Lab 1 Analysis - Noboru Hayashi

I. Data Structure

In this Lab assignment, I defined a Node class and used it to implement a Stack class as a linked list. So all stack operations such as pop, push and peek are designed to manipulate the node class 'Head' inside the stack data structure.

The node class contains fields: data (Character) and next (Node), so by using this class, the linked list can be implemented with a pointer towards the next Node object. Hence inside the stack class, the field is : head (Node), and having isEmpty(), push(), pop() and peek() methods. (* Please see source codes Node.java, Stack.java)

For examining the given string is matching specific pattern of languages, such as the string is of the form of A^nB^n, I took approaches that check if the stack is empty after series of push and pop operations: if the character read from the left is A, the element is kept pushing into the stack, and if the next character is B then start to pop elements at each B is read, until no character left. Finally checking if the stack is empty or not to examine the string follows the pattern of the language. Also the LIFO characteristic allows us to compare the character that has been read previously with the current character. In this case, the space and time complexity would be O(n), since the method reads the entire length of the string, and half of the string is pushed into the stack.

Therefore, the linked list implementation for stack is good enough since I don't need any extra helper operation such as random access which would be available if the stack is implemented with the array. Also the linked list implementation accomplishes dynamic data size.

For some other cases such as L1, I used two stacks that each store character A or B in LIFO manner, and after the entire string is read, popping an element until one of two stacks is empty, and then check whether both stacks are empty or not. In this situation, the time and space complexity would be O(n) as well, because the string is linearly scanned, and all characters are stored once into the two separate stacks.

II. Iteration vs Recursion

Since the push and pop operations are iteratively run inside while loops, and its condition is checking both the character index and the next character is 'A' or 'B', so once the program exits that loop, it represents the all string characters are read or the program has detected non-A and non-B alphabets.

In the other hand recursive approach could be implemented with following algorithm:

1. Assume the program is evaluate the string is of the form of L2: AnBn for some n>0
2. We can define a helper method that takes the first and the last index of the string (i = 0, j = str.length() -1), then the method checks the character at the first and last index is 'A' and 'B' accordingly
3. Then this helper method recursively calls itself with the arguments of the indices shifted towards the middle of the string (i++, j--). If the condition (str[i] == 'A' && str[j] == 'B') fails within the

recursive operations, the method will return false, otherwise once the indices passes the midpoint (i>= j) then return true.

However, for L1 and L4 the string might not be symmetrical, so this recursive approach would not work, or needs special handlings. So I think the stack implementation would be more helpful in these cases.

III. Enhancement

This lab assignment requires to choose non-trivial languages for L5 and L6, so I chose the L5= { w: w is of the form BnAn for some n>0 } and L6 = { w: w contains equal numbers of A's, B's and C's (in any order) and no other characters}. The methods to evaluate these language forms can be extended from one for L1 and L2.

IV. Conclusion

In this lab assignment, I learned how to utilize a linked list class to implement the stack data structure, in order to accomplish complex operations like string pattern detections with one pass. However, for the L4 evaluator I examined the number of A’s and B’s in the first substring to determine the pattern, n and m for (AnBm)p specifically, I might try to figure out if there is any other way to examine the pattern without courting next time.