EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATH. - NATURWISS. FAKULTÄT
FACHBEREICH INFORMATIK
KOGNITIVE SYSTEME · PROF. A. ZELL

Artificial Intelligence

Assignment 4

Assignment due by: 23.11.2016, Discussion: 29.11.2016

Question 1 Recursive Best-First Search (RBFS) (6 points)

For this question please draw the trees for recursive best-first search. Please represent each node as a circle with its name inside, write the f_limit in a rectangle above the corresponding node, and express $f(n) = g(n) + h(n)$ under each node. The solution path should be highlighted.

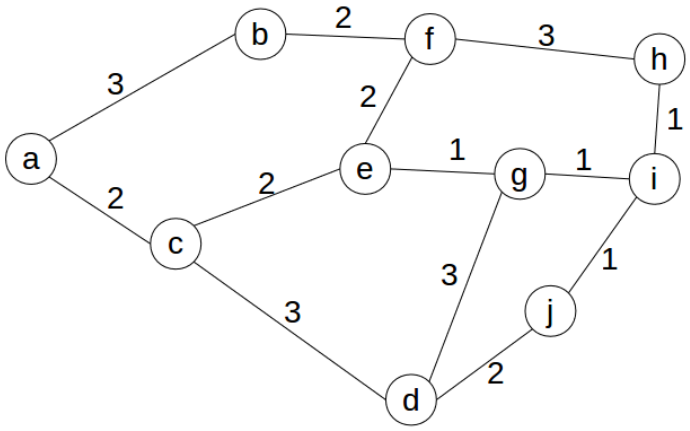


Table 1.

n	$h(n)$	n	$h(n)$
a	10	f	3
b	5	g	1
c	5	h	1
d	4	i	0
e	5	j	1

Table 2.

n	$h(n)$	n	$h(n)$
a	6	f	4
b	5	g	3
c	3	h	5
d	0	i	4
e	3	j	2

Table 3.

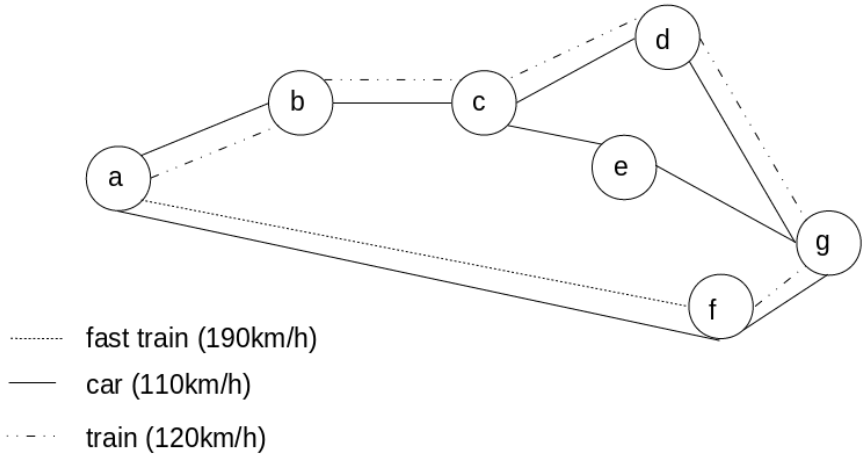
n	$h(n)$	n	$h(n)$
a	0	f	2
b	3	g	7
c	4	h	9
d	6	i	12
e	4	j	10

(a) Using the heuristic function shown in Table 1, find a path between the node $\mathbf{a}$ and the target $\mathbf{i}$.

(b) Using the heuristic function shown in Table 2, find a path between the node $\mathbf{b}$ and the target $\mathbf{d}$.

(c) Using the heuristic function shown in Table 3, find a path between the node $\mathbf{i}$ and the target $\mathbf{a}$.

Question 2 Pathfinding (6 points)



We assume each node of the graph above represents a city and we further assume cities can be reached using car, train and/or fast train as indicated on the graph. Using the following inter-city distances $ab = bc = cd = ce = eg = fg = 200$ km, $dg = 300$ km and $af = 1100$ km and using the A* algorithm, please find and specify the fastest way to go from a to g. The mean of transportation from a to g should not change (i.e. either (i) the car from a to g or (ii) the train/fast train from a to g). Detail intermediate steps and indicate the f-values of the different nodes. For this question, you are required to use the heuristic $h(n) = 0$.

Question 3 Programming in LISP (4+2+2 = 8 points)

When doing these exercises you are **not** allowed to use general looping constructs or built in functions that solve the exercise by themselves. You can use the specific `(loop for x in list collect (expression with x))` construct, since this is basically just list comprehension.

Similar to last week, download the file *graphsearch-rbfs.lisp*, which contains a graph of German cities and the distances between them, as well as the coordinates of each city and some functions to access that information: `(expand city)` returns a list of all the cities connected to `city`, `(get-distance city1 city2)` returns the distance between two adjacent cities in km (or `nil` if they are not adjacent) and `(get-coordinates city)` returns the xy position (in km) of `city` relative to a flat coordinate system. The file also contains a function stub for the exercise. The answers to (b) and (c) can either be put into the lisp file (in comments) or with the answers for Q1 and Q2.

(a) Implement RBFS in LISP, using the straight line heuristic.

(b) Run the function to find paths for the following trips: Karlsruhe → Aachen, Trier → Dresden, Passau → Wilhelmshaven. How many cities were visited and what paths were found? *Note:* RBFS can visit quite a large number of cities.

(c) Compare the routes found and the number of cities visited by A* graph search and RBFS. Explain your observations.