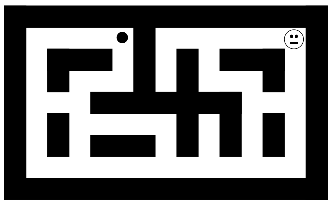
Artificial Intelligence

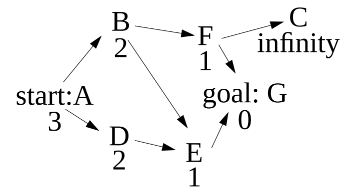
Informed Search

Nilsson - Chapter 9

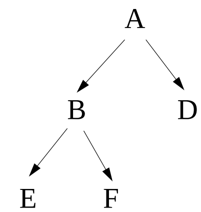
Russell and Norvig - Chapter 4



state space



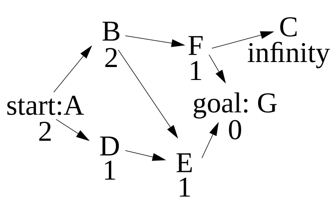
search tree



best-first search:
start with a tree that contains only the start state
pick a fringe node n (somehow using estimates of the goal distances)
if fringe node n represents a goal state: stop
expand fringe node n
go to

h: heuristic function
h(n): heuristic estimate of the goal distance of n (really: of s(n))
for example: h(A) = 3

Greedy Search

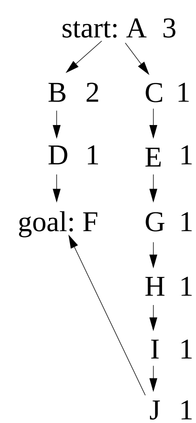


search tree?

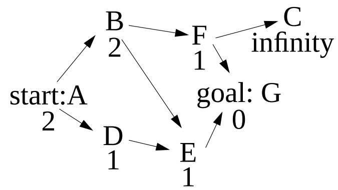
start with a tree that contains only the start state
pick a fringe node n **with the smallest h(n)**
if fringe node n represents a goal state: stop
expand fringe node n
go to

Greedy Search

works often well in practice
but has problems



A*



search tree?

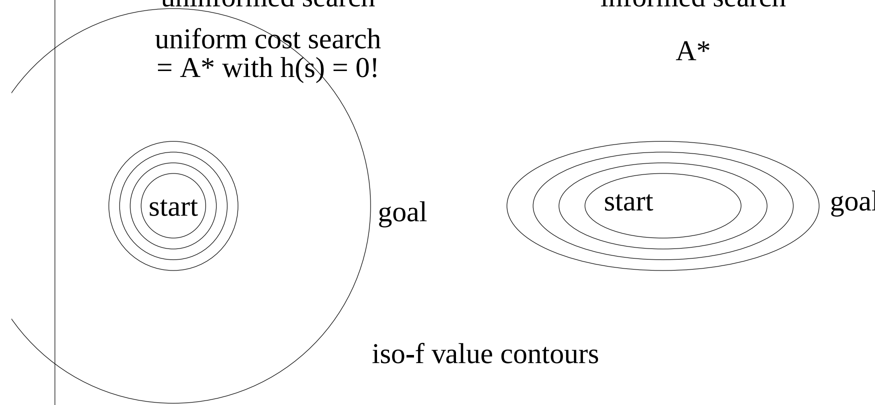
start with a tree that contains only the start state
pick a fringe node n **with the smallest f(n) = g(n) + h(n)**
if fringe node n represents a goal state: stop
expand fringe node n
go to

g(n) = depth of n (or weighted distance from root node to n)
h(n) = heuristic estimate of the goal distance of n
f(n) = heuristic estimate of the length of a shortest path
from the root to a goal node that goes through n
(really: from the start state to a goal state that goes through s(n))

Uninformed Search vs. Informed Search

uninformed search
uniform cost search
= A* with h(s) = 0!

