SAN FRANCISCO STATE UNIVERSITY
Computer Science Department

CSC510 Section 04 – Analysis of Algorithms
Algorithm Challenge 1: Complexity Functions

Instructor: Jose Ortiz

08/04/2020

Full Name: Robert Cacho Ruiz
Student ID: 917624960

Assignment Instructions. Must read!

Note: Failure to follow the following instructions in detail will impact your grade negatively.

1. This algorithm challenge is worth 10%, and will be graded using a grading point scale where the maximum possible grade is 100 points. For instance, if your grade in this assignment is 85/100, then this is equivalent to 0.85*10%=8.5% of 10%
2. The deadline of this assignment will be announced by the instructor in class.
3. Each section of this algorithm challenge is worth 25 points
4. Take into account that in this type of assignments, I am more interested in the way you approach the problem rather than your final solution.

Problem Statement

1. Create an optimized function "print s(n,s)" that prints the given argument **s** (representing a string) **k** times. **k** represents the number of iterations of **'for j'** for each iteration of **i** based on the increments $i = i + 1$ and $j = j * 2$

   (a) Initial conditions: $i = 1$, $j = 1$, $i <= n$, and $j <= i$

   (b) input as arguments in the function: n (an integer representing the size of the input), and s (the string)

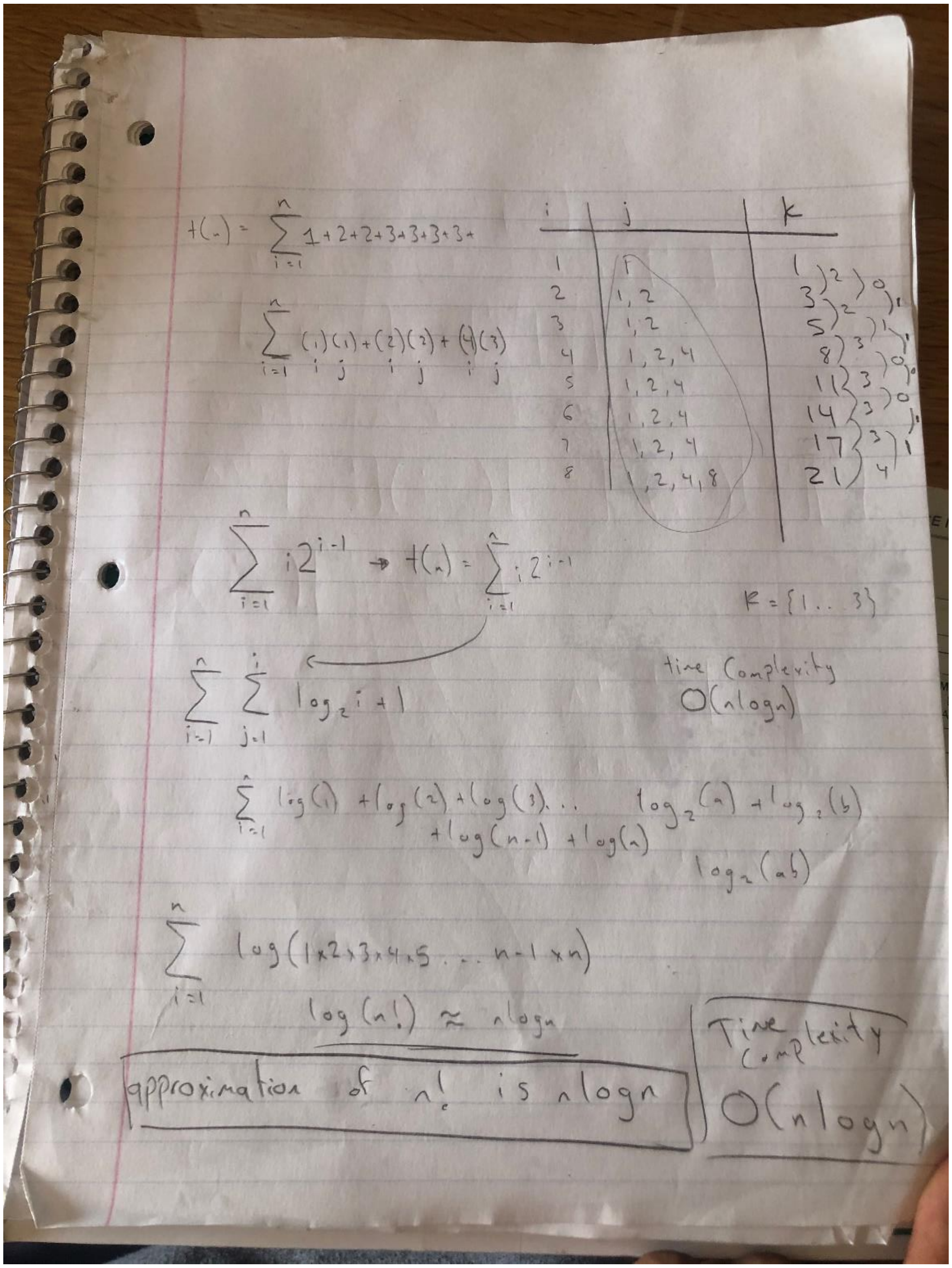
   (c) output: print s k times

   (d) example: n=5, s="hello CSC510-01 class", s will be printed 11 times

1

Your work here

1. Describe the algorithm to solve the problem. Use n=5 as your base example, and then based on your solution, define a general algorithm for all the values of n. Finally, state the complexity of your algorithm as (1) a function of T(n), and (2) time complexity with big O notation



$t(n) = \sum_{i=1}^{n} 1+2+2+3+3+3+3+$

$\sum_{i=1}^{n} \underset{i}{(1)}\underset{j}{(1)} + \underset{i}{(2)}\underset{j}{(2)} + \underset{i}{(4)}\underset{j}{(3)}$

i	j	k		
1	1	1	2	0
2	1, 2	3	2	1
3	1, 2	5	3	1
4	1, 2, 4	8	3	0
5	1, 2, 4	11	3	0
6	1, 2, 4	14	3	1
7	1, 2, 4	17	3	1
8	1, 2, 4, 8	21	4	

