①

Judy Chen #51748144 -x1b0b
Yueyue Zhang #49084130 -a5a9

Question #1.

Assume List1 is the first list, List2 is the second list
a points to the head of list1, b points to the head of list2

```
if (List1 is not equal to NULL and list2 is not NULL) {
    for ( int i=0 ; i< size of List2 ; i++) {
        b = b->next;
        b->next = a,
        a = a->next;
        b->next = b;
    }
    return List1;
}
if (List1 = NULL)
    return List2,
if (List2 = NULL)
    return List1;
```

Question #2.

(a) For a sorted array, it would take O(logn) steps to search for the given house # on the street; since the array is sorted, simply return the house # of the next two elements of the array.

it would take O(n) steps to insert a new house in the array because every element past that point in the array would have to be shifted down.

(b) For a sorted singly linked list, it would take O(n) steps to search for the given house #, as you must traverse down the linked-list in order to find it; In order to insert a new house at the position after searching, it would take O(1) steps to create a new node and assign pointers to it. It takes O(1) steps because the work done is constant and there is no need to move the rest of elements down the list.

(c) singly linked list would be prefered

① Adding a new house is more efficient. There is no need to shift every element

② O(1) indicate the insertion into the list will be equally fast no matter how large the list is.

②

Question #3:

① $\lg 32^n = \lg 2^{5n} = 5n$

② $2^{\lg(n^2m^2) - \lg m^2} = \frac{2^{\lg(n^2m^2)}}{2^{\lg(m^2)}} = \frac{n^2m^2}{m^2} = n^2$

③ $-(\lg(\frac{1}{8})) = -\lg 2^{-3} = 3$

④ $\log_p(\frac{1}{p}) = \log_p p^{-1} = -1$

⑤ $64^{\lg(n^2)} = 2^{6\lg(n^2)} = n^{6\times 2} = n^{12}$

Question #4:

$\log n < \sqrt{n} < n < \lg(n!) < n\log n < \sum_{i=1}^{n} i^3 < 2^{2\lg n} < 2^n < 2^{2n} < n^n$

Question #5:

(b) $T(n) = (4n+12)(6n+2) = 24n^2 + 80n + 24 \in \Theta(n^2)$

$O: T(n) \leq 38n^2$ for $n \geq 6$ $(c = 38,\ n_0 = 6)$

$\Omega: T(n) \geq 0 \cdot n^2$ for $n \geq -3$ $(c = 0,\ n_0 = -3)$

(c) $T(n) = \sum_{i=0}^{n} 2^{i+c} = 2^c \cdot (2^{n+1} - 2) = 2^k \cdot (2^n - 1)$, $k = c+1$

$T(n) \in \Theta(2^n)$

$O: T(n) \leq 2^{c+1} \cdot 2^n = 2^{n+c+1}$

$\Omega: T(n) \geq 2^{c\ 1} \cdot 2^n = 2^{n+c\ 1}$

(d) $T(n) = \sum_{i=1}^{n} \sum_{j=i^2}^{n^2} c = n \cdot [n^2 - \frac{n}{6}(n+1)(2n+1)]c = (\frac{n^3}{2} - \frac{n^2}{6} - \frac{n^4}{3})c \in \Theta(n^4)$

$O: T(n) \leq -n^4$ for $n \leq -1$ $(c = -1,\ n_0 = -1)$

$\Omega: T(n) \geq 0 \cdot n^4$ for $n \geq 1$ $(c = 0,\ n_0 = 1)$

(e) $T(n) = \begin{cases} 1, & n = 0 \\ 1, & n = 1 \\ T(n-2) + 4, & n \geq 2 \end{cases}$ $\Rightarrow T(n) = 5 \cdot 2^{\frac{n-3}{2}}((\sqrt{2}-1)(-1)^n + 1 + \sqrt{2}) - 4$

$T(n) \in \Theta(2^n)$

f) $T(n) = \begin{cases} 1 & , n=1 \\ T(\frac{n}{2})+3 & , n>1 \end{cases} \Rightarrow T(n) = \frac{3\log(n)}{\log(2)} + 1 \in \Theta(\log n)$.

Question # 6.

(a) $T(n) = 54n^3 + 17 \in \Theta(n^3)$

$O: 54n^3+17 \leq 71n^3$, for $n \geq 1$ $\quad (C = 71, n_0 = 1)$

$\Omega: 54n^3+17 \geq 37n^3$, for $n \leq -1$ $\quad (C = 37, n_0 = -1)$

(b) According to the defn of $\Theta(f(n))$, there must exist more than one constant c such that $T(n) \leq O(f(n))$ and $T(n) \geq \Omega(f(n))$. In this case, we have.

$O: 54n^3+17 \leq 71n^2$ for $n \geq 1$

$\Omega: 54n^3+17 \geq 71n^2$ for $n \geq 0$

only when $c = 71$, we have $O(f(n)) = \Omega(f(n))$ and other integer constant would fall to $O(f(n))$ or $\Omega(f(n))$, so we fail to prove $T(n) \in \Theta(f(n))$.

(c) $T(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots a_1 n + a_0$, given that $f(n) = n^d$

O: let $c = a_d + a_{d-1} + \cdots a_1 + a_0$, $n_0 = 1$, we have

$T(n) \leq (a_d + a_{d-1} + a_{d-2} + \cdots a_1 + a_0) n^d$ for $n \geq 1$

Ω = let $c = 0$, $n_0 = -1$, we have.

$T(n) \geq 0 \cdot n^d$ for $n \leq -1$

$\therefore T(n) \in \Theta(n^d)$ as required.

④

Question #7:

(a)
```
double pow(double x, unsigned int n){
    if(n==0) return 1;              -1
    double result = pow(x*x, n/2);
    if(n&1) result *= x;            -1
    return result;                  -1
}
```

$T(n) = 3 + \lg n \in \Theta(\lg n)$

(b)
```
longestIncreasing(A)
    ResultForPrefix = new Array[A.length]   -1
    for i=0 to A.length-1 {                 ①
        r=1                                 ②
        for j=0 to i-1 {
            if A[j]<A[i] and r<ResultForPrefix[i]+1 then      3i
                r = ResultForPrefix[j]+1
        }
        ResultForPrefix[i]=r                ③
        if bestOverall < r then bestOverall = r    ④
    }
    return bestOverall   -1
```

$\sum_{i=0}^{n-1}(4+3i)$

$T(n) = 2 + \sum_{i=0}^{n-1}(4+3i) \in \Theta(n^2)$