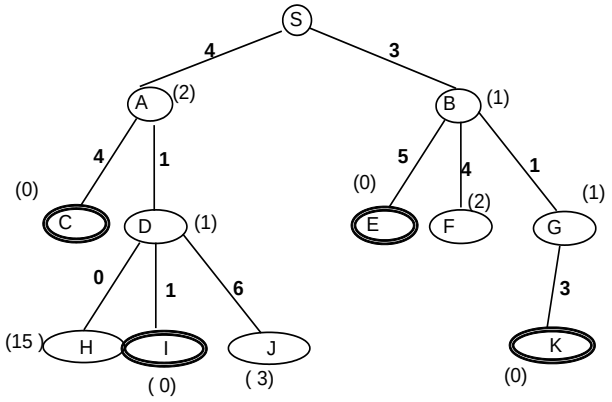
COMP3308/COMP3608 Introduction to Artificial Intelligence

Week 3 Tutorial exercises
A* algorithm. Heuristics. Local search algorithms.

Exercise 1 (Homework). **A* search**

Consider the tree below. Step costs are shown along the edges and heuristic values *h* are shown in brackets. The goal nodes are circled twice, i.e. they are C, I, E and K.



Run the A* search algorithm. Show the list of expanded nodes and the solution path found with its cost. If there are nodes with the same priority for expansion, expand the last added first.

Expanded nodes:
Path found:
Path cost:

Solution:

Expanded nodes: SBGADI
Path found: SADI
Path cost: 6

Explanation:

A* selects nodes for expansion based on the *f*-cost, which is the sum of the cost so far (*g*) and estimated cost to reach a goal node (*h*), i.e. *f(n)= g(n) + h(n)*

We use the following notation: {expanded} | {fringe}. In **bold** we will show the node selected for expansion, i.e. the one that is at the front of the fringe.

empty | S
S | A[6], **B[4]**
S, B | A[6], E[8], F[9], **G[5]**
S, B, G | **A[6],** E[8], F[9], K[7]
S, B, G, A | E[8], F[9], K[7], C[8], **D[6]**
S, B, G, A, D | E[8], F[9], K[7], C[8], H[20], **I[6],** J[14]
S, B, G, A, D, I | [8], F[9], K[7], C[8], H[20], J[14]

Check the figure to verify that SADI is the shortest path to a goal node!

Exercise 2. **Admissible heuristics – 4-queens puzzle**

You are given a 4x4 chess board and the goal is to place 4 queens on the board so that no queen can capture any other. That is, only one queen can be in any row, column or diagonal of the 4x4 array. Suppose we try to solve the puzzle using the following problem space: the start node corresponds to an empty 4x4 array; the successor function creates new 4x4 arrays containing one additional legal placement of a queen anywhere in the array; the goal test is satisfied if and only if there are 4 queens in the array that are legally positioned. Note that all goal nodes are precisely 4 steps from the start node as at each step we place 1 queen and the total number of queens is 4.

Explain whether each of the following is an admissible heuristic:

a) *h(n) = depth (n),* i.e. where *depth(n)* is the depth of the node *n* in the search tree

b) *h(n) = 4 -queens(n),* where *queens(n)* is the number of queens on the board in state *n* (i.e. the number of queens already placed)

c) *h(n) = 4 -queens(n) -available(n)/16,* where *available(n)* is the number of available squares in the array that can receive a new queen in state *n*

d) Can you solve the 4-queens puzzle?

Answer:

a) No, as it increases when it should decrease, leading to an overestimate. For example, at depth 4, on a goal node, it should be 0 instead of 4.

b) Yes, the heuristic is admissible.

