Ackermann

Christopher Rose: Assignment 04b

04/03/2020

1 Problem

We were given an assignment to implement a stack using either an Array list or a Linked list. Then we used a stack to implement Ackermann's function non-recursively. Once we have it solved then we are supposed to record the data.

2 Hypothesis

The time cost of the non-recursive function is determined by a while loop that keeps putting data on the stack depending on the input. I hypothesize that the run time for the non-recursive function is $\Theta(logn)$ where n is the first input. The time cost of the recursive function is $\Theta(n)$ where n is the first input. Therefore, the run time for the recursive function is smaller than the non-recursive function.

3 Methods

```
public static long ackermann(long m, long n) {
   Stack<Long> s = new ackermann<Long>();
   s.push(m);
   while(s.length() > 0) {
       m=s.pop();
       if(m==0) {
           n++;
       } else if (n==0) {
           s.push(--m);
           n++;
       } else {
           s.push(--m);
           s.push(++m);
           n--;
       }
   }
   return n;
}
```

For this method I used a while loop to check each element in the stack until there were no elements left. In the while loop I made an if statement that checked for the cases of m=0 or n=0. If these cases were true, then n would go up by 1 and the appropriate m would be pushed onto the stack. If they were not true then the value of n would be reduced by 1 and 2 more m values onto the stack. For data structures we were told to implement our own stack, which I did using an Array list. I then used the stack to hold the value of m. For the experiments we used 15 sets of variables given to us to test the run time of our code.

4 Results and Discussion

For the first experiment, I used the sets of variables to test the run time of the recursive function.

m value	n value	answer	runtime
0	0	1	1
1	0	2	1
1	10	12	1
1	20	22	0
2	0	3	1
2	10	23	0
2	20	43	1
3	1	13	1
3	2	29	0
3	3	61	1
3	4	125	0
3	5	253	1
3	6	509	2
3	7	1021	5
3	8	2045	11

For the second experiment, I used the same sets to test the non-recursive function.

m value	n value	answer	runtime
0	0	1	3
1	0	2	4
1	10	12	2
1	20	22	3
2	0	3	2
2	10	23	2
2	20	43	4
3	1	13	2
3	2	29	2
3	3	61	3
3	4	125	5
3	5	253	12
3	6	509	19
3	7	1021	38
3	8	2045	48

For the recursive function the results indicate that the run time is $\Theta(logn)$. For the non-recursive function the run time can be defined as $\Theta(nlogn)$.

5 Conclusion

In conclusion, according to the results my hypothesis was partially wrong and partially right. The run times were different from what I expected, but I was right about recursion being the faster option.