CS 401 Algorithms
Assignment1
Due Date: 04/22/2019

04/12/2019

Part I: Algorithm Analysis

1. Give the order of growth (as a function of n) of the running times of each of the following code fragments. Please show your work.

a.
```
while (n > 0) {
    for (int k = 0; k < n; k++)
        printReport(); // runs in O(N/2) time
    n = n / 2;
}
```

$n+1$ $2n+1$ $\log_2(n)+1$ $O(n/2)$

$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{6} \cdots + 1$

$n(1 + \frac{1}{2} + 1/4) + 1$

$2n+1$

b.
```
for (int i = 0; i < n; i++)
    for (int j = i+2; j > i; j--)
        for (int k = n; k > j; k--)
            System.out.println(i+j+k);
```

$2n+1$ $2 \cdot 2(n+1)$ $2n$ $3n$ $3n$

$2\left[\frac{(n-2)(n-1)}{2} + \frac{(n-1)(n+2)}{2}\right]$

c.
```
void aMethod(int n){
    if (n <= 1)
        return;
    anotherMethod(); //runs in O(1) time
    aMethod(n/2);
    aMethod(n/2);
}
```

d.
```
public static int mystery(int n) {
    int i = 1;
    int j = 0;
    while (i < n) {
        j++;
        if (j == i) {
            i++;
            j = 0;
        }
    }
    return j;
}
```

$n=5$

$1+2+3+4$

$i=2, 3,$

e.
```
public static int compute(int n) {
    int total = 0;
    for (int i = 1; i < n; i++) {
        int k = 1;
        while (k < i) { k = k + k;          }
        while (k > 1) { k /= 2;    total++; }
    }
    return total;
}
```

$n+1$ $2n$ $2n$

$\log_2 2^x = n$

$\log_2(n) = x$

$\log_2(n) \cdot \frac{1}{2x} = 1$

$\log_2(n) = 2^x$

$x = \log_2(\log_2(n))$

f. *//n, the length of arr*

```
public static int removeDuplicates(char[] arr) {
    int len = arr.length;
    int i = 0;              // index of current item to find
    while (i < len) {
        int j;              // will be index of duplicate of arr[i]
        for (j = i + 1; j < len; j++) {     n(n+1)/2 + 1
            if (arr[i] == arr[j]) break;  - n
        }
        if (j == len) {   n  // no duplicate of arr[i] found; go to next i
            i++;  - 1
        } else {        n  // duplicate found; shift array over arr[j]
            for (int k = j + 1; k < len; k++) {    (n(n+1)/2 + 1)
                arr[k - 1] = arr[k];     n(n+1)/2
            }
      n     len--;
            arr[len] = 0;
        }
    }
    return len;
}
```

Part II: Programming

1. Write a method in Java that returns the total number of trailing zeroes for all integers from 1 to its parameter n; given 5, it returns 0 + 1 + 0 + 2 + 0 = 3. The *trailing zeroes* of an integer *a* are the zeroes following the last 1 in *a*'s binary representation (ex: 0100 ->2; 0101->0). Using big-O notation in terms of its parameter *n* and provide the time complexity of your algorithm.

2. Write a program that, given two sorted arrays of n in values, prints all elements that appear in both arrays, in sorted order. The **running time** of your program **should be proportional to n** in the worst case (Exercise 1.4.12)

3. Write a program that, given an array `a[]` of n distinct integers, finds a strict local minimum: an entry `a[i]` that is strictly less than its neighbors. Each internal entry (other than `a[0]` and `a[n-1]`) has 2 neighbors. Your program should use **~2(lgn)** compares in the worst case (Exercise 1.4.18). Provide tests for all possible cases.

Sample Cases:

```
Array-> {5, -4, 10, 16, 11, 20, 24, -10};
Local minimum: -4            //it can also output 11 as local minimum

Array-> {-8, -6, 18, 8, 20, 4, 40};
Local minimum: 8             //it can also output -8 or 4 as local minimum
```

a.) If $n < 0$, then either the loop will end, if n is an int, or it will continue for some finite divisions as the ability to hold the decimal places is finite (and this amount of divisions has an upper limit)

$$n\left(\tfrac{1}{2}\right)^k = 1$$

$$n = 2^k$$

$$k = \log_2(n) + \alpha$$

constant which is the max limit to get the number to equal 0

$$\sum_{i=0}^{\log_2(n)+\alpha} (2n+1)\left(\frac{n}{2}\right) = \sum_{i=0}^{\log_2(n)+\alpha} \frac{2n^2+n}{2}$$

$$= (\log_2(n) + \alpha)\left(\frac{2n^2+n}{2}\right)$$

$$= \frac{1}{2}\left(2n^2\log_2(n) + n\log_2 n + \alpha 2n^2 + \alpha n\right)$$

$$\boxed{O(n^2\log_2(n))}$$

b.) Adding all of the statement operations

$$1 + n + 2n + 1 + 2(n) + 2(n+1) + 2\cdot 2(n+1) + 2n + 3n + 3n$$

$$+ \underbrace{\frac{(n-2)(n-1)}{2}}_{\text{for } j=i+2} + \underbrace{\frac{(n-1)(n)}{2}}_{\text{for } j=i+1}$$

$$= (\text{some multiple of } n) + \frac{n^2 - 3n + 2 + n^2 - n}{2}$$

$$= \alpha n + \frac{1}{2}(2n^2 - 4n + 2)$$

$$\boxed{O(n^2)}$$

c.) $T(n) = 1 + 1 + T(n/2) + T(n/2)$

$$T(n/2) = 2 + T\left(\frac{n}{4}\right) + T\left(\frac{n}{4}\right)$$
$$T(n/4) = 2 + T\left(\frac{n}{8}\right) + T\left(\frac{n}{8}\right)$$
$$T(n/8) = 2 + T\left(\frac{n}{16}\right) + T\left(\frac{n}{16}\right)$$

$$= 2 + 2 + T\left(\frac{n}{4}\right) + T\left(\frac{n}{4}\right) + 2 + T(n/4) + T(n/4)$$
$$= 6 + 4T\left(\frac{n}{4}\right)$$
$$= 6 + 4\left(2 + T\left(\frac{n}{8}\right) + T\left(\frac{n}{8}\right)\right)$$
$$= 14 + 8T\left(\frac{n}{8}\right)$$
$$2 + \sum_{i=0}^{k} 2^{i+1} + 2^{k+1}T\left(\frac{n}{2^{k+1}}\right)$$

$$\frac{n}{2^{k+1}} = 1$$

$$n = 2^{k+1}$$

$$\log_2(n) = k+1$$

$$k = \log_2(n) - 1$$

$$2 + \sum_{i=0}^{\log_2(n)} 2^{\log_2 n+1} + 2^{\log_2(n)+1} (1)$$

$$2 + \log_2(n) \cdot 2^{\log_2 n+1} + 2^{\log_2 n+1}$$

$$\boxed{O(\log_2(n)\, 2^{\log_2 n})}$$

d.) $\frac{(n-1)(n)}{2} \approx T(n)$

$$\boxed{O(n^2)}$$

e.) $1 + 1 + n+1 + 2n + 2n + \log_2(n) + \log_2(\log_2 n)$
$+ 1$

$$= 4 + 4n + \log_2(n) + \log_2(\log_2 n))$$

$$\boxed{O(n)}$$

f.) $T(n) \approx \alpha n + 4\left(\frac{n(n+1)}{2}\right) + 4$

$\approx \alpha n + 2n^2 + 2n + 4$

$\boxed{O(n^2)}$