- For nodes that are on a goal path, *h(n)* gives the exact number of steps to reach a goal state as the goal nodes are precisely 4 steps from the start node, i.e. *h(n)= h*(n) = 4 - queens(n)*.
- Note that some nodes will not lead to a goal given the definition of the problem. For such nodes, i.e. nodes that are not on a goal path, a goal state cannot be reached and *h*(n)* is infinite and *h(n)< h*(n).*

c) Yes, because it subtracts a positive number from an already admissible heuristic, i.e. decreases the heuristic value even more.

d) Here is a possible solution:

		X	
X			
			X
	X		

Exercise 3. **Admissibility and consistency**

a) Does an admissible heuristic *h* lead to non-decreasing *f* values along any path? In other words, are all admissible heuristics consistent? Give an example to illustrate your answer.

b) Does a non-decreasing *f* value along any path imply admissibility of *h*?

c) A* uses admissible heuristics. What happens if we use a non-admissible one? Is it still useful to use A* with a non-admissible heuristic?

Answer:

a) No, see the example below.

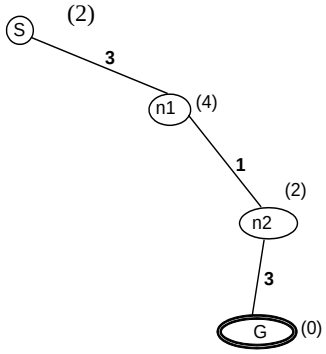
h is admissible as:

h(S) = 2 <= 3+1+3=7

h(n1) = 4 <= 1+3=4

h(n2) = 2<= 3

Let's calculate the *f*-values: *f(n1) = 4+3=7, f(n2*) = 3+1+2=6, *f(n3*) = 3+1+3+0 = 7. Hence, *f* is not non-decreasing as *f(n2)<f(n1),* i.e. it decreases at *n2.*



=> admissibility (h) ≠> decreasing *f*-values (i.e. consistent heuristic)

Note: non-decreasing *f* <=> triangle equation for *h*

b) Yes, non-decreasing *f*-values along any path imply admissibility of *h*. What did we prove at the lecture about consistency and non-decreasing *f*-values?

We proved that if the heuristic is consistent, than the f-values are non-decreasing. The opposite is also true.

c) Not optimal anymore. But it could still find a good answer in a reasonable time, depending on how good the heuristic is.

Exercise 4*. A

A * does not terminate until a goal node is selected for expansion from the fringe. However, a path to a goal node might be reached long before the node is selected for expansion, i.e. there may be already a goal node in the fringe. Why not terminate as soon as a goal node has been added to the fringe? Illustrate your answer with an example.

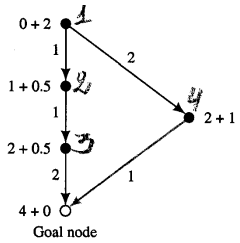
Answer:

When a goal node is placed in the fringe it means that A* has found a path to a solution but this may not be the path to the optimal solution (i.e. the shortest path to a goal).

Example 1. In your homework for this week (Ex.1), the goal nodes E, C and K are added to the fringe before the optimal goal node I, and they are not selected for expansion.

Example 2. In the Romania example, Bucharest (goal node) is added to the fringe after expanding Fagaras but it is not selected for expansion. Instead Riminicu Vilcea is expanded, then Pitesti which adds Bucharest again in the fringe (with different f-cost), and this last Bucharest is selected for expansion, and it is the optimal goal node.

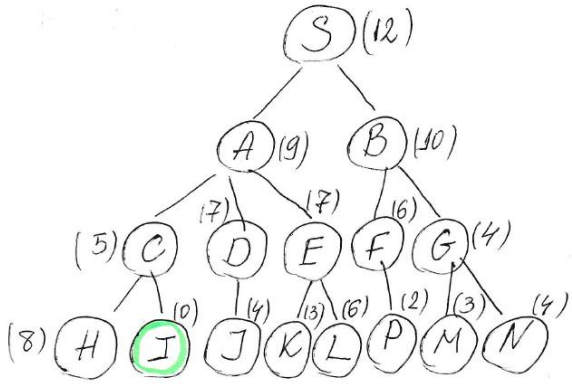
Example 3. See the figure below for another example. At each node the *f* value is indicated *(g + h)*.



