Santa Monica College
Spring 2018

CS 20A: Data Structures with C++

Assignment 5

Name: Nashir Jammohamed **ID:** 1325605

Problem 1:

For all of the following, determine the **total complexity** 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++;
```

(1) $f(n) = 1 + n + n + n + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} + \frac{n(n-1)}{2} = \boxed{\frac{3}{2}n^2 + \frac{3}{2}n + 1}$

b. (2) $O(n^2)$

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

$\frac{q^4 - q^2}{2}$

(1) $f(q) = 1 + q^2 + q^2 + q^2 + \frac{q^2(q^2-1)}{2} + \frac{q^2(q^2-1)}{2} + \frac{q^2(q^2-1)}{2} = \boxed{\frac{3}{2}q^4 + \frac{3}{2}q^2 + 1}$

(2) $O(q^4)$

c.

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

$(n-1)^2 - 1 = j_{max}$

$i \times i - 1 = k_{max}$

(1) $f(n) = 1 + 3n + 3\left(\frac{n((n-1)^2-1)}{2}\right) + 3\left((n^2-2n-1)\left(\frac{1}{2}(n^3-n^2)\right)\right) = \boxed{\frac{3}{2}\left(n^5-3n^4+2n^3-n^2\right)+3n+1}$

$\frac{3}{2}(n^3 - 2n^2) + \frac{3}{2}(n^5 - 3n^4 + n^3 + n^2)$

(2) $O(n^5)$

d.

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

$i \times 1 = (p-1)^2$

$p - 1$

(1) $f(p) = 1 + p + p + p + 3\left(\frac{p((p-1)^2-1)}{2}\right) + 3\left((p-1)\left(\frac{1}{2}(p^3-2p^2)\right)\right) = \boxed{\frac{3}{2}(p^4-2p^3)+3p+1}$

$\frac{3}{2}(p^3 - 2p^2) + \frac{3}{2}(p^4 - 2p^3 - p^3 + 2p^2)$

(2) $O(p^4)$

Name: Nashir Janmohamed **ID:** 1325605

e.

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

① $f(n) = 1 + n + n + n^2 + n = n^2 + 3n + 1$

② $O(n^2)$

f.

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

① $f(n) = 1 + n + n + n + n\log_2 n + n\log_2 n + n\log_2 n = 3n\log_2 n + 3n + 1$

② $O(n\log_2 n)$

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?

accessing first set → $O(1)$
entering first set → $O(1)$
searching set → $O(\log_2 Q)$

$\log_2 Q + 1 + 1 \Rightarrow \log_2 Q$

$\boxed{O(\log_2 Q)}$

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

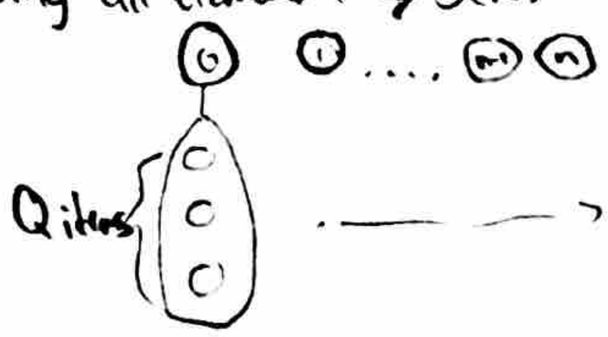
iterating through vector v → $O(N)$ searching set → $O(\log_2 N)$

$\boxed{O(N\log_2 Q)}$

Name: Nashir Jamohamed ID: 1325605

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

iterating through V → O(N)
accessing all elements → O(N)



iterating through each set in V → O(Q)

(N+N)Q = 2NQ

O(NQ)

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?

finding the first set with a value of 7 → O(N log₂Q)

counting the number of even values → O(Q)

N log₂Q + Q

O(N log₂Q + 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.

map<string, set<string>> nameToAddresses;

What is the Big-O cost of:

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

iterating through everyone → O(P)
accessing everyone → O(P)

searching for "Levering Street" → O(log₂E)

(P + P) × log₂E = 2P log₂E

O(P log₂E)

Name: Nashir Janmohamed **ID:** 1325605

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

finding Bill → $O(\log_2 P)$
accessing Bill → $O(1)$
searching for Westwood Blvd → $O(\log_2 E)$

$1 + \log_2 P + \log_2 E$

$O(\log_2 P + \log_2 E)$

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

iterating through everyone → $O(P)$
accessing everyone → $O(P)$
printing everyone's name → $O(P)$
printing addresses → $O(E)$

$(P+P+P)E = 3PE$

$O(PE)$

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

finding Tala → $O(\log_2 P)$
printing streets → $O(E)$

$\log_2 P + E$

$O(\log_2 P + E)$

Name: Nashir Jamohamed ID: 1325605

Problem 4:

Fibonacci numbers are a series of numbers given by the relationship:

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

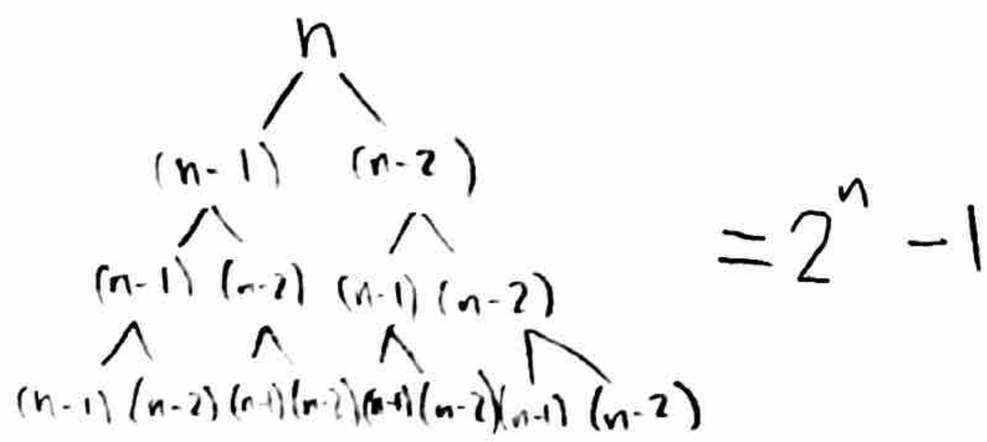
With $F_0 = 0 \; and \; F_1 = 1$. In other words, the nth Fibonacci number is given by the sum of the two Fibonacci numbers before it. For Example, the first 13 Fibonacci numbers are:

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
~~void~~ fibonacci(int n) {
    if (n <= 1)
        return 1;
    return fibonacci(n-1) + fibonacci(n-2);
}
```

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



$= 2^n - 1$

$O(2^n)$

Name: Nashir Jamohamed **ID:** 1325605

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 outer most 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