Eleanor Wong k1w8
Lawrence Garcia
41018128
j3x8

CPSC 221: Written Assignment 1

1)
```
struct Node <T> {
    T key;
    Node* next;
}
```

```
Node* insertList ( Node* a, Node* b) {
    if (b == NULL) return a;   ← if (a == NULL) return b;
    else:
        Node* bNext = b→next;
        Node* aCursor = a;   b→next = a;
        while ( aCursor→next != NULL)
            aCursor = aCursor→next;
        aCursor→next = bNext;
        return b
    return NULL;
}
```

2 a) House Numbers

As an array, the houses are simply listed: [9, 10, 11, 12, 13]

To output neighbours, first have to find the index of the house n, then return the houses at index n-1 and n+1. If house can't be found, can return NULL index. To find this index, we can use binary search, which will take O(log n) time. Accessing the neighbours is simple O(1) retrieval given an index.

To add a house, even if the array size is dynamic, every additional house added shifts everything to its right:

Find index to add house @ → [1|3|4|5|6|7] → make new array one size larger
4
add house here
↳ add everything before new house,
add new house,
add larger house numbers.

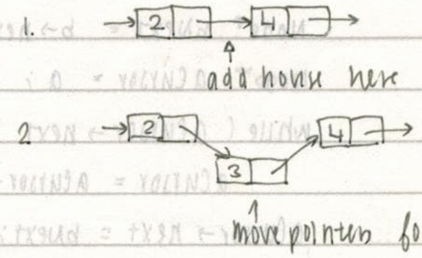
Each addition of a house takes O(n) operation where n is the # of houses.

b) As a singly linked list, the house # list looks as follows: [1|]→[2|]→[3|]→

Since the list is ordered, to find neighbours, have to iterate through the list, saving previous iteration (as left neighbour) until one finds the house. Then call house→next to get it's right neighbour. At worse case, if the house you are looking for is at the end, this is n operations to finish where n = # of houses. ⇒ O(n).

Eleanor Wong k1w8
Lawrence Garcia j3x8

To add a house, we need to find where to add the house next to, then just change the next pointers to accomodate the new house



This only takes two short O(1) operations.

c) Since new construction happens very frequently, we should pick a data structure that is faster to insert new houses: the singly linked list. This structure is better because we do not need to create the list again each time we add a house. $\underbrace{O(1) + O(n)}_{\text{singly linked}}$ vs. $\underbrace{O(\log n) + O(n)}_{\text{array}}$

Eleanor Wong klw8
Lawrence Garcia j3x8

3. (a) $\lg 32^n = \lg 2^{5n}$
$= 5n \lg 2$
$= 5n$

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

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

(d) $\log_p(1/p) = \log_p(p^{-1})$
$= -\log_p p$
$= -1$

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

4. In non-decreasing order of growth rate:

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

- First three are obvious $\log n \le \sqrt{n}$ for some $n \ge n_0$
$n^{1/2} \le n$ for $n \ge 1$
- Comparing $\lg n!$ and $n\log n$ note that $n^n \ge n!$ so $\log n^n \ge \log(n!)$ and the base 2 is just a change of constant.
- $\sum_{i=1}^{n} i^3 = \left[\frac{n(n+1)}{2}\right]^2$ which is $O(n^4)$ so we put it between $n\log n$ and
- For $n^{2\lg n}$ and 2^n consider the log of both which gives us $2\lg n \log n$ and $n\log 2$. This means we're comparing $(\log n)^2$ and n. In this case, n grows faster.

Eleanor Wong k1w8
Lawrence Garcia j3x8

- $n^n \geq 2^n$ for $n \geq 2$
- The last element is easily placed. 2^{2^n}

5. (a) $T(n) = 47$, $\Theta(1)$

(b) $T(n) = (4n+12)(6n+12)$
$= 24n^2 + 120n + 144$

Ans: $\Theta(n^2)$

(c) $T(n) = \sum_{i=0}^{n} 2^{i+c} = \sum_{i=0}^{n} 2^i 2^c = 2^c \sum_{i=0}^{n} 2^i$

$= 2^c(2^n - 1)$

Thus, we have $\Theta(2^n)$

(d) $T(n) = \sum_{i=1}^{n} \sum_{j=i^2}^{n^2} c = \sum_{i=1}^{n} (n^2 - i^2 + 1) c$

$= n^3 c - \sum_{i=1}^{n} i^2 + nc$

$= n^3 c - \frac{n(n+1)(2n+1)}{6} + nc$

So we see that we get $\Theta(n^3)$

(e) $T(0) = 1$, $T(1) = 1$ and $T(n) = 2T(n-2) + 4$ for $n \geq 2$

$T(n) = 2T(n-2) + 4$
$= 2[2T(n-4) + 4] + 4 = 2^2 T(n-4) + 2 \cdot 4 + 4$
$= 2^3 T(n-6) + 2^2 \cdot 4 + 2 \cdot 4 + 4$
$= 2^{\lceil \frac{k}{2} \rceil - 1} T(n-k) + 4(2^{\lceil \frac{k}{2} \rceil - 1} - 1)$

Need $k = n, n-1$ to get $T(0) = T(1) = 1$ (base case)

