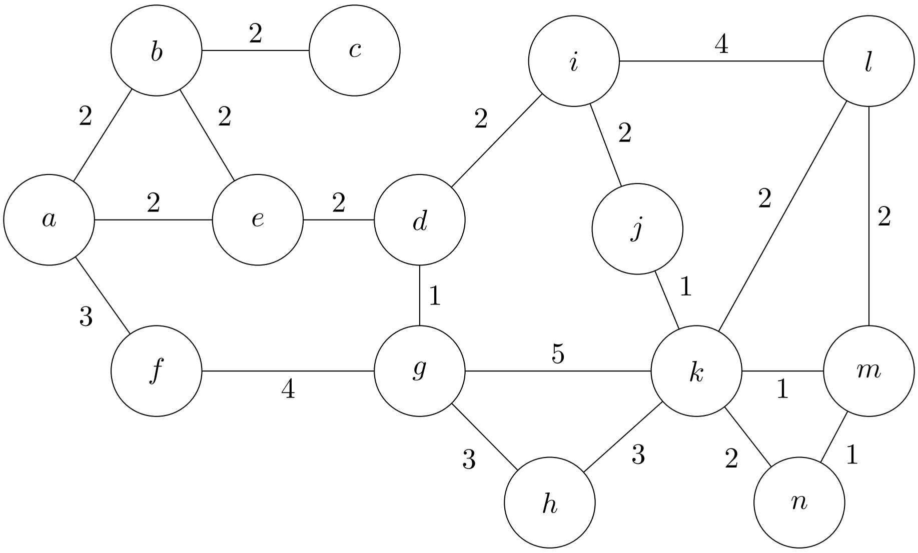
Katrin Bott (katrin.bott@student.uni-tuebingen.de, 3944875)
Mila Gorecki (mila.gorecki@student.uni-tuebingen.de, 3962077)

Assignment Nr. 3

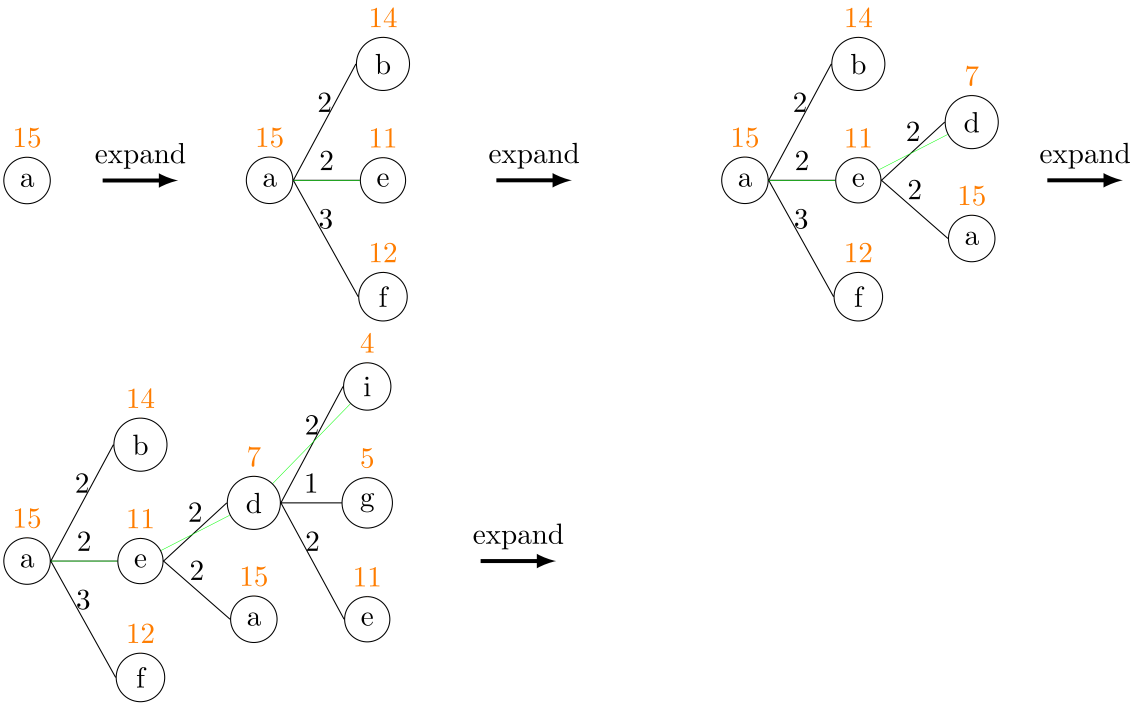
(Abgabetermin 16.11.2016)

