CSCI 104

Mika Lim lslim@usc.edu

January 2025

1 Written Homework 1

2 Part 1

2.1 Part a

$\sum_{i=2}^{x} O(1)$ X represents the number of times the while loop is run. When x= 0, n is 2, and when x=1, n= 4, x=2 –¿ n=16. So n is equal to $(2^{(2^x)})$. Solving for x , the loop runs log(logn) times. The final summation is then:
$\sum_{i=2}^{log(logn)} O(1)$
which becomes =log(logn) * O(1) **Final answer is then Θ (log(logn))**

2.2 Part b

1. setup: $(\sum_{i=1}^{n}((\Theta(1) + \sum_{k=0}^{i^3-1} O(1)))$. Each summation represents one of the loops.

2. z(the number of times the inner loop an) was derived by testing cases and finding a relationship in terms of n:

z	i
1	$sqrt(n)$
2	$2(sqrt(n))$
3	$3(sqrt(n))$
z	$z(sqrt(n))$

3. $((\Theta(1) + \sum_{k=0}^{\sqrt{n}}((i)^3)))$

4. $((\Theta(1) + \sum_{k=0}^{\sqrt{n}}((z\sqrt{n})^3)))$

5. If the loop stops when i = n, then i = $z\sqrt{(n)}$, so z = $\sqrt{(n)}$

6. $\Theta(n) + (\sqrt{n})^3(\Theta(\sqrt{n})^4)$

7. **Final answer:** $\Theta(\sqrt{n})^7$

2.3 Part c

1. setup: $(\sum_{i=2}^{n}(\sum_{k=1}^{n}(O(1)+\sum_{m=1}^{k}O(1)))$, where each summation was derived from the three for loops, and the O(1) is from the if-statement, and the assumed lines that take O(1)
2. To find the k value, which represents the number of times the innermost loop runs, I did tests to find the relationship between k and n.
3.

k	n
1	1
2	2
3	4
4	8

4. to find the relationship to be n = $2^(k-1)$
5. This means k in terms of n is log(n) + 1, or essentially log(n)
6. $(\sum_{i=2}^{n}(\sum_{k=1}^{n}(O(1)+\sum_{m=1}^{log(n)}O(1)))$
7. Distributing the outer summation: $(\sum_{i=2}^{n}(\sum_{k=1}^{n}O(1))+\sum_{i=2}^{n}(\sum_{m=1}^{log(n)}O(1))$
8. However the if statement means that the outermost loop will not run the innermost loop every time. The innermost loop only runs when A[k] = i. For every set of times i loops, A[k] can only be equal to i once, and the max value of i is n therefore the total number of times the if-condition is passed is n.
9. $(\sum_{i=2}^{n}(\sum_{k=1}^{n}O(1))+\sum_{i=1}^{n}(\sum_{m=1}^{log(n)}O(1))$
10. Simplified, this is $\Theta(n^2)+\Theta(nlogn)$
11. **Final answer:** $\Theta(n^2)$

2.4 Part d

1. Each outer loop iteration is $\sum_{i=0}^{n-1}O(1)+[inner\ loop\ resizing\ cost]$
2. The relationship that describes the resizing is $n = 10 \times (\frac{3}{2})^k$, where k is the number of resizes. Solving for k: $k = O(\log_{\frac{3}{2}}(n)) = O(\log(n))$
3. $\sum_{k=0}^{O(\log(n))}O((\frac{3}{2})^k)$
4. Solving as a geometric series to find the sum of each resize: $10\times\sum_{k=0}^{O(\log(n))}$ Simplifies to: $10 \times O((3/2)^{O(\log(n))}) = 10 \times O(n^{\log_2(3/2)}) = 10 \times O(n^{O(1)})$
5. So the total cost of resizing is $10 \times O(n^{O(1)}) = O(n)$.
6. Non-resizing iterations take $O(1)$ so $O(1) + O(n) + O(n)$
7. **Final answer s $\Theta(n)$.**

3 Part 1

3.1 Question a

in1 = 1,2,3,4 and in2 = 5,6

The total linked list will be represented as (1,2,3,4), where 1 is the head and each item points to the following as next.

1. The first call would be ***llrec((1,2,3,4),(5,6))***
   (a) After checking that the heads of both in1(1) and in2(5) are not null, ***in1-¿next is set to llrec(in2,in1-¿next)***.
2. So the recursive call is ***llrec((5,6),(2,3,4))***
   (a) This call evaluates that in1 and in2 are not null
3. The next recursive call is ***llrec((2,3,4), (6))***
   (a) This call evaluates that in1 and in2 are not null
4. The next recursive call is then ***llrec((6), (3,4))***
   (a) This call evaluates that in1 and in2 are not null
5. ***llrec((3,4), (null))***
   (a) the base case is fulfilled and in2 is now null
   (b) **This function returns (3 ,4)**

   Following the returned values of each recursive call:
6. The returned value of llrec((3,4),(null)) is (3,4)
7. In the previous call ***llrec((6), (3,4))***
   (a) in1-¿next is set to this returned val (3,4), meaning that the in1 list is now (6 , 3 ,4)
   (b) **This call returns in1 (6 ,3,4)**
8. In the previous call of ***llrec((2,3,4),(6))***
   (a) in1-¿next is set to this returned val (6 , 3 , 4), meaning that the in1 list is now (2 , 6 , 3 , 4)
   (b) **This call returns in1(2 ,6,3,4)**
9. In the previous call of ***llrec((5,6),(2,3,4))***
   (a) in1-¿next is set to this returned val (2 ,6,3,4), meaning that the in1 list is now (5 ,2,6,3,4)
   (b) **This call returns (5,2,6,3,4)**

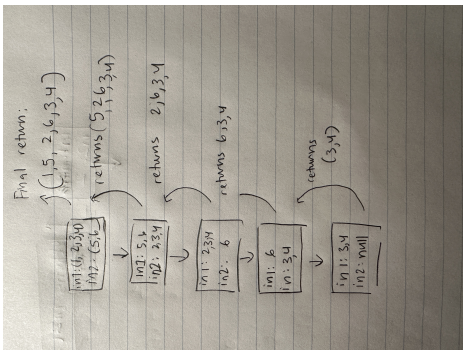

Figure 1: Visualization of Answer to Problem a

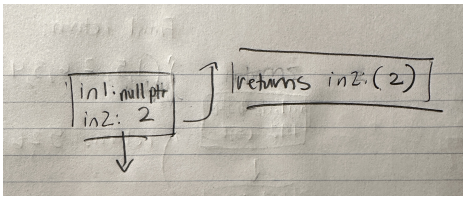

Figure 2: Visualization of Answer to Problem b

10. In the first call llrec((1,2,3,4),(5,6))
    (a) in1-¿next is set to the returned val of (5,2,6,3,4)
    (b) **Returning the final list (1,5,2,6,3,4)**

3.2 Question b

Since in1 is nullptr, the basecase is immediately met, so in2(2) is returned.