Assignment 1

Luigi Sacco (h2y9a) 21139143 | Andre Tertzakian (y6l0b) 17682148

CPSC 221 | October 3, 2016

1.

```
void insert(Node *a, Node *b)
{
    Node a_final = a;
    while (a_final->next)
        a_final = a_final->next;
    a_final->next = b->next;
    b->next = a;
}
```

2.

(a) If the structure were an array, then there would n elements in the array, where n is the number of houses in Vancouver. In this way, the house would be ordered within the array according to their house number. This would mean that it would take $O(n)$ time to find the house with the given number, as well as its neighbours, since you would have to loop through the list (because we cant assume the numbers are in ascending order, so a search algorithm such as Binary Search, could not be used).

In order to insert a house between two others would also take $\Theta(n)$, time since you would insert a house at the beginning and have to shift all of the elements of the array over to accommodate the new entry. It is $\Theta(n)$, since every time you perform this algorithm, you will have to perform n iterations, no matter where the house to be inserted lies within the list. This is because you will have to shift $n - k$ elements over once the insertion is complete, where k is the position at which the house was inserted.

(b) If the structure was chosen to be a singly-linked list, then there would still be n houses within the list, since there are n houses in Vancouver. The list would be ordered such that each house/node points to its right-hand neighbour. If no right-hand neighbour exists, then the list points to the next cluster of houses. For this reason, we cannot assume that the list is sorted. This means that it take $O(n)$ to find the neighbours of a house, since would have to loop through the entire list in order to find the house which points to the desired house, and then grab the house after the given house as well.

It is slightly different when it comes to inserting a house. In this case, it would take $O(n)$ instead of $\Theta(n)$, you do not need to shift over all of the house afterwards after inserting the new node. You only need to change two pointers.

(c) If given the choice of a data structure to use, and memory space was not a consideration, I would opt for a hash-map with a key-value pair, of house number to house object. In this case, the house object would contain a list of all of its neighbours, which can be obtained by calling `house.getNeighbours()`, for example. This would allow for a wort-case of $O(n)$ and a best-case of $O(1)$.

3.

(a) $\lg 32^n = n \lg 2^5 = 5n$

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

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

(d) $\log_p(1/p) = -\log_p p = -1$

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

4.

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

5.

(a) $T(n) = 47 \quad \therefore \boxed{T(n) \in \Theta(1)}$

(b) $T(n) = (4n+12)(6n+2) = 24n^2 + 80n + 24 \quad \therefore \boxed{T(n) \in \Theta(n^2)}$

(c) $T(n) = \sum_{i=0}^{n} 2^{i+c} = \sum_{i=0}^{n} 2^c \cdot 2^i = 2^c(1 + 2^1 + 2^2 + 2^3 + ... + 2^n) \quad \therefore \boxed{T(n) = \Theta(2^n)}$

(d) $T(n) = \sum_{i=1}^{n}\sum_{j=i^2}^{n^2} c = \sum_{i=i}^{n}\left(\sum_{j=1}^{n^2} c - \sum_{j=1}^{i^2-1} c\right) = \sum_{i=i}^{n}(cn^2 - c(i^2-1)) = \sum_{i=1}^{n}(cn^2+1) - ci^2$
$= (cn^2+1)n - c(n(n+1)(2n+1)/6) \ \sim \ n^3 \quad \therefore \boxed{T(n) = \Theta(n^3)}$

(e) $T(0) = 1, T(1) = 1, T(n) = 2T(n-2) + 4 = 4T(n-4) + 12 = 2^kT(n-2k) + 4(2^k-1)$. Let $n = 2k \implies 2^{n/2} = 2^k \implies T(n) = 2^{n/2}T(0) + 4\cdot 2^{n/2} - 4 \ \sim \ 5\cdot 2^{n/2} \quad \therefore \boxed{T(n) \in \Theta(2^{n/2})}$

(f) $T(1) = 1, T(n) = T(\lceil n/2\rceil) + 3 = T(\lceil n/4\rceil) + 6 = T(\lceil n/8\rceil) + 9 = T(\lceil n/2^k\rceil) + 3k$. Let $n = 2^k \implies k = \lg n \implies T(1) + 3\lg n = 1 + 3\lg n \geq \lg n \quad \therefore \boxed{T(n) \in \Theta(\lg n)}$

6.

(a) $54n^3 + 17 \in O(n^3) \because$ choose $c = 71, n_0 = 1 \implies 54n^3 + 17 \leq 71n^3 \implies 1 \leq n^3 \ \forall n \geq 1$. $(1*)$
$54n^3 + 17 \in \Omega(n^3) \because$ choose $c = 54, n_0 = 0 \implies 54n^3 + 17 \geq 54n^3 \implies 17 \geq 0 \ \forall n \geq 0$. $(2*)$
$\therefore (1*) \wedge (2*) \implies 54n^3 + 17 \in \Theta(n^3)$ $\square$

(b) $T(n) \notin \Theta(n^2) \because T(n) \notin \Omega(n^2)$.
Negation for existence in $\Omega(n^2)$: $\forall c, n_0 \in \mathbb{R}^+, \exists n \geq n_0$ s.t. $54n^3 + 17 < cn^2$
Choose $n = n_0$. Then $54n^3 - cn^2 + 17 < 0$. Since n is positive, we can divide forth by n^2. $\implies 54n - c + \frac{17}{n^2} < 0$ which with large values of n acts like $54n - c$, and since n will be fixed, there will be a value of c that makes the expression negative (e.g. $55n$). $\therefore T(n) \notin \Theta(n^2)$ $\square$

(c) Proof by induction. Base case $d = 0 \implies a_0 \in \Theta(1)$ because it is a constant. Then, if $\left(\sum_{i=0}^{d-1} a_i n^i \in \Theta(n^{d-1})\right)$ is true, it must be the case that $\sum_{i=0}^{d} a_i n^i = a_d n^d + \sum_{i=0}^{d-1} a_i n^i \in \Theta(n^d)$ because $a_d > 0$ and hence it follows that $P(n+1) \geq P(n)$ gives the desired result. $\square$

7.

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

$T(1) = 1$

$T(n) = T(n/2) + 2 = T(n/4) + 4 = T(n/8) + 6 = T(n/2^k) + 2k$

Let $n = 2^k \implies k = \lg n \implies T(1) + 2\lg n = 1 + 2\lg n \implies \boxed{T(n) \in \Theta(\lg n)}$

(b)
```
// Returns the length of the longest subsequence of strictly increasing
// elements in the array A. A subsequence of array A is not necessarily
// contiguous.
longestIncreasing(A)
    ResultForPrefix = new Array[A.length]
    for i = 0 to A.length - 1 {
        r = 1
        for j = 0 to i - 1 {
            if A[j] < A[i] and r < ResultForPrefix[j] + 1 then
                r = ResultForPrefix[j] + 1
        }
        ResultForPrefix[i] = r
        if bestOverall < r then bestOverall = r
    }
    return bestOverall
```

$$\begin{aligned} T(n) &= 1 + \sum_{i=0}^{n-1} \left(3 + \sum_{j=0}^{i-1} 2 \right) \\ &= 1 + \sum_{i=0}^{n-1} (3 + 2i) \\ &= 1 + 3n + 2\frac{n(n+1)}{2} \\ &= n^2 + 4n + 1 \end{aligned}$$

$\therefore \boxed{T(n) \in \Theta(n^2)}$