$\sum_{i=1}^{n} i2^{i-1} \rightarrow t(n) = \sum_{i=1}^{n} i2^{i-1}$

$k = \{1 .. 3\}$

$\sum_{i=1}^{n} \sum_{j=1}^{i} \log_2 i + 1$

time Complexity

$O(n\log n)$

$\sum_{i=1}^{n} \log(1) + \log(2) + \log(3) \ldots + \log(n-1) + \log(n)$

$\log_2(a) + \log_2(b)$

$\log_2(ab)$

$\sum_{i=1}^{n} \log(1 \times 2 \times 3 \times 4 \times 5 \ldots n-1 \times n)$

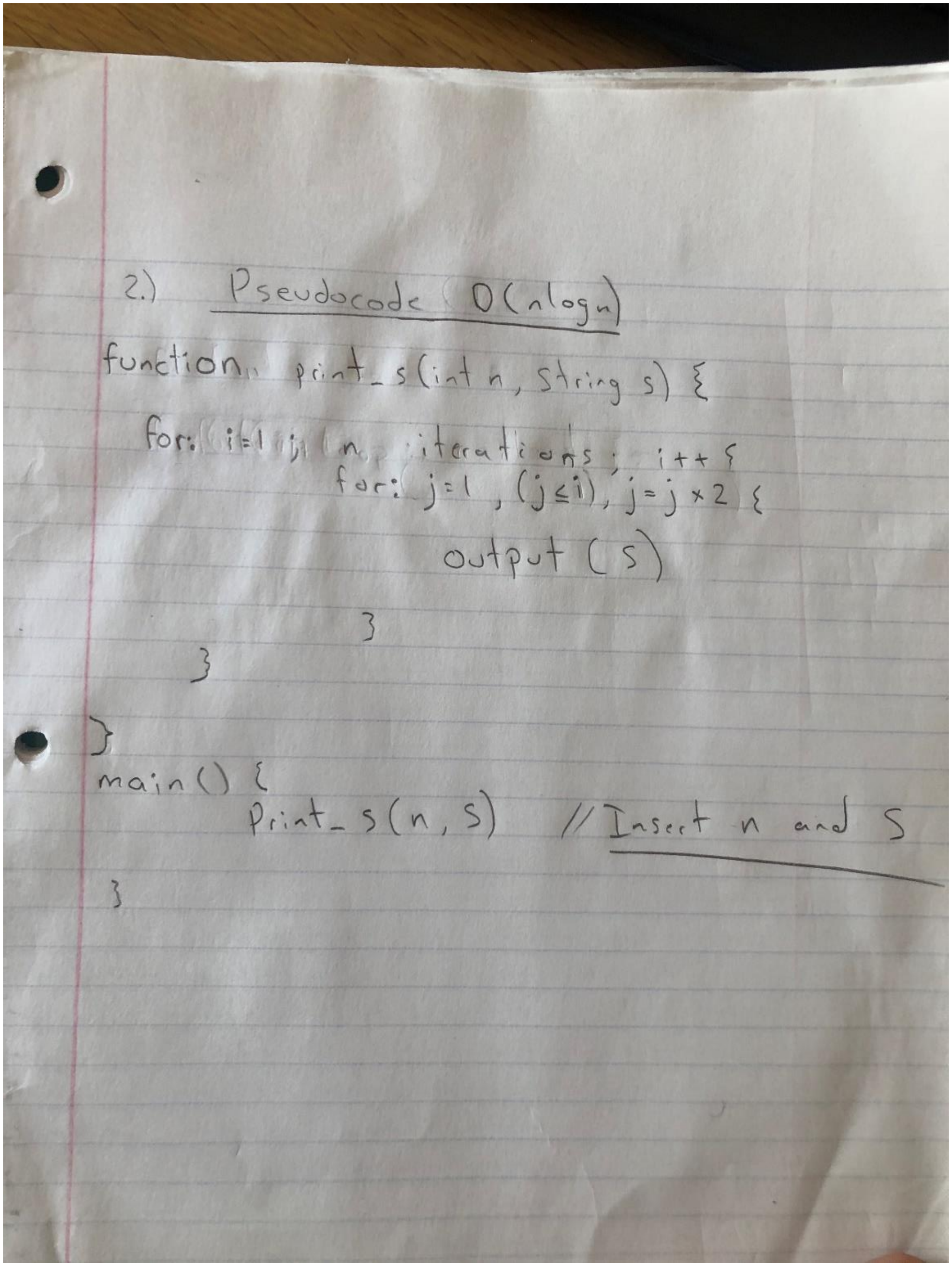
$\log(n!) \approx n\log n$

approximation of n! is nlogn

Time Complexity

$O(n\log n)$

2. Write the pseudocode to that defines the algorithm in part (1) for $T(n)$



2.) Pseudocode O(nlogn)

```
function print_s(int n, String s) {
  for: i=1 ; n iterations ; i++ {
              for: j=1 , (j≤i), j=j×2 {
                        output ( s)

              }
      }
}
main() {
           Print_s(n, S)    //Insert n and S

 }
```

3. Provide an optimization for the pseudocode in part (2). Note that there is always a way to optimize your algorithm. I want you to think hard about this.

