Introduction to Artifitial Intelligence

Assignment 1

Author: Sergey Makarov

To test:

There are two possible ways to run the program:

1. `swipl -s game.pl -g $methodname$_entry_point -t halt` , where instead of `$methodname$` should be substituted either `random` , `backtrack` or `bfs` . In this case input will be imported from `input.pl` file.
2. Comment `:-[input].` in the very beginning of the file and run `swipl -f $path_to_input$ -s game.pl -g $methodname$_entry_point -t halt` where instead of `$path_to_input` path to the input file (or it’s name if it is in the same directory as game.pl (http://game.pl)) should be specified.

Assumptions made during implementation:

- Field is always 20x20. However, it can become smaller size using orcs as margins.
- Player cannot pass directly to the touchdown, only to another human
- All search algorithms are uninformed. They try to go/throw on each direction
- Random search make 100 tries, each take at most 100 turns. By turn we mean throw, going to adjacent cell, but not rotation and not passing by ball. If it didn’t succeed after 100 tries, it terminates.
- Backtracking return the best possible path, not the first found. Thus, it runs until all possible paths observed.
- As third search algorithm I use bfs, which is basically A^*, but with zero heuristic. Heuristic cannot be determined for uninformed search, and that is why this algorithm is applied.
- There can be several touchdowns, but not zero

Part 1.

I’ve implemented three search algorithms: Random Search, Backtracking and Breadth-first search. Before going through algorithms, I’d like to mention some functions that are reused in all algorithms:

This function is used to append to existing path new turn. It just prints to `Output` existing path + new turn :

```
add_turn_to_string(Throw, X, Y, Path, Output) :-
    swritef(Output, '%w\n%w%w %w', [Path, Throw, X, Y]).
```

The following function performs logic of throw: it recursively checks all coordinates in given direction and if there is a human in this direction, terminates with true and assigns to passed value new coordinates:

```
    throw(X, Y, Xnew, Ynew, DirX, DirY) :-

    Xn is X+DirX, %find new coordinates
    Yn is Y+DirY,
    not(orc(Xn, Yn)),(
    (   human(Xn, Yn) % if it is human, return values
    ->  Xnew is Xn,
        Ynew is Yn
    ;   ( Xn>=0, %if no, check validity of coordinates and recursively perform cl
        Yn>=0,
        Xn=<19,
        Yn=<19
        ),

        throw(Xn, Yn, Xnew, Ynew, DirX, DirY))
    ).
```

Random Search

This Algorithm is very easy. It doesn’t make any assumptions, just pick random turn out of all possible (for example, we exclude going downwards if $y = 0$). All rules are preserved: if we come to the cell with human, we do not count next turn, on orc we lose, on touchdown win. Ball can be randomly thrown out in the direction where there is no human, this counts as lost try. This leads to not really high performance on big maps, where touchdown is many turns away from (0, 0). Also, as throws have probability of $\frac{1}{3}$ to be performed on the first turn, if there is a human nearby to the touchdown that is one throw away from (0, 0), there is very big chance to succeed in 100 tries. Runtime is almost constant $\approx 0,035 sec$. For example, consider the following map `input1.pl` :

.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
T	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	T	.	.	.	.	.	.	.
H	.	.	.	.	.	.	.	.	.	H	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	H	.	.	T	.	.	.	.	.	.

Here, with best-possible score 4, we get such statistics:

mean	7.364
first quartile	4
median	6
third quartile	8
mode	4

95% of got scores are between 6 and 9.

Obviously, if length of optimal path is longer, for example this map

`input2.pl`

.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
T	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	T	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	T	.	O	.	.	.	.	.	.	.

Then only in $\approx \frac{1}{3}$ runs some path, not always optimal, is found.

But adding some humans to the same map we can increase this chance to $\approx \frac{1}{2}$:

`input3.pl`

.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	H	.	.	.	.	.	.	O	.	.	.	.	.	.	.
T	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	H	H	.	.	.	.	O	.	.	.	.	.	.	.
.	.	H	.	.	.	.	H	.	.	O	.	.	.	.	.	.	.
.	.	.	.	H	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	H	.	H	.	.	.	.	.	O	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	H	O	.	.	.	.	.	.	.
.	H	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
.	H	.	.	.	.	.	.	T	.	O	.	.	.	.	.	.	.

