Informed Search

Chapter 3

3.15 Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k + 1$.

a. Draw the portion of the state space for states 1 to 15.

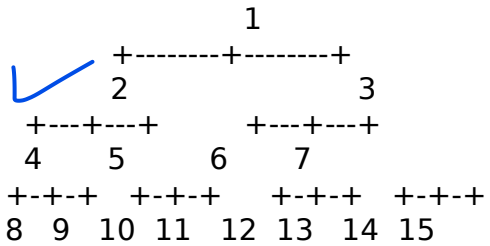
b. Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

c. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

d. Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

e. Call the action going from k to $2k$ Left, and the action going to $2k + 1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

a.



B.
BFS: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
DFS: 1, 2, 4, 8, 9, 5, 10, 11 assuming we stop at the limit of the drawn portion, otherwise, we will go through 1,2,4,8,16,32,... and so on forever.
Depth Limited with limit 3: same as DFS if we stop at the limit of the drawn portion.
Iterative Deepening: 1, 1,2,3, 1,2,4,5,3,6,7, 1,2,4,8,9,5,10,11

C. It will work well since the branching factor in the backward direction is only 1 (compared to 2 in the forward direction). So the backward search will quickly reach a state visited by the forward search. Overall, this will decrease the search complexity from O(b^d) for BFS to O(b^(d/2)) for Bidirectional search where b is the branching factor and d is the goal depth.

D.
Initial state: n
Goal Test: state = 1
Action: Go to parent
Transition model: new_state = floor(state / 2)

3.3 Suppose two friends live in different cities on a map, such as the Romania map shown in Figure 3.2. On every turn, we can simultaneously move each friend to a neighboring city on the map. The amount of time needed to move from city i to neighbor j is equal to the road distance $d(i, j)$ between the cities, but on each turn the friend that arrives first must wait until the other one arrives (and calls the first on his/her cell phone) before the next turn can begin. We want the two friends to meet as quickly as possible.

a. Write a detailed formulation for this search problem. (You will find it helpful to define some formal notation here.)

b. Let $D(i, j)$ be the straight-line distance between cities i and j. Which of the following heuristic functions are admissible? (i) $D(i, j)$; (ii) $2 \cdot D(i, j)$; (iii) $D(i, j)/2$.

c. Are there completely connected maps for which no solution exists?

d. Are there maps in which all solutions require one friend to visit the same city twice?

a.
State space: States are all possible city pairs (i, j). The map is not the state space.
Initial State: (i, j)
Goal: Be at (i, i) for some i.
Action: {Go((i,j) → (x,y)) where x ∈ neighbors(i) and y ∈ neighbors(j)}
Transition Model: if State=(i, j) and action=Go((i,j) → (x,y)), then the successor is (x, y).
Step cost function: The cost of Go((i,j) → (x,y)) is max(d(i, x), d(j, y)).

b. In the best case, the friends head straight for each other in steps of equal size, reducing their separation by twice the time cost on each step. Hence (iii) is admissible.

c. Yes: e.g., a map with two nodes connected by one link. The two friends will swap places forever. The same will happen on any chain if they start an odd number of steps apart. (One can see this best on the graph that represents the state space, which has two disjoint sets of nodes.) The same even holds for a grid of any size or shape, because every move changes the Manhattan distance between the two friends by 0 or 2.

d. Yes: take any of the unsolvable maps from part (c) and add a self-loop to any one of the nodes. If the friends start an odd number of steps apart, a move in which one of the friends takes the self-loop changes the distance by 1, rendering the problem solvable. If the self-loop is not taken, the argument from (c) applies and no solution is possible.

3.14 Which of the following are true and which are false? Explain your answers.

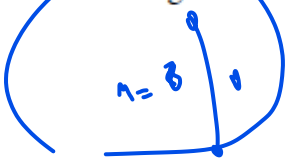
a. Depth-first search always expands at least as many nodes as A* search with an admissible heuristic.

b. $h(n) = 0$ is an admissible heuristic for the 8-puzzle.

c. A* is of no use in robotics because percepts, states, and actions are continuous.