Question 1

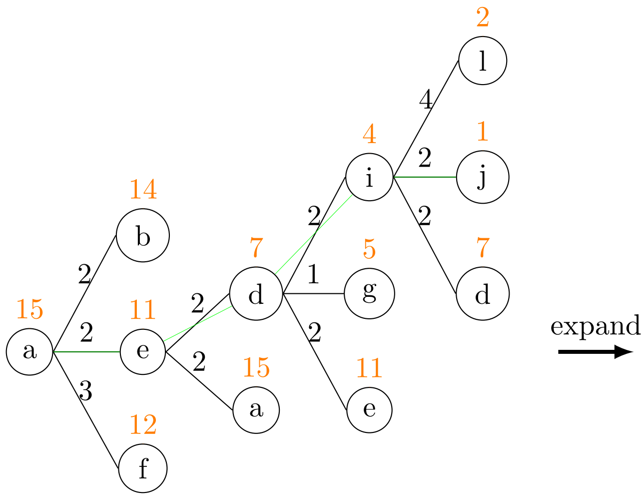


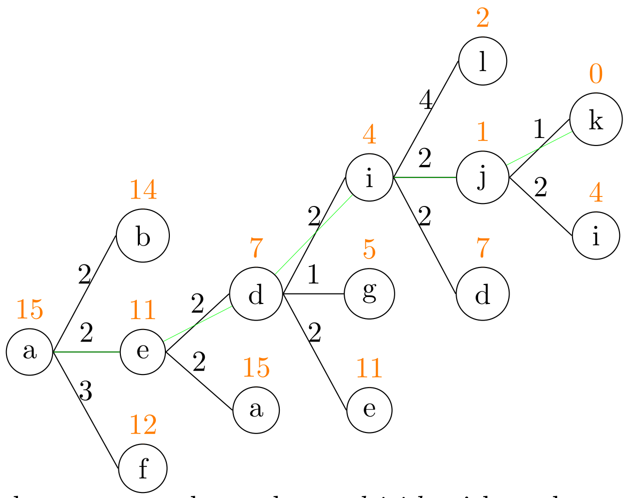
(a)

Greedy Best-First Search expands the node with the lowest heuristic value h(n). As the algorithm is greedy, the next node is allways the node with minimal h(n). In the following pictures, greedy best-frist search ist visualized. The orange values represent the value given by the heruistic function, while the green path is the path chosen so far.



Katrin Bott (katrin.bott@student.uni-tuebingen.de, 3944875)
Mila Gorecki (mila.gorecki@student.uni-tuebingen.de, 3962077)

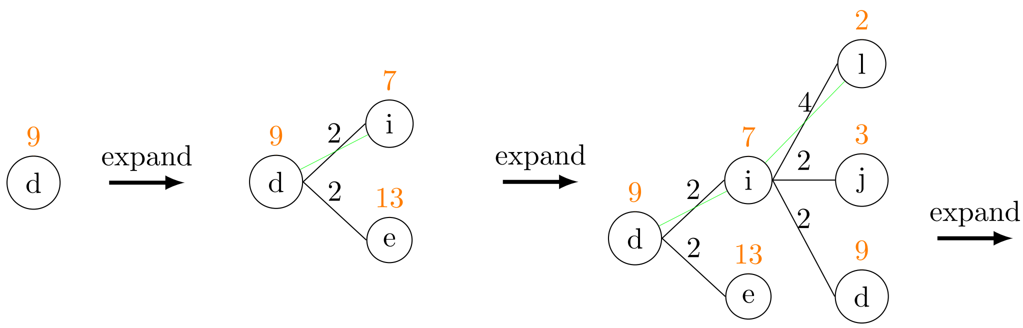


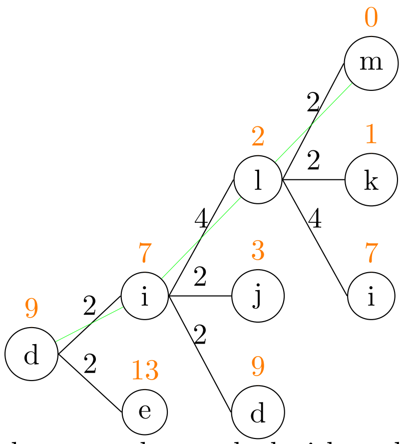


The algorithms returns the path *a-e-d-i-j-k* with path costs $2 + 2 + 2 + 2 + 1 = 9$, which is optimal for the path from a to k. The way is optimal because every alternative to the subpaths from a to antother node is more expensive (e.g. a-e (costs 2) could be replaced by a-b-e (costs 4) at expense of path costs). Therefore the way along any other path is more expensive and *a-e-d-i-j-k* is optimal.

