Announcements

- Programming assignment 1 is on the webpage
  - Start early
  - Due on Sunday April 20

- Any other Python/version issues?
Recap

- Rational Agents
- Problem state spaces and search problems
- Uninformed search algorithms
  - DFS
  - BFS
  - Iterative Deepening
  - UCS
Recap

- Heuristics

- Greedy Solutions
  - Best First

- Can we do better?
Example: Pancake Problem

Action: Flip over the top $n$ pancakes

Cost: Number of pancakes flipped
Example: Pancake Problem

BOUND FOR SORTING BY PREFIX REVERSAL

William H. GATES
Microsoft, Albuquerque, New Mexico

Christos H. PAPADIMITRIOU*†
Department of Electrical Engineering, University of California, Berkeley, CA 94720, U.S.A.

Received 18 January 1978
Revised 28 August 1978

For a permutation $\sigma$ of the integers from 1 to $n$, let $f(\sigma)$ be the smallest number of prefix reversals that will transform $\sigma$ to the identity permutation, and let $f(n)$ be the largest such $f(\sigma)$ for all $\sigma$ in (the symmetric group) $S_n$. We show that $f(n) \leq (5n + 5)/3$, and that $f(n) \geq 17n/16$ for $n$ a multiple of 16. If, furthermore, each integer is required to participate in an even number of reversed prefixes, the corresponding function $g(n)$ is shown to obey $3n/2 - 1 \leq g(n) \leq 2n + 3$. 
Example: Pancake Problem

State space graph with costs as weights
General Tree Search

function Tree-Search \( (\text{problem, strategy}) \) returns a solution, or failure
initialize the search tree using the initial state of \text{problem}
loop do
  if there are no candidates for expansion then return failure
  choose a leaf node for expansion according to \text{strategy}
  if the node contains a goal state then return the corresponding solution
  else expand the node and add the resulting nodes to the search tree
end

- Action: flip top two
  Cost: 2

- Path to reach goal:
  Flip four, flip three
  Total cost: 7
Uniform Cost Search

- **Strategy:** expand lowest path cost

- **The good:** UCS is complete and optimal!

- **The bad:**
  - Explores options in every “direction”
  - No information about goal location
Uniform Cost

- Cost of 1 for each action
- Explores all of the states, but one
Search Heuristics

- Any estimate of how close a state is to a goal
- Designed for a particular search problem

Examples: Manhattan distance, Euclidean distance
Example: Heuristic Function

$h(x)$: assigns a value to a state
Example: Heuristic Function

Heuristic: the largest pancake that is still out of place

h(x)
Best First Search (Greedy)

- Expand the node that seems closest...

- What can go wrong?
Best First (Greedy)

- **Strategy**: expand a node that you think is closest to a goal state
  - **Heuristic**: estimate of distance to nearest goal for each state

- **A common case**: Best-first takes you straight to the (wrong) goal

- **Worst-case**: like a wrongly-guided DFS
Greedy Solution
Combining UCS and Greedy

- **Uniform-cost** orders by path cost, or *backward cost* \( f(n) = g(n) \)
- **Best-first** orders by goal proximity, or *forward cost* \( f(n) = h(n) \)
- **A* Search** orders by the sum: \( f(n) = g(n) + h(n) \)
When should A* terminate?

- Should we stop when we enqueue a goal?

  - **No:** only stop when we dequeue a goal
Is A* Optimal?

What went wrong?
- Actual bad goal cost < estimated good goal cost
- We need estimates to be less than actual costs!
Admissible Heuristics

- A heuristic $h$ is admissible (optimistic) if:

$$h(n) \leq h^*(n)$$

where $h^*(n)$ is the true cost to a nearest goal.

- Examples:

- Coming up with admissible heuristics is most of what’s involved in using A* in practice.
Assume:
- G* is an optimal goal
- G is a sub-optimal goal
- h is admissible

Claim:
- G* will exit fringe before G
Optimality of A*: Blocking

Notation:
- $g(n) =$ cost to node $n$
- $h(n) =$ estimated cost from $n$
  to the nearest goal (heuristic)
- $f(n) = g(n) + h(n) =$
  estimated total cost via $n$
- $G^*$: a lowest cost goal node
- $G$: another goal node
Optimality of A*: Blocking

Proof:
- What could go wrong?
  - We’d have to have to pop a suboptimal goal G off the fringe before G*

- This can’t happen:
  - For all nodes $n$ on the best path to G*
    - $f(n) < f(G)$
  - So, G* will be popped before G

$$f(n) = g(n) + h(n)$$
$$g(n) + h(n) \leq g(G^*)$$
$$g(G^*) < g(G)$$
$$g(G) = f(G)$$
$$f(n) < f(G)$$
Properties of A*

Uniform-Cost

A*
UCS vs A* Contours

- Uniform-cost expanded in all directions

- A* expands mainly toward the goal, but will hedge its bets to ensure optimality
Astar
UCS

- 9000 States
Astar

- 180 States
Creating Admissible Heuristics

- Most of the work in solving hard search problems optimally is in coming up with admissible heuristics.

