AI Assignment 1
Boganova Alina BS18-04

Implementation details:

1. Random search

The search is implemented as 100 attempts, during which the direction of the next step is chosen in two-way random decision: during the first decision one of four or five direction is chosen (up, right, left, down or pass, if it hasn't been made during the previous steps of the attempt), the second decision is only made if the 'pass' direction was chosen at the previous stage, it affects the direction of the pass. Such an algorithm was chosen to increase the probability of finding the target point (passes are kind of risky).

Also the choice of the step's direction was made according to the sensors of the agent, who can see one step around. If the agent sees an orc in front of him, he does not make stp to this direction; if he sees the touchdown point, he moves to it directly (no random choice in this case); if agent wants to make a pass, he does not perform it in the direction, where he sees the orc or the edge of the field, if the passes on the diagonal are denied, if agent see the walls at the directions, close to the diagonal direction (e.g. in initial position agent cannot make pass to the pint (-1, -1), because he sees the walls at (0, -1) and (-1, 0)). Such step-checking is performed for any other algorithm.

Another assumption, made during the implementation is that the agent does not go away. In case, when the agent cme to the point where he already was, the branch is stopped (marked, as failed). The same assumption made for this algorithm is applied to all next.

To find the best path after each attempt the path and the score of the lowest-score attempt is saved and propagated.

2. Search based on prolog tree (tree search)

The algorithm is similar to the random search, but the random component, now instead of choosing a random member from a list of possible decisions, all of them are tested. Thus the algorithm always finds the lowest score.
THe algorithm is implemented with the usage of prolog tree: it finds all possible paths and then finds the minima. However, such an approach requires quite a lot of RUM (during the tests 10Gb was given to prolog for 6x6 maps).

3. Backtracking search

The algorithm itself is a tree with a cut: in case, when during computations the agent reaches the path, with score greater, then current minima, the branch is stopped. This method also returns the most optimal path.

Tests

To test the system 10 tests were created (except unit one):

1. An empty map with only a touchdown point on it
2. The map with several orcs and touchdown point (orcs are sparse - distributed)
3. The map with several humans and touchdown pint (humans are sparse - distributed)
4. Map with both humans, orcs and touchdown point
5. The map with a path to the touchdown made by humans only (score should be 0)
6. The map with orcs, who do not make a possibility except the track between them
7. The test with touchdown point at the origin
8. Test, where an agent ought to pass to reach the touchdown point
9. Unsolvable map, with impossible movement of the agent
10. Empty map without even touchdown point
11. - 20. Randomly chosen tests with a few (up to 8) agents

These maps were selected to first of all test all possible movements of the agent, then check the speed of finding path in possible conditions and also to test wrong input cases (10).

All the tests are 6x6 maps due to the low speed of the tree algorithm.

Total score is evaluated as sum of steps/passes on the cells without humans.

Impact of region of vision

What is changed by the region of seeing:

1. Faster search of the touchdown point
2. More checks before pass making (up, right, down or left direction only)

Significant increase in time savings are reached in many tests, especially, on those, where the target point was located far away from the origin, (tests 1, 5, 11, 19). Another example of such a test is presented below:

				T	
O	O	O	O		
					O

Region of vision makes no unpossible maps to be solved, due to the fact that it does not affect the possibility of making steps and passes, it only makes pass more safe and fuster converges to touchdown. That is also the reason why it does not make it vice versa (does not make solvable maps unsolvable). The only algorithm, where it affects solvability is Random search (e.g. test 1, 2, 4, 13, 14 and 18).

Testing and collecting statistics

The results of the testing are presented in attachment at the end of the document.