$T(n) = 2^{\lceil \frac{n}{2} \rceil - 1} T(0) + 4(2^{\lceil \frac{n}{2} \rceil - 1} - 1)$

$= (4 + T(0)) 2^{\lceil \frac{n}{2} \rceil - 1} - 4$

Thus we have $\Theta(2^{\frac{n}{2}})$ or $\Theta(\sqrt{2}^n)$

Eleanor Wong KIW8
Lawrence Garcia j3x8

(f) $T(1) = 1$ and $T(n) = T(\lceil n/2 \rceil) + 3$ for $n \geq 2$

$$\begin{aligned} T(n) &= T(\lceil n/2 \rceil) + 3 \\ &= T(\lceil n/4 \rceil) + 6 \\ &= T(\lceil n/8 \rceil) + 9 \\ &= T(\lceil n/2^k \rceil) + 3k \qquad \text{let } n = 2^k \\ &= T(1) + 3\lg n \qquad\quad k = \lg n \end{aligned}$$

Thus, we have $\Theta(\log n)$.

6. (a) Prove that $54n^3 + 17$ is $\Theta(n^3)$

- O-case : $c = 71$, $n_0 = 1$ gives
  $54n^3 + 17 \leq 71n^3$,
  $17n^3 \geq 17$
  $n^3 \geq 1$ which is certainly true for all $n \geq n_0$
  Thus we have $O(n^3)$ ①
- Ω-case : $c = 54$, $n_0 = 1$ gives
  $54n^3 + 17 \geq 54n^3$
  $17 \geq 0$ which is true for all $n \geq n_0$
  Thus we have $\Omega(n^3)$ ②

①, ② $\Rightarrow \Theta(n^3)$

(b) Consider again $T(n) = 54n^3 + 17$.
For $T(n) \in \Theta(n^2)$ we must have $T(n) \in O(n^2)$.
- Let us then assume $T(n) \in O(n^2)$.
- By definition, there are positive constants c and n_0 such that $T(n) \leq cn^2$ for all $n \geq n_0$

So $54n^3 + 17 \leq cn^2$
$n^2(54n - c) + 17 \leq 0$

Looking at the equation, choosing $n > \frac{c}{54}$ will result in a contradiction.
Thus, there does not exist a positive constant n_0 such that $54n^3 + 17 \leq cn^2$ holds for all $n \geq n_0$
Thus, $T(n) \notin O(n^2) \Rightarrow T(n) \notin \Theta(n^2)$

Eleanor Wong K1W8
Lawrence Garcia j3x8

(c) $T(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$
where a_i is constant for all $i = 0, \ldots, d$
and $a_d > 0$. Prove $T(n) \in \Theta(n^d)$

① O-case: $T(n) \leq |a_d| n^d + |a_{d-1}| n^{d-1} + \cdots + |a_1| n + |a_0|$
(For $n \geq 1$) $\leq (|a_d| + |a_{d-1}| + \cdots + |a_1| + |a_0|) n^d$

This means that there exist $C = |a_d| + |a_{d-1}| + \cdots + |a_1| + |a_0|$
and $n_0 = 1$ s.t. $T(n) \leq Cn^d \ \forall\ n \geq n_0$

Thus, $T(n) \in O(n^d)$

② Ω-case: $T(n) = a_d n^d + \underbrace{a_{d-1} n^{d-1} + \cdots + a_1 n + a_0}_{\text{from above: } O(n^{d-1})}$

Thus, $T(n) \geq a_d n^d - C_1 n^{d-1}$ for some $n \geq n_0$
$= \left(a_d - \frac{C_1}{n}\right) n^d$

So therefore we see that there will always exist a value K s.t. $T(n) \geq C_2 n^{d-1}$ for all $n \geq K$
Specifically, we just need $K > \frac{C_1}{a_d}$
with $C_2 = a_d - \frac{C_1}{n}$

Thus, $T(n) \in \Omega(n^d)$

①, ② thus prove that $T(n) \in \Theta(n^d)$

Eleanor Wong K1W8
Lawrence Garcia j3x8

7. Runtime Analysis

(a) - Base case of recursion is $T(1) \le b$ where b is some constant.
- Recursion gives:

$$
\begin{aligned}
T(n) &\le T(n/2) + 1 \quad \text{(one is for odd case)} \\
&\le T(n/4) + 2 \\
&\le T(n/8) + 3 \\
&\vdots \\
&\le T(n/2^k) + k \quad \text{let } n = 2^k,\ k = \lg n
\end{aligned}
$$

Thus,
$$
\begin{aligned}
T(n) &\le T(1) + \lg n \\
&\le b + \lg n
\end{aligned}
$$

The order of growth is thus: $\Theta(\log n)$

(b) - Nested for loops.
- Let the runtime of operations in the inner and outer loop be constants b and c respectively.

Thus,
$$
\begin{aligned}
T(n) &\le \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i-1} b + c \right) \\
&= \sum_{i=0}^{n-1} (bi + c) \\
&= [b + 2b + 3b + \cdots (n-1)b] + nc \\
&= \frac{n(n-1)}{2} b + nc
\end{aligned}
$$

The order of growth is $\Theta(n^2)$.