Search Trees

Applies best to fully observable, deterministic, unchanging environments.
    but still used for just about everything

A search tree is not the same as the state space
    each state may appear multiple times.

The tree doesn't necessarily exist.

Nodes are states. Edges are actions.

Exploration methods:

- Open List, Closed List
- Depth-first search
- Breadth-first search
- Best-first search: minimum value of an evaluation function
- Heuristic search:
    - Rule of thumb, e.g.
        - Straight line distance
        - Number of out-of-place tiles
        - Sum of distances from desired position
            - Manhattan distance
    - Admissible - never overestimate
    - A* search - eval. fn = cost so far (g) + est. cost of rest of path (h)
        - Dijkstra's is A* with h always = 0
    - Finding an heuristic - sometimes consider a relaxed problem
- Bidirectional search
- Frontier set
- Beam search - limit size of frontier set
- Land marks - short cuts

Redundant paths - a waste of memory

- Remember all reached states - closed set
- Don't worry about them
- Only check for cycles - easier

Performance

- Completeness - always get solution or signal failure
    - e.g. DFS for chess: moving back and forth
- Optimality - does solution have optimal cost?
- Time complexity
- Space complexity