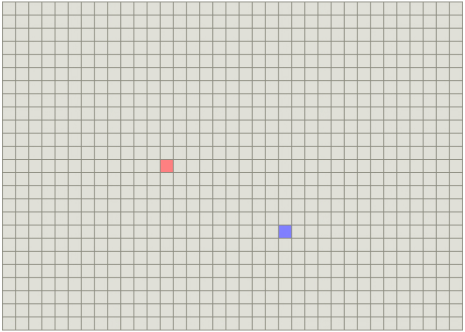
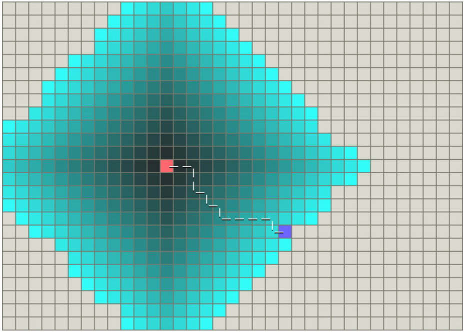
Heuristic Search

Dijkstra's Algorithm Behavior



start state: red, goal state: blue

Dijkstra's Algorithm Behavior



generated nodes: cyan (darker = earlier in priority queue)

Heuristic Estimation

Many of the generated nodes are unnecessary – they can't possibly be part of a good solution.

Heuristic knowledge is an estimation or educated guess, but not necessarily correct

Heuristic algorithms use heuristic knowledge to solve a problem

Heuristic functions (or just *heuristics*) take a state as an input and outputs an estimate of the cost to a goal state

Greedy Search

```
Open ← ordered list containing initial state
while true do
    if Open is empty then
        return failure
    Node ← Open.Pop()
    if Node is goal then
        return Node (or path to Node)
    else
        Children ← Expand(Node)
        Add Children to Open, sorting on heuristic
```

8-Puzzle

$h(n) =$ number of tiles out of place

Start state:

2	8	3
1	6	4
7	␣	5

Goal state:

1	2	3
8	␣	4
7	6	5

Draw a level of the search tree from expanding the start node, and determine *h(n)* for the child states.

Greedy Evaluation

Assume branching factor b and solution depth d

Completeness:
Time:
Space:
Admissibility:

A* Search

$f(n) = g(n) + h(n)$: $g(n)$ is cost from start state to current state, $h(n)$ is heuristic estimate from current state to goal

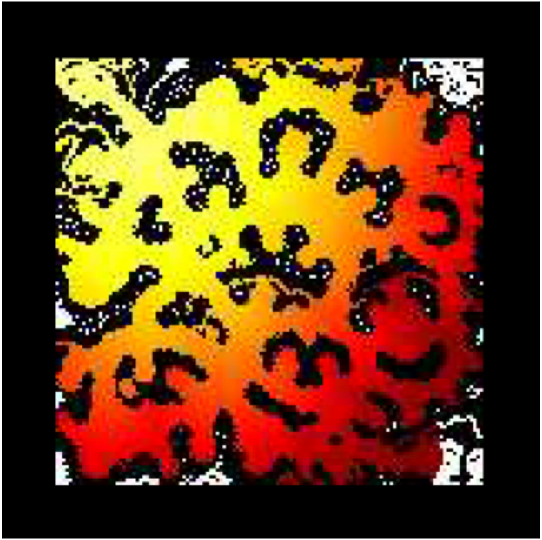
```
Open ← priority queue containing initial state
while true do
    if Open is empty then
        return failure
    Node ← Open.Pop()
    if Node is goal then
        return Node (or path to Node)
    else
        Children ← Expand(Node)
        Add Children to Open, sorted on f(n)
```

A* Evaluation

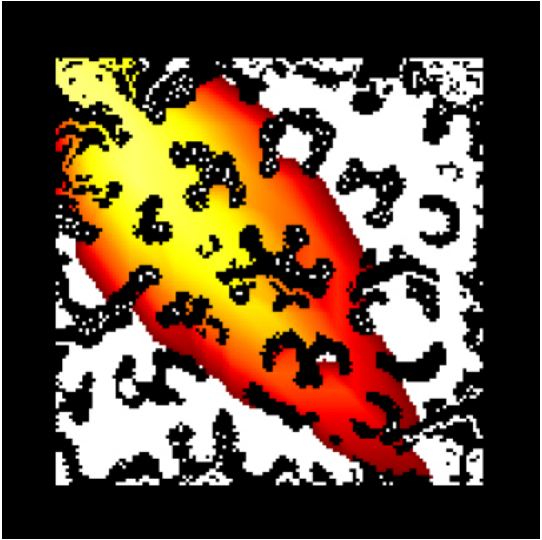
Assume branching factor b and solution depth d

Completeness:
Time:
Space:
Admissibility:

UCS Behavior



A* Behavior



A* Admissibility

- Assume A* finds a goal node s using an admissible heuristic.
- Suppose node p is a node on a path to a better solution b.
- $f(p) = g(p) + h(p) \leq f(b) = g(b)$
- But $g(s) = f(s) \leq f(p)$
- Therefore $g(s) \leq g(b)$ and s is optimal.

A* Optimality

- For admissible h, nodes in open list can be sorted on f
- A* expands these nodes in sorted order
- A* must examine all nodes with $f < f^*$

Heuristic Functions

Function values are based on solving a simpler related problem.
Function design:

- Relaxation: fewer or weaker constraints
- Abstraction: simplify search space by leaving out details

Admissibility:

- We want $h(n)$ to give the largest value that does not exceed the actual cost to the goal of our original problem
- If $h(n)$ never overestimates the cost to the goal, it is admissible and A* gives an optimal solution
- If $h_1(n) \leq h_2(n) \forall n$, h_2 dominates h_1 and is a better heuristic function

We want $h(n)$ to be easy to compute