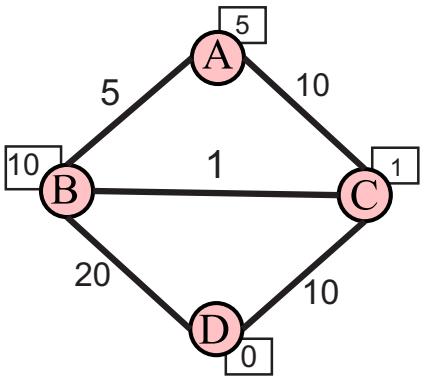
Grundlagen der Künstlichen Intelligenz
Exercise 2b: Informed Search – Solutions 8th and 14th November 2019
Anna-Katharina Rettinger (last updated 15th November 2019)

1 Presented Problems

Problem 2.5: Heuristics for Informed Search

Consider the following graph. We start from node A and our goal node is D. The path costs are shown on the arcs and the heuristic values are shown at each node.



Problem 2.5.1: What are the requirements on a heuristic for A* tree search and graph search to be optimal?

- *A* tree search requires the heuristic to be admissible.*
- *A* graph search requires the heuristic to be consistent.*

For its definitions, we use $h(n)$ as the heuristic value at node n and $g(n, n')$ as the path cost from node n to node n'.

An admissible heuristic always under-approximates the actual cost:

$$h(n) \leqslant g(n, goal) \tag{1}$$

A consistent heuristic fulfills this triangle inequality for all nodes:

$$h(n) \leqslant g(n, n') + h(n') \tag{2}$$

Problem 2.5.2: Is the given heuristic admissible? Is it consistent? If not, why not?

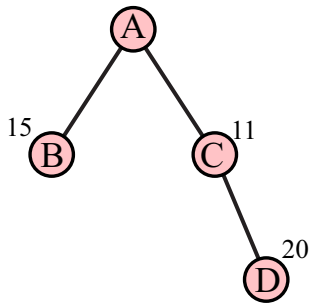
Admissible? : Yes
Consistent? : No, since for nodes B and C the triangle inequality does not hold:

$$h(B) \nleqslant c(B, C) + h(C) \tag{3}$$

Problem 2.5.3: Perform A* graph search and tree search to verify your previous answers

Recall that for A, $f(n) = g(n) + h(n)$.*

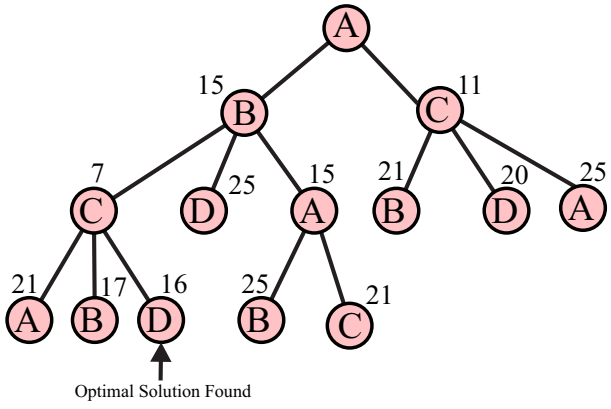
- *A* graph search:*



node with $f(n)$	A(5)	C(11)	B(15)	D(20)
frontier with $f(n)$	C(11) B(15)	B(15) D(20)	D(20)	
explored	A	AC	ACB	ACB

A graph search did not find the optimal solution. Note that when exploring C in step 2, we do not add its child B to the frontier, since its $f(n) = 21$ is higher than the value of f of the B already on the frontier.*

- *A* tree search:*



node with $f(n)$	A(5)	C(11)	B(15)	C(7)	A(15)	D(16)
frontier with $f(n)$	C(11) B(15)	B(15) D(20) B(21) A(25)	C(7) A(15) D(20) B(21) A(25) D(25)	A(15) D(16) B(17) D(20) A(21) B(21) A(25) D(25)	D(16) B(17) D(20) A(21) B(21) C(21) A(25) B(25) D(25)	B(17) D(20) A(21) B(21) C(21) A(25) B(25) D(25)

A tree search found the optimal solution.*

Problem 2.6: Application of Search Algorithms: Train Journey

We want to travel from Aberdeen to Birmingham. The (semi-fictional) map of the British rail system is available in Fig. 1. Routes have the same cost in both directions for both time and price. If there is no clear preference for the next node to explore, we explore first the node which is first alphabetically.

