Question 1

a) i $f(n) = n^2 \log(n) + n(\log(n))^2 \approx n^2 \log(n) \in \Theta(n^2 \log(n))$

$g(n) = n \log(n) \in \Theta(n \log n)$

$\Rightarrow f(n) \in \Omega(g(n))$

ii $f(n) = \sum_{i=0}^{n} 2^i \quad g(n) = 2^n$

$\sum_{i=0}^{n} 2^i = \frac{2^n - 1}{2 - 1} = 2^n - 1 \in \Theta(2^n)$

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

$\Rightarrow f(n) \in \Theta(g(n))$

iii $f(n) = 3^n \in \Theta(3^n) \quad g(n) = 3^{3n} = 27^n \in \Theta(27^n)$

$\Rightarrow f(n) \in O(g(n))$

iv $f(n) = (\sqrt{2})^{\log n} = 2^{\frac{1}{2}\log n} \quad g(n) = \sqrt{n} = n^{1/2}$

$2^{1/2 \log n} \in \Theta(2^n)$

$n^{1/2} \in \Theta(n^\varepsilon)$ where $0 < \varepsilon < 1$

$\Rightarrow f(n) \in \Omega(g(n))$

b)

2. $\Omega(1)$ is not a subclass of $O(n^2)$

c) basic operation = ~~sum~~ if i % 10 == 0

this algorithm is input-insensitive

$$C(n) = \sum_{i=0}^{n^2-1} \sum_{j=0}^{i-1} 1$$

$$= \sum_{i=0}^{n^2-1} (i - 1 - 0 + 1)$$

~~$$= \sum_{i=0}^{n^2-1} i = i(n^2 - 1 - 0 + 1)$$~~

~~$$\Rightarrow C(n) \in \Theta(n^2)$$~~

$$= \frac{n^2(n^2+1)}{2}$$ (triangle numbers formula)

$$= \frac{n^4 + n^2}{2} \Rightarrow C(n) \in \Theta(n^4)$$

Question 2.

a) $T(n) = T(n-1) + k^n, \quad T(0) = 1$

$= (T(n-2) + k^{n-1}) + k^n$

$= ((T(n-3) + k^{n-2}) + k^{n-1}) + k^n$

$\vdots$

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

let $c = n$

$= T(n-n) + \sum_{i=1}^{n} i$

$= T(0) + \frac{n(n+1)}{2}$

$= 1 + \frac{n^2+n}{2} = \frac{1}{2}n^2 + \frac{1}{2}n + 1$

b) $T(n) = 4T(n/8) + \sqrt[3]{n^2}$

$(+ n^{2/3}) \quad \in \Theta(n^{2/3})$

$a = 4, \quad b = 8, \quad d = 2/3$

$\Rightarrow T(n) = \Theta(n^{2/3} \log n)$ as $a = \sqrt[3]{8^2}$

$T(n) \in O(n^{2/3} \log n)$

Question 3.

a) If we use an adjacency list, we simply have to iterate through the list once and note the number of occurrences of each vertex (number of edges going to it in other words)

An adjacency list for a directed graph has $|V| + |2E|$ elements to ~~traverse~~ so we have $\Theta(|V| + |E|)$

b) Yes, as T is the minimum span

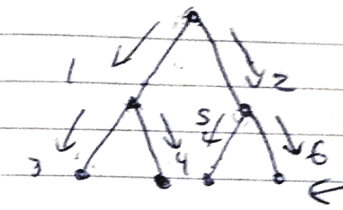
c) $w(e) \in [1, 8]$

d)

Covered	A	B	C	D	E	F
—	∞/nil	∞/nil	0/nil	∞/nil	∞/nil	∞/nil
C	11/C	4/C	~~0/nil~~	9/C	a/nil	∞/nil
C, B	5/B			9/C	6/B	∞/nil
C, B, A				9/C	6/B	8/A
C, B, A, E				8/E		8/A
C, B, ~~A~~, E, D						8/A
C, B, A, E, D, F						

Question 4.

a)



n nodes ~~height~~ = $n = 7$

worst case path finding

Professor Breadth is correct, the worst case scenario would involve trying to find a path to the lowest rightmost leaf, ~~of~~ which can always be reached ~~at least~~ in linear time ($n-1$ steps more precisely) as we are essentially looking at every node once and there are n nodes. ($n-1$ if we exclude the root).