informed search
A*



iso-f value contours

f-value

10.5	9.0	7.5	6.0	6.5	7.0	7.5	9.0	10.5
9.0	7.5	6.0	4.5	5.0	5.5	6.0	7.5	9.0
7.5	6.0	4.5	3.0	3.5	4.0	4.5	6.0	7.5
6.0	4.5	3.0	start 1.5	2.0	2.5	goal 3.0	4.5	6.0
7.5	6.0	4.5	3.0	3.5	4.0	4.5	6.0	7.5
9.0	7.5	6.0	4.5	5.0	5.5	6.0	7.5	9.0
10.5	9.0	7.5	6.0	6.5	7.0	7.5	9.0	10.5

I don't guarantee that I have
not made mistakes -
if you find one,
please let me know

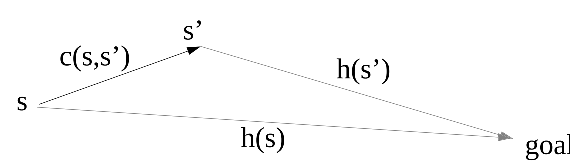
gridworld - the robot can move north, east, south, or west
h value = goal distance / 2

Properties of Heuristic Functions

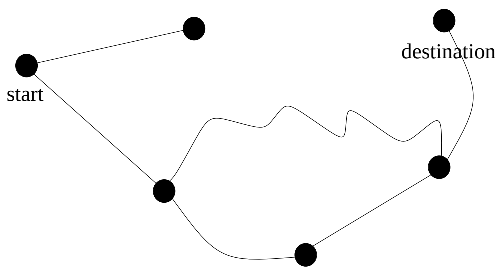
Let gd(s) be the goal distance of state s.
Let c(s,s') be the action cost of the action going from s to s'.

h is **admissible** iff, for all states s,
0 <= h(s) <= gd(s)
"the estimates never overestimate the true goaldistances"

h is **consistent** (or, synonymously, **monotonic**) iff, for all states s,
h(s) = 0 if s is a goal state
0 <= h(s) <= h(s') + c(s,s') for all succ states s' of s otherwise
"the triangle inequality holds"



Examples of Heuristic Functions



Straight-Line Distance:

given a map, use the straight-line distance from a city to the destination as its heuristic value

Examples of Heuristic Functions

4	1	3
7		5
8	2	6

1	2	3
4	5	6
7	8	

goal state

Tiles-Out-Of-Order heuristic:

the smallest number of moves needed to achieve the goal configuration if two or more tiles can occupy the same square and it counts as one move when a tile is removed from its current square and placed on any other square

admissible? consistent?

Manhattan heuristic:

the smallest number of moves needed to achieve the goal configuration if two or more tiles can occupy the same square and it counts as one move when a tile is slide from its current square to an adjacent square (up, down, left, or right)

admissible? consistent?

How to Obtain Heuristic Functions

Problem Relaxation

same states
same actions
add some states and actions

use the correct goal distances for this problem
as heuristic values for the original problem
(you need to be able to determine the correct goal distances WITHOUT search)

the resulting heuristic function is consistent and admissible!

How “Admissible” and “Consistent” Relate

h is consistent implies h is admissible

why?

prove h(s) <= gd(s) by induction on the goal distances:

the statement is true for all s with gd(s) = 0
pick an s. assume the statement is true for all s’ with gd(s’) < gd(s).
pick a shortest path from s to a goal.
let s’ be the successor state of s on that path.
then, h(s) <= c(s,s’) + h(s’) <= c(s,s’) + gd(s’) = gd(s)

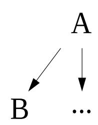
How “Admissible” and “Consistent” Relate

h is admissible does NOT imply h is consistent

start: A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ goal: E

0 0 2 0 0

but we can convert an admissible heuristic function h
into a different heuristic h’
that is consistent (and thus also admissible)



when A* expands A and generates B, it sets
h’(B) := max(h’(A) - c(A,B), h(B))
or, alternatively,
f’(B) := max(f’(A), g(B)+h(B))
“the pathmax equation”

(there are some problems with this in Russell and Norvig)

if h is consistent,
then the f-values of all nodes along a path in the search tree
are nondecreasing

A f(A) = g(A) + h(A)

c(A,B)

B f(B) = g(B) + h(B)
= g(A) + c(A,B) + h(B)

thus,
if h(A) <= c(A,B) + h(B),
then f(A) <= f(B)

A* expands nodes in order of non-decreasing f-values!

