DATA STRUCTURE & ALGORITHMS

SOLUTIONS

1. Which of the following is not an application of Greedy approach?

(i) 0/1 Knapsack
(ii) Fractional Knapsack
(iii) All pair shortest path
(iv) Single Source shortest path
(v) Optimal Merge Patterns

(a) Only (ii), (iv) & (v)
(b) Only (ii), (iii) & (v)
(c) Only (i) & (iii)
(d) Only (i), (iii) & (iv)

Solution: Option (c)

Explanation:
(i) & (iii) uses dynamic programming approach.

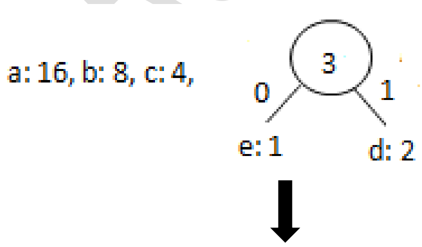
2. Suppose the letters a, b, c, d, e has probability 1/2, 1/4, 1/8, 1/16, 1/32 respectively. Which one of the following is the Huffman code for the letters a, b, c, d, e respectively?

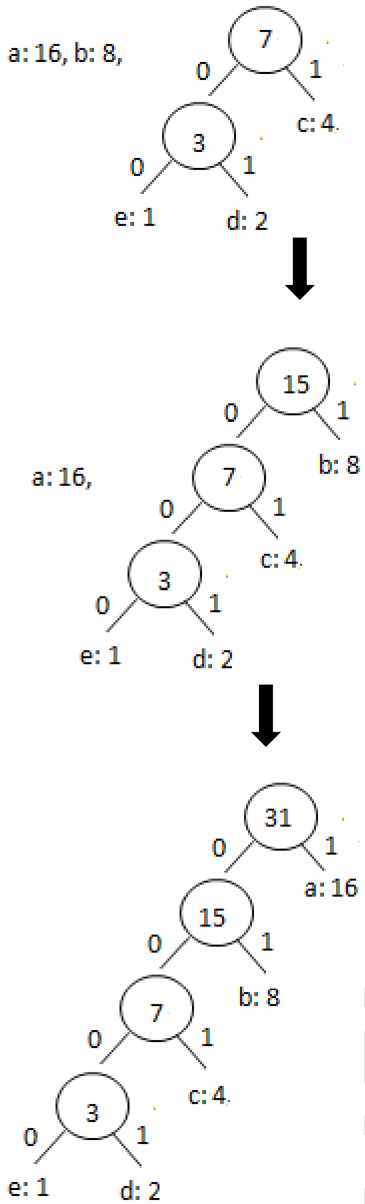
(a) 11, 10, 011, 010, 001
(b) 0, 10, 110, 1110, 1111
(c) 1, 10, 01, 001, 0001
(d) 110, 100, 010, 000, 001

Solution: Option (b) or 1, 01, 001, 0001, 0000

Explanation:
Given Probabilities occuring: a, b, c, d, e as 1/2, 1/4, 1/8, 1/16, 1/32 respectively.
To get the frequencies, we just multiply each probability with 32.

∴ Frequencies are as follows:
a – 16, b – 8, c – 4, d – 2, e – 1





∴ a: 1
b: 01
c: 001
d: 0001
e: 0000

3. Suppose the letters a, b, c, d, e have probabilities 1/2, 1/4, 1/8, 1/16, 1/32 respectively. What is the average number of bits required per message if Huffman code is used for encoding a, b, c, d & e?

(a) 3	(b) 3.75
(c) 1.75	(d) 2.75

Solution: Option (c)

Explanation:
Huffman codes for a, b, c, d, e are 1, 01, 001, 0001, 0000 respectively.