- Often, admissible heuristics are solutions to relaxed problems, where new actions are available.

- Inadmissible heuristics are often useful too (why?)
Creating Heuristics

8-puzzle:

- What are the states?
- How many states?
- What are the actions?
- What states can I reach from the start state?
- What should the costs be?
8 Puzzle I

- Heuristic: Number of tiles misplaced
- $h(\text{start}) = 8$
- Is it admissible?
What if we had an easier 8-puzzle where any tile could slide any direction at any time, ignoring other tiles?

- Total *Manhattan* distance
- \( h(\text{start}) = 3 + 1 + 2 + \ldots \)
  \[ = 18 \]
- Admissible?
8 Puzzle III

- How about using the *actual cost* as a heuristic?
  - Would it be admissible?
  - Would we save on nodes expanded?
  - What’s wrong with it?

- With A*: a trade-off between quality of estimate and work per node!
Trivial Heuristics, Dominance

- Dominance: \( h_a \geq h_c \) if
  \[
  \forall n : h_a(n) \geq h_c(n)
  \]

- Heuristics form a semi-lattice:
  - Max of admissible heuristics is admissible
  \[
  h(n) = \max(h_a(n), h_b(n))
  \]

- Trivial heuristics
  - Bottom of lattice is the zero heuristic (what does this give us?)
  - Top of lattice is the exact heuristic
Which Search Strategy?
Which Search Strategy?
Which Search Strategy?
Which Search Strategy?
Which Search Strategy?
Tree Search: Extra Work!

- Failure to detect repeated states can cause exponentially more work. Why?
Graph Search

- In BFS, for example, we shouldn’t bother expanding some nodes (which, and why?)
Graph Search

- Idea: never expand a state twice

- How to implement:
  - Tree search + list of expanded states (closed list)
  - Expand the search tree node-by-node, but…
  - Before expanding a node, check to make sure its state is new

- Python trick: store the closed list as a set, not a list

- Can graph search wreck completeness? Why/why not?

- How about optimality?
A* Graph Search Gone Wrong

State space graph

Search tree

S (0+2)
A (1+4)
B (1+1)
C (2+1)
G (5+0)

A (1+4)
C (3+1)
G (6+0)

G (5+0)
Consistency

- Wait, how do we know parents have better f-values than their successors?

- **Consistency** for all edges \((A,a,B)\):
  - \(h(A) \leq c(A,a,B) + h(B)\)

- **Proof that** \(f(B) \geq f(A)\),
  - \(f(B) = g(B) + h(B) = g(A) + c(A,a,B) + h(B) \geq g(A) + h(A) = f(A)\)
Optimality of A* Graph Search

Proof:

- Main idea: Show nodes are popped with non-decreasing f-scores
  - for n’ popped after n:
    - \( f(n’) \geq f(n) \)
  - is this enough for optimality?

- Sketch:
  - assume: \( f(n’) \geq f(n) \), for all edges \((n,a,n’)\) and all actions a
  - is this true?
  - proof: A* never expands nodes with the cost \( f(n) > C^* \)
  - proof by induction(1) always pop the lowest f-score from the fringe, (2) all new nodes have larger (or equal) scores, (3) add them to the fringe, (4) repeat!
Optimality

- **Tree search:**
  - A* optimal if heuristic is admissible (and non-negative)
  - UCS is a special case (h = 0)

- **Graph search:**
  - A* optimal if heuristic is consistent
  - UCS optimal (h = 0 is consistent)

- Consistency implies admissibility

- In general, natural admissible heuristics tend to be consistent
Summary: A*

- A* uses both backward costs and (estimates of) forward costs.
- A* is optimal with admissible (and/or consistent) heuristics.
- Heuristic design is key: often use relaxed problems.
A* Applications

- Pathing / routing problems
- Resource planning problems
- Robot motion planning
- Language analysis
- Machine translation
- Speech recognition
- …
Which Algorithm?
Which Algorithm?
Which Algorithm?
Which Algorithm?

- Uniform cost search (UCS):
Which Algorithm?

- A*, Manhattan Heuristic:
Which Algorithm?

- Best First / Greedy, Manhattan Heuristic: