FizzBuzz Big O Problem 4

September 22, 2017

Question

Given the following code, explain the run-time and Big O. The following code performs integer division. What is it's runtime (assume a and b are both positive).

Explanation and Algorithm

This code computes the product of a and b.

An explanation of Big-O:

Big O is how we describe the efficiency of algorithms. This is an important concept, because a program that is time and space efficient saves a company time and money. For time complexity, think of Big O in terms of best case (fastest), worst case (slowest), and average/expected case. For many coding algorithms, the worst case and average case will actually be the same.

In addition to time complexity, you will often be asked in an interview about space complexity. The Big O of time complexity and the Big O of space complexity can often be different.

When thinking about the Big O of your algorithm, write a formula for your run time and then drop the constants. For example, if you have an algorithm with two for loops, the run time is $0(n)$, not $O(2n)$. In addition to this, it is also important to remember to drop the non dominant terms. For example, $O(N^2 + N)$ should simplify to $O(N^2)$.

The Big O times in order from most efficient to least efficient are the following: $O(1)$, $O(logn)$, $O(nlogn)$, $O(n^2)$, $O(2^n)$, $O(n!)$.

Hints

1. Think about how many times an operation is conducted in this problem. What are the main operations?

2. An explanation of $O(n)$: This means an algorithm is being run in constant time.

3. An explanation of $O(logn)$: If n number of operations get divided at each run time.

4. An explanation of $O(n^2)$: If n times n operations are performed.

5. An explanation of $O(2^n)$: If the number of operations increases exponentially.

Code

```
/* Code */

int div(int a, int b){
   int count=0;
    int sum=b;
   while (sum<=a){
      sum+=b;
      count++;
    }
    return count;
}
```

Run time analysis

The best case run time of this algorithm is O(1). If b¿a, then the algorithm runs in constant time.

For our worst case run time, we keep increasing the value of the sum by b increments until the sum is greater than the value of a. This means that the while loop iterates a/b number of times, and this comes out to O(a/b).