CSCI 2300
Introduction to Algorithms: HW1

Sean Hyde
RIN: 662096071

DUE: September 5th, 2024

1 Question 1

`Solution`:

When `foo(n)` is called, it checks if $n = 0$; if true, it ends the program without printing "*". Otherwise, it enters a loop, printing "*" n times, from 0 to $n-1$, ensuring we count n since the index starts at 0. Then it recursively calls `foo(n-1)`. Therefore, $T(n) = n + T(n-1)$, where n corresponds to the number of "*" printed in the first loop, plus the recursive call for `foo(n-1)`.

$$\texttt{Expressing } T(n) = n + T(n-1) \texttt{ in terms of } n:$$

$$n + T(n-1) + T(n-2) + T(n-3) + T(n-4)\ldots \text{ until we reach } T(0).$$

1.1 Answer

In terms of only n:

$$\frac{n(n+1)}{2}$$

2 Question 2

When `bar(n)` is called, it immediately prints "*". It then checks if $n = 0$; if true, it ends the program. Otherwise, it enters a loop from 0 to $n-1$, recursively calling `bar(i)` for each i.

2.1 Step-by-Step

```
Start with n = 1
- It Prints a *
- Checking if n equal to 0; false
- For i = 0 to n-1:
    - it will call bar(i) in this case bar(0)
      - It starts over and prints a *
      - Check if n equals 0; true
      - End recursion
Result: n = 1, * printed 2 times

Start with n = 2:
- It prints a *
- Check if n == 0; false
- For i = 0 to n-1:
    - Call bar(i) in this case bar(0):
      - Prints a *
      - Check if n == 0 -> true
      - End recursion
i = 1 now and does bar(1) and we already know that prints 2 stars, so
Result: n = 2, "*" printed 4 times
```

`Solution`: Based on this, I can conclude:

$$T(n) = n^2$$

The number of "*" printed grows exponentially. For $n = 3$, 8 stars are printed, and so on.

3 Question 3

`Explanation:`

```
Check each value of i in the range of A:
  - If i modulos 2 is equal to 0:
      - i is even
      - Multiply it by any number and the result is even
  - Assign even i values to a list and return the list
```

`Mathematical Notation:`
For each i in A:

$$O(A)$$

Check if $i \mod 2 = 0$ and assign i to a list:

$$+1$$

`Final Equation:`

$$O(A) + 1$$

`Or if "+ 1" is trivial:`

$$O(A)$$

4 Question 4

The following table shows the results from `fib1` and `fib2`, including the index (n), Fibonacci number, time in seconds for `fib1`, and the average time for `fib2`.

Index (n)	Fibonacci Number F(n)	Time in Sec for "fib1"	Avg. Time Per Execution for "fib2"	Avg Time Per Execution for "fib2" in Secs
1	1	0	1.384E-07	0.000000138400
2	1	0	1.684E-07	0.000000168400
3	2	0	1.906E-07	0.000000190600
4	3	0	2.120E-07	0.000000212000
5	5	0	2.361E-07	0.000000236100
6	8	0	2.736E-07	0.000000273600
7	13	0	2.945E-07	0.000000294500
8	21	0	3.182E-07	0.000000318200
9	34	0	3.376E-07	0.000000337600
10	55	0	3.604E-07	0.000000360400
11	89	0	3.804E-07	0.000000380400
12	144	0	4.034E-07	0.000000403400
13	233	0	4.257E-07	0.000000425700
14	377	0	4.552E-07	0.000000455200
15	610	0	4.726E-07	0.000000472600
16	987	0	5.077E-07	0.000000507700
17	1597	0	5.397E-07	0.000000539700
18	2584	0	5.762E-07	0.000000576200
19	4181	0	6.126E-07	0.000000612600
20	6765	0	6.329E-07	0.000000632900
21	10946	0	6.606E-07	0.000000660600
22	17711	0	6.888E-07	0.000000688800
23	28657	0	7.145E-07	0.000000714500
24	46368	0	7.479E-07	0.000000747900
25	75025	0.01	7.750E-07	0.000000775000
26	121393	0.01	8.031E-07	0.000000803100
27	196418	0.02	8.367E-07	0.000000836700
28	317811	0.02	8.599E-07	0.000000859900
29	514229	0.04	8.910E-07	0.000000891000
30	832040	0.06	9.126E-07	0.000000912600
31	1346269	0.1	9.375E-07	0.000000937500
32	2178309	0.17	9.721E-07	0.000000972100
33	3524788	0.27	1.002E-06	0.000001002000
34	5702887	0.44	1.030E-06	0.000001030000
35	9227465	0.71	1.052E-06	0.000001052000
36	14930352	1.13	1.072E-06	0.000001072000
37	24157817	1.83	1.101E-06	0.000001101000
38	39088169	2.96	1.124E-06	0.000001124000
39	63245986	4.81	1.151E-06	0.000001151000
40	102334155	7.87	1.178E-06	0.000001178000
41	165580141	12.82	1.206E-06	0.000001206000
42	267914296	20.76	1.240E-06	0.000001240000
43	433494437	33.63	1.267E-06	0.000001267000