All tests

Here are statistical arguments against all (20) tests for each algorithm:

	Rand (1)	Rand (2)	Tree (1)	Tree (2)	Backtrack (1)	Backtrack(2)
Mean	15.4	16.35	328191.65	136147.4	93405.95	97950.60
Std	7.03	8.9	725081	443894	417474	437900
Min	0	0	0	0	0	0
Max	26	32	2495118	1979690	1867059	1958382
Median	17.5	17.5	6731.5	394	14	8

Student - T test between two sample means:

	Rand(1)	Rand(2)	Tree(1)	Tree(2)	Backtrack(1)	Backtrack(2)
Rand (1)	0	-0.37419	-2.02411	-1.3715	-1.00043	-1.00018
Rand (2)	0.37419	0	-2.02411	-1.37149	-1.00042	-1.00017
Tree (1)	2.02411	2.02411	0	1.01021	1.25496	1.21559
Tree (2)	1.3715	1.37149	-1.01021	0	0.31368	0.27396
Backtrack (1)	1.00043	1.00042	-1.25496	-0.31368	0	-0.03359
Backtrack (2)	1.00018	1.00017	-1.21559	-0.27396	0.03359	0

Green color indicates values with low significant difference, yellow - medium difference and red - absolutely different.

Due to the fact that the first 10 tests were “stress - test of the algorithm”, a lot of values for tree and backtrack algorithms are too high. It can be seen by the high difference between mean and median values of the data.

That is the reason why there also will be separately considered statistics about the last 10 tests.

Random sparse easy tests:

Here are statistical arguments against 10 last tests for each algorithm:

	Rand (1)	Rand (2)	Tree (1)	Tree (2)	Backtrack (1)	Backtrack(2)
Mean	16.8	15.7	35802	7590	64.3	34.7
Std	5.88	7.134	60510	11875	94	48
Min	5	5	1	1	0	1
Max	25	30	191605	35620	281	143
Median	18.5	17.5	9883.5	142.5	19	10

Student - T test between two sample means:

	Rand(1)	Rand(2)	Tree(1)	Tree(2)	Backtrack(1)	Backtrack(2)
Rand (1)	0	0.37614	-1.87018	-2.01671	-1.59435	-1.15018
Rand (2)	-0.37614	0	-1.87023	-2.01701	-1.62978	-1.21678
Tree (1)	1.87018	1.87023	0	1.44680	1.86769	1.86924
Tree (2)	2.01701	2.01701	-1.44680	0	2.00400	2.01193
Backtrack (1)	1.59435	1.62978	-1.86769	-2.00400	0	0.88333
Backtrack (2)	1.15018	1.21678	-1.86924	-2.01193	0.88333	0

Green color indicates values with low significant difference and red - absolutely different.

Statistical conclusions

From all the resulting data it is possible to conclude, that tree search is the slowest algorithm from these three, while random algorithm is fast, but not precise (on 7 out of 20 tests it did not find any path, while on 2/20 it found not the best one).

Random algorithm has the lowest standard deviation, it means, that it takes approximately the same time (for 0 ms to 32 ms) to be finished. Tree algorithm has the highest standard deviation, thus it has the most sparse time distribution.

Talking about correlations between algorithms - the highest similarities are, predictably the one with the similar algorithm, but with different regions of seeing.

Impossible arrangements

The difficulty of the solution of the arrangement depends on the algorithm, we consider. For random it is hard to find the target point, which is far from the initial position, especially in situations, when there are no humans, for whom the agent can pass (test 1, 2). For tree algorithms the complexity grows with the amount of non-orcs fields in a map (test 1, 10), the worst case for such algorithm - full map of humans except the touchdown point. For backtracking the worst case situation is, when attempts are directed not in desired way, e.g. the longest path to the target was found the first, and the shortest - the last.

Talking about the impossible maps, thera are a few categories of them:

1. No target point
2. Origin point is cycled by the orcs in such way, that agent cannot neither move, nor pass to the touchdown point area
3. The touchdown point is cycled by the orcs and it is impossible to successfully pass in such area

The common thing between 2 and 3 is that such arrangements provide separation of the area in two unconnected pieces.