A* expands all nodes n with f(n) < f*,
some nodes n with f(n) = f*,
and no nodes with f(n) > f*.

h is consistent implies that A* is complete

why?

If the goal distance of the start state s is finite,
then there is at least one node g which represents a goal state
in the search tree with f(g) = g(g) + h(g) = g(g) + 0 = g(g) being finite.

A* expands nodes in order of non-decreasing f-values
and thus will reach g eventually.

(assumptions?)

h is consistent implies that A* is optimal

why?

Assume that A* stopped in node g2 (which represents a goal state)
but it could have found a shorter path from the root of the
search tree to a different node g1 (which also represents a goal state)

Let n be an unexpanded node on the path from the root to g1

Then,
g(g1) = g(g1) + h(g1) = f(g1) >= f(n) >= f(g2) = g(g2) + h(g2) = g(g2)
this means that a path from the root to g2 is optimal.

h is consistent implies A* is efficient

no other search method
that uses the same heuristic values,
that is optimal,
and that is also complete,
can expand fewer nodes than A*
(except for breaking ties at nodes n with f(n) = f*)

why?

“if the search method does not expand a node n with f(n) < f*,
make n represent a goal state.” (some hand waving here)

a consistent heuristic function simplifies the implementation of A*

if h is consistent (NOT: admissible),
then the f-values of all nodes along a path in the search tree
are nondecreasing

...
expand a node, say n, that corresponds to state s
...
expand another node, n’, that corresponds to state s

f(n) <= f(n’)
g(n)+h(s) <= g(n’)+h(s)
g(n) <= g(n’)

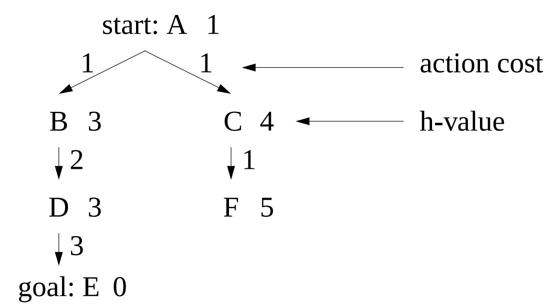
we do not need to expand node n’!
(= the first path found from the start state to state s is a shortest path)

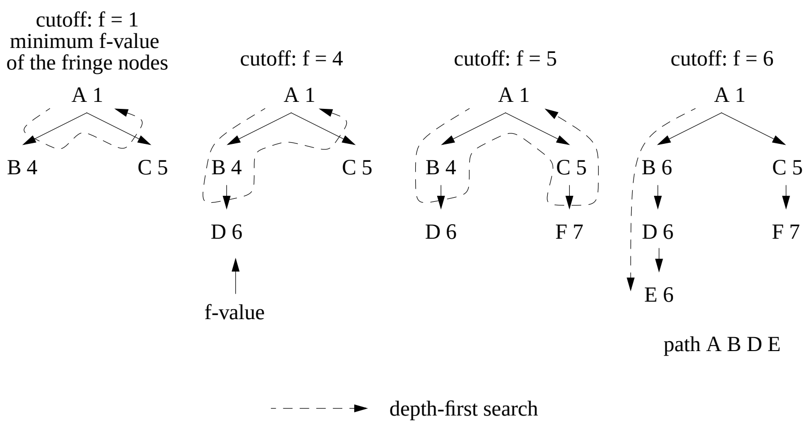
A* still needs an exponential amount of memory
we need memory-bounded heuristic search methods

combine A* with iterative deepening:
iterative deepening A* (IDA*)

or use
linear best-first search (LBFS, SMA*)

