Name: Wenhao Li ID:#14328112

Introduction:

Every character in this report and code is written ONLY by myself. The programming language I am using is C++. The code is run and tested on my own computer (Windows x64 platform). I only used standard C++ library, so the code should be able to compile on any platform.

How to compile:

The code directory is as follows
./HW2.sln ···Visual studio 2019 solution file
./HW2 ···The source codes (Timer.h Map.h Map.cpp main.cpp)
./HW2/Debug ···The executable file

If you are on Windows and have Visual studio, open the .sln solution file and hit ctrl+f5 to run or run the executable in ./HW2/Debug directly. If you are on Unix, use g++ to compile without any additional flags.

Algorithm description:

0. Cost Function:
   As given in the homework description:
   +4 for cleaned up
   -1 for move ←
   -1.1 for move →
   -1.2 for move ↑
   -1.3 for move ↓
   -0.2 for suck
   0 for NoOp
   The smallest cost is the one with highest performance points.
1. Uniform cost tree search:
   We construct a fringe as a normal array. First, expand the initial state and push the result into the fringe. And we enter in a loop. Each time, only the smallest cost node in the fringe is expanded and push the expanded nodes into the fringe. The expanded nodes are removed from the fringe and pushed into the result. Each expansion is counted as 1 more depth. If the fringe is empty before expansion, the search will ended as failure.
2. Uniform cost graph search:
   Similar to uniform cost tree search stated above. The difference is that we created a new

array called 'visited'. And every time we pick a node in the fringe, we first compare it with all visited nodes and check whether the node has been visited before. If it is, delete it from the fringe to avoid revisiting.

3. Depth-limited depth-first tree search:
   We construct the fringe as a stack (LIFO). First, expand the initial state and push the one of the resulting nodes randomly into the fringe. And we enter in a loop. Each time, we expand the node at the top of the fringe, choose one of the resulting nodes randomly (to break the tie of the nodes at the same depth) and push it into the fringe. Each expansion is counted as 1 depth. If no more expansion is possible, we pop off this node from the fringe.
4. Depth-limited depth-first tree
   Similar to depth-limited depth-first tree stated above, the only difference is we created a new array called 'visited'. Each time, we choose the node at the top of the fringe and we compare the resulting expanded nodes with all visited nodes to check whether the node has been visited before. If it is, delete it from the resulting nodes to avoid revisiting.

Result:

The - character in the screenshot represents No-op, + represents suck.

Test Case1:

The initial state is under follows, X representing a dirty room and S for the initial position of the vacuum cleaner.

X			X	X
X			X	X
	S			X
X		X		
X			X	X

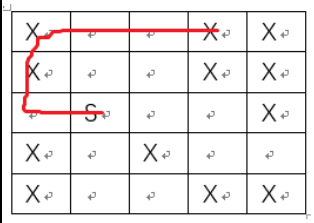
For uniform cost tree search, the program produce this result:

```
--------------------Test Case 1----------------------
-------------------Runing Uniform Cost Tree Search---------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
-        Cost=0
There are 2 movements in total.
--       Cost=0
There are 3 movements in total.
---      Cost=0
There are 4 movements in total.
----     Cost=0
Total expansion= 51

The best performance within 10 movements is:
There are 10 movements in total.
----------       Cost=0
Finished in 0.003 seconds.
```

It works as expected because according to the algorithm, it always expand the nodes with a highest performance. In this situation, No-op will produce the highest performance of 0, and all other options will produce a negative performance. Therefore, the result is not optimal.

For uniform cost graph search:

```
--------------------Runing Uniform Cost Graph Search----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
←        Cost=-1
There are 1 movements in total.
→        Cost=-1.1
There are 1 movements in total.
↑        Cost=-1.2
There are 1 movements in total.
↓        Cost=-1.3
Total expansion= 52

The best performance within 10 movements is:
There are 9 movements in total.
←↑+↑+→→→+  Cost=4.7
Finished in 0.008 seconds.
```



This is the only search algorithm that produce the optimal solution. And the path found is shown at the right.

For depth-first tree search:

```
--------------------Runing Depth First Tree Search----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
←        Cost=-1
There are 2 movements in total.
←↑       Cost=-2.2
There are 3 movements in total.
←↑→      Cost=-3.3
There are 4 movements in total.
←↑→←          Cost=-4.3
Total expansion= 60

The best performance within 10 movements is:
There are 9 movements in total.
←↑→←+-↑-↓  Cost=-3
Finished in 0.01 seconds.
```

It expands one branch of the search tree all the way down, which is certainly not optimal. Because the ties are broken randomly, the results will be different every time the program is run.

For depth-first graph search:

```
--------------------Runing Depth First Graph Search----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
↑        Cost=-1.2
There are 2 movements in total.
↑→       Cost=-2.3
There are 3 movements in total.
↑→→      Cost=-3.4
There are 4 movements in total.
↑→→→          Cost=-4.5
Total expansion= 30

The best performance within 10 movements is:
There are 9 movements in total.
↑→→→↓+↓←↑         Cost=-5.5
Finished in 0.012 seconds.
```

It expands one branch of the search tree unless all the nodes has been visited, which is also not optimal. Unlike depth-first tree search, it could never perform no-op, because it will have been visited.

