Lecture 8

- Heuristic search
- Motivational video
- Assignment 2

Informed Search

DAA Ch. 4

- Search methods so far based on expanding nodes is search space *based on distance from the start node.*
  - Obviously, we always know that!

- How about using the estimated distance *h'(n)* to the goal!
  - What's the problem?

If we knew *h(n) exactly*, it wouldn't even be "search". We would just expand the *d* needed nodes that define the solution.

Heuristic Search

- What is we just *have a guess* as to the distance to the goal: a **heuristic**. (like "Eureka!")

Best-First Search

- At any time, expand the most promising node.
- Recall our general search algorithm (from last lecture).
  - We can re-order to list *L* to put the best node at the front of the list.

Compare this to uniform-cost search
which is, in some sense, the opposite!

Best-First

- Best-First is like DFS

- HOW much like DFS depends on the character of the heuristic

  evaluation function *h'(n)*

  – If it's zero all the time, we get BFS

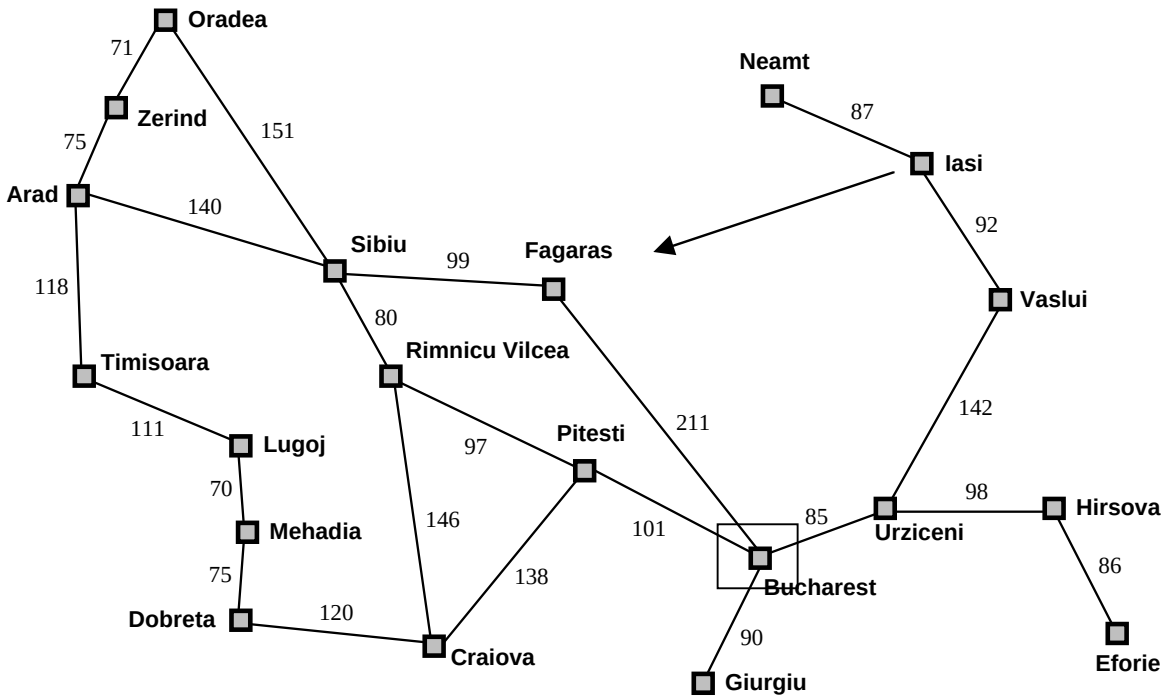
- Best-first is a **greedy** method.

  – Greed methods maximize short-term advantage without worrying about long-term consequences.

Example: route planning

- Consider planning a path along a road system.
- The straight-line distance from one place to another is a reasonable heuristic measure.
- Is it always right?
  - Clearly not: some roads are very circuitous.

Example: The Road to Bucharest



Straightline 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

Problem: Too Greedy

- From Arad to Sibiu to Fagaras --- but to Rimnicu would have been better.
- Need to consider: cost of getting from start node (Arad) to intermediate nodes!

Intermediate nodes

- Desirability of an intermediate node
  - How much it costs to get there
  - How much farther one has to go afterwards
- Leads to evaluation function of the form:

$$e(n) = g(n) + h'(n)$$

  - As before, $h'(n)$
  - Use $g(n)$ express the accumulated cost to get to this node.
- **This expresses the *estimated* total cost of the best solution that passes through state n.**

Details...

- Set *L* to be the initial node(s).
- Let *n* be the node on L that minimizes to *e(n)*.
- If *L* is empty, fail.
- If *n* is a goal node, stop and return it (and the path from the initial node to n).
- Otherwise, remove n from *L* and add all of n's
- children to *L* (labeling each with its path from the initial node).

Finds Optimal Path

- Now expands Rimnicu

(f = (140 + 80) + 193 = 413)

over

Faragas

(f = (140 + 99) + 178 = 417).

- Q. What if *h(Faragas) = 170* (also an underestimate)?

Admissibility

- An **admissible** heuristic always finds the best (lowest-cost) solution **first.**
  - Sometimes we refer to the algorithm being admissible -- a minor “abuse of notation”.
  - Is BFS admissible?

- If

$$h'(n) <= h(n)$$

then the heuristic is admissible.

- It means it never overestimates the cost of a solution through *n*.

A*

- The effect is that is it never overly optimistic (overly adventurous) about exploring paths in a DFS-like way.

- If we use this type of evaluation function with and admissible heuristic, we have algorithm A*
  - A* search

If for any triple of nodes,
cost(a,b) + cost(b,c) >= cost(a,c)
the heuristic is **monotonic.**

Monotonicity

- Let's also also assume (true for most admissible heuristics that *e* is **monotonic**, i.e., along any path from the root f never decreases.
- Can often modify heuristic to become monotonic if it isn't already.
- E.g. let *n* be parent of *n'*. Suppose that *g(n)* = 3 and *h(n) = 4* , so *f(n) = 7*.
  and *g(n') = 4* and *h(n') = 2*, so *f(n') = 6*.
- But, because any path through *n'* is also a path through n$, we can set *f(n') = 7*.

A* is good

- If *h'(n) = h(n)* then we know "everything" and *e(n)* is exact.
- If *e(n)* was exact, we could expand only the nodes on the actual optimal path to the goal.
- Note that in practice, *e(n)* is always an **underestimate**.

- So, we always expand more nodes than needed to find the optimal path.
- Sometimes we expand extra nodes, but they are always nodes that are "too close" to the start.

- A* finds the optimal path (first)
- A* is complete
- A8 is *optimally efficient* for a given heuristic: if we ever skipped expanding a node, we might make a serious mistake.

Video

- Nick Roy (former McGill grad student) describes tour-guide robot on the discovery channel.

- Note: such a mobile robot might well use A* search.
  - Obstacles are a major source of the mismatch between h' and h.
  - Straight-line distance provides a natural estimate for h'

- Learning about the environment is important.
  - We'll talk more about that soon….

Hill-climbing search

- Usually used when we do not have a **specific** goal state.
- Often used when we have a solution/path/setting and we want to improve it:
  an **iterative improvement algorithm.**

- Always move to a successor that that increases the desirability (can minimize or maximize cost).