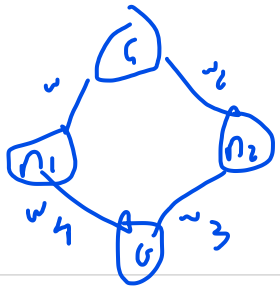
d. Breadth-first search is complete even if zero step costs are allowed.

e. Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.



a. False: a lucky DFS might expand exactly d nodes to reach the goal. A * largely dominates any graph-search algorithm that is guaranteed to find optimal solutions.
b. True: h(n) = 0 is always an admissible heuristic, since costs are nonnegative.
c. False: A* search is often used in robotics; the space can be discretized or skeletonized.
d. True: depth of the solution matters for breadth-first search, not cost.
e. False: a rook can move across the board in move one, although the Manhattan distance from start to finish is 8.

3.29 Prove that if a heuristic is consistent, it must be admissible. Construct an admissible heuristic that is not consistent.



A heuristic h is consistent if for any transition $(S, A) \rightarrow S_n$, $h(S) - h(S_n) \leq Cost(A, S)$
Where S_n is the state after we do the action A in state S.

So if the path from any state S_0 to S_g (a goal state) is the action sequence $[A_0, A_1, \ldots A_n]$, then
$h(S_0) - h(S_1) \leq Cost(A_0, S_0)$
$h(S_1) - h(S_2) \leq Cost(A_2, S_2)$
And so on till we reach
$h(S_n) - h(S_g) \leq Cost(A_n, S_n)$

And by summing all these inequalities, we get:
$h(S_0) - h(S_g) \leq Cost(A_0, S_0) + Cost(A_1, S_1) + \ldots + Cost(A_n, S_n) = PathCost(S_0, S_g)$

Since $h(S_g) = 0$ (the heuristic estimate at the goal is 0)
So $h(S_0) \leq PathCost(S_0, S_g)$

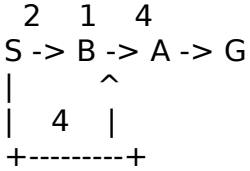
So if h is consistent, it is admissible (never overestimates the cost to reach the goal).

3.25 The **heuristic path algorithm** (Pohl, 1977) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this complete? For what values is it optimal, assuming that h is admissible? What kind of search does this perform for $w = 0$, $w = 1$, and $w = 2$?

It is complete whenever $0 \leq w < 2$. $w = 0$ gives $f(n) = 2g(n)$. This behaves exactly like uniform-cost search—the factor of two makes no difference in the ordering of the nodes.

$w = 1$ gives A $*$ search.
$w = 2$ gives $f(n) = 2h(n)$, i.e., greedy best-first search.
We also have $f(n) = (2 - w)[g(n) + w/(2 - w)*h(n)]$ which behaves exactly like A $*$ search with a heuristic $w/(2-w)*h(n)$.
For $w \leq 1$, this is always less than $h(n)$ and hence admissible, provided $h(n)$ is itself admissible.

3.24 Devise a state space in which A* using GRAPH-SEARCH returns a suboptimal solution with an $h(n)$ function that is admissible but inconsistent.



h(S)=7, h(B)=5, h(A)=1, h(G)=0

Shortest path: S->B->A->G (Path cost = 7)
A* Tracing:
Q=[S (f=7+0)] => Explore S to get A & B and add to closed set
Q=[A (f=1+4), B (f=5+2)] => Explore A to get G and add to closed set
Q=[B (f=5+2), G (f=0+8)] => Explore B to get A and add to closed set.
Since A is already in closed set, it will not be added to the queue.
Q=[G (f=0+8)] => Dequeue G and traceback the path to S => Path = S->A->G

Result: S->A->G (Path cost = 8) Suboptimal

3.21 Prove each of the following statements, or give a counterexample:

a. Breadth-first search is a special case of uniform-cost search.

b. Depth-first search is a special case of best-first tree search.

c. Uniform-cost search is a special case of A* search.

a. When all step costs are equal, $g(n) \propto \text{depth}(n)$, so uniform-cost search reproduces breadth-first search.

b. Depth-first search is best-first search with $h(n) = -\text{depth}(n)$ or $1/\text{depth}(n)$. (Extra: Similarly, Breadth-first search is best-first search with $h(n) = \text{depth}(n)$).

c. Uniform-cost search is A $*$ search with $h(n) = 0$.