Project 2: Big-O Notation

CSE3333

The University of Texas Rio Grande Valley

Spring 2018

Dr. Liyu Zhang

Juan Bermudez

September 24, 2018

The program can be run on visual studio. The zip file contains the project 2 and the RunningTimes.cpp file contains the code for the program. The code can be executed by pressing ctrl+ F5.

The comments in the code describe the functions. When the program is compiled and executed, a menu will appear. The user will then type a number from 1 -8, and then press enter. The size of the recursion will then be prompted in another menu, where the user will have to type a number then press enter again so the results will be displayed.

This program will compute the code segments in running time notation using Big-O. The segments to be analyzed will be the following:

a) The for loop in the 2nd line of code executes for a total of n number of times. This makes the running time for the code fragment f(N) = O(N). The function name is One_linear

```
 sum = 0;
for (int i = 0; i < n; ++i)
    ++sum;
```

b) The for loop in the inner and outer loop are multiplied together, n * n, making the running time for the code fragment f(N) = $O(N^2)$. The function name is Two_quadratic

```
  sum = 0;
for (int i = 0; i < n; ++i)
    for (int j = 0; j < n; ++j)
        ++sum;
```

c) The inner for loop is multiplied with the outer one, making the running time n^2*n, f(N)= $O(N^3)$. The function name is Three_cubic:

```
 sum = 0;
for (int i = 0; i < n; ++i)
    for (int j = 0; j < n * n; ++j)
        ++sum;
```

d) The inner loop is of θ(i), proportional to $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$, this will make the running time f(N) = $O(N^2)$. The function name is Four_quadratic

```
 sum = 0;
for (int i = 0; i < n; ++i)
    for (int j = 0; j < i; ++j)
        ++sum;
```

e) The inner most loop executes for i^4 * the outer loop, making the running time f(N) = $O(N^5)$ The function name is Five

```
 sum = 0;
for (int i = 0; i < n; ++i)
    for (int j = 0; j < i * i; ++j)
        for (int k = 0; k < j; ++k)
            ++sum;
```

f) The running time for this statement will be f(N) = O(N^4). The function name is Six

```
 sum = 0;
for (int i = 1; i < n; ++i)
    for (int j = 1; j < i * i; ++j)
        if (j % i == 0)
            for (int k = 0; k < j; ++k)
                ++sum;
```

g) This statement is the “regular” way of calculating exponents. The result will multiply by itself based on the number of iterations, the block statement has a running time f(N) = O(N). The function name is Exponentiation

```
  for (int i = 1; i <= n; i++)
  {

     result = result*x;

  }

  return result;
```

h) This statement uses the power function. It is the fast exponential. Because the inner for loop cuts each statement by half, the running time is f(N) = O(log N). The function name is FastExponentiation

```
if (n == 0)
    return 1;
if (n == 1)
    return x;
if (isEven(n))
    return pow(x * x, n / 2);
else
    return pow(x * x, n / 2);
```

The program opens with a menu where the user can select a recursive function to calculate the running time, the user can then choose n, the number of iterations N in 10^2, 10^3, and 10^4:

```
Choose a recursive function to calculate it's running time
1. This recursive function is linear of running time f(N) = O(N)
2. This recursive function is of quadratic of running time f(N) = O(N^2)
3. This recursive function is of cubic of running time f(N) = O(N^3)
4. This recursive function is of quadratic of running time f(N) = O(N^2)
5. This recursive function is of running time f(N) = O(N^5)
6. This recursive function is of running time f(N) = O(N^6)
7. This is a recursive function for regular exponentiation of type f(N)= O(N)
8. This is a recursive function for fast exponentiation of running time f(N)= O(log N)
1
What size of n do you want?
1. 100
2. 1,000
3. 10,000
```

The following screen shots are in order of the trial on the menu, each trial will have the 3 screen shots for the number of iterations N, when N is 10^2, 10^3, and 10^4.

Running times for a)

```
The sum is 1000
The result of the exponentiation was: 0
The function took 3 milliseconds
Press any key to continue . . .
```

```
The sum is 10000
The result of the exponentiation was: 0
The function took 2 milliseconds
Press any key to continue . . .
```

Running times for b)

```
The sum is 10000
The result of the exponentiation was: 0
The function took 2 milliseconds
Press any key to continue . . .
```

```
The sum is 1000000
The result of the exponentiation was: 0
The function took 5 milliseconds
Press any key to continue . . .
```

```
The sum is 100000000
The result of the exponentiation was: 0
The function took 244 milliseconds
Press any key to continue . . .
```

Running times for c)

```
The sum is 1000000
The result of the exponentiation was: 0
The function took 6 milliseconds
Press any key to continue . . .
```

```
The sum is 1000000000
The result of the exponentiation was: 0
The function took 2369 milliseconds
Press any key to continue . . .
```

Running time for 10^4 took to long to appear.