Therefore $\Omega(n)$ is correct.

b) This statement is correct as we can ~~combine the two arrays in a linear pass across both at the same time requiring $\Theta(2n) = \Theta(n)$ and then we~~

Question 5

a)

```
function LONGEST SUBSTRING(S)
    longestSubLen ← 0
    longestSubStart ← -1
    longestSubEnd ← -1
    currentSubLen ← 0
    currentSubStart ← -1
    currentSubEnd ← -1
    for each character in S do
        if character = A or T then
            if currentSubLen = 0 then
                currentSubStart ← character index (i)
                currentSubEnd ← character index (i)
            currentSubLen ← currentSubLen + 1
        else
            currentSubLen ← currentSubLen + 1
            currentSubEnd ← currentSubEnd + 1
    else                    //we have found the end of the su
        if currentSubLen > longestSubLen then
            longestSubLen ← currentSubLen
            longestSubstart ← currentSubStart
            longestSubEnd ← currentSubEnd
        currentSubLen ← 0

    Return (longestSubStart, longestSubEnd)
```

Question 5.

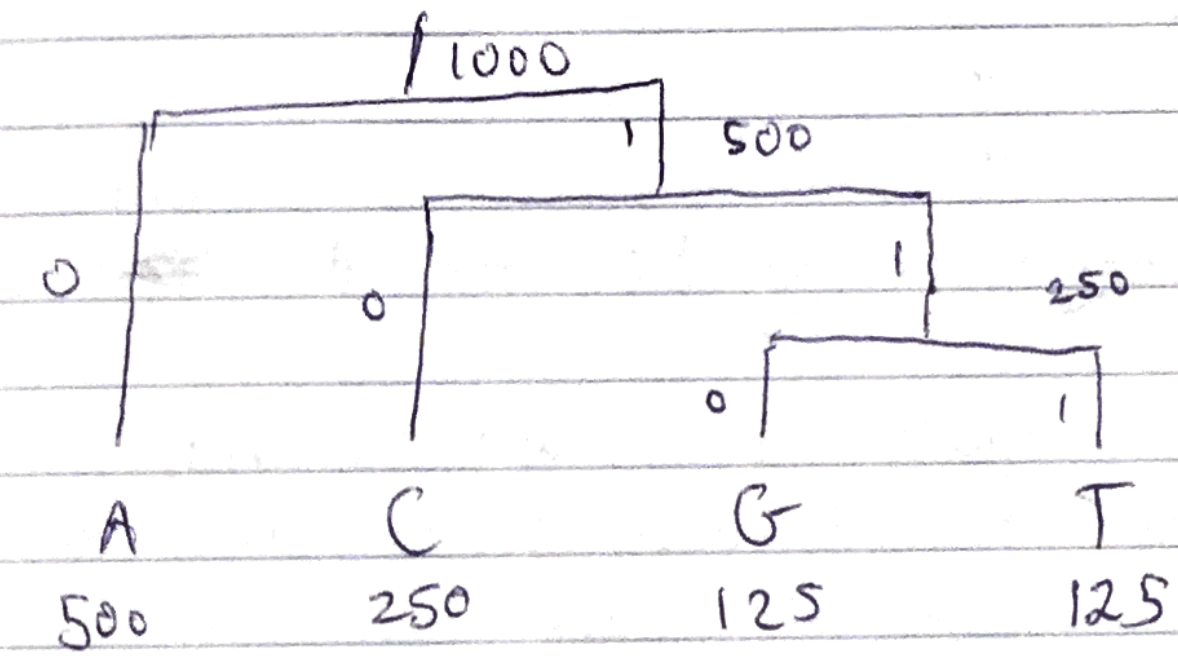
~~b) We ar.~~

b) We are simply traversing through the S string by checking each character one at a time and therefore the runtime will grow linearly with the length of input n.