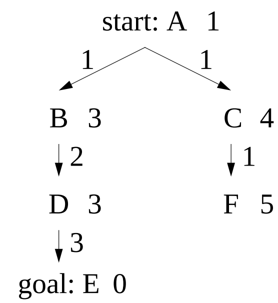
Example: Iterative-Deeping A*

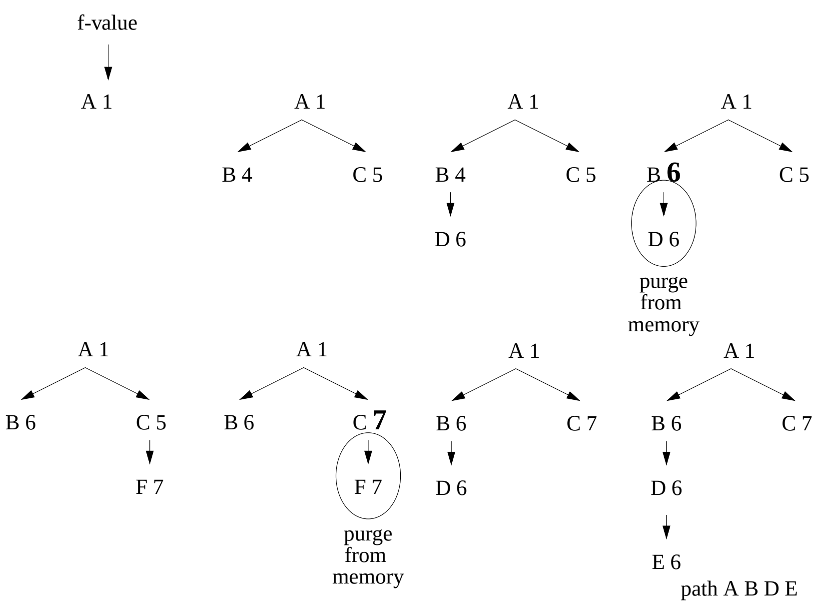




note: the depth-first search goes just BEYOND the cutoff for the f-values

Example: Linear-Best First Search





note: we do not need to purge parts of the tree if we still have memory available!

h **dominates** h' if, for all states, h(s) >= h'(s)

given two consistent heuristic functions h and h'
if h dominates h',
then A* with h cannot expand more nodes than A* with h'

why?

A* expands all nodes n with f(n) < f*, that is, h(n) < f* - g(n)

- does tiles-out-of-order dominate manhattan or vice versa?
- what do you do if you have two consistent heuristic functions
    - and one of them dominates the other?
    - and neither of them dominates the other?

Summary: How to Solve Search Problems with A*

- encode the search problem (states, actions, goal test and so on)
- design an admissible heuristic function (and check consistency)
  example: problem relaxation (no need to check consistency)
- apply A*

admissible: optimistic (never overestimates the true goal distance)
consistent: satisfies the triangle inequality

consistency implies admissibility
admissibility does NOT imply consistency
but in practice most admissible heuristics are consistent

an admissible heuristic function guarantees correctness and optimality!
a consistent heuristic function simplifies the implementation of A*!

larger consistent heuristic values make A* more efficient
example: if h1 and h2 are consistent, use h3(s) = max(h1(s), h2(s))!

A* Self-Test

Just for fun. Nothing to turn in. On the other hand, we won't provide sample solutions either.

Question 1:

Which problem(s) can arise if A* is used with a non-admissible heuristic function? Illustrate the problem(s) with a simple example.

Question 2:

Assume that A* uses a consistent heuristic function. Assume further that it expands node n at some point in time, the f-value of node n is f(n), and the node corresponds to state s in the state space. The path from the root of the search tree to n corresponds to a path from the start state of the state space to s. Show that subsequent node expansions of A* cannot find a shorter path from the start state of the state space to s. (Hint: Assume that A* later expands a different node n', the f-value of node n' is f(n'), and the node also corresponds to state s in the state space. Show first that f(n) <= f(n') and then that $g(n) <= g(n')$.) WHY IS THIS RESULT IMPORTANT? Which problems can arise if A* is used with an inconsistent but admissible heuristic function? Illustrate the problem(s) with a simple example. How can A* be changed to deal with the problem(s).

Question 3:

Assume that you are given two consistent heuristic functions h1 and h2. Show that h3 is consistent if h3(s) = max(h1(s), h2(s)) for all states s. WHY IS THIS RESULT IMPORTANT?