I believe the best way to optimize this is to calculate T(n) then run a loop T(n) amount of times. This makes the code O(n) because there are no nested for loops or anything else.
The first thing to do in the function print_s(int n, string n) is to set a variable equal to T(n)

```
int ans = <formula to calculate T(n)>
```

After we set a variable equal to the amount of T(n) we run the loop "ans" amount of times

```
for(int i=0; i<=ans; i++)
  cout << s << endl;
```

This should make it O(n)

4. Create/implement the method based on your work above and provide several unit test for your (optimized)solution

I went to a study group for CSC 510 and I told them my thinking, they thought it was wrong, they were looking to solve with if statements etc.. Overall, I left the study group and tried finding the formula, took me hours but eventually I think I got it right.

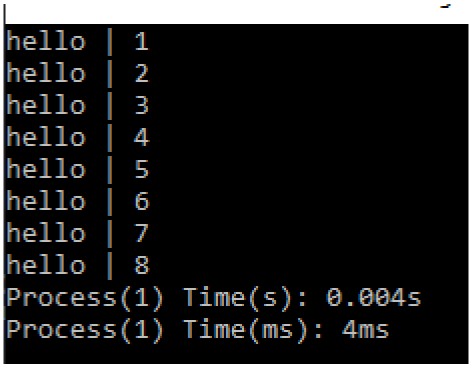
K related to n is 2^(k-1)

T(n) = N * Log2n – 2^(log2n) + 1

Since I and j = 1 to start off in the loop I adjusted each n to (n+1)

