The search problem

Suppose our problem is:

▶ fully observable
▶ discrete
▶ known (modeled)
▶ deterministic

Then, we can use classical search!

Assume we are given:

▶ a known initial state
▶ a set of actions and a model of the effects of those actions (the transition model)
▶ cost function
▶ goal test
An example of a search problem

- fully observable?
- discrete?
- known (modeled)?
- deterministic?
An example of a search problem

- initial state?
- actions, transition model?
- cost function?
- goal test?
Another example of a search problem

- fully observable?
- discrete?
- known (modeled)?
- deterministic?
Another example of a search problem

- initial state?
- actions, transition model?
- cost function?
- goal test?
Breadth-first search (BFS)

Search strategy: expand *shallowest* node first.

▶ This is an *uninformed* search strategy. Why?
▶ Where/what is the *frontier*?
▶ BFS can be implemented by a FIFO queue of frontier nodes (illustrate on the board).
▶ Computational complexity: How big does the FIFO queue get for a graph w/ branching factor \( b \) and maximum depth \( d \)?
  ▶ what is the asymptotic complexity?
▶ Is BFS complete (is it guaranteed to find a path if one exists)? Is it optimal (does it find the shortest path to the goal)?
▶ For which problem is BFS appropriate?
Breadth-first search (BFS): example
Uniform-cost search (UCS)

But, does BFS make sense for the road map problem?

UCS: always choose to expand the frontier node with lowest cost.

- Generalizes BFS to graphs with weighted edges.
- Can be implemented by a priority queue.
- Computational complexity?
Depth first search (DFS)

What can we do to improve the space requirements of BFS?

DFS: expand the deepest node first.

- what is the storage requirement w/ BFS (in terms of branching factor, $b$, and max depth, $m$)?
- what is the storage requirement w/ DFS?
- is it complete? optimal? Finite vs. infinite search spaces...
Iterative deepening search

Can we combine the space advantages of DFS w/ the optimality of BFS?

Iteratively do depth limited search (what’s that?) w/ successively increasing depth limits

▶ what is the asymptotic complexity now?
Heuristic search: greedy best-first search

So far, we have covered covered *uninformed* search strategies, where search is undirected. But, what if we have heuristics that can guide search?

Greedy best-first search:
- Assume the search algorithm has access to a heuristic that evaluates how useful it will be to expand a particular fringe node.
- Always choose to expand the fringe node w/ the best heuristic value.
Greedy best-first search: example

- Is this algorithm complete? Is it optimal?
Heuristic search: $A^*$

Combine greedy search w/ UCS. Consider the application to search in a weighted graph:

- $n$: number of a particular node in the tree.
- $h(n)$: heuristic value of that node.
- $g(n)$: cost so far to get to that node.
- UCS: evaluate $g(n)$ for each node on fringe. Expand node w/ highest value.
- Greedy: evaluate $h(n)$ for each node on fringe. Expand node w/ highest value.
- $A^*$: evaluate $g(n) + h(n)$ for each node on fringe. Expand node w/ highest value.
Is this algorithm more or less efficient than UCS?

Is this algorithm complete? Is it optimal?
The heuristic function needs to satisfy certain conditions in order for A* to be optimal:

- **Admissibility**: the heuristic function cannot *overestimate* the cost to goal: the heuristic should be ”optimistic”.
  - what’s an example of an admissible heuristic for the map problem?

- **Consistency** (monotonicity): $h(n) \leq c(n, a, n') + h(n')$.
  - This is a kind of triangle inequality.
  - If $h(n)$ is consistent, then the values of $h(n) + g(n)$ along any path are nondecreasing.
Optimality of A* 

Map of Romania showing contours at \( f = 380 \), \( f = 400 \), and \( f = 420 \), with Arad as the start state. Nodes inside a given contour have \( f \)-costs less than or equal to the contour value.

- Notice that consistency implies admissibility. Why?
- Notice that consistency implies optimality of A*. Why?
A*: the power of heuristics

The 8-puzzle.
Are there general strategies for creating heuristics:
▶ Solve a relaxed version of the problem.
▶ Solve a subproblem.
▶ Can you suggest a good heuristic for the 8-puzzle?
A*: the power of heuristics

Start State

Goal State
How many nodes are expanded to solve an 8-puzzle problem w/ optimal solution depth 12?

- IDS: 3.6M
- A* w/ displaced tile heuristic: 227
- A* w/ manhattan heuristic: 73