Test Case 2:

Similar to test case 1, only uniform cost graph search can give us the optimal solution.

```
--------------------Test Case 2-----------------------
--------------------Runing Uniform Cost Tree Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
→        Cost=0
There are 2 movements in total.
→→       Cost=0
There are 3 movements in total.
→→→      Cost=0
There are 4 movements in total.
→→→→     Cost=0
Total expansion= 51

The best performance within 10 movements is:
There are 10 movements in total.
→→→→→→→→→→       Cost=0
Finished in 0.007 seconds.
--------------------Runing Uniform Cost Graph Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
←        Cost=-1
There are 1 movements in total.
→        Cost=-1.1
There are 2 movements in total.
→+       Cost=2.7
There are 3 movements in total.
→+←      Cost=1.7
Total expansion= 71

The best performance within 10 movements is:
There are 10 movements in total.
→+↑+←↑+←←+         Cost=8.7
Finished in 0.01 seconds.
--------------------Runing Depth First Tree Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
→        Cost=-1.1
There are 2 movements in total.
→↓       Cost=-2.4
There are 3 movements in total.
→↓←      Cost=-3.4
There are 4 movements in total.
→↓←+     Cost=-3.6
Total expansion= 60

The best performance within 10 movements is:
There are 10 movements in total.
→↓←+↓←↓↓←+       Cost=-5.7
Finished in 0.008 seconds.
```

```
--------------------Runing Depth First Graph Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
         Cost=0
There are 1 movements in total.
↓        Cost=-1.3
There are 2 movements in total.
↓→       Cost=-2.4
There are 3 movements in total.
↓→↓      Cost=-3.7
There are 4 movements in total.
↓→↓←         Cost=-4.7
Total expansion= 26

The best performance within 10 movements is:
There are 9 movements in total.
↓→↓←←←+↑+         Cost=-0.3
Finished in 0.009 seconds.
```

Copied from the console (in case of letter encoding issues)

```
--------------------Test Case 1-----------------------
--------------------Runing Uniform Cost Tree Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
            Cost=0
There are 1 movements in total.
-           Cost=0
There are 2 movements in total.
--          Cost=0
There are 3 movements in total.
---         Cost=0
There are 4 movements in total.
----        Cost=0
Total expansion= 51

The best performance within 10 movements is:
There are 10 movements in total.
----------          Cost=0
Finished in 0.003 seconds.
--------------------Runing Uniform Cost Graph Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
            Cost=0
There are 1 movements in total.
←           Cost=-1
There are 1 movements in total.
→           Cost=-1.1
There are 1 movements in total.
↑           Cost=-1.2
There are 1 movements in total.
↓           Cost=-1.3
Total expansion= 52

The best performance within 10 movements is:
There are 9 movements in total.
←↑+↑+→→→+   Cost=4.7
Finished in 0.008 seconds.
--------------------Runing Depth First Tree Search-----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
            Cost=0
There are 1 movements in total.
←           Cost=-1
```

There are 2 movements in total.
←↑ Cost=-2.2
There are 3 movements in total.
←↑→ Cost=-3.3
There are 4 movements in total.
←↑→×← Cost=-4.3
Total expansion= 60

The best performance within 10 movements is:
There are 9 movements in total.
←↑→×←+-↑-↓ Cost=-3
Finished in 0.01 seconds.
----------Runing Depth First Graph Search----------
The first 5 nodes to be expanded:
There are 0 movements in total.
Cost=0
There are 1 movements in total.
↑ Cost=-1.2
There are 2 movements in total.
↑→ Cost=-2.3
There are 3 movements in total.
↑→⟶ Cost=-3.4
There are 4 movements in total.
↑→⟶⟶ Cost=-4.5
Total expansion= 30

The best performance within 10 movements is:
There are 9 movements in total.
↑→⟶⟶↓+↓←↑ Cost=-5.5
Finished in 0.012 seconds.

----------Test Case 2----------
----------Runing Uniform Cost Tree Search----------
The first 5 nodes to be expanded:
There are 0 movements in total.
Cost=0
There are 1 movements in total.
- Cost=0
There are 2 movements in total.
-- Cost=0
There are 3 movements in total.
--- Cost=0
There are 4 movements in total.

---- Cost=0
Total expansion= 51

The best performance within 10 movements is:
There are 10 movements in total.
---------- Cost=0
Finished in 0.007 seconds.
--------------------Runing Uniform Cost Graph Search----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
  Cost=0
There are 1 movements in total.
← Cost=-1
There are 1 movements in total.
→ Cost=-1.1
There are 2 movements in total.
→+ Cost=2.7
There are 3 movements in total.
→+← Cost=1.7
Total expansion= 71

The best performance within 10 movements is:
There are 10 movements in total.
→+↑+←↑+←←+ Cost=8.7
Finished in 0.01 seconds.
--------------------Runing Depth First Tree Search----------------------
The first 5 nodes to be expanded:
There are 0 movements in total.
  Cost=0
There are 1 movements in total.
→ Cost=-1.1
There are 2 movements in total.
→↓ Cost=-2.4
There are 3 movements in total.
→↓← Cost=-3.4
There are 4 movements in total.
→↓←+ Cost=-3.6
Total expansion= 60

The best performance within 10 movements is:
There are 10 movements in total.
→↓←+↓←↓↓←+ Cost=-5.7
Finished in 0.008 seconds.
--------------------Runing Depth First Graph Search----------------------

The first 5 nodes to be expanded:
There are 0 movements in total.
Cost=0
There are 1 movements in total.
↓ Cost=-1.3
There are 2 movements in total.
↓→ Cost=-2.4
There are 3 movements in total.
↓→↓ Cost=-3.7
There are 4 movements in total.
↓→↓← Cost=-4.7
Total expansion= 26

The best performance within 10 movements is:
There are 9 movements in total.
↓→↓←←←+↑+ Cost=-0.3
Finished in 0.009 seconds.