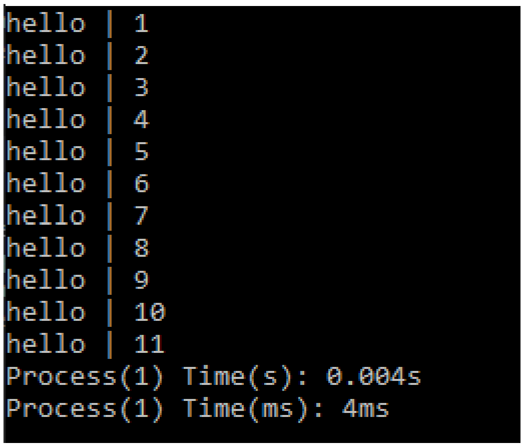
```
#include <iostream>
using namespace std;

void print_s(int n, string s) {
    int ans = (((n + 1) * ceil(log2(n + 1))) - pow(2, ceil(log2(n + 1))) + 1);

    for (int i = 1; i <= ans; i++)
           cout << s << " | " << i << endl;
    }

int main(){
    print_s(10, "hello");
}
```

N = 4;



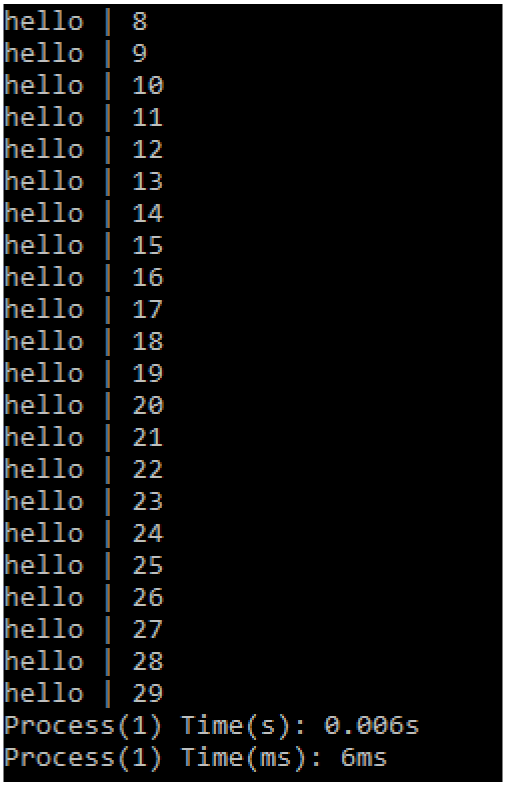
```
hello | 1
hello | 2
hello | 3
hello | 4
hello | 5
hello | 6
hello | 7
hello | 8
Process(1) Time(s): 0.004s
Process(1) Time(ms): 4ms
```

N=5;



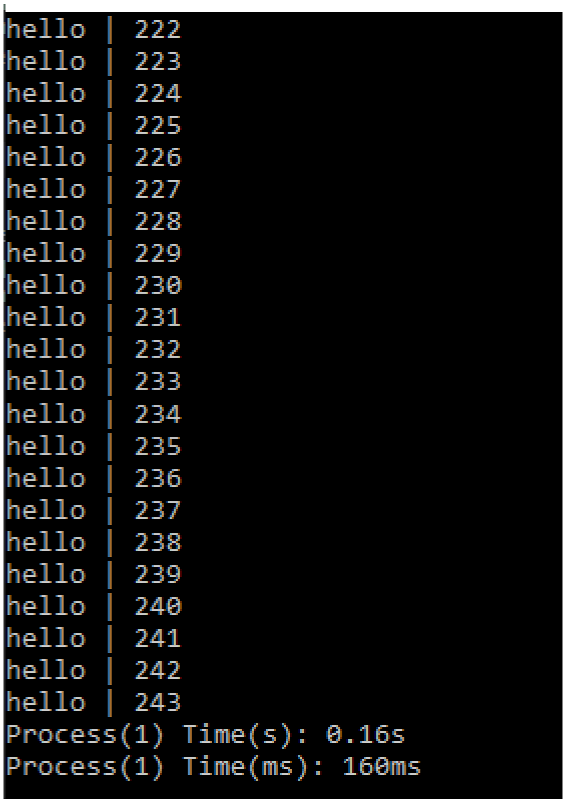
```
hello | 1
hello | 2
hello | 3
hello | 4
hello | 5
hello | 6
hello | 7
hello | 8
hello | 9
hello | 10
hello | 11
Process(1) Time(s): 0.004s
Process(1) Time(ms): 4ms
```

N=10;