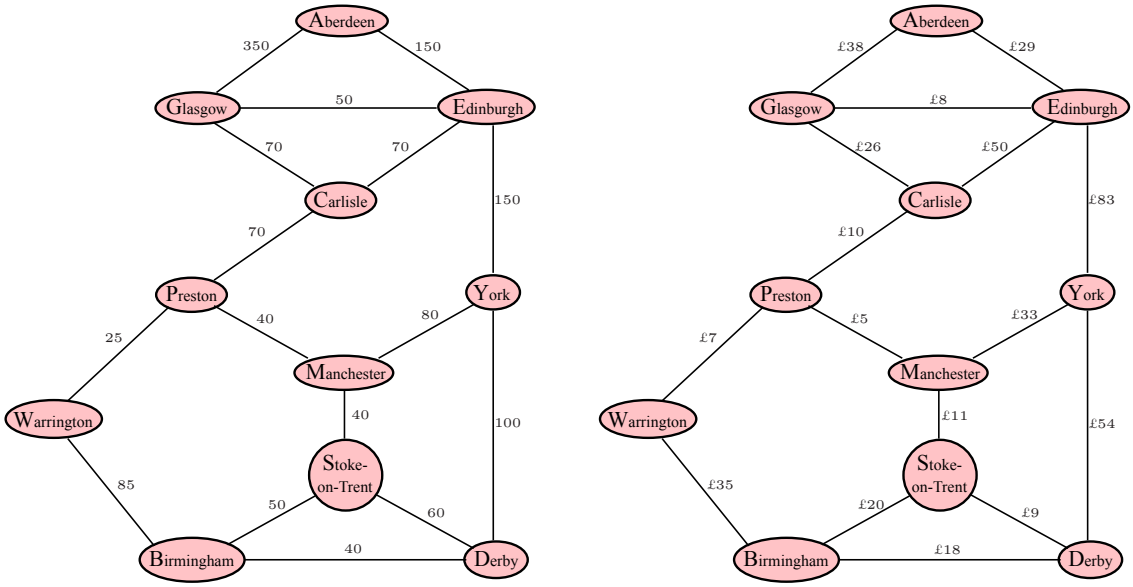


Figure 1: Rail map: left, cost in time (minutes); right, cost in price (£ sterling)

Problem 2.6.1: In the table below, we are given a heuristic for cost in time, which is based on the straight-line distances to Birmingham and the maximum speed of the train being no more than 120km/h. Perform Greedy Best-First and A* Search for time cost using tree search. For each step, write down the node which is currently expanded and the nodes in the frontier set, each with its value of the evaluation function $f(n)$.

node n	A	G	E	C	Y	P	M	W	S	D	B
heuristic function $h(n)$	259	203	197	138	86	76	56	55	31	28	0

Greedy Best-First tree search (recall that $f(n) = h(n)$): The solution is A-E-Y-D-B with cost in time of 440 minutes and in price of £184.

node with $f(n)$	A(259)	E(197)	Y(86)	D(28)	B(0)
frontier with $f(n)$	E(197), G(203)	Y(86), C(138), G(203), G(203), A(259)	D(28), M(56), C(138), E(197), G(203), G(203), A(259)	B(0), S(31), M(56), Y(86), C(138), E(197), G(203), G(203), A(259)	S(31), M(56), Y(86), C(138), E(197), G(203), G(203), A(259)

Note that node G(203) *is added to the frontier in step 2 again, since in tree search (according to slide 17 of lecture 3) we add nodes to the frontier without checking whether they are already in the frontier.*