And if we make our map even more complicated, let’s say only one touchdown far away from start, the chance of getting a solution is neglectable. Needs to say, that other methods as well do not lead us to solution, as they take to This is example:
`input4.pl`

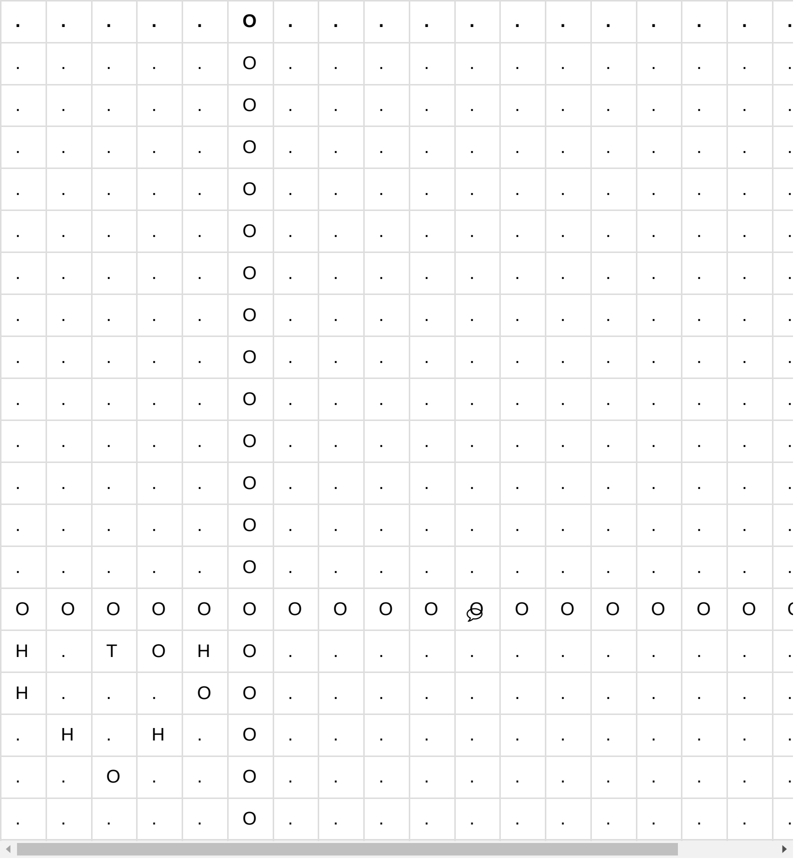
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	T	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	H	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	H	.	.	.	.	.	.	.	.	.	.
.	H	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	H	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

Only 2 runs out of 200 was successful.

More maps with statistics of each method on every map will be in appendix, here are just some basics.

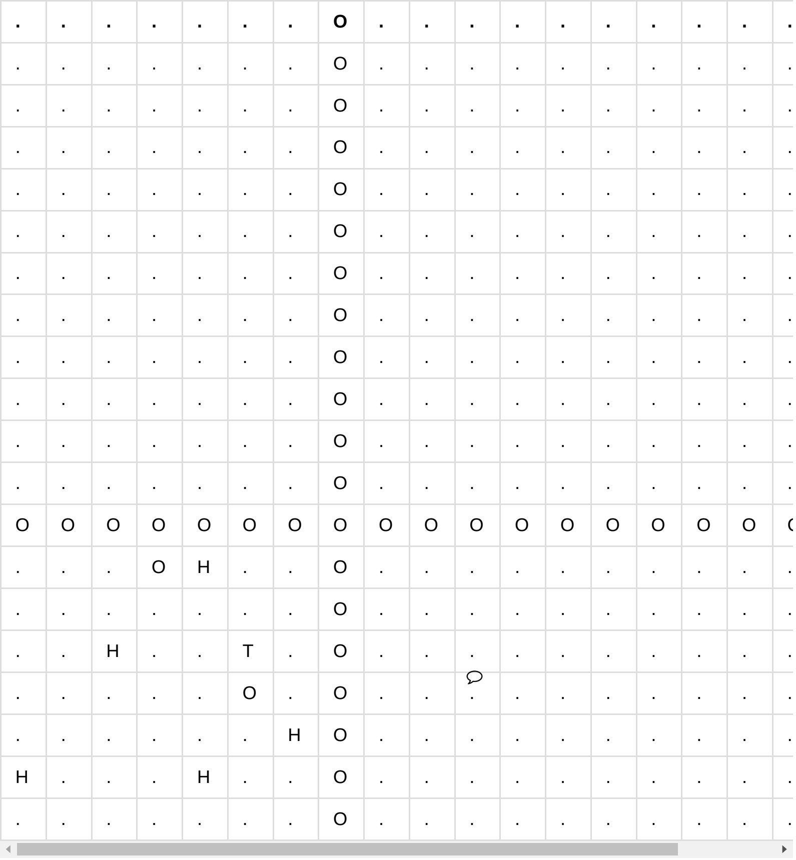
Backtracking

This method has really good performance for small maps. For example, on 10x10 map it usually takes no more than 3 seconds to run. However, for `input1.pl` it takes 33 seconds to finish, as map is quite big. Of course, by our assumption all maps are 20x20, but still we can place orcs as a borderline. Some comparison of performance:
`input5.pl` takes 13-15 ms to traverse all paths.