$\Rightarrow C(n) \in \Theta(n)$

c) A: 00
C: 01
T: 10
G: 11

d)



A: 0
C: 10
G: 110
T: 111

Total bits for fixed-length = $1000 \times 2 = 2000$ bits

Total bits for Huffmann Encoding = $500 \times 1 + 250 \times 2 + 125 \times 3 + 125 \times 3$

$= 1750$ bits

$2000 - 1750 = 250$ bits saved

Question 5.

e) Considering that we want to use ~~small~~ amounts of memory but be as fast as possible we will likely want to utilise radix sort ~~in conjunction with fixed length encoding~~. While we do have to trade memory, it is only $\Theta(n+k)$ (which is known in this case) and can give us the fastest speed possible ($\Theta(n)$ sorting).

f)

	Shifts
A	1
C	4
G	4
T	4

AAAC

Character comparisons = $500 - 3 + 4 = 501$ comparisons

as we don't check the first 3 As, and C will continue to mismatch with the remaining 497 As (as the pattern will shift right by 1 each time) and then we will do four last comparisons at the end of the sequence of As and start of the sequence of Cs.

AAAA ..... AAA C CCC...

AAAC
× →
AAA C
✓ ✓ ✓ ✓

match here ↗

501 comparisons

Question

a)

Insert L

-1

B

Rotate B

L

B

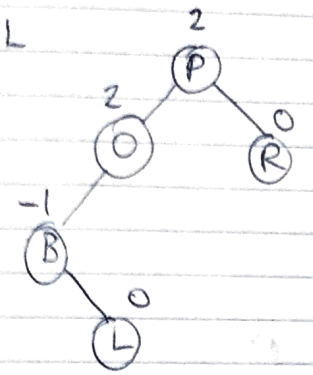
Rotate O

B

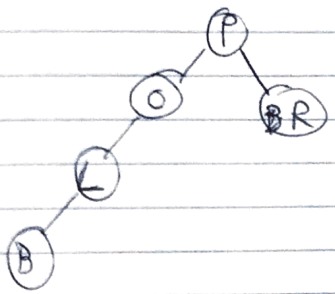
Question 6.

a)

