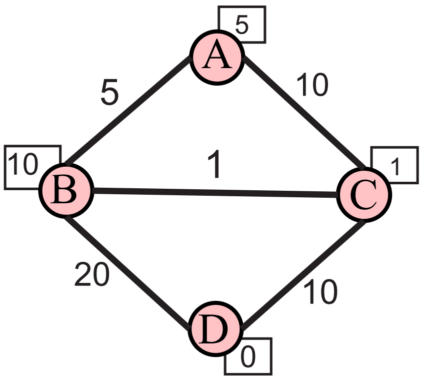
FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE
Exercise 2b: Informed Search – Problems   Winter Semester 2023/24
Sebastian Mair

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

Problem 2.5.2: Perform A* search. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.
Hint: You can write $X(C, P, F)$ for a node n, where $X = \texttt{n.STATE}, C = \texttt{n.PATH-COST}, P = \texttt{n.PARENT.STATE}$, and $F = f(n)$. F can be omitted if it is not needed.

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 which node to expand next from the frontier, we choose the node first which is first alphabetically.

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. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.
Hint: You can again write $X(C, P, F)$ for a node n, where $X = \texttt{n.STATE}, C = \texttt{n.PATH-COST}, P = \texttt{n.PARENT.STATE}$, and $F = f(n)$. F can be omitted if is not needed.

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

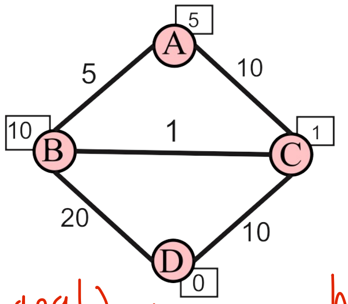
Problem 2.6.2: For the problem with cost in price (see Fig. 1 right), would a heuristic based on distance be valid?

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)?

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

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.



$10 \le 1 + 1$ X

$h(n) \le g(n, goal)$ ✓ $\qquad h(n) \le c(n,a,n') + h(n')$

Problem 2.5.1: Is the given heuristic admissible? Is it consistent? If not, why not? X

Problem 2.5.2: Perform A* search. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.
Hint: You can write $X(C, P, F)$ for a node n, where X = `n.STATE`, C = `n.PATH-COST`, P = `n.PARENT.STATE`, and $F = f(n)$. F can be omitted if it is not needed.

A^* $\quad f(n) = h(n) + g(n)$

node	frontiner	reached set
A(0,-,5)	∅	A(0,-,5)
A(0,-,5)	B(5,A,15) C(~~10,A,11~~)	B(5,A,15) C(~~10,A,1~~)
C(10,A,11)	B(5,~~A,15~~) D(20,C,20)	D(~~20,C,20~~) 覆写
B(5,A,15)	D(20,C,20) C(6,~~B~~,7)	C(6,B,7)
C(6,B,7)	~~X~~ → D(16,~~C~~,16)	D(16,C,16)
D(16,C,16)	⇐ Find 覆写	D ⇐ C ⇐ B ⇐ A