.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
H	.	T	O	H	O	.	.	.	.	.	.	.	.	.	.	.	.
H	.	.	.	O	O	.	.	.	.	.	.	.	.	.	.	.	.
.	H	.	H	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	O	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.

for `input6.pl` it takes 30-40 ms



.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	O	H	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	H	.	.	T	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	H	O	.	.	.	.	.	.	.	.	.	.
H	.	.	.	H	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.

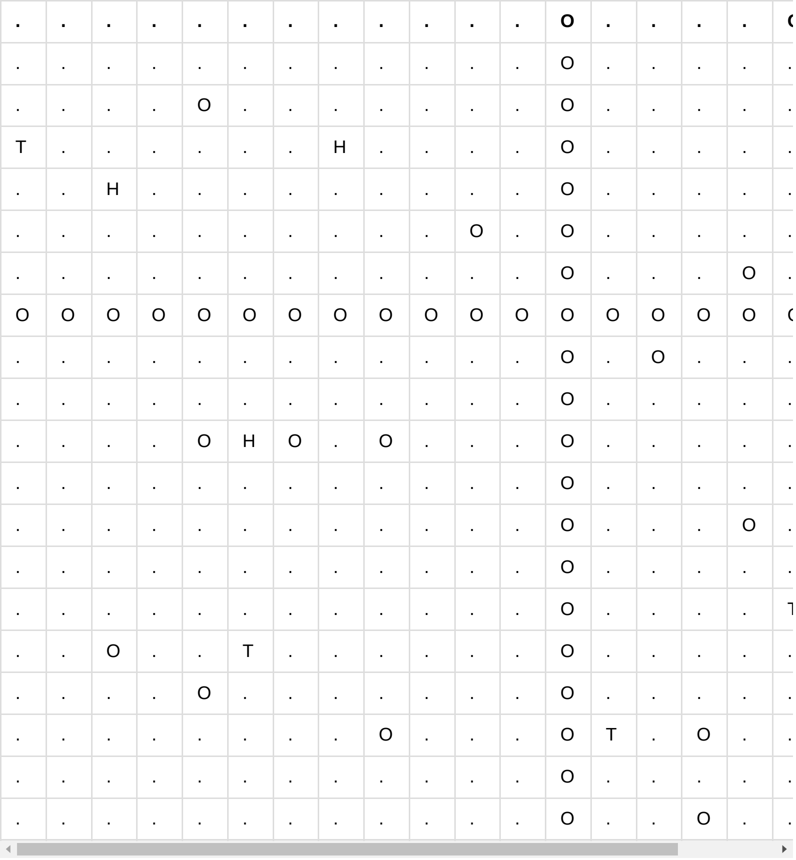
Observe `input7.pl` . There are quite a lot of paths, but as there is a path of length one, which can be easilly found, when our backtracking finds this path it throws away all paths that are longer, so the runtime is just 10 ms

.	.	O	O	.	.	.	.	.	O	.	.	.	.	.	.	.	.
O	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	O	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	O	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	O	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	O	.	.	.	.	.
T	.	.	.	.	.	O	.	O	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	O	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	O	.	.	T	.	.	O	.	.	.	.	T	.	.	.
.	.	.	.	.	.	.	.	.	T	.	.	.	.	.	.	.	.
.	T	.	.	.	.	.	.	.	O	.	.	.	.	T	.	.	.
.	.	.	.	.	O	.	.	O	O	.	.	.	.	.	.	.	.
.	.	.	.	.	T	.	.	.	O	.	.	.	.	.	.	.	.
.	O	.	.	.	.	H	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	T	.	T	.	T	.	.	.
.	.	.	.	.	O	.	H	T	O	.	.	.	O	.	.	.	.
.	T	.	.	.	.	H	.	.	O	.	.	.	H	.	.	.	.