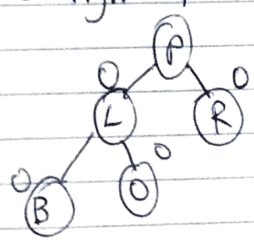
Insert L



Rotate B left



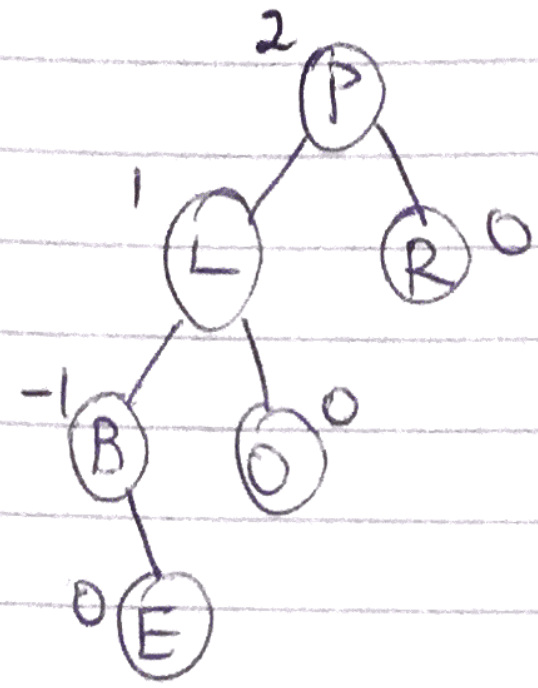
Rotate O right



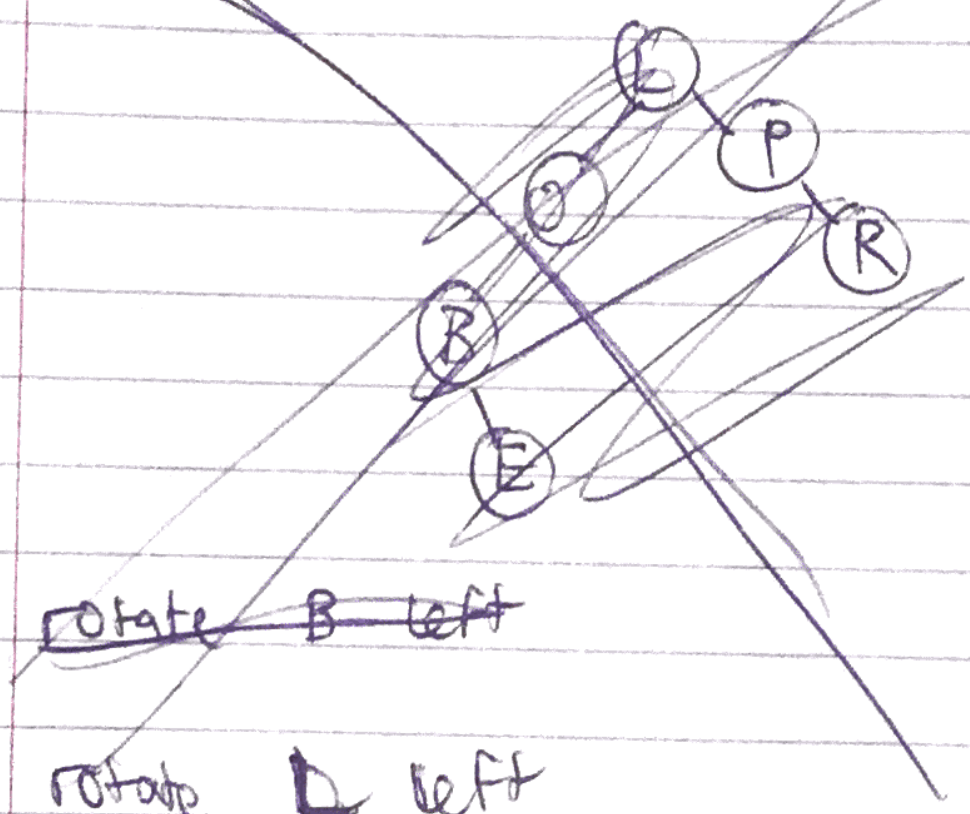
→ Balanced

Question 6.

b)

