CSCI 104 written HW #1 Breeze Pickford jan 2024

Q1 a)

```
void f1(int n)
{
  int i=2
  while(i<n){
    /* do smth that takes O(1) time */
    i = i * i;
  }
}
```

how many iterations until i exceeds n?

find largest k such that

$2^k < n$ ← take log of both sides

$k \log 2 < n$ ← solve for k

$k < \frac{\log n}{\log 2}$

sum from k=0 to x: 2^k.

whats happening? i is increasing exponentially from 2 until it is => than n at which point the loop stops. each time the loop runs it takes O(1) time.

what kind of series is this? 2 2*2=4 4*4=16 $i^{2k} < n$ geometric. i is increasing exponentially

How many times i doubles until it reaches n? $\Theta(\log_2 n)$

$n = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdots$

$n = i^k$

Q1 b)

```
void f2(int n)
{
  for(int i=1; i<=n; i++){
    if( (i % (int)sqrt(n)) == 0 ){
      for(int k=0; k<pow(i,3); k++){
        /* do smth that takes O(1) time */
      }
    }
  }
}
```

whats happening? inner most loop has k increase from 0 to i^3 and does a O(1) operation each time. so that means inner loop $\sum_{k=0}^{i^3} O(1) \rightarrow i^3 \rightarrow \Theta(i^3)$.

if $\frac{i}{\sqrt{n}}$ has a remainder of 0. AKA $\sqrt{n}$ is a factor of i. ← needs to check if statement every time → O(1)

outter loop has i go from 1 to <= n. so that means $\sum_{i=1}^{n}$ ↳ this goes a total of n times.

whats the worst case of the if statement?

work it out: outter loop; how many times this runs worst case → $\sqrt{n}$

$$\sum_{i=1}^{n} \left(\Theta(1) + O\left(\sum_{k=0}^{i^3} O(1) \right) \right)$$

$\Theta(n) + \sqrt{n} \cdot \Theta(n) \cdot O(\Theta(i^3))$

$\Theta(n) + \Theta(\sqrt{n} \cdot n^3) = \Theta(n) + \Theta(n^2)$

outter inner

run time of $\Theta(n^2)$

Breeze Bickford

Q1c) What is happening:

```
for(int i=1; i≤n; i++){
    for(int k=1; k<=n; k++){
        if(A[k]==i){
            for(int m=1; m≤n; m=m+m){
                // do smth that takes O(1) time
                // assume contents of array are unchanged
            }
        }
    }
}
```

whats happening? 3 nested loops + 1 if statement around innermost loop.

outer loop: runs from $i=1$ to $i=n$. total of n times.

2nd loop: runs from $k=1$ to $k=n$, total of n times.

if: constant time to check $\Theta(1)$.

inner loop: runs from $m=1$ to $m=n$ but m doubles each time → so it runs for $\log_2 n$ iterations

$\sum_{i=0}^{k} 2^i \dots \log_2 n$ from $i=0$ to $i=k$ of 2^i

$$\sum_{i=1}^{i=n} \left(\sum_{k=1}^{k=n} \left(\Theta(1) + \sum_{i=0}^{k} 2^i \left(\Theta(1)\right) \right) \right)$$

$$\left(\Theta(1) + \Theta(\log_2 n)\right)$$

$$\Theta(n) + \Theta(n)\Theta(\log_2 n)$$

$$\Theta(n^2) + \Theta(n^2)\Theta(\log_2 n)$$

↑ take highest order term

$$\boxed{\Theta\left(n^2 \log_2(n)\right)}$$

Breeze Pickford

(Q1D)

```
int f(int n) {
    int *a = new int[10];
    int size = 10;
    for (int i=0; i<n; i++) {
        if (i == size){
            int newsize = 3*size/2;
            int* b = new int[newsize];
            for(int j=0; j<size; j++) b[j] = a[j];
            delete[] a;
            a=b;
            size=newsize;
        }
        a[i] = i*i;
    }
```

whats happening?

array a of size 10 (dynamic)

outer loop: runs from i=0 to n-1. a total of n times.

if statement: if i = size resizes array by making new array b of size $\frac{3}{2}$ · currentsize

it copies elements from a to b and makes a point to b. size variable is updated to new size.

inner for loop, at worst $\Theta(n)$ because goes from j=0 to size-1

other: regardless of if statement. a[i] = i*i happens, which takes $\Theta(1)$

$$\sum_{i=0}^{n-1} \left(\quad \text{// } O\left(\Theta(n) + \Theta(1)\right) \right.$$

