CSCI 104 Participation Quiz 4: February 21, 2019

Name, ID: $T(1) = c_1$ $T(0) = c_2$

```
bool ispowertwo(double x){
    if (x == 1) return true;
    if (x < 1) return false;
    if (x > 1) return ispowertwo(x/2);
}
void function1(int x){
    if (ispowertwo(x)){
        for (int i = 0; i < x ; i++)
            cout << i << endl;
    } else {
        cout << x << endl;
    } C4
}
void function2(int x){
    if (!(x & (x-1))){ /*condition is true if and only if x is a power of 2. Checks using bitwise and */
        for (int i = 0; i < x ; i++)
          c cout << i << endl;
    } else {
        cout << x << endl;
    }
}
void function3(int n){
    for (int i = 1; i <= n; i++){
        function1(i);
    }
}
void function4(int n){
    for (int i = 1; i <= n; i++){
        function2(i);
    }
}
```

$$T(x) = T\left(\frac{x}{2}\right) + c_3$$

$$= \left(T\left(\frac{x}{4}\right) + c_3\right) + c_3$$

$$\vdots$$

$$= T\left(\frac{x}{2^k}\right) + kc_3$$

$$\sum_{i=0}^{x-1} c = xc$$

$$\frac{x}{2^k} = 1 \Rightarrow k = \lg x \quad = \Theta(\log x)$$

$$\sum_{i=1}^{n} c_5 + \sum_{k=1}^{n} \lg k + c\sum_{j=0}^{\lg n} 2^j + \sum_{l=1}^{n-\lg n} c_4$$

$$\log 1 + \log 2 + \log 3 + \dots + \log n$$

$$\log 1 * 2 * 3 * \dots * n = \log n!$$

by Sterling's approximation $n \log n$

$n \lg n$

1. What is the worst case runtime analysis of function3?
   Suggested Answer:

$$\sum_{i=1}^{n} c_1 + \sum_{k=1}^{n} \log k + \sum_{j=1}^{\log(n)} 2^j + \sum_{l=1}^{n-\log n} c_3 = \Theta(n \log n)$$

Possible Multiple Choice Answers:

(a) $\Theta(n)$
(b) $O(n^2)$
(c) $\Theta(n^2)$
[✓ (d) $\Theta(n \log n)$]

if power of 2

not power 2

2. What is the worst case runtime analysis of function4?
   Suggested Answer:

$$\sum_{i=1}^{n} c_1 + \sum_{k=1}^{n} c_2 + \sum_{j=0}^{\log(n)} 2^j + \sum_{l=1}^{n-\log n} c_3 = \Theta(n)$$

geometric series

$$\Theta(2^{\lg(n)}) = \Theta(n)$$

Possible Multiple Choice Answers:

[✓ (a) $\Theta(n)$]
(b) $O(n^2)$
(c) $\Theta(n^2)$
(d) $\Theta(n \log n)$

(4) [10 points]

```
void runTimeFun(int n){
    for (int i = 1; i <= n; i++){
       for (int j = 0; j < n; j += i) {
         for (int k = 0; k < n ; k++) {
               if (i % 2 == 0) {
                  for (int m = 1; m <= n ; m *= 2){
                       cout << m << endl;
                  }
              } else {
                 cout << i << endl;
             }
         }
       }
    }
}
```

What is the worst case runtime analysis of runTimeFun? You must show your work and mathematical derivation to receive credit.

Let's take a first pass by translating code directly to summations

$$\sum_{i=1}^{n}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1}[\sum_{m=1}^{n} c_1 + c_2]$$

There are a few problems with this first attempt. First for the loop indexed by j, we are incrementing by i not 1, but the inner for loop indexed by k does not depend on j, so we can replace that with the number of iterations, $\frac{n}{i}$. Second the innermost for loop indexed by m is not incremented by 1. Instead, m is multiplied by 2, but the work inside is constant and does not depend on m, so we can replace that by the number of iterations $\log n$. Let's see how this simplifies summation:

$$\sum_{i=1}^{n}\frac{n}{i}\sum_{k=0}^{n-1}[c_1\log n + c_2]$$

This is certainly better, but we have a problem with how we are handling the for loop indexed by k, namely we are considering all the possible work that can happen, but we have an if statement so all of that work will not always happen. If i is even, then the $\log n$ work of the for loop indexed by m will happen. When i is odd, then only a constant amount of work is done. In either case, the work does not depend on the actual value of k in the outer loop. From this summation is simpler still

$$\sum_{i=1}^{n}\frac{n}{i}n[c_1\log n + c_2]$$

We can remove from within the summation any terms that do not depend on i,

$$n^2[c_1\log n + c_2]\sum_{i=1}^{n}\frac{1}{i}$$

When we now recognize the harmonic series and get a closed form for this summation and apply an asymptotic bound

$$n^2\log n[c_1\log n + c_2] = O(n^2\log^2 n)$$

This is counting more work than we need since we are adding all the work if i is odd or even. We can argue that we are justified in doing this for the upper bound or big O since if we do the work for when i is even in worst case for all i, then our summation is

$$n^2\log n[c_1\log n] = O(n^2\log^2 n)$$

Our asymptotic bound holds. However, although this code works the same for any n, to get a tighter lower bound would require more careful analysis than is necessary. You would have to argue that for the $n/2$ even iterations the work of the above sum occurs and that for the $n/2$ odd iterations the work is $n^2\log nc_2$. When you add those terms you would still get the same bound for the lower bound, but the subtlety is in convincing yourself that the number of iterations of the for loop indexed by i is still $\log n$. That is tricker to show rigorously than is necessary for this problem or than we showed you in class. If you argued for Ω and Θ, we will also give credit.

- translation of code to summation - 2 pts
- work to get closed form - 2 pts
- counting iterations of j loop and recognizing harmonic series - 1 pt
- counting iterations of m loop - 1 pt
- considering effect of the if statement - 2 pts
- asymptotic bound 2 pts