University of Washington Bothell

CSS 340: Applied Algorithmics

Program 3: Algorithm Analysis

Purpose

This assignment will focus on algorithm analysis (Big O). It will have both a written part as well as a programming part. The goal is to clearly show the impact of algorithms with different complexity.

Written Problems:

1. What is the Big O upper bound of the func() below as a function of n assuming that the Func1(n) is O(n)? Show your work by first finding a function g(n) which quantifies the number of operations as a function of n.

```
def func(n):
    j = n
    while j >= 1:
        for i in range(j):
            val = func1(n)
        j = j // 3
```

2. Determine the BigO complexity of func2() as a function of n assuming the task(a,b) is O(1).

```
def func2(n):
    for i in range(1, n+1):
        for j in range(i,1 + (i*n)):
            task(i+1,j)
```

Programming Problem:

Create a module called bigo which has four functions, find1(list, val), find2(list, val), find3(list, val), and find4(list, val). Each of the functions will take as arguments a list followed by a value. The functions will return a boolean as to whether the val is a member of the list.

The specification for each of the functions is as follows:

find1(list, val): The unsorted list is searched linearly to see if the val is in the list

find2(list, val): A deep copy is made of the list; the copied list is then sorted using the **sort** built-in function and then a binary search is performed on the list to find if the val is in the list

find3(list, val): The **in** built-in is used to determine if the val is in the unsorted list

find4(list, val): This function requires the list to be sorted before it is called. A binary search is performed on the pre-sorted list to find val.

Code the four functions in module as described above.

Write a report which:

1) Determines the BigO complexity for each function
2) Graphically depicts the running time of each of the functions as the size of the list increases. Do this using the Timer in the python TimeIt module. Note: the graphs do not need to be generated programmatically; you can just graph these collecting the data and using something like excel.

<u>Turn In</u>
A **.zip file** with

- A module names bigo.py which has the four functions above implemented
- Driver code which shows how you tested the functions to illustrate their complexity
- A doc, docx or .pdf with the report and graphs for functions