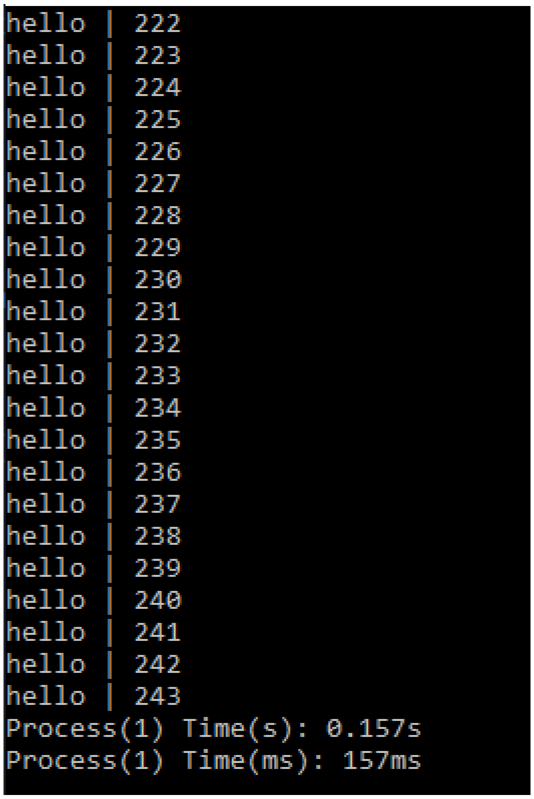
```
hello | 8
hello | 9
hello | 10
hello | 11
hello | 12
hello | 13
hello | 14
hello | 15
hello | 16
hello | 17
hello | 18
hello | 19
hello | 20
hello | 21
hello | 22
hello | 23
hello | 24
hello | 25
hello | 26
hello | 27
hello | 28
hello | 29
Process(1) Time(s): 0.006s
Process(1) Time(ms): 6ms
```

N=50;



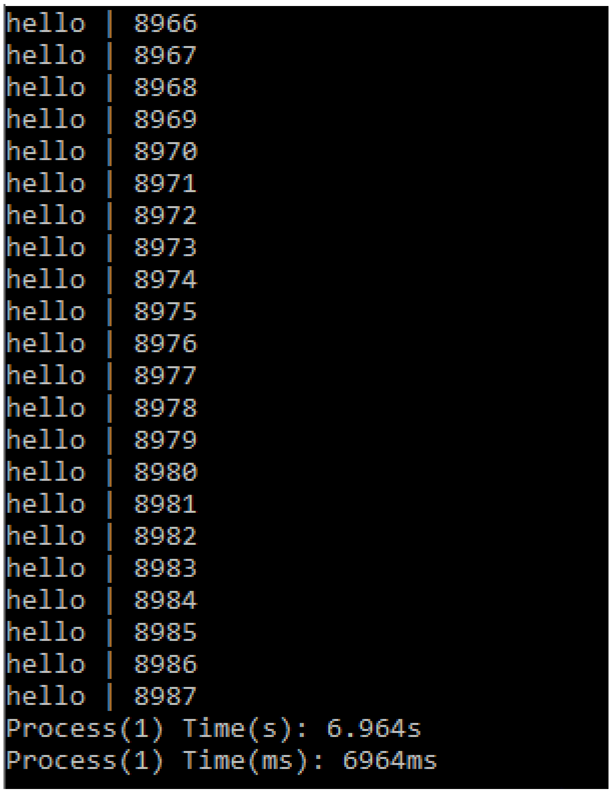
```
hello | 222
hello | 223
hello | 224
hello | 225
hello | 226
hello | 227
hello | 228
hello | 229
hello | 230
hello | 231
hello | 232
hello | 233
hello | 234
hello | 235
hello | 236
hello | 237
hello | 238
hello | 239
hello | 240
hello | 241
hello | 242
hello | 243
Process(1) Time(s): 0.16s
Process(1) Time(ms): 160ms
```

N=100;



```
hello | 222
hello | 223
hello | 224
hello | 225
hello | 226
hello | 227
hello | 228
hello | 229
hello | 230
hello | 231
hello | 232
hello | 233
hello | 234
hello | 235
hello | 236
hello | 237
hello | 238
hello | 239
hello | 240
hello | 241
hello | 242
hello | 243
Process(1) Time(s): 0.157s
Process(1) Time(ms): 157ms
```

N=1000



```
hello | 8966
hello | 8967
hello | 8968
hello | 8969
hello | 8970
hello | 8971
hello | 8972
hello | 8973
hello | 8974
hello | 8975
hello | 8976
hello | 8977
hello | 8978
hello | 8979
hello | 8980
hello | 8981
hello | 8982
hello | 8983
hello | 8984
hello | 8985
hello | 8986
hello | 8987
Process(1) Time(s): 6.964s
Process(1) Time(ms): 6964ms
```