Avg. No. of bits required per message = Σ Pi · No. of bits used by letter i,
where Pi is the probability of occurring of each letter

∴ Average number of bits = $\left(\frac{1}{2} \times 1\right) + \left(\frac{1}{4} \times 2\right) + \left(\frac{1}{8} \times 3\right) + \left(\frac{1}{16} \times 4\right) + \left(\frac{1}{32} \times 4\right) = 1.75$

4. A file contains the following characters and their corresponding frequencies as shown below:
a: 45, b: 13, c: 12, d: 16, e: 9, f: 5

If we use Huffman coding for data comparisons, the average length will be:

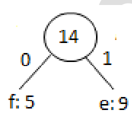
(a) 2.24
(b) 2.48
(c) 1.24
(d) 1.48

Solution: Option (a)

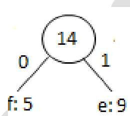
Explanation:
a: 45, b: 13, c: 12, d: 16, e: 9, f: 5

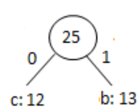
Sort them accordingly the lowest frequency first.

<u>Iteration (i)</u> f: 5, e: 9, c: 12, b: 13, d: 16, a: 45

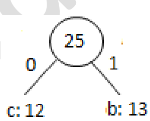
<u>Iteration (ii)</u> c: 12, b:13, 

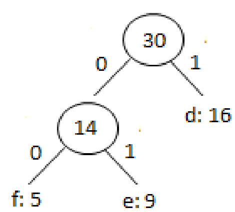
, d: 16, a: 45

<u>Iteration (iii)</u> 

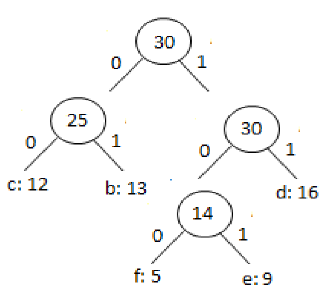
, d: 16, 

,a: 45

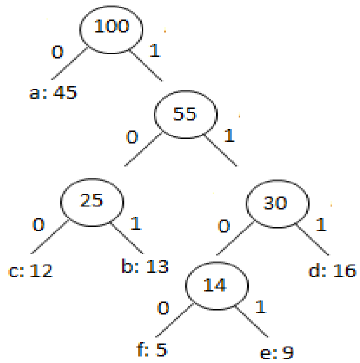
<u>Iteration(iv)</u> 



, a: 45

<u>Iteration (v)</u> a: 45, 

Iteration (vi)



$\therefore$ Average length = $\frac{(45\times1)+(13\times3)+(12\times3)+(9\times4)+(5\times4)}{100} = \frac{224}{100} = 2.24$

5. Total running time of Huffman coding is
(Hint: Consider using Minheap)

(a) $O(n^2)$
(b) $O(n \log n)$
(c) $O(n^2 \log n)$
(d) $O(n^3)$

Solution: Option (b)

Explanation:
To analyze the running time of Huffman's algorithm,
Huffman (c) Algorithm

1. n = |c| //No. of characters in the alphabet

2. Q = c // Build a Min-heap

3. For i =1 to n – 1

4. allocate a new node z

5. z.left = x = Extract – Min(Q)

6. z.right = y = Extract – Min(Q)

7. z.freq = x.freq + y.freq

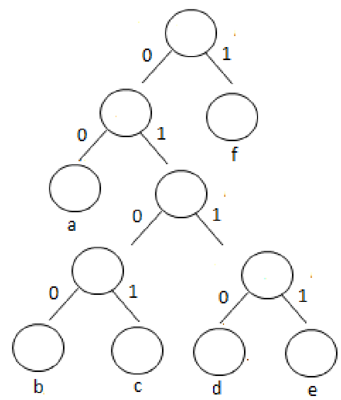
8. Insert (Q, z)

9. return Extract min(Q) // Return the root of the tree.

In the above algorithm, line 2 takes O(n) time to build Minheap. The for loop execute exactly n – 1 times during which Extract – Min(Q) takes O(log n) time. Thus the for loop contributes O(n log n) to the running time.