The following scatter plots compare the performance and visual differences between `fib1` (recursive) and `fib2` (linear time) functions:

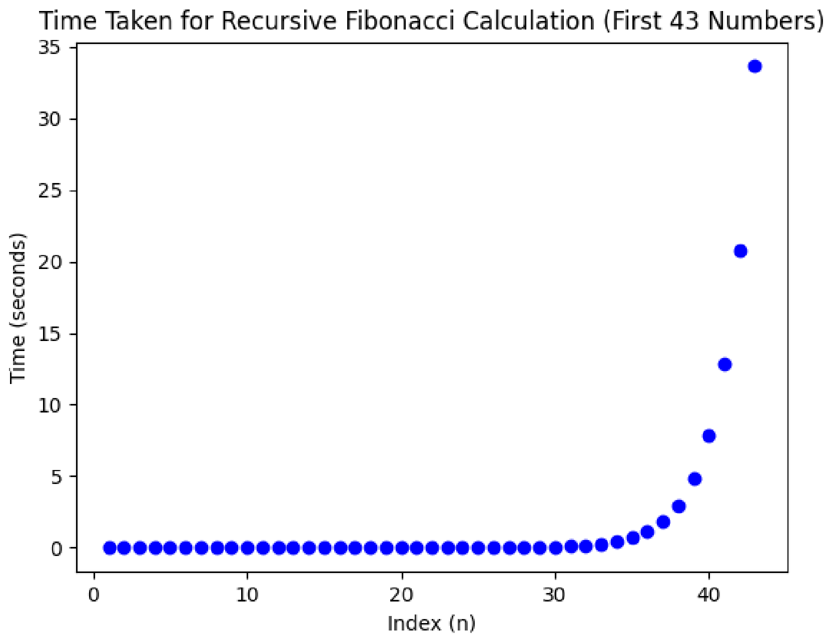

Figure 1: Scatter plot of `fib1` (recursive)

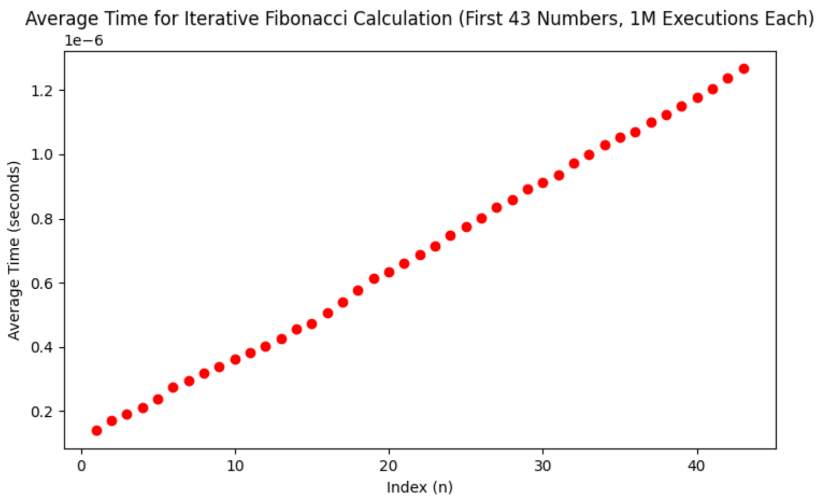

Figure 2: Scatter plot of `fib2` (linear time)