Appendix

Time data:

The table below presents the time and results data, taken during the tests. Tree and backtrack algorithms always find the best solution, while random can find not the best one or find no such. The successfulness of the random is denoted with +, - or +/- signs at their column ("-" sign means, that the result wasn't reached at all, and "+/-" - that not the smallest possible result was reached and "+" that the shortest path to the touchdown point was found). The symbol in braces represent the distance of vision of the algorithms. Also in columns of random algorithms the grey cells denote unsuccessful searches.

Runtime of the algorithms

	Random (1)	Random (2)	Tree (1)	Tree (2)	Backtrack (1)	Backtrack (2)
1	-, 9ms	+, 15 ms	301268 ms	91922 ms	326 ms.	218 ms
2	-, 13 ms	+, 32 ms	1099 ms	564 ms	34 ms.	28 ms
3	+, 20 ms	+, 29 ms	2495118ms.	390144 ms	9 ms.	7 ms
4	+/-(9/6), 20 ms.	+/-(7/6), 22 ms	12364 ms.	3542 ms	29 ms.	23 ms
5	+, 8 ms.	+, 25 ms	1611866 ms.	181185 ms	19 ms.	7 ms

6	-, 26 ms.	-, 11 ms	26 ms.	1 ms	0 ms.	0 ms
7	+, 0ms	+, 0ms	0 ms.	0 ms	0 ms.	0 ms
8	+, 15ms	+, 23 ms	0 ms.	0 ms	0ms.	0 ms
9	-, 7ms	-, 6 ms	0 ms.	0 ms	0ms.	0 ms
10	-, 22ms	-, 7 ms	1784065 ms.	1979690 ms	1867059 ms.	1958382ms
11	+, 22 ms	+, 30 ms	43665 ms	224 ms	32 ms	11 ms
12	+, 5 ms	+, 18 ms	64 ms	9 ms	6 ms	6 ms
13	+/- (3, 2), 18 ms	(-/+)(4/2), 12 ms	48 ms	61 ms	0 ms	2 ms
14	-, 17 ms	+, 19 ms	19546 ms	11075 ms	87 ms	99 ms
15	+, 20 ms	+, 6 ms	78677 ms	10154 ms	55 ms	27 ms
16	+, 19 ms	+, 17 ms	221 ms	30 ms	4 ms	9 ms
17	+, 12 ms	+, 14 ms	109 ms	1 ms	3 ms	1 ms
18	-, 19 ms	+, 18 ms	24091 ms	18725 ms	281 ms	48 ms
19	+/- (4/ 3), 25 ms	+, 4 ms	191605 ms	35620 ms	174 ms	143 ms
20	+, 11 ms	+, 18 ms	1 ms	1 ms	1 ms	1 ms

Tests:

At the next two pages the tests are visualized as a 6x6 cell squares with symbols inserted:
O - orc
H - human
T - target point
The origin point is the left low cell

Test 1:

					T

Test 2:

					T
					O
		O		O	
	O				
					O

Test 3:

H				T	
				H	
				H	

Test 4:

				O	
					T
H	O				
H	O				O
					H

Test 5:

H	T				
H	H				
H	H				
H	H				
H	H				

Test 6:

		O		O	
		O	T	O	
		O		O	
O	O	O		O	
				O	
	O	O	O	O	

Test 7:

T					

Test 8:

			T		
	O	H			
O		O			
	O				

Test 9:

O	O				
	O				

Test 10:

Test 11:

				H	
		H			
			T		
	H		O		
		O			H

Test 12:

H		T		O	
				O	
	H				O
	O				

Test 13:

				T	
			O		
		T,H			
	O				
O					
			O		

Test 14:

					H
	O				
			O		
		O		T	

Test 15:

	H				
				O	O
	O		H		T
				H	

Test 16:

H					
		T		O	
		H			O
	O	H			

Test 17:

		T	T		
		T			

Test 18:

		O	T		
		H	O		O
		O			
		H			

Test 19:

			H		
		H			
			O	T	
					O

Test 20:

			H		
			O		
	O				
			T	O	
	O			O	