∴ O(n) + O(n log n) ≃ O(n log n)

6. The tree shown below is a Huffman code tree for the letters a, b, c, d, e & f. What would be the encoding of the string edaa?



(a) 011011100000
(b) 011101100000
(c) 011101110000
(d) 011011000000

Solution: Option (b)

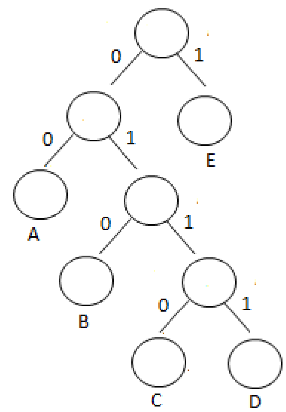
Explanation:

a: 00
b: 0100
c: 0101
d: 0110
e: 0111
f: 1

Now, we encode for "edaa"
011101100000

7. Given a Huffman code tree for the letters A, B, C, D & E. What is the last character in the string encoded by

01110101110110



(a) A (b) B
(c) C (d) D

Solution: Option (c)

Explanation:

0111 010 1 1 1 0110
E B E E E C

8. Which of the following is True about Huffman's Coding?

(i) Huffman code is a data compression, encoding technique that follows greedy approach
(ii) An optimal code is always represented by a full binary tree

(a) Only (i) (b) Only (ii)
(c) Both (i) & (ii) (d) Neither (i) nor (ii)

Solution: Option (c)

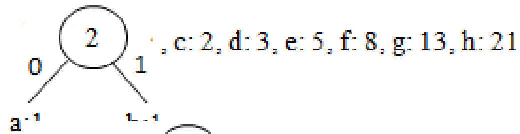
9. What would be the optimal Huffman code for the string "abbbce" for the given set of frequencies, based on the first 8 Fibonacci numbers?

(a) 1111100000111110111111111110
(b) 1000001111000011100000011111
(c) 1111111000111110001111111110
(d) 1000001110000111100100111111

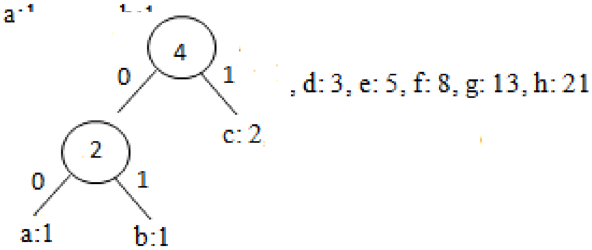
Solution: Option (a)

Explanation:

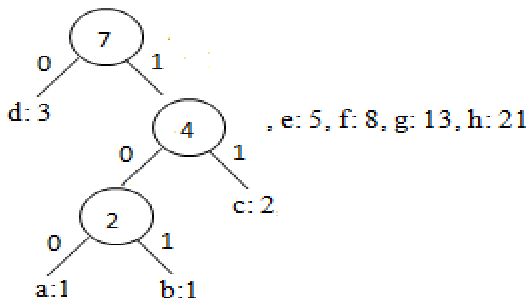
Iteration 1:



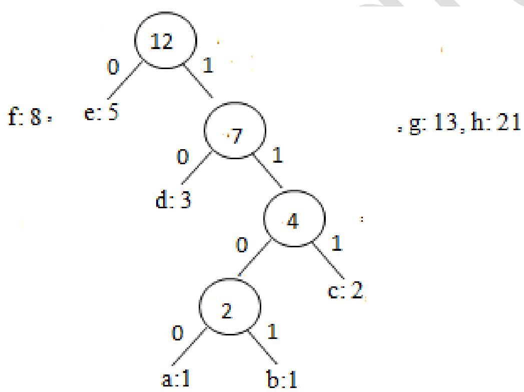
Iteration 2:



