Project 1 Report

Java:

The Java code is the longest out of all the implementations because the syntax is very verbose. It includes a lot of keywords that just make it very long to code in, so it might be harder for people who are learning the language to remember all the rules. But because there are so many rules and the language is so verbose, it makes it very easy to understand what's going on in the code; what you read is what is actually happening. I also liked how we could assign variables. Variables allowed me to keep track of multiple values in a function and global variables helped me keep track of and compare the best move set. It is very useful having the different scopes in order to compare values in different subroutines.

Implementation of the algorithm was very easy since Java allows for both functional and OO programming, so using the recursive functional approach for the algorithm while representing states and the PegBoard as objects was very easy. The language also made it easy to optimize the algorithm which included DP and caching of states because it included HashMaps.

There wasn't any trouble implementing the algorithm since I was very used it and it was very flexible. The only complain is that it takes a long time to write since the other languages took a fraction of the time.

Haskell:

The Haskell code was very interesting since i've never been exposed to functional programming. Readability is terrible; it's very hard to read what's going on since everything follows a mathematical approach to programming, so people not sufficient in math may have a problem. Writability was very hard because I could not assign many variables to keep track of different function values.

It was rough not being able to use loops and only doing recursion, but list operations are really interesting. List comprehensions let me manipulate the lists very easily and acted like a loop. It's also useful that list comprehensions would create a new list and we wouldn't have to keep track of the old list. Also list operations like `filter` and `map` were very useful

Python:

The Python code is basically a simpler version of the Java implementation. The readability is pretty similar to that of Java, so if you know Java/C/C++ then you can read Python. Writability is a little better than Java since you have the syntactic sugar of Java but you don't need to write as much.

It's very nice how Python is very lightweight and doesn't need to be compiled like Java since it's a scripting language. A small annoyance was not being able to coerce an int to a String

when formatting the output; I had to call str(someInt) to concatenate strings. I was also annoyed by the fact that every function takes in the self object as a parameter.

Prolog:

I thought prolog was the most interesting out of all the languages. The language is very readable since everything is listed as facts and rules and is very English friendly. The language may be easy to write in if the program is not complicated; the hardest but most important concept was variables since a variable could be assigned as multiple values.

The most interesting feature that helped in implementing the algorithm was the automatic backtracking. This automatic backtracking gets rid of loops and basically will assign a variable to every possible value, so reaching every state of a board was very easy. Another interesting feature was the omission of if statements since prolog will keep traversing through rules whether the rules are true or false. Another very useful feature was how every rule could basically return multiple variables. Since every parameter variable could be either input or output, it was very useful when I had to return tuples of values such as the state of a board and the number of moves.

The hardest thing was actually following the code because of the backtracking; because rules will rely on other rules and almost 90% of the rules take in variables, it is very hard to keep track of which values are being passed into which rules. Not being able to access directly into a list was a problem since we had to update the 2-d array.

All the languages had a way to implement recursion, so the algorithm was not a problem. The problem was representing the PegBoard which would change my algorithm up. Python and Java represented the board as a 2-D triangle array, Prolog also represented as a 2-D array, but Haskell was represented as a list of coordinates which takes functionality of Filter, Map and the many other features of list operations. Accessing the lists for Prolog were a pain since I had to recursively call a function to get the correct rows and columns. Java made the most sense for me since it was as if I were using a real board; however half the code is the board implementation whereas Prolog is just a 2-D array and Haskell is a list of coordinates.

We cannot compare solutions since each solution, except for java and python, did not implement the algorithm the same. With a more efficient optimization, I would think prolog would have the fastest solution. Prolog was the only language where the algorithm takes in a number of allowed moves instead of searching through all movesets and recording the smallest move set. However, my implementation makes Prolog search through the same states multiple times, so it is still slow.