	Plan length	expansions	Goal test	New nodes	Time
BFS	6	43	56	180	0.026860407830703203
BFTS	6	1458	1459	5960	0.6823091510843168
DFS	12	12	13	48	0.007290360681002112
DFS limited	50	101	271	414	0.07142405992717533
UCS	6	55	57	224	0.03397115436237372
Recursice	6	4229	4230	17029	1.8551575709535102
greeady	6	7	9	28	0.004530224126877856
a* h_1	6	55	57	224	0.03325546105965242
a* ignore	6	41	43	170	0.033684561236187474
a* lvelsum	6	11	13	50	0.334701295748316

So looking at these results I can say that almost all search algorithms are optimal(except dfs and dfs limited...) For first problem greedy search is really good choise. Also BFS and A* are the good options. BFTS works too long because it doesn't consider visited nodes. In terms of memory consumption DFS is good.

All heuristic searched made a good job also. A* levelsum is great in terms of optimality.

	Plan length	expansions	Goal test	New nodes	Time
BFS	9	3343	4609	30509	12.197592538788761
BFTS	Too long				
DFS	1444	1669	1670	14863	11.74676694531203

DFS limited	Too long				
UCS	9	4853	4855	44041	11.820613625095138
Recursice	Too long				
greeady	17	998	1000	8982	2.3914577879116123
a* h_1	9	4853	4855	44041	14.31584138161728
a* ignore	9	1450	1452	13303	5.05178021183027
a* lvelsum	9	86	88	841	33.204141316559856

For problem number 2 not all non-heuristic searched coped with the problem in an acceptable time. From non-heuristic BFS and UCS are still optimal. BFS is also fast enough but uses a lot of memory.
A* ignore is really fast and A* levelsum is still super optimal(looks like it knows where to go:))

	Plan length	expansions	Goal test	New nodes	Time
BFS	12	14663	18098	129631	123.68751567538713
BFTS	Too long				
DFS	571	592	593	4927	3.6587543214316867
DFS limited	Too long				
UCS	12	18223	18225	159618	75.10463798306166
Recursice	Too long				
greeady	22	5578	5580	49150	24.188024165191422
a* h_1	12	18223	18225	159618	77.39354213558694
a* ignore	12	5040	5042	44944	22.848290932843938
a* lvelsum	12	314	316	2894	167.56909535054484

Well, here results are exactly the same as in previous problem.