion</th>
<th>Breadth-First</th>
<th>Uniform-cost</th>
<th>Depth-First</th>
<th>Depth-limited</th>
<th>Iterative deepening</th>
<th>Bidirectional search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Time</td>
<td>$b^d$</td>
<td>$b(C^*/e)+1$</td>
<td>$b^m$</td>
<td>$b^l$</td>
<td>$b^d$</td>
<td>$b^{d/2}$</td>
</tr>
<tr>
<td>Space</td>
<td>$b^d$</td>
<td>$b(C^*/e)+1$</td>
<td>$bm$</td>
<td>$bl$</td>
<td>$bd$</td>
<td>$b^{d/2}$</td>
</tr>
<tr>
<td>Optimal?</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Assumes $b$ is finite**

---

CIS 521 - Intro to AI - Spring 2016
Outline for today’s lecture

**Uninformed Search**

- Iterative deepening search (AIMA 3.4.4-3.4.5)
  - Strange Subroutine: Depth-limited search
  - Depth-limited search + iteration = WIN!!
- Briefly: Bidirectional Search
- “Uniform Cost” Search (UCS)

**Informed Search**

- Introduction to Informed search
  - Heuristics
- 1st attempt: Greedy Best-first search
Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad.
Is Uniform Cost Search the best we can do?
Consider finding a route from Bucharest to Arad.

WRONG WAY!!!!!
A Better Idea…

- Node expansion based on an **estimate** which **includes distance to the goal**

- General approach of informed search:
  - **Best-first search**: node selected for expansion based on an **evaluation function** $f(n)$
    - $f(n)$ includes **estimate** of distance to goal (**new idea!**)

- Implementation: Sort frontier queue by this new $f(n)$.
  - Special cases: greedy search, **A* search**
Simple, useful estimate *heuristic*: straight-line distances
Heuristic (estimate) functions

Heureka! ---Archimedes

[dictionary]“A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood.”

Heuristic knowledge is useful, but not necessarily correct.

Heuristic algorithms use heuristic knowledge to solve a problem.

A heuristic function $h(n)$ takes a state $n$ and returns an estimate of the distance from $n$ to the goal.

(graphic: http://hyperbolegames.com/2014/10/20/eureka-moments/)
Breadth First for Games, Robots, …

- Pink: Starting Point
- Blue: Goal
- Teal: Scanned squares
  - Darker: Closer to starting point…

Graphics from
http://theory.stanford.edu/~amitp/GameProgramming/
(A great site for practical AI & game Programming)
vs. an optimal *informed search* algorithm (A*)

- We add a *heuristic estimate* of distance to the goal
- Yellow: examined nodes with *high estimated* distance
- Blue: examined nodes with *low estimated* distance
Breadth first in a world with obstacles
Greedy best-first search in a world with obstacles
Informed search (A*) in a world with obstacles
Outline for today’s lecture

Uninformed Search

- Briefly: Bidirectional Search
- “Uniform Cost” Search (UCS)

Informed Search

- Introduction to Informed search
  - Heuristics
- 1st attempt: Greedy Best-first search (AIMA 3.5.1)
Review: Best-first search

Basic idea:

- **select node for expansion** with minimal evaluation function $f(n)$
  - where $f(n)$ is some function that includes *estimate heuristic* $h(n)$ of the remaining distance to goal

- Implement using priority queue
- Exactly UCS with $f(n)$ replacing $g(n)$
Greedy best-first search: \( f(n) = h(n) \)

- Expands the node that *is estimated* to be closest to goal
- Completely ignores \( g(n) \): the cost to get to \( n \)
- Here, \( h(n) = h_{SLD}(n) = \) straight-line distance from ` to Bucharest
Greedy best-first search example

- **Initial State** = Arad
- **Goal State** = Bucharest

<table>
<thead>
<tr>
<th>City</th>
<th>Distance</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>Neamn</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>100</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 best-first search example

Frontier queue:
- Sibiu 253
- Timisoara 329
- Zerind 374

<table>
<thead>
<tr>
<th>City</th>
<th>Distance</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>Dobroesti</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>Neamţ</td>
<td>234</td>
</tr>
<tr>
<td>Oradea</td>
<td>380</td>
</tr>
<tr>
<td>Pitesti</td>
<td>100</td>
</tr>
<tr>
<td>Rimnicu</td>
<td>193</td>
</tr>
<tr>
<td>Vilcea</td>
<td></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 best-first search example

Frontier queue:
- Fagaras 176
- Rimnicu Vilcea 193
- Timisoara 329
- Arad 366
- Zerind 374
- Oradea 380

<table>
<thead>
<tr>
<th>City</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<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>Arad</td>
<td>366</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>100</td>
</tr>
<tr>
<td>Rmn Nicu 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 best-first search example

Frontier queue:
- Bucharest 0
- Rimnicu Vilcea 193
- Sibiu 253
- Timisoara 329
- Arad 366
- Zerind 374
- Oradea 380

Goal reached!!
Properties of greedy best-first search

• **Optimal?**
  • No!

  — Found: *Arad → Sibiu → Fagaras → Bucharest (450km)*
  — Shorter: *Arad → Sibiu → Rimnicu Vilcea → Pitesti → Bucharest (418km)*
Properties of greedy best-first search

- **Complete?**
  - No – can get stuck in loops,
  - e.g., Iasi → Neamt → Iasi → Neamt → …
Properties of greedy best-first search

- **Complete?** No – can get stuck in loops,
  - e.g., Iasi → Neamt → Iasi → Neamt → …

- **Time?** $O(b^m)$ – worst case (like Depth First Search)
  - But a good heuristic can give dramatic improvement of average cost

- **Space?** $O(b^m)$ – priority queue, so worst case: keeps all (unexpanded) nodes in memory

- **Optimal?** No
Next time: A* search

- Best-known form of best-first search.
- Key Idea: avoid expanding paths that are already expensive, but expand most promising first.
- **Simple idea:** \( f(n) = g(n) + h(n) \)
  - \( g(n) \) the cost (so far) to *reach* the node
  - \( h(n) \) estimated cost to *get from the node to the goal*
  - \( f(n) \) estimated *total cost* of path through \( n \) to goal
- Implementation: Frontier queue as priority queue by increasing \( f(n) \) (*as expected...*)
The Goat, Cabbage, Wolf Problem

PROBLEM: THE BOAT ONLY HOLDS TWO, BUT YOU CAN'T LEAVE THE GOAT WITH THE CABBAGE OR THE WOLF WITH THE GOAT.

SOLUTION:
1. TAKE THE GOAT ACROSS.
2. RETURN ALONE.
3. TAKE THE CABBAGE ACROSS.
4. LEAVE THE WOLF.

WHY DID YOU HAVE A WOLF?

(From xkcd.com)