A tree search (recall that $f(n) = g(n) + h(n)$): A* finds the optimal solution, which is A-E-C-P-W-B with cost in time of 400 minutes and in price of £131.*

node with $f(n)$	A(259 = 0 + 259)	E(347 = 150 + 197)	C(358 = 220 + 138)
frontier with $f(n)$	E(347 = 150 + 197) G(553 = 350 + 203)	C(358 = 220 + 138) Y(386 = 300 + 86) G(403 = 200 + 203) G(553 = 350 + 203) A(559 = 300 + 259)	P(366 = 290 + 76) Y(386 = 300 + 86) G(403 = 200 + 203) E(487 = 290 + 197) G(493 = 290 + 203) G(553 = 350 + 203) A(559 = 300 + 259)

P(366 = 290 + 76)	W(370 = 315 + 55)	M(386 = 330 + 56)
W(370 = 315 + 55) M(386 = 330 + 56) Y(386 = 300 + 86) G(403 = 200 + 203) E(487 = 290 + 197) G(493 = 290 + 203) C(498 = 360 + 138) G(553 = 350 + 203) A(559 = 300 + 259)	M(386 = 330 + 56) Y(386 = 300 + 86) B(400 = 400 + 0) G(403 = 200 + 203) P(416 = 340 + 76) E(487 = 290 + 197) G(493 = 290 + 203) C(498 = 360 + 138) G(553 = 350 + 203) A(559 = 300 + 259)	Y(386 = 300 + 86) B(400 = 400 + 0) S(401 = 370 + 31) G(403 = 200 + 203) P(416 = 340 + 76) P(446 = 370 + 76) E(487 = 290 + 197) G(493 = 290 + 203) Y(496 = 410 + 86) C(498 = 360 + 138) G(553 = 350 + 203) A(559 = 300 + 259)

Y(386 = 300 + 86)	B(400 = 400 + 0)
B(400 = 400 + 0) S(401 = 370 + 31) G(403 = 200 + 203) P(416 = 340 + 76) D(428 = 400 + 28) M(436 = 380 + 56) P(446 = 370 + 76) E(487 = 290 + 197) G(493 = 290 + 203) Y(496 = 410 + 86) C(498 = 360 + 138) G(553 = 350 + 203) A(559 = 300 + 259) E(647 = 450 + 197)	S(401 = 370 + 31) G(403 = 200 + 203) P(416 = 340 + 76) D(428 = 400 + 28) M(436 = 380 + 56) P(446 = 370 + 76) E(487 = 290 + 197) G(493 = 290 + 203) Y(496 = 410 + 86) C(498 = 360 + 138) G(553 = 350 + 203) A(559 = 300 + 259) E(647 = 450 + 197)

Problem 2.6.2: For the problem with cost in price (see Fig. 1 right), would a heuristic based on distance be valid?

Since the price cost is not necessarily correlated with distance, such a heuristic might not be admissible. For example, Carlisle is further from Birmingham than York, but costs less to travel from, whereas Aberdeen is further than York and costs more to travel from. An effective heuristic could be obtained by analyzing rail companies' pricing strategies.

Problem 2.6.3: Which search algorithm would we use if we want to minimize train changes (assuming that we change train at every station)?

To minimize train changes, we have to find the shallowest solution. Breadth-First search finds this solution.

Problem 2.6.4: Is bidirectional search a good option for the train journey search?

Yes, as all actions are reversible and there is only one goal, we can use bidirectional search. Since the cost in time and price are the same in both directions, the backwards search can be performed just like the forward search.