1 | 2[1.5], 4[3]
1, 2 | 4[3], 3[2.5]
1, 2, 3 | 4[3], Goal[4] //goal node in the fringe but it will not be selected
1, 2, 3, 4 | Goal[4], Goal[3]
1, 2, 3, 4, Goal | Goal[4]
=>The path found is 1,4,Goal, not 1,2,3,Goal

The condition of stopping when a goal node is selected for expansion as opposed to stopping when a goal node is added to the fringe guarantees that we will find the shortest path to a goal (i.e. the optimal solution), provided that *h* is admissible.

***Exercise 5*. Hill-climbing and beam search**
Consider the tree below. The *v*-value for each node is shown in brackets (this is the value of the heuristic evaluation function). The goal node is I, shown in green.



a) Run the **hill-climbing** algorithm; the smaller the value, the better (i.e. hill-climbing descent).

Show the list of expanded nodes. Was the goal node found?

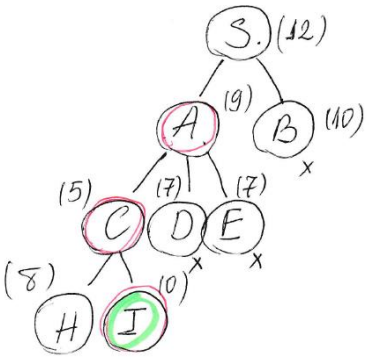
b) Now run the **beam search** algorithm with k=2. Again - the smaller the value, the better.

Show the list of expanded nodes. Was the goal node found?

Solution:
a) Hill climbing

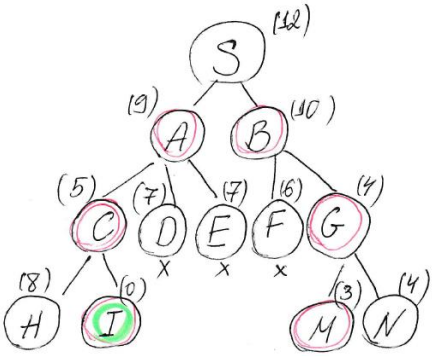
List of expanded nodes: SACI
Yes, the goal node I was found

The figure below shows how the search is conducted. Each time we select the best child (shown in red) and ignore the other children. The best child also has to be better than the parent (i.e. with smaller *v*-value) for the search to continue, otherwise the search stops.



A change in the graph to see a case of early stopping (local minimum) - let's change the *v*-value of C, D and E to 10, 11 and 12, respectively. The search will stop before reaching I. After expanding A, the best child is selected – C, but it is not better than the parent (10 is not smaller than 9), so the algorithm stops and returns A as a solution (local minimum).

b) Beam search
nodes in red = the *k* best nodes selected at each level (*k*=2)



List of expanded nodes: SABCGI
Yes, the solution node I was found

Note: The beam search algorithm that we studied (as in the textbook) does not check if the child is better than a parent. It simply selects the best *k* nodes at each level, no matter if they are better than the parent. There are different variations, so it is possible to add this check too.

***Exercise 6*. Genetic algorithms** (adapted from Dunham, Data Mining, 2003, Pearson Education))
Given is the following initial population: {101010, 001100, 010101, 000010}. Apply the following genetic algorithm to find a better population:

Fitness function: It is defined as the sum of the bit values of each individual.
Selection: At each iteration select for crossover: 1) the best individual and the second best individual and 2) the best individual and the third best individual. If there are ties, resolve them randomly.
Crossover: Always choose the same crossover point, between the 3rd and 4th bit.

Mutation: Mutation means negating a bit. Always mutate bit 2 (the bit numbering starts from 1, from the left).
Stopping condition: The average fitness for the entire population is greater than 4.

Show the new population. How many iterations were needed?

