Name: Ebony Warren ID:1521067

Problem 1: For all of the following, determine the total operation count and then the Big-O of the given code segments:

a.

```
for (int j = 0; j < n; j++)
        for (int k = 0; k < j; k++)
                Sum++;
```

Total operation count: $n^2/2*(n-1)$

Big-O: $O(n^2)$

b.

```
for (int i = 0; i < q*q; i++)
        for (int j = 0; j < i; j++)
                Sum++;
```

Total operation count: $q^4/2*(q-1)$

Big-O: $O(q^4)$

For all of the following, just determine the Big-O of the given code segments:

C.

```
for (int i = 0; i < n; i++)
        for (int j = 0; j < i*i; j++)
                for (int k = 0; k < j; k++)
                        Sum++;
```

Big-O: $O(n^5)$

Name: Ebony Warren ID:1521067

D.

```
for (int i = 0; i < p; i++)
        for (int j = 0; j < i*i; j++)
                for (int k = 0; k < i; k++)
                        Sum++;
```

Big-O: $O(p*i^3)$

e.

```
for (int i = 0; i < n; i++) {
        Circ arr[n];
        arr[i].setRadius(i);
}
```

Big-O: O(n)

F.

```
for (int i = 0; i < n; i++) {
        int k = i;
        while (k > 1) {
                sum++;
                k = k / 2;
        }
}
```

Big-O: $O(n*\log_2(i))$

Name: Ebony Warren ID:1521067

Problem 2: Given a vector of sets of ints, vector< set<int> > v, assume the vector v has N total sets and that each set has an average of Q items.

a. What is the Big-O of determining if the first set, v[0], contains the value 7?

Big-O: $O(\log_2(Q))$

b. What is the Big-O of determining if any set in v has the value 7?

Big-O: $O(N*\log_2(Q))$

c. What is the Big-O of determining the number of even values in all of v?

Big-O: $O(N+(N)\log_2(Q))$

d. What is the Big-O of finding the first set with a value of 7 and then counting the number of even values in that set?

Big-O: $O(N*\log_2(Q) +(N+(N)\log_2(Q)))$

Problem 3: Determine the data structure needed if we wanted to maintain a bunch of peoples' names and for each person, allows us to easily get all of the streets they lived on. Assume there are P total people and each person has lived on average E former streets.

data structure: map<names, vector<string> address> list;

What is the Big-O cost of:

a. Finding the names of all people who lived on "Levering Street"?

Big-O: $O(\log_2(P)*E)$

b. Determining if "Bill" ever lived on "Westwood Blvd"?

Big-O: $O(\log_2(P)*E)$

c. Printing out every name along with each person's street addresses in alphabetical order?

Big-O: $O(P*\log_2(P)+E*\log_2(P))$

d. Printing out all the streets that "Tala" has lived on?

Big-O: $O(\log_2(P)*1)$

Name: Ebony Warren ID:1521067

Problem 4: Fibonacci numbers are a sequence of numbers given by the relationship:

$$F_n = F_{n-1} + F_{n-2}$$

With $F_0 = 0$ and $F_1 = 1$. In other words, the nth Fibonacci number is given by the sum of the two Fibonacci numbers before it. For Example, the first 13 Fibonacci numbers are:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

a. Implement a recursive function to compute the nth Fibonacci number:

```
int fibonacci(int n) {
        If (n == 0) return 0;
        If (n == 1) return 1;
        Return (fibonacci( n - 2)+fibonacci( n - 1));
}
```

b. What is the Big-O of the recursive Fibonacci function?

$O(2^n)$

Problem 5:

Given the following array show the result after one round of the each of the sorting algorithms indicated. One round being one full iteration of the algorithm's outermost for/while loop.

a. Selection Sort:

99 16 3 19 13 0 13 12 6

0 16 3 19 13 99 13 12 6

b. Insertion Sort:

99 16 3 19 13 0 13 12 6

16 99 3 19 13 0 13 12 6

c. Bubble Sort

99 16 3 19 13 0 13 12 6

16 3 19 13 0 13 12 6 99