CS4033/6033 - Artificial Intelligence – Assignment 4 – Blocks World

Joe Ferguson, Brad Greene, Brett Marchese

All Contributed Equally

1. Execution: How to Run the Program

   The program can be run in two ways, using the predicate **solveWithIDS/1**, or **solveWithIDS/3**. The first option can just be ran as "solveWithIDS(Path).", which uses the program defined start and goal states (which can be changed in the code), while the other option you need to provide the start and goal states. These states must be valid, or else the program will fail. They are represented the same way they are shown in the assignment description, [[on, d, a], [clear, d], [on, a, table]], and so on.

2. Notes About Program: Programming and Considerations

   The first note to be considered is that both given predicates, path/2 and connect/2, are not used and instead action() is used. This was because, during development, both path and connect do not allows us to track the moves we are making while action() does. It also allows us to change the names of our rules/moves for easier understanding.

   The second note is that ordered sets, which are build-in for prolog, are used. This allows us to compare via these ordered sets instead of permutations, which results in notYetVisited being changed accordingly, and results in substitute/ not being used.

   The third note is that Rules 2-4 (not 1 as it does not need to be defined in the context of this program), were developed via the given Rule 2 in the assignment description and format statements for output tracking were added. This will showup when running the program as the steps the search algorithm is taking.

   The final change is that depth-first search was somewhat changed and iterative deepening was added. This was because, as said in the, assignment description, it was supposed to be tested first on the Romanian data, and because of the previous changes already mentioned.

3. Results Of 4 Runs:
   Run1
   start([[on, i , table],[on, h, table],[on, g, table], [on, f, table],[on, e, table], [on, d, table],
   [clear, i], [clear, h], [clear, g], [clear, f], [clear, e], [clear, d]]).
   goal([[on, i, h],[on, h, g], [on, g, f], [on, f, e], [on, e, d], [on, d, table], [clear, i]]).

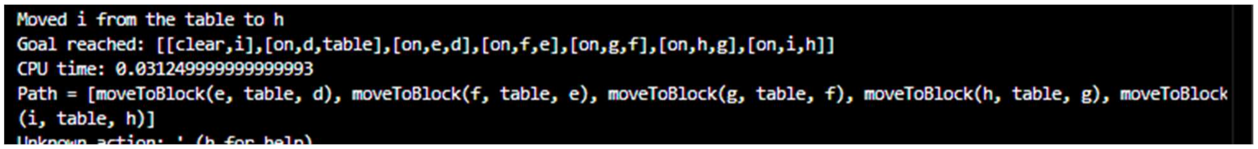
Start:

I	H	G	F	E	D
---	---	---	---	---	---

Goal:

					I
					H
					G
					F
					E
					D
---	---	---	---	---	---

Output:



```
Moved i from the table to h
Goal reached: [[clear,i],[on,d,table],[on,e,d],[on,f,e],[on,g,f],[on,h,g],[on,i,h]]
CPU time: 0.031249999999999993
Path = [moveToBlock(e, table, d), moveToBlock(f, table, e), moveToBlock(g, table, f), moveToBlock(h, table, g), moveToBlock
(i, table, h)]
Unknown action: ' (h for help)
```

Note: This solved it optimally in the least amount of moves

Run2 - Opposite

start([[on, i, h],[on, h, g], [on, g, f], [on, f, e], [on, e, d], [on, d, table], [clear, i]]).
goal([[on, i , table],[on, h, table],[on, g, table], [on, f, table],[on, e, table], [on, d, table],
    [clear, i], [clear, h], [clear, g], [clear, f], [clear, e], [clear, d]]).