Running time for d)

```
The sum is 4950
The result of the exponentiation was: 0
The function took 2 milliseconds
```

```
The sum is 499500
The result of the exponentiation was: 0
The function took 4 milliseconds
```

```
The sum is 49995000
The result of the exponentiation was: 0
The function took 106 milliseconds
```

Running time for e)

```
The sum is 975002490
The result of the exponentiation was: 0
The function took 2120 milliseconds
```

Running time for 10^3 and 10^4 took to long to calculate

Running time for f)

```
The sum is 12087075
The result of the exponentiation was: 0
The function took 32 milliseconds
```

Running times for 10^3 and 10^4 took to long to calculate

Running time for g)

```
The sum is 0
The result of the exponentiation was: 5.15378e+47
The function took 117 milliseconds
```

```
The sum is 0
The result of the exponentiation was: inf
The function took 3 milliseconds
```

Running time for 10^4 was also infinity

Running time for h)

```
The sum is 0
The result of the exponentiation was: 5.15378e+47
The function took 10 milliseconds
```

```
The sum is 0
The result of the exponentiation was: inf
The function took 2 milliseconds
```

```
The sum is 0
The result of the exponentiation was: inf
The function took 3 milliseconds
```

The tables below will compare the running times between the code fragments in 2.7 and the power functions

Table 2.7

The output time for the algorithm is in milliseconds.

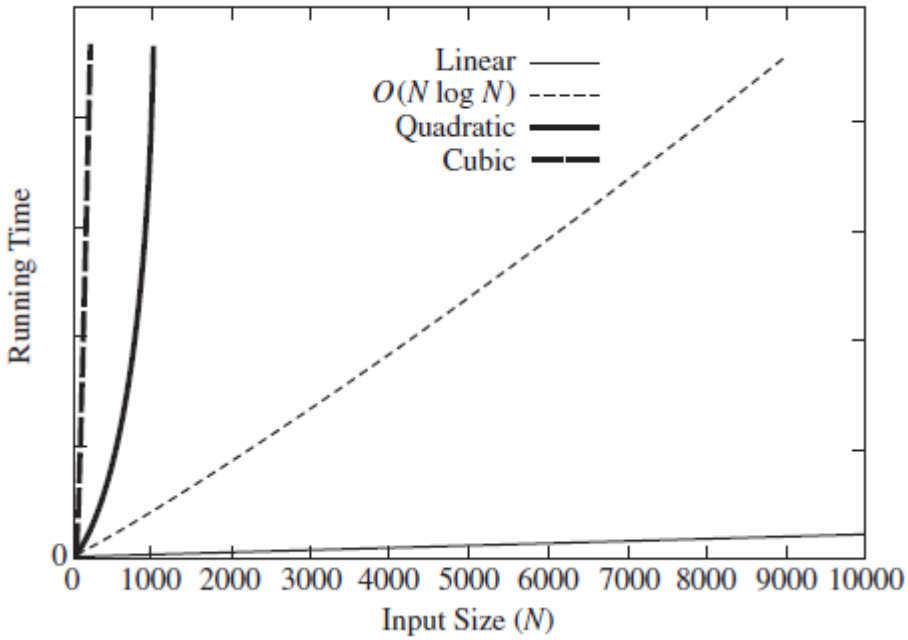
Input Size	One_linear O(N) a)	Two_quadratic $O(N^2)$ b)	Three_cubic $O(N^3)$ c)	Four_quadratic $O(N^2)$ d)	Five $O(N^5)$ e)	Six $O(N^4)$ f)
$N = 10^2$	3	2 milli sec	6	2	2120	32
$N = 10^3$	2	5 milli sec	2369	4	NA	NA
$N = 10^4$	2	244 milli sec	NA	106	NA	NA

The table for the power functions, the output time for the algorithms was 4 x 3

Input Size	Exponentiation O(N) g)	FastExponentiation (log N) h)
$N = 10^2$	117	10
$N = 10^3$	3 (infinity calculation)	2 (infinity calculation)
$N = 10^4$	3 (infinity calculation)	2 (infinity calculation)

Conclusion:

The running times show to be consistent with the sum of the outputs and they seem to output the function behaviors. This may be explained by the following graphs:



	Algorithm Time			
Input Size	1 $O(N^3)$	2 $O(N^2)$	3 $O(N \log N)$	4 $O(N)$
$N = 100$	0.000159	0.000006	0.000005	0.000002
$N = 1{,}000$	0.095857	0.000371	0.000060	0.000022
$N = 10{,}000$	86.67	0.033322	0.000619	0.000222
$N = 100{,}000$	NA	3.33	0.006700	0.002205
$N = 1{,}000{,}000$	NA	NA	0.074870	0.022711

The statements that were of linear time were the regular Exponentiation function and One_linear. Their behavior is of linear running time, even if the output is different. Similar to table 2.7, the results are of running time $O(N^3)$, $O(N^4)$, and $O(N^5)$ will have a NA on the time outputted by the function because the numbers become to large to calculate in real time. Like the graph above, these values showed NA on the program when it was run. Thus, if the results from the program are compared with the theoretical results from the graph above, then the results seem to be consistent enough with the theoretical analysis.