2 Additional Problems

Problem 2.7: Graph Search

Consider the graph of Problem 2.1 (of Exercise 2a). With the heuristic given in the table below, perform Greedy Best-First and A* Search, each with tree search and with graph search. For each step, write down the node which is currently expanded and the nodes in the frontier set, each with its value of the evaluation function $f(n)$ (and the nodes in the explored set in case of graph search).

node n	A	B	C	D	E	F
heuristic function $h(n)$	4	5	2	1	7	0

Greedy Best-First tree search:

node with $f(n)$	A(4)	D(1)	F(0)
frontier with $f(n)$	D(1), C(2), B(5)	F(0), C(2), A(4), B(5)	C(2), A(4), B(5)

A tree search:*

node with $f(n)$	A(4 = 0 + 4)	C(4 = 2 + 2)	D(5 = 4 + 1)	F(5 = 5 + 0)
frontier with $f(n)$	C(4 = 2 + 2), D(5 = 4 + 1), B(6 = 1 + 5)	D(5 = 4 + 1), B(6 = 1 + 5), A(8 = 4 + 4)	F(5 = 5 + 0), B(6 = 1 + 5), A(8 = 4 + 4), A(12 = 8 + 4)	B(6 = 1 + 5), A(8 = 4 + 4), A(12 = 8 + 4)

Greedy First-Best graph search:

node with $f(n)$	A(4)	D(1)	F(0)
frontier with $f(n)$	D(1), C(2), B(5)	F(0), C(2), B(5)	C(2), B(5)
explored	A	AD	AD

A graph search:*

node with $f(n)$	A(4 = 0 + 4)	C(4 = 2 + 2)	D(5 = 4 + 1)	F(5 = 5 + 0)
frontier with $f(n)$	C(4 = 2 + 2), D(5 = 4 + 1), B(6 = 1 + 5)	D(5 = 4 + 1), B(6 = 1 + 5)	F(5 = 5 + 0), B(6 = 1 + 5)	B(6 = 1 + 5)
explored	A	AC	ACD	ACD

Problem 2.8: General Questions on Search

Problem 2.8.1: Can Uniform-Cost search be more time-effective or memory-effective than the informed search algorithms mentioned in the lectures? What about cost-effectiveness?

Greedy Best-First is not optimal, so may be less cost-effective than Uniform-Cost, and may be less time-effective and memory-effective, since the heuristic may lead the algorithm onto paths that appear to be dead ends. A is optimal, so will give the same (optimal) cost as Uniform-Cost. Similarly, with a bad heuristic (e.g. $h = 0$ for all nodes), A* can perform as badly in memory or time as Uniform-Cost, but not worse (since for $h \geq 0$, any nodes explored in A* will be explored in Uniform Cost, but not vice-versa.)*

Problem 2.9: Application of Search Algorithms in Daily Life

Which search algorithm (out of those covered in lectures 3 and 4) do you think your brain uses in the following cases, and what are the advantages of this algorithm? (Note that there are no hard-and-fast solutions for these; it is a matter of personal preference. The purpose of this exercise is to think about situations where you use search algorithms in daily life.)

a. Planning a route from Arad to Bucharest with a map.

Since you can see the whole map, you are informed and have some sort of heuristic (you would tend to prefer the roads leading towards Bucharest, for example). Therefore, you might first use Greedy First-Best, as you do not know the exact path cost; perhaps also using Bidirectional Search. To find the optimal solution, you might use A or Uniform-Cost afterwards.*

b. Finding an Easter egg:

i. in a building you are unfamiliar with, on your own.

ii. in a building you are familiar with, on your own.

iii. in a building you are familiar with, with a team of people who all have mobile phones.

iv. if the Easter egg has a bell attached (which constantly rings).

On your own, only nodes close together can be explored (you cannot jump halfway across the building to explore another node). Therefore, some sort of Depth-First search would be sensible.
If you do not know the building, you may opt for Depth-Limited Search (as you do not want to get lost) or Iterative Deepening. If you do know the building, you may opt for Depth-First Search, Depth-Limited Search, or Iterative Deepening.

If you have a team of people, nodes all along the frontier can be explored. You could opt for Breadth-First Search.

With the attached bell, you have a heuristic - how loud or soft the sound is. On your own, perhaps you would prefer Greedy First-Best, as it limits the amount of backtracking you need to do. If you have a team of people, you could perform A Search.*

c. Playing a strategy game, e.g. chess.

This is very individual! It is left to you to decide what you do. Personally, I use some kind of Depth-Limited Search.