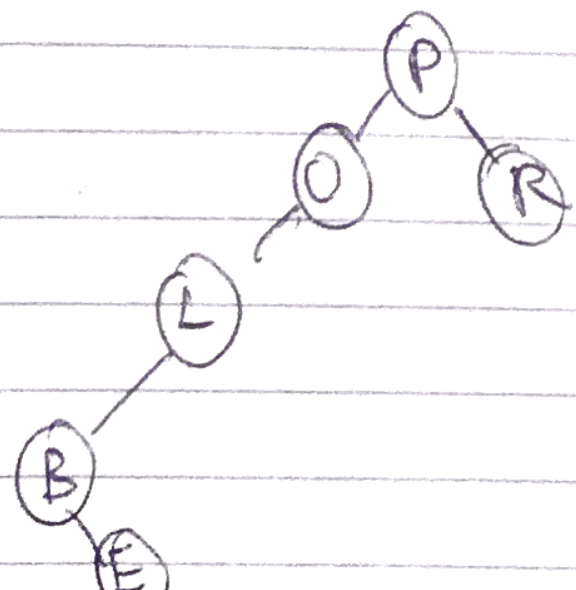


~~rotate B right~~ ~~rotate P right~~

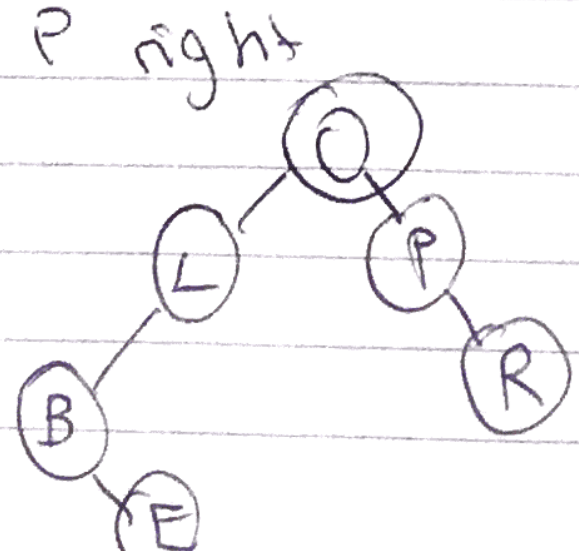


~~rotate B left~~

rotate L left



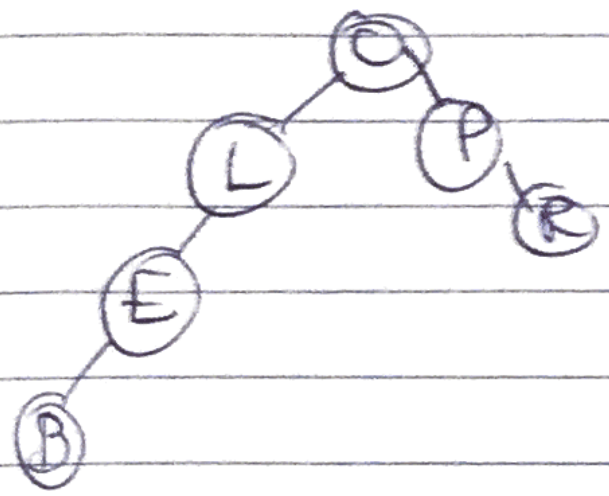
rotate P right



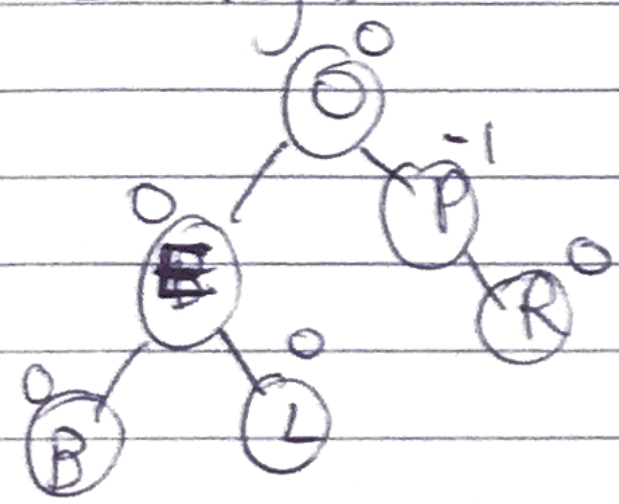
Question 6.

b) cont.

rotate B left



rotate L right



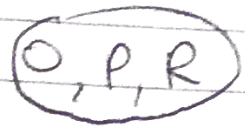
→ balanced

Question 6.

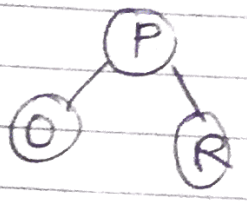
c) insert P

insert R

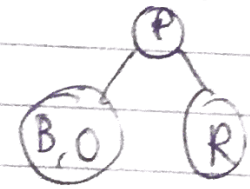
insert O



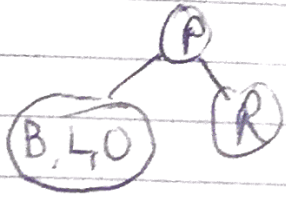
split with P as parent



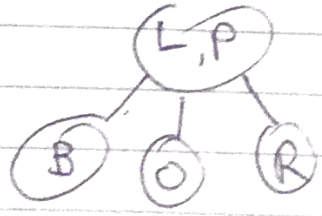
insert B



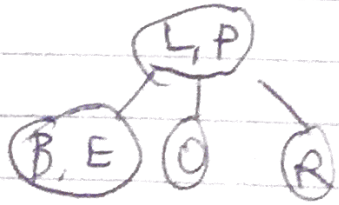
insert L



split with L as parent

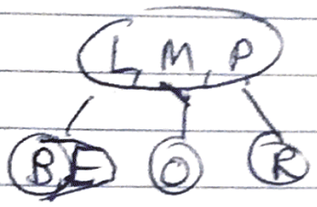


insert E

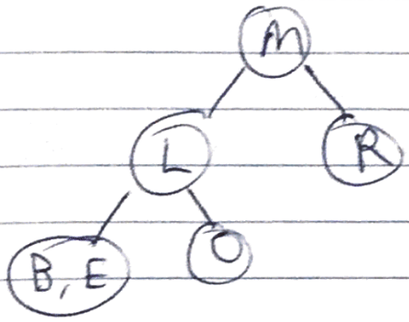


Question 6.