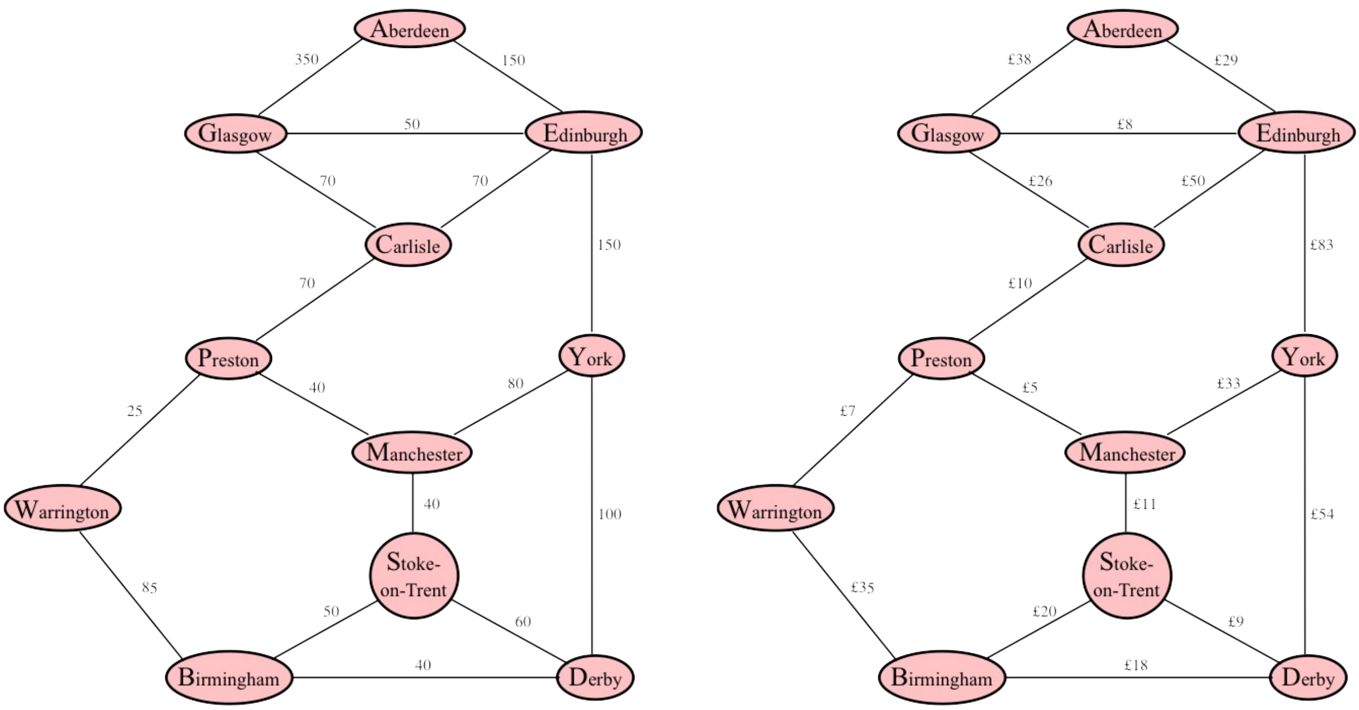


Figure 1: Rail map: left, cost in time (minutes); right, cost in price (£ sterling)

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 which node to expand next from the frontier, we choose the node first which is first alphabetically.

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. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.
Hint: You can again write $X(C, P, F)$ for a node n, where $X =$ `n.STATE`, $C =$ `n.PATH-COST`, $P =$ `n.PARENT.STATE`, and $F = f(n)$. F can be omitted if is not needed.

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

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

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)?

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

2.6.1

GBS $f(n) = h(n)$

node	Frontiren	Reached set
A(0,-,259)	φ	A(0,-,259)
A(0,-,259)	E(~~150, A~~,197) G(350, A, 203)	E(150, A, 197) G(~~350, A~~, 203) G(200, E, 203)
E(150, A, 197)	G(200, E, 203) C(220, E, 138) Y(~~300, E~~, 86)	C(220, E, 138) Y(300, E, 86)
Y(300, E, 86)	G(200, E, 203) C(220, E, 138) M(380, Y, 56) D(400, ~~Y~~, 28)	M(380, Y, 56) D(400, Y, 28)
D(400, Y, 28)	G(200, E, 203) C(220, E, 138) M(380, Y, 56) S(460, D, 31) B(440, D, 0)	S(460, D, 31) B(440, D, 0)
B(440, D, 0)	⇐ Find.	

B ⇐ D ⇐ Y, E ⇐ A

2.6.1

A^* $f(n) = h(n) + g(n)$

node	Frontiren	Reached set
A(0, –, 259)	∅	A(0, -, 259)
A(0, -, 259)	E(150, ~~A~~, 347) G(350, A, 553)	E(150, A, 347)
E(150, A, 347)	G(200, E, 403) C(220, ~~E~~, 358) Y(300, E, 386)	~~G(350, A, 553)~~ G(200, E, 403) C(220, E, 358) Y(300, E, 386)
C(220, E, 358)	G(200, E, 403) Y(300, E, 386) P(~~290, C~~, 386)	P(290, C, 386)
P(290, C, 386)	G(200, E, 403) Y(300, E, 386) M(330, P, 406) W(~~315~~, P, 370)	M(330, P, 406) W(315, P, 370)
W(315, P, 370)	G(200, E, 403) Y(300, ~~E~~, 386) M(330, P, 406) B(400, W, 400)	B(400, W, 400) D(400, Y, 428)
Y(300, E, 386)	G(200, E, 403) D(400, Y, 428) M(330, P, 406) B(~~400, W~~, 400)	
B(400, W, 400)		