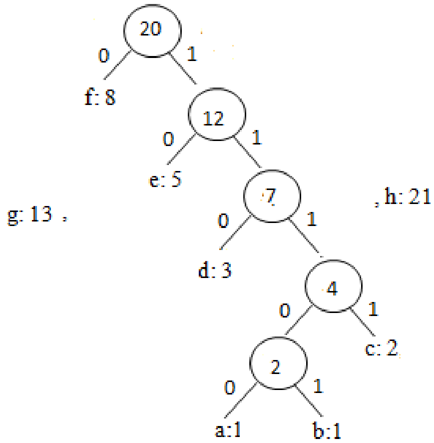
Iteration 3:



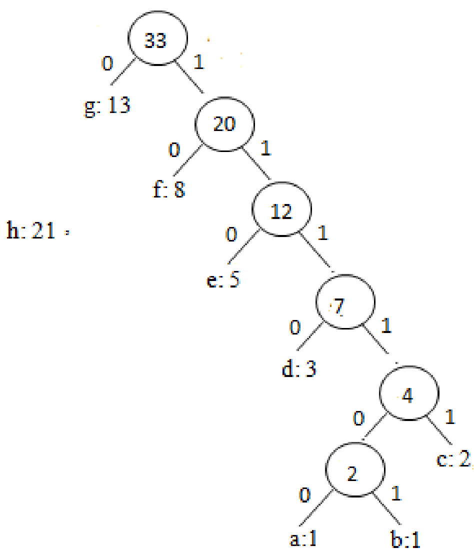
Iteration 4:



Iteration 5:

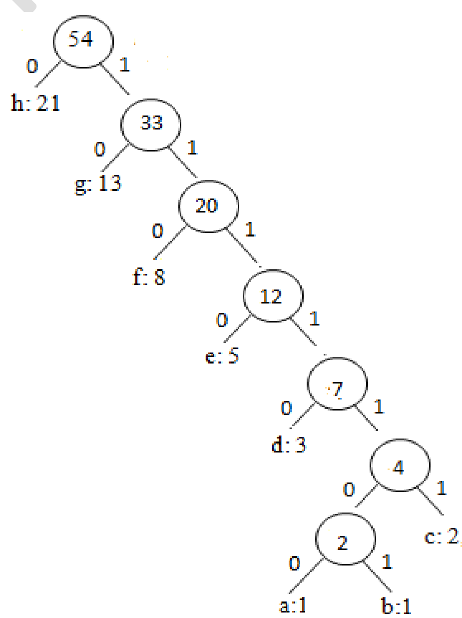


Iteration 6:



Iteration 7:

a: 1111100
b: 1111101
c: 111111
d: 11110
e: 1110
f: 110
g: 10
h: 0



ahhhbce

↓

1111100000111101111111111110

10. Consider a set of 4 messages (M_1 – M_4) whose frequency of occurrences in the text is as given:

(0.37, 0.51, 0.05, 0.07)

Using frequency dependent Huffman Coding the codes of the messages M_2 and M_3 respectively.

(a) 0, 110
(b) 0, 011
(c) 1, 000
(d) 1, 001

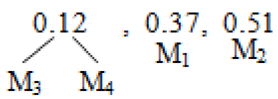
Solution: Option (c)

Explanation:

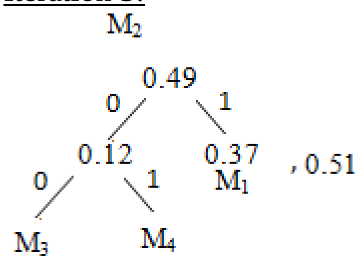
Iteration 1:

0.05, 0.07, 0.37, 0.51
M_3 M_4 M_1 M_2

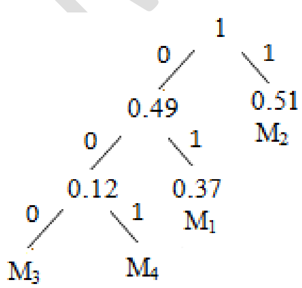
Iteration 2:



Iteration 3:



Iteration 4:



M_1 – 01
M_2 – 1
M_3 – 000
M_4 – 001