# Data Structures and Algorithms

# INFO 6205

# Homework 1. Hongxi Li 001893090

# Due: January 20, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework1.zip and submit it via the drop box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. Estimate the running time (or memory) as a function of input size *N*. Explain as to why the results are for the following three examples.

⅙ *N* 3 + 20 *N* + 16 ~ ⅙ *N* 3

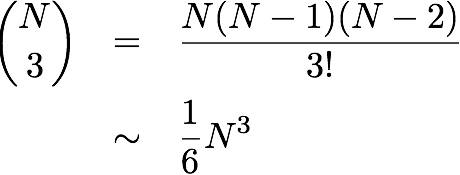
⅙ *N* 3 + 100 *N* 4/3  + 56 ~ ⅙ *N* 3

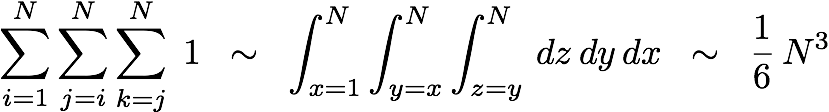
⅙ *N* 3 - ½ *N* 2+ ⅓ *N* ~ ⅙ *N* 3

In this expression, the other items after the first item are relatively small. For example, when N=1000, -1/2N2+1/3N��499667, but 1/6N3��166666667. It is much smaller than the after.

Compared to the 20N+16 and 100N4/3+56, they are much smaller than 1/6N3. Therefore, They cannot make some difference. According to the Exponential growth curve chart, ignore lower order terms. When N is large, terms are negligible. When N is small, we don't care.

As the result, These three results are close to the 1/6N3.

2. Write the code samples with running time of: (constant 1, logN, N, NlogN, N^2, N^3, 2^N). Mathematically, what do you describe each one of these examples?

3. Write the code that results to following running time. Describe the math.

4. The 3-Sum Triple loop has the following running time estimate. Explain.

Note: I do not want to prove the math. I want only the description of the math, what it

represents and why the result is 1/6 N^3

5. What are all Stack operations, explain.

* **Push:**Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
* **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
* **Peek or Top:** Returns top element of stack.
* **isEmpty:**Returns true if stack is empty, else false.

6. Consider String “It was the best of time”. Start with the first word, design a Stack such that when you read back the words, the order of string does not change. Provide code for all necessary operations of Stack. Compile and run the code.

7. Consider the following Node data structure, build a Stack linkedList with the following data: {31, “Name1”}, {24, “Name2”}, {10, “Name3”}, {44, “Name4”}, {81, “Name5”}.

a) Provide java implementation for all necessary Stack operations including stack pointers.

b) Compile and run your program.

c) What is Stack linkedList time and space complexity?

They are all the O(n).

class Node {

int Age;

String Name;

Node next;

}

8. Consider data: {31, “Name1”}, {24, “Name2”}, {10, “Name3”}, {44, “Name4”}, {81, “Name5”}.

a) Provide Array implementation of Stack.

b) Compile and run the code.

c) What is Stack Array implementation time and space complexity?

They are all the O(n).

9. Suppose in problem-8 the array size was: a) too large, or b) too small. How would you manage resizing the array for (a) and (b). Write the code, compile and test the program. Also discuss the running time/space complexity of your approach.

10. Human use Infix expression and computers use Postfix expression. You are to write a simple Calculator. There are three steps: a) Read Infix expression, b) Convert Infix expression to Postfix, and c) Evaluate Postfix expression. Write Java code, compile and run with five Infix expression examples.

(1 + 2 \* (20 / 5 ))

(1 + 2 - 3)

(1 + 3 + ( ( 4 / 2 ) \* ( 8 \* 4 ) ))

(300+23)\*(43-21)/(84+7)

(4+8)\*(6-5)/((3-2)\*(2+2))

References:

<http://www.cs.nthu.edu.tw/~wkhon/ds/ds10/tutorial/tutorial2.pdf>

<https://www.includehelp.com/c/infix-to-postfix-conversion-using-stack-with-c-program.aspx>

<http://ice-web.cc.gatech.edu/ce21/1/static/audio/static/pythonds/BasicDS/InfixPrefixandPostfixExpressions.html>