Solution:

Initial population:
ind.1 101010 f=3
ind.2 001100 f=2
ind.3 010101 f=3
ind.4 000010 f=1

iteration 1:
ind.1 101|010 crossover-> 101101 mutation -> 111101 f=5
ind.3 010|101 crossover-> 010010 mutation -> 000010 f=1

ind.1 101|010 crossover-> 101100 mutation ->111100 f=4
ind.2 001|100 crossover->001010 mutation ->011010 f=3

average fitness = (5+1+4+3)/4=13/4 < 4, stopping condition is not satisfied

iteration 2:
ind.1 111101 f=5
ind.2 000010 f=1
ind.3 111100 f=4
ind.4 011010 f=3

ind.1 111|101 crossover->111100 mutation -> 101100 f=3
ind.3 111|100 crossover->111101 mutation -> 101101 f=4

ind.1 111|101 crossover->111010 mutation -> 101010 f=3
ind.4 011|010 crossover->011101 mutation -> 001101 f=3

average fitness = (3+4+3+3)/4=13/4 < 4, stopping condition is not satisfied

iteration 3:
ind.1 101100 f=3
ind.2 101101 f=4
ind.3 101010 f=3
ind.4 001101 f=3

ind.2 101|101 crossover->101100 mutation -> 111100 f=4
ind.1 101|100 crossover->101101 mutation -> 111101 f=5

ind.2 101|101 crossover->101010 mutation -> 111010 f=4
ind.3 101|010 crossover->101101 mutation ->111101 f=5

average fitness = (4+5+4+5)/4=18/4 > 4, stopping condition is satisfied
3 iterations were needed

Exercise 7 (Advanced students only) **Gaschnig's heuristic**

At the lectures we defined a relaxation of the 8-puzzle problem in which a tile can move from square A to square B if B is blank. The exact solution of this problem defines the Gaschnig's heuristic. One way to compute this heuristic is the following:

Let B be the location of blank in the current state.
Compare the current state with the goal state:
If B in the current state is occupied by a tile X in the goal state, move X to B in the current state (i.e. swap X and B);
Otherwise, move any misplaced tile to B.

The value of the heuristic for a given state is the number of switches required to reach the goal state.

a) Apply the Gaschnig heuristic to solve the 8-puzzle from this initial state:

start	goal
2B6	123
134	456
758	78B

b) Explain why the Gaschnig's heuristic dominates the heuristic *h1* from the lectures (*h1* - the number of misplaced tiles).
c) Recall the heuristic *h2* from the lectures - the Manhattan distance. Show cases when the Gaschnig heuristic is more accurate than both *h1* and *h2*, i.e. returns a higher value.

Answer:

a)

start:	s1	s2	s3	s4	s5	s6	s7
2B6	B26	126	126	12B	123	123	123
134	134	B34	43B	436	4B6	456	456
758	758	758	758	758	758	7B8	78B

In yellow are the changes between 2 successive states. The number of switches needed is 7.

a) The Gaschnig heuristic is exact for the relaxed version of the problem that a tile can move from A to B is B is blank. The misplaced tiles heuristic *h1* is exact for the relaxed version of the problem that a tile can move from A to B. Hence, the Gaschnig heuristic is more restrictive than *h1* and cannot be less than *h1*.

c) Consider the goal state and permute (swap) 2 adjacent tiles, e.g. tiles 1 and 2:
213
456
78B

Both heuristics *h1* and *h2* return 2 but the Gaschnig's heuristic returns 3:
h1=2 (2 misplaced tiles: 1 and 2)
h2=2 (Manhatten distance, values of 1 for tiles 1 and 2 and 0 for the remaining tiles)
Gaschnig's heuristic =3 (3 swaps are needed to get to the goal state):

B13	1B3	123
456	456	456
782	782	78B

Gaschnig dominates *h1*, *h2* dominates *h1* but we don't know the relationship between Gaschnig and *h2*.