However, if we remove this touchdown, runtime will be greater:
`input8.pl` takes about 40 ms

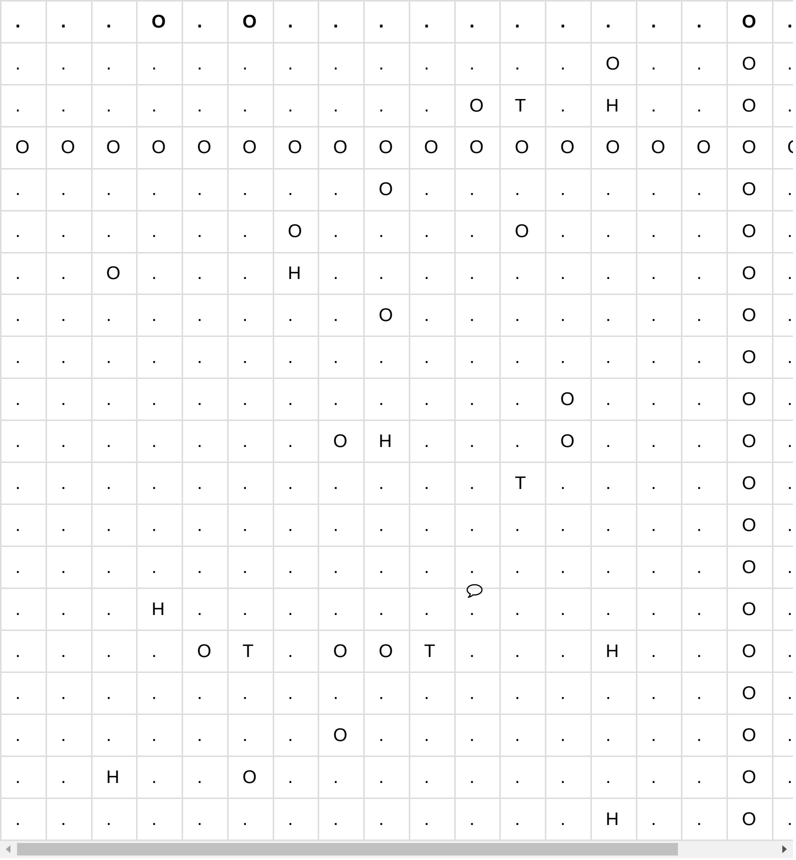
.	O	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
O	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	O	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	O	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	O	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	O	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	O	.	.	.	.	.
T	.	.	.	.	.	O	.	O	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	O	.	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	O	.	.	.	.	.	O	.	.	.	.	T	.	.	.
.	.	.	.	.	.	.	.	.	T	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	T	.	.	.
.	.	.	.	.	O	.	.	O	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	O	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	T	.	.	O	T	.	T	.	T	.	.	.
.	.	.	.	.	O	.	.	.	O	.	.	.	O	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	H	.	.	.	.

`input9.pl` , which is 12x12, takes $1.5s$ to run



.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	O
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	O	.	.	.	.	.
T	.	.	.	.	.	.	H	.	.	.	.	O	.	.	.	.	.
.	.	H	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	O	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	.	.	.	.	.	.	.	.	.	O	.	O	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	O	H	O	.	O	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	T
.	.	O	.	.	T	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	O	.	.	.	O	T	.	O	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	O	.	.

`input10.pl` , which is 16x16, takes about to $15s$



.	.	.	O	.	O	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	O	.
.	.	.	.	.	.	.	.	.	.	O	T	.	H	.	.	O	.
O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	O	.	.	.	.	O	.	.	.	.	O	.
.	.	O	.	.	.	H	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.	O	.
.	.	.	.	.	.	.	O	H	.	.	.	O	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	T	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	H	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	O	T	.	O	O	T	.	.	.	H	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	O	.
.	.	H	.	.	O	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	H	.	.	O	.

just a random 20x20 map `input11.pl` even more, $4m\ 30s$

.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	T	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	T	.	.	.	O	.	.	H	.	.	.	.	.	O
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.
.	.	.	O	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.
O	T	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.	.	.
.	O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	O	.	.	.	T	O	.	.	.	H	H
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	H	.	.	O	O	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.	.
.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	O	.	.
.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	T	.

Maps like `input4.pl` are too hard to solve, after 40 min I've dropped test as it is unreasonable long to count as solution.

Breadth-first search

This algorithm in prolog requires a lot of unnicessary in other languages calculations, it's quite imperative, but still works. For `input1.pl` runtime is also about to $30ms$, but in `input2.pl` it takes more than 1 minute. Again, all important statistics will be in appendix, but it is important to say that bfs has quite similar to random search requirements, but it does not fail on hard maps, just take too long to run.