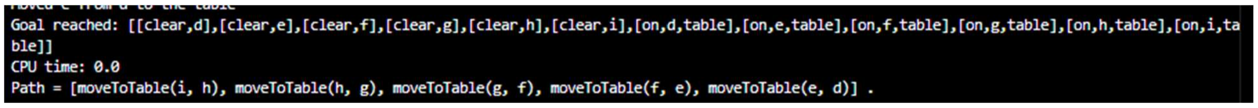
Start:

					I
					H
					G
					F
					E
					D
---	---	---	---	---	---

Goal:

I	H	G	F	E	D
---	---	---	---	---	---

Output:



```
Moved e from d to the table
Goal reached: [[clear,d],[clear,e],[clear,f],[clear,g],[clear,h],[clear,i],[on,d,table],[on,e,table],[on,f,table],[on,g,table],[on,h,table],[on,i,ta
ble]]
CPU time: 0.0
Path = [moveToTable(i, h), moveToTable(h, g), moveToTable(g, f), moveToTable(f, e), moveToTable(e, d)] .
```

Note – This is way faster than the last even though it is the same amount of blocks and moves. This let us know that we are truly doing depth-first search

Run3 – Harder

start([[on, f , d],[on, d, g],[on, g, h], [on, h, table],[on, i, table], [on, k, table], [on, e, table],
    [clear, i], [clear, k], [clear, f], [clear, e]]).
goal([[on, d, f],[on, f, table], [on, i, k], [on, k, table], [on, e, g], [on, g, h], [on, h, table]

,[clear, d],[clear, i], [clear, e]]).

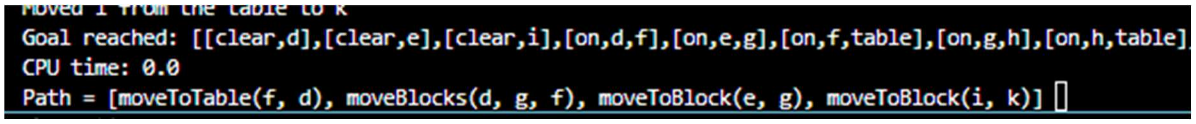
Start:

F					
D					
G					
H		I	K		E
---	---	---	---	---	---

Goal:

		E			
D	I	G			
F	K	H			
---	---	---	---	---	---

Output:



```
Moved i from the table to k
Goal reached: [[clear,d],[clear,e],[clear,i],[on,d,f],[on,e,g],[on,f,table],[on,g,h],[on,h,table]
CPU time: 0.0
Path = [moveToTable(f, d), moveBlocks(d, g, f), moveToBlock(e, g), moveToBlock(i, k)]
```

Note: Even though we added a block, the result was still incredibly fast

Run4: Hardest

start([[on, i , f],[on, f, d],[on, d, g], [on, g, h],[on, h, table], [on, o, k], [on, k, e],
    [on, e, table], [on, a, b], [on, b, c], [on, c, table], [clear, i], [clear, o], [clear, a]]).
goal([[on, h, g],[on, g, d], [on, d, f], [on, f, i], [on, i, table], [on, o, k], [on, k, e]
    ,[on, e , table], [on, a, table], [on, b, table], [on, c, table],[clear, h],[clear, o], [clear, c],
[clear, b], [clear, a]]).

Start:

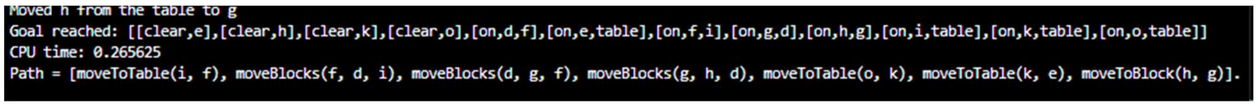
I					
F					
D		O			
G		K			
H		E			
---	---	---	---	---	---

Goal:

H					
G					
D					

F					
I		O	K	E	
---	---	---	---	---	---

Output:



```
Moved h from the table to g
Goal reached: [[clear,e],[clear,h],[clear,k],[clear,o],[on,d,f],[on,e,table],[on,f,i],[on,g,d],[on,h,g],[on,i,table],[on,k,table],[on,o,table]]
CPU time: 0.265625
Path = [moveToTable(i, f), moveBlocks(f, d, i), moveBlocks(d, g, f), moveBlocks(g, h, d), moveToTable(o, k), moveToTable(k, e), moveToBlock(h, g)].
```

Note: Despite the supposed, cpuTime, this took a few minutes. This had the longest amount of moves and the most blocks yet, and this is depth-first, so it is to be expected. Some optimizations that could be made are adding a heuristic and then an informed search like greedy, pruning moves/states that inefficient or that get nothing accomplished, and so on. These optimizations were not defined in the assignment description, just dfs and implementing the problem, so it was not done here.