(b)

The way nodes are chosen is the same as in (a) to find a path from node d to target m.

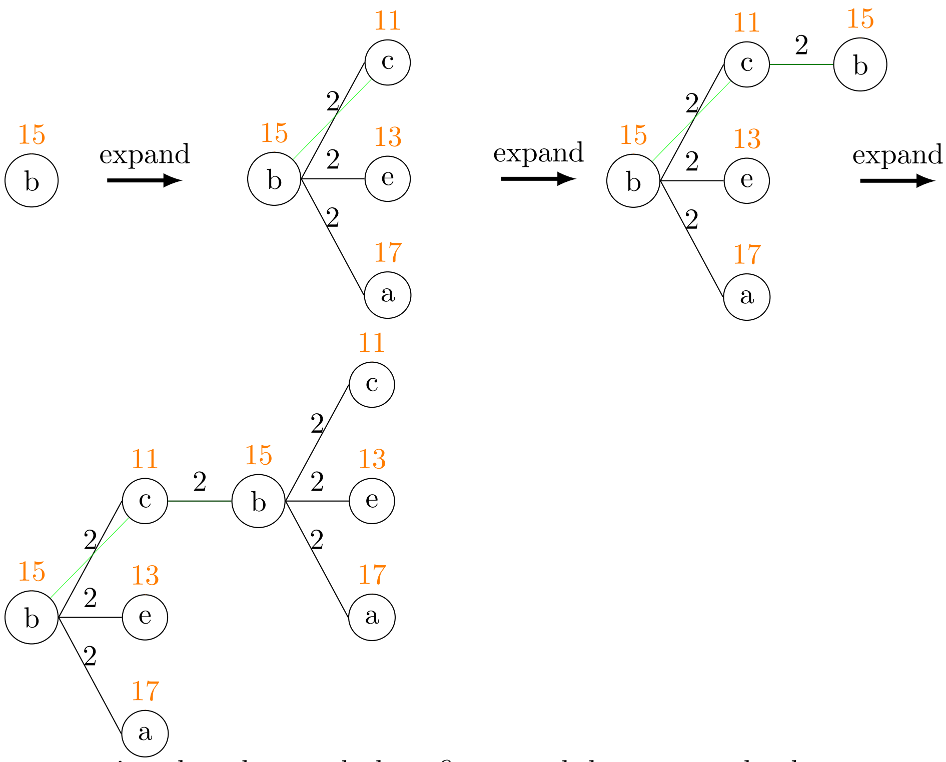




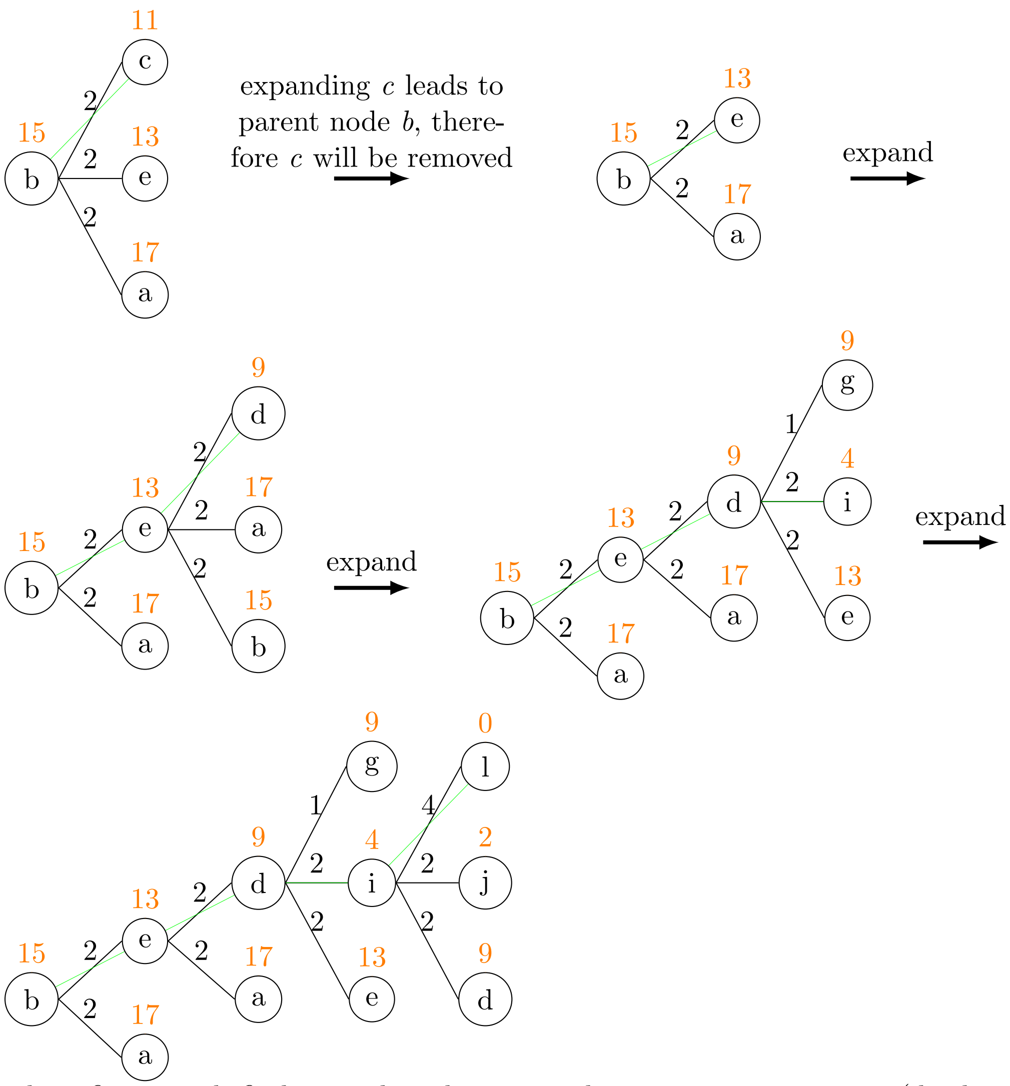
The algorithm started on node d with goal m returns the path d-i-l-m with costs 2+4+2 = 8. This path is not optimal as there exists a path with lower costs: $w(d\text{-}i\text{-}j\text{-}k\text{-}m) = 2+2+1+1 = 6 < 8 = w(d\text{-}i\text{-}l\text{-}m)$, where w is the weigth of the path (costs).

(c)

The way nodes are chosen is the same as in (a) to find a path from node b to target l.



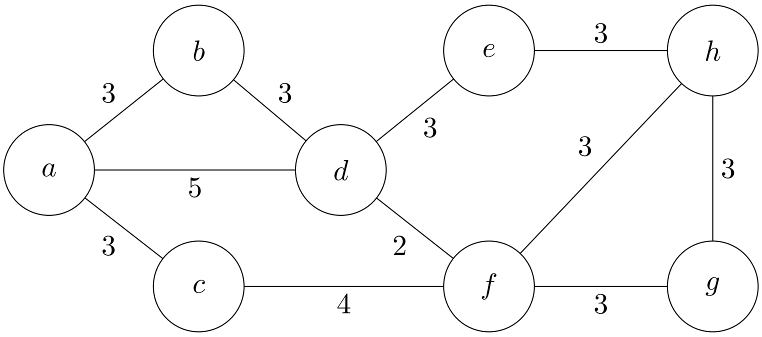
Under the assumption that the greedy best-first search has no saved values e.g. arepresentation of the parent and therefore realises, that it returns to the parent node, the algorithm will never stop expanding node b and c and stays in that loop forever.
Otherwise, if node c will be visited, but the removed from the path and no longer expanded, the algorithm would find the optimal pat b-e-d-i-l-m with costs 10:



The greedy best-first search finds a path as long as it does not get into a trap (dead end, cycle). The whole performance depends on the heuristic function which has to be chosen carrefully. Though there is no guarantee that greedy best-first search finds the shortest path between two nodes.