B ⇐ W ⇐ P ⇐ C ⇐ E ⇐ A

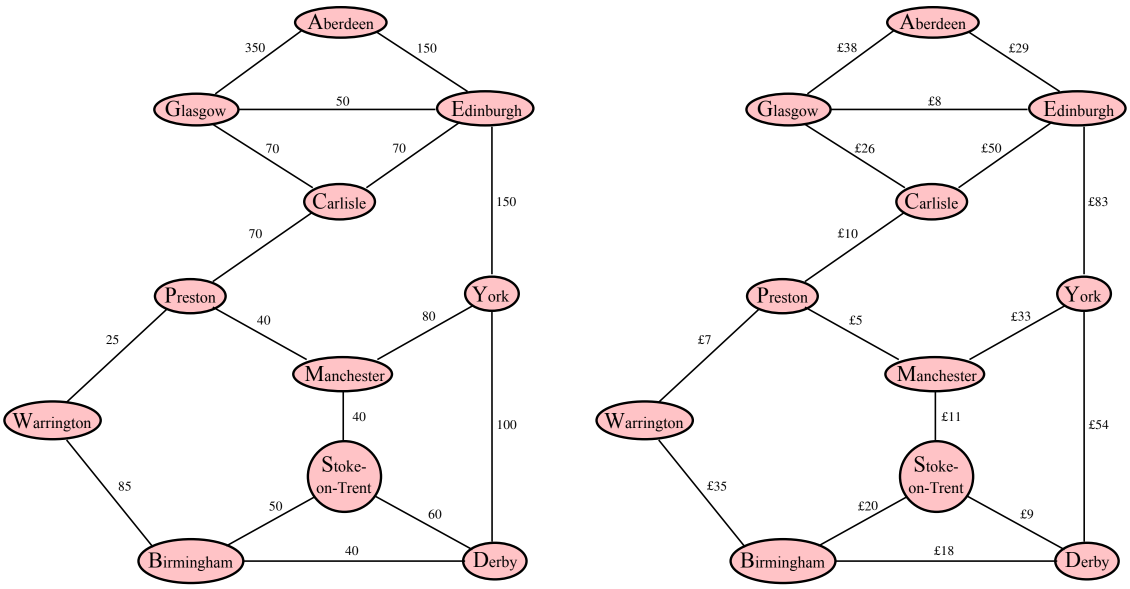


Figure 1: Rail map: left, cost in time (minutes); right, cost in price (£ sterling)

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. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.

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

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? What about cost-effectiveness?

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.)

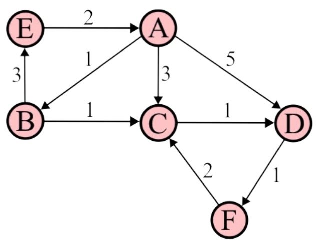
a. Planning a route from Arad to Bucharest with a map.

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 has a bell attached (which constantly rings).

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

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. For each search step, write down the node that is currently expanded, the frontier, and the lookup table *reached*.

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

GBF $f(n) = h(n)$

node	Frontier	Reached set
A(0, -, 4)	∅	A(0, -, 4)
A(0, -, 4)	B(1, A, 2) C(3, ~~A~~, 1) D(5, A, 1)	B(1, A, 2) C(3, A, 1) D(5, ~~A~~, 1)
C(3, A, 1)	D(4, C, 1) B(1, A, 2)	D(4, C, 1)
D(4, C, 1)	B(1, A, 2) F(5, ~~D~~, 0)	F(5, D, 0)
F(5, D, 0)	F $\Leftarrow$ D $\Leftarrow$ C $\Leftarrow$ A	

A* $f(n) = h(n) + g(n)$

node	Frontier	Reached set
A(0, -, 4)	∅	A(0, -, 4)
A(0, -, 4)	B(1, ~~A~~, 3) C(3, A, 4) D(5, A, 6)	B(1, A, 3) C(3, ~~A~~, 4) D(5, ~~A~~, 6)
B(1, A, 3)	C(2, ~~B~~, 3) D(5, A, 6) E(4, B, 9)	C(2, B, 3)
C(2, B, 3)	D(3, C, 4) E(4, B, 9)	D(3, C, 4)
D(3, C, 4)	F(4, D, 4) E(4, B, 9)	F(4, D, 4)
F(4, D, 4)	F $\Leftarrow$ D $\Leftarrow$ C $\Leftarrow$ B $\Leftarrow$ A	