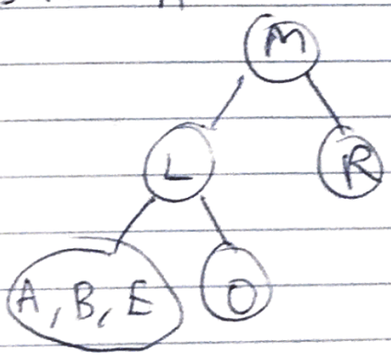
insert M



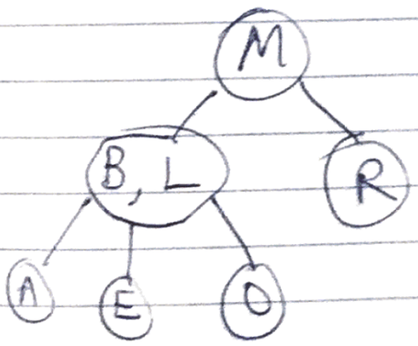
split with M as parent



d) insert A

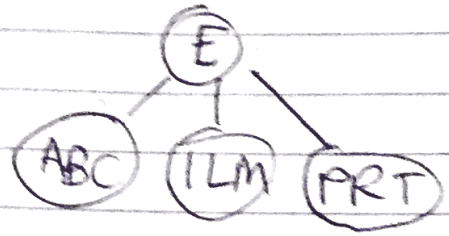


split with M as parent



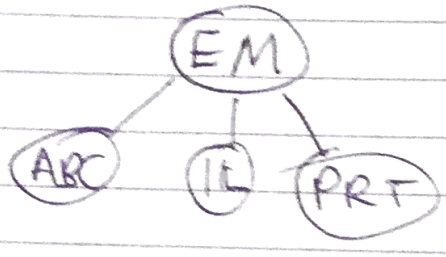
Question 6.

e) Delete O



~~B-Tree is still valid.~~

Root needs ~~at least~~ 2 keys as ~~there~~ we 3 children



Question 7.

a) [9, 4, 8, 1, 2, 3, 5] 1 swap (3 with 9)

b) p = 3

[8, 7, 9, 5, 4, 3, 1, 2]
- swap 2 with 5
- swap 1 with 4
- swap 8 with pivot 3

c) The worst case for insertion sort occurs when an array is ~~sorted~~ in reverse order $\Theta(n^2)$

The worst case for regular mergesort ~~occurs~~ is $\Theta(n \log n)$ as runtime is input insensitive

Therefore the worst case for hybrid sort would occur when each subarray of size 10 ~~or smaller~~ is reverse sorted. ~~There are~~

We stop splitting at $n \leq 10$ so mergesort complexity becomes n elements processed/compared $\log(\frac{n}{10})$ times

Question 7 cont.

c) There are then $\frac{n}{10}$ ~~elements~~ arrays to sort with insertion sort, each with 10 elements max

$\Rightarrow$ Worst case complexity $= \frac{n}{10} \times 10^2 + n \log\left(\frac{n}{10}\right)$

$= 10n + n\log\left(\frac{n}{10}\right) \in \Theta(n \log \frac{n}{10})$

$\in \Theta(10n + n\log\left(\frac{n}{10}\right))$

$\in \Theta(n\log\left(\frac{n}{10}\right)$

house i

houses visited

Question 8.

a)

profit = $P(i, j)$

$P(i, j) = 0$ if $i = 0$ or $j = 0$

i = added house , J = total number of houses

$P(i, 1) = p_i$ = profit of house i