Dania Etienne Stacks for Regular Languages 06/22/2018

Project Commentary

//Commentary:

//Discuss your data structures and their implementation and why they make sense.

//Why is a stack a reasonable choice to solve this problem?

//What implementation of a stack did you choose? Why?

//Include the computational complexity of the operations in your code.

In this project, we are not overwhelmingly concerned with memory or speed. So, there is not too much difference between using a linked list and an array. Although an array would have been much easier to implement.

An array based stack typically provides faster search speed, an easier time inserting, uses less memory, and allows for random access. But none of these were pressing concerns for this project. A concern in this project was the size of the array. We were in the dark about the size of the array that would be given but we could get around this by using a dynamic array or a vector. (Typically creating a new element would take O (1) amortized time and worst case O(n) time).

If speed or random access was a pressing concern, an array would hand down be the best option. Arrays are stored in a contiguous block in memory versus linked lists which are stored wherever in memory. So, an array would have the locality of reference advantage even with the cost of resizing.

Besides that, the only small (or possibly large) draw back with using a linked list would be the memory required for references. With a linked list, you would be using extra memory to store a pointer or two. But with a linked list allocating a set size is not a concern. In the end, I decided to gain a better understanding of a linked list stack by implement one myself. I did not see a big advantage over using a double linked list over a single linked list. I used a singly linked list. In L3-L4 I was using more than one stack but the other stacks would not have been any bigger than the one “original” stack so again there wasn’t much of an advantage in terms of size but there was slight disadvantage in terms of memory.

Moving on, a stack was a great overall choice to solve this project. Stacks are usually used in language processing. For example, a compiler’s syntax check for matching braces is implemented by using a stack. Another example would be undo in a word processor. Implementing a stack was simple compared to a queue implementation. In this project, specifically we are taking advantage of the stacks LIFO.

Finally, all the operations we used for the stack implementation were insertion and deletion which have worst case computational complexity of O (1). Access and Search would have had a worst case of O(n) but we did not need these operations. In L1-L4 operations we are going through the entire string and we have a computational complexity of O(n).