$$\downarrow$$

$$\sum_{i=0}^{k} 10i \quad \left(\Theta(1)\right) + \Theta(1)$$

$$\downarrow$$

$$\Theta(n) \left(\log_{10} n + \Theta(1) \right)$$

$$\Theta(n \log_{10} n) + \Theta(1)$$

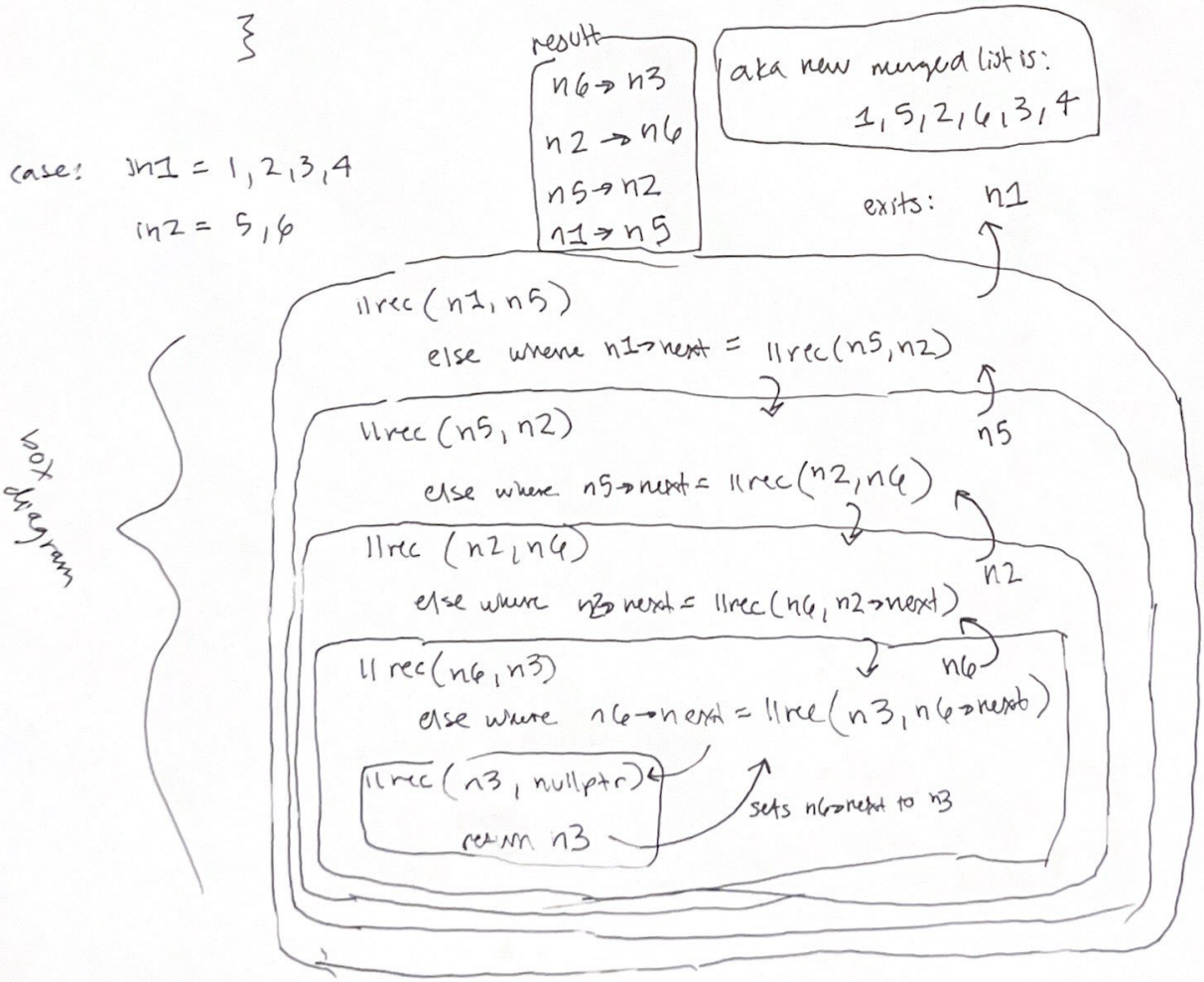
↑ take highest order term

runtime: $\Theta(n \log_{10} n)$

(2 a)

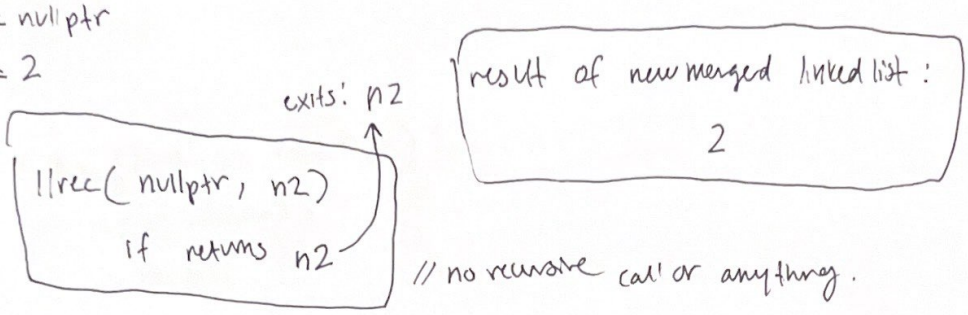
```
Struct Node {
    int val;
    Node* next;
};

Node* llrec(Node* in1, Node* in2) {
    if( in1 == nullptr) {
        return in2;
    }
    else if (in2 == nullptr){
        return in1;
    }
    else {
        in1->next = llrec( in2, in->next);
        return in1;
    }
}
```

case: in1 = 1, 2, 3, 4
in2 = 5, 6



Q2b)

case: in1 = nullptr
in2 = 2



// no recursive call or anything.