Question 2

(a)



n	$h(n)$
a	10
b	4
c	7
d	3
e	0.5
f	2
g	1
h	0

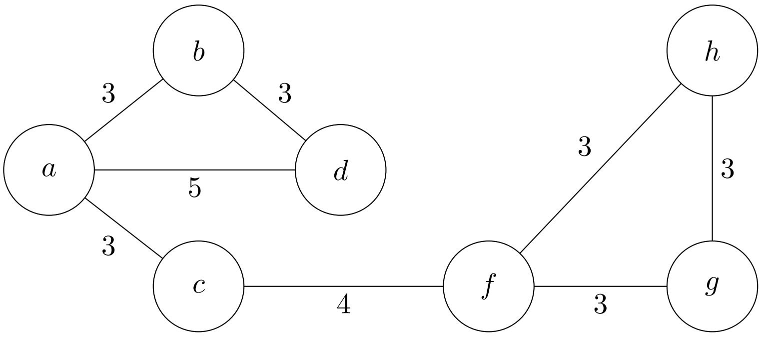
$$f(n) = \sum_{\text{path to } n} g_{(i,j)} + h(n)$$

The table contains the minimal calculated *f(n)*-values at each step for node n. The purple colored *f(n)*-value is the one with minimal costs for the subpath and will be expanded in the next step. The gray values are values of already visited nodes, which do not occur anymore in a priority queue.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
0	$0+10$ $= \mathbf{10}$	/	/	/	/	/	/	/
1	(10)	$3+4$ $= \mathbf{7}$	$3+7$ $= \mathbf{10}$	$5+3$ $= \mathbf{8}$	/	/	/	/
2	(10)	(7)	10	8	/	/	/	/
3	(10)	(7)	10	(8)	5+3+0.5 $= \mathbf{8.5}$	/	/	/
4	(10)	(7)	10	(8)	(8.5)	/	/	$5+3+3$ $= \mathbf{11}$
5	(10)	(7)	(10)	(8)	(8.5)	$3+4+2$ $= \mathbf{9}$	/	11
6	(10)	(7)	(10)	(8)	(8.5)	(9)	3+4+3+ 1 $= \mathbf{11}$	$4+3+3$ $= \mathbf{10}$

There is no unique shortest path, because the weight of path a-d-f is the same as the weight of path a-c-f. Therefore it doesn't matter wich subpath to take to then get from f to h. There are all toegether two possibilities to get from a to h with minimal costs.

(b)



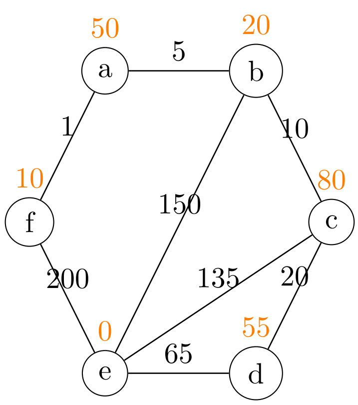
n	$h(n)$
a	10
b	4
c	7
d	3
e	0.5
f	2
g	1
h	0

The table contains the minimal calculated *f(n)*-values at each step for node n. The purple colored *f(n)*-value is the one with minimal costs for the subpath and will be expanded in the next step. The gray values are values of already visited nodes, which do not occur anymore in a priority queue.

	a	**b**	**c**	**d**	**f**	**g**	**h**
0	$0+10$ $= \mathbf{10}$	/	/	/	/	/	/
1	(10)	$3+4$ $= \mathbf{7}$	$3+7$ $= \mathbf{10}$	$5+3$ $= \mathbf{8}$	/	/	/
2	(10)	(7)	10	8	/	/	/
3	(10)	(7)	10	(8)	/	/	/
4	(10)	(7)	(10)	(8)	$3+4+2$ $=\mathbf{9}$	/	/
5	(10)	(7)	(10)	(8)	$3+4+2$ $= \mathbf{9}$	/	/
6	(10)	(7)	(10)	(8)	(9)	3+4+3+1 $= \mathbf{11}$	$4+3+3$ $= \mathbf{10}$

We get the path $a-c-f-h$ as the shortest path. In step 2) we encountered a cycle $a-b-d$, since we

(c)



In this example the f-value of the target node was changed four times, it is therefore possible to make it change three times (remove one of the edges (b-e, c-e, f-e)) and even four times.

	a	**b**	**c**	**d**	**e**	**f**
0	$0 + 50 = \mathbf{50}$	/	/	/	/	/
1	(50)	$5 + 20 = \mathbf{25}$	/	/	/	$1 + 10 = \mathbf{11}$
2	(50)	25	/	/	$200 + 0 = \mathbf{200}$	(11)
3	(50)	(25)	$5 + 10 + 80 = 95 = \mathbf{95}$	/	$5 + 150 = \mathbf{155}$	(11)
4	(50)	(25)	(95)	$5 + 10 + 20 + 55 = \mathbf{90}$	$5 + 10 + 135 = \mathbf{150}$	(11)
5	(50)	(25)	(95)	(90)	$5 + 10 + 20 + 65 = \mathbf{100}$	(11)

Question 3

(a)

(b)

(c)