Charles Kolozsvary Period 2 Algorithms 12-23-20
4 Analysis of Algorithms - Order of Growth Classifications

Q 1. Describe the best and the worst order-of-growth for an algorithm.

The best and worst order of growth for an algorithm refers to the best and worst time complexities for a given algorithm under varying circumstances where the number of samples N is the same but the data itself is different. In the method below, simpleSearch, an array of length N and an int key are given. The method returns the index of where the key is located in the array or -1 to indicate that no such value exists in the array. This is a simple method but it can be used to understand how best and worst case order of growths are present in algorithms. If the key does not exist in the array, then every element in the array must be accessed and therefore the order of growth is N; this is the worst order of growth for the algorithm. If key == a[0] then only one element in the array had to be accessed and therefore the order of growth is 1 (single access, essentially single statement); this is the best order of growth for the algorithm. Therefore the method simpleSearch with the same number of samples N with varying data can provide different orders of growth, creating best and worst case time complexities and orders of growth.

```
private int simpleSearch(int[] a, int key) //simple search method example
{
    //given an array and a value you are looking for, return the index where the key
    //is located or return -1 indicating that no such key exists in the array
    int N = a.length;
    int i = 0;
    do {i++;} //do while loop, keep incrementing while there is still more to be seen (i<N) and
    //while the key is not found (a[i] != key)
    while(i<N&&a[i]!=key);
    if(i<N) return i; //return the index
    else return -1; //return -1
}
```

Another example of best and worst order of growth can be seen in the quick sort algorithm. I won't show the algorithm because it is a lot more complex than this simple search method, so I'll just use it to illustrate my point. The Quicksort algorithm is a divide and conquer algorithm which will sort a given array to be in order of lowest to highest. The best order of growth for quick sort is N log (N) and the worst order of growth is N^2. The average order of growth for this algorithm is therefore Nlog(N). Even with the same number of samples, N, the quick sort algorithm can have orders of growth N log N or N^2 depending on the actual state of the data given (in this case the state of the array).

Q 2. What is the order of growth for two nested for loops with N as their upper bound?

The order of growth for this description would be quadratic, given the framework looks like

```
for(int i = 0; i<N; i++)
{
        for(int j = 0; j<N; j++)
        {
                ...
        }
}
```

Therefore the order of growth in this instance is N^2 or quadratic

Q 3. Give an example of a set of instructions with order-of-growth approximately N^3.

```java
public int example(int N, int[] a, int[] b, int[] c)
{
    //a, b, and c are all size N
    int count = 0;
    for(int i = 0; i<N; i++)
    {
        for(int j = 0; j<N; j++)
        {
            for(int k = 0; k<N; k++)
            {
                //Three for loops = N^3 = cubic order of growth
                if(a[i]+b[j]+c[k] == 0)
                {
                    count++; //increment
                }
            }
        }
    }
    return count; //returns the number of triplet pairs in the three arrays (number of terms which sum to zero)
}
```

The code snippet above is an example of a set of instructions where the order of growth is N^3

Q 4. Does better order of growth mean more efficient algorithm?

Yes, in most instances better order of growth (where log(N) is better than N^3) means less runtime, and therefore the algorithm is more efficient.