Part 2

In my implementation there is no vision, we just observe all accessible cells and test them. For sure, if implementation taked into account environment around cell, for some maps that take too long to complete there can be improvements (not for random, it is still random). But on average it gives only constant time improvement, while complexity depends on size of map exponentially, so this couple of maps that will run faster are neglectable. Example of such map is `input4.pl` , where backtracking can see touchdown from edge and quite rapidly find the shortest path (but only if we throw away paths longer than current, full backtracking takes ages almost on every big map). For algorithms I have there is no map that become unsolvable because if the solution exist both backtracking and bfs will find it, even though it may take ages.

Part 3

As I said in part 1, there is no completely unsolvable maps (except maps like `input12.pl` and `input13.pl` , where there is no way to reach touchdown), but there are some maps that will take quite a long to process.
`input12.pl` :

.	H	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	H	.	.	H	O	.
O	.	.	.	O	.	.	.	.	.	.	.	O	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	.	.	O	.	.	.
.	.	.	.	O	H	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	H	.	.	.	.	H	.	.	O
.	.	.	.	O	.	.	.	.	.	.	.	.	.	T	.	.	O
.	.	.	.	O	.	.	.	O	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
O	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	O
.	.	.	O	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	.	.	.	.	O	.	.	.	.	.	.
O	O	O	O	O	T	O	O	O	O	O	O	O	O	O	O	O	O
.	.	.	O	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	O	.	.	H	.	.	.	.	.	.	.
.	.	.	O	O	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	O	.	.	H	.	.	.	.	O	.	.	H	.	.

`input13.pl`:

.	.	O	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	H	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	H	.	.	.
.	.	.	.	.	.	.	.	.	H	.	.	H	.	.	.	.	.	.
O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	H	.	.	.
O	.	.	H	.	.	.	.	.	.	.	.	O	.	.	.	.	.	O
.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.
.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	O	.	O	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	O	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
O	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	H	.	.	.

Hard to solve maps are big, sparse (not a lot of humans and orcs), has no short solution, so that backtracking cannot process it in a minute, must take more than 8 turns for optimal solution, so that random search will rarely find such a solution and bfs will take to long to process. The easiest example is just one touchdown somwhere close to the center.

Appendix 1: Statistics for maps and methods

file	random search	backtracking	bfs	comment
`input1.pl`	almost 100% success;$0.02s$ runtime	$33s$ runtime	$0.029s$ runtime	map is big, but pash is short, so random ans bfs perform better than dfs
`input2.pl`	$0.022s$ runtime, but only 30%success	$0.022s$ runtime	$1min20s$ runtime	map is bounded, so backtrack is very fast
`input3.pl`	same runtime, 40% success	$2.8s$ runtime	$> 10min$	more humans - better for random search, but a bit harder for others because of a lot of successful throws. Especially for bfs
`input4.pl`	same runtime, less than 1% success	~	~	very hard map for dfs, bfs
`input5.pl`	about to 50% success	$0.013ms$	$0,009ms$	very easy map for all
`input6.pl`	13% success	$0,032ms$	$2.8s$	again, bounded map with reasonably long path is ok for all except of random, for which path of length 6 is rather complicated
`input7.pl`	100% success	$0.008ms$	$0.009s$	nothing to say, next
`input8.pl`	$5-7\%$	$0.04s$	$6min50s$	optimal path of length 8, hard for bfs and random
`input9.pl`	$2-3\%$	$1.8s$	~	path of 9 - too hard for bfs and random
`input10.pl`	10%	$15s$	$22s$	path of length 6
`input11.pl`	couple of percent	$4min30s$	~	20x20 map
`input12.pl`	~	~	~	impossible
`input13.pl`	~	~	~	impossible
`input14.pl`	9%	~	$6.77s$	randomly generated 20x20map, ok for bfs, but too hard for backtrack
`input15.pl`	100%	$0.018s$	$0.010s$	path of length 2, easy for all
`input16.pl`	almost always optimal solution	$0.008s$	$0.015s$	lentgh 4, again nothing interesting
`input17.pl`	almost 100% to find solution, but very often it is twice longer than optimal	$0.06s$	$0.94s$	length 5, but a lot of humans and touchdowns randomly spreaded, so random shows different result each time
`input18.pl`	see input17	$0.05s$	$0.94s$	see input17