Semester: 5th
Subject Name:- DAA & Code:- CS 2012
Branch (s): CSE, IT, CSCE, CSSE, ECS

AUTUMN MID SEMESTER EXAMINATION-2022

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Design & Analysis of Algorithms
[CS 2012]

Time: 1 1/2 Hours **Full Mark: 20**

Answer any four Questions including Q.No.1 which is Compulsory.
The figures in the margin indicate full marks. Candidates are required to give theiranswers in their own words as far as practicable and all parts of a question should beanswered at one place only.

Solution and Evaluation Scheme

1. Answer all the questions. [1 x 5]

a) log(n!) = O(n log n) True/False? Justify your answer.

Answer/Solution:

True

Justification

log 1 ≤ log n
log 2 ≤ log n
… … …
log n ≤ log n

Adding up both the LHS and the RHS, we get

⇒ log1 + log 2+ . . .+ log n ≤ n log n

⇒ log(1 · 2 · . . . n) ≤ n log n

⇒ log(n!) ≤ n log n

As per definition of O-Notation the above inequality is valid for c=1 and n_0=1.

Scheme

- Correct answer with explanation -1 Mark
- Written answer as true without explanation -0.5 mark

b) Consider the following function:

```
int FUN(int n)
{
  int i, j, k = 0;
  for (i  = n; i>0; i--)
      for (j = 1; j <= n; j = j * 2)
          k = k + 2/n;
  return k;
}
```

What is the most approximate returned value of the above function?

A) Θ(log n) B) $\Theta(n^2 \log n)$ C) $\Theta(n^3)$ D) $\Theta(n^3 \log n)$

Answer/Solution:A

Scheme

- Correct choice-1 Mark

c) Where you can find the largest element in a min-heap? Justify your answer by taking a suitable example.

Answer/Solution:

One of the leaf nodes

Scheme

- Correct answer with example-1 Mark

d) Given items as {value, weight} pairs {{30,10},{40,20},{20,5}}. The capacity of knapsack=10. Find the maximum value output assuming items to be divisible and non-divisible respectively.

A.30, 30 B.30, 35 C.35, 30 D.35, 35 E.NONE

Answer/Solution:

C or E

Scheme

- Correct choice-1 Mark

e) Given 5 activities, A=< a1, a2,….,a5 > along with their start time (s_i) and finish time (f_i) are given as follows:

i	1	2	3	4	5
si	3	2	5	4	6
fi	4	4	7	8	9

Which of the following activity will never participate in finding all possible schedules in activity selection problem.

A.a2 B.a3 C.a4 D.a5 E.NONE

Answer/Solution:

E

Scheme

- Correct choice-1 Mark

2. In a class, there are m boys and n girls. Their CGPAs are stored in two arrays B & G, one for the boys(B) in descending order, the other for the girls(G) in ascending order. Devise an efficient O(m+n) algorithm to find out the set (A) of duplicate CGPAs that are common between both the boys and girls in ascending order.

Sample Input=>

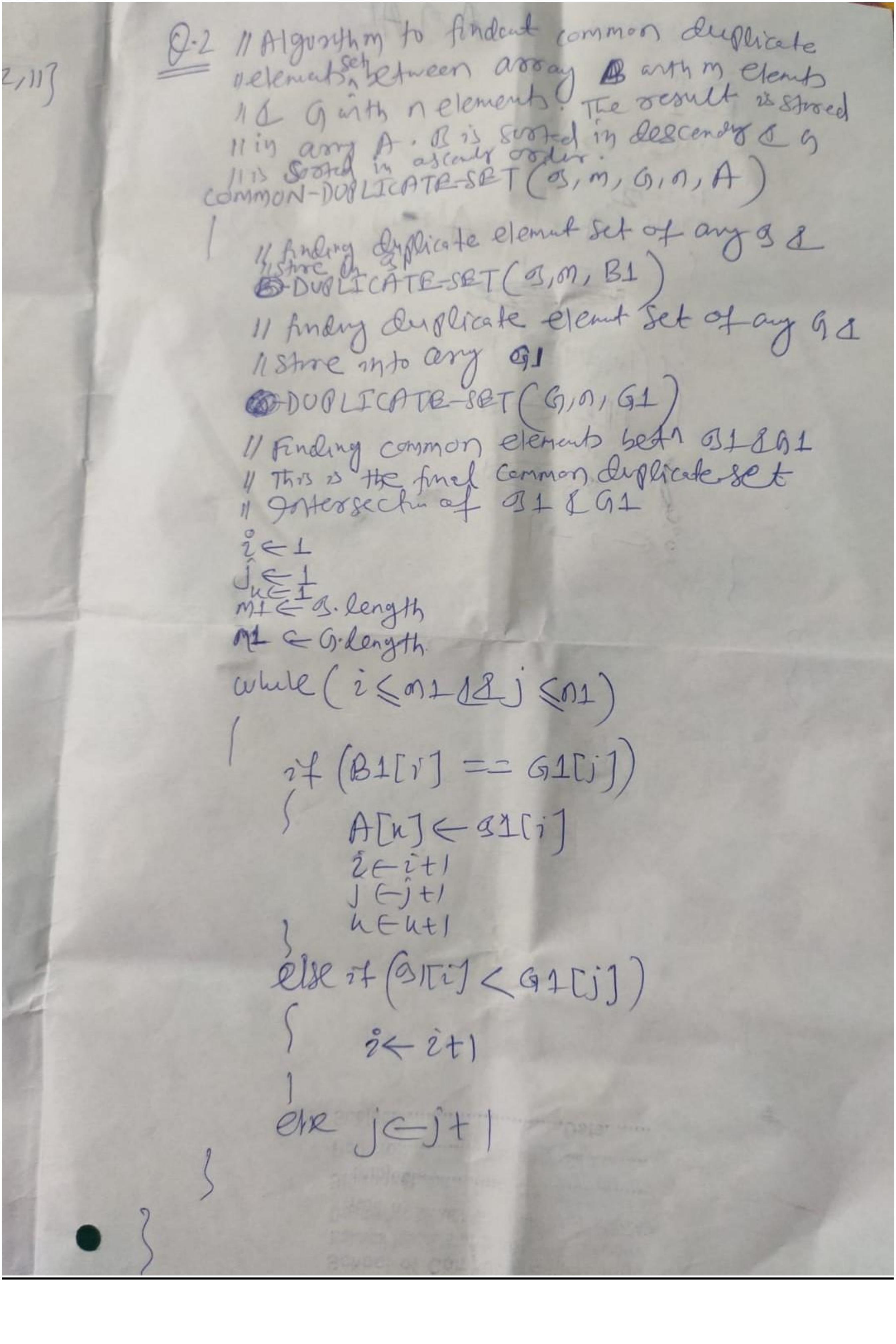
Array B={9.5, 9.0, 8.0, 8.0, 8.0, 7.5, 6.5, 5.5, 4.5, 4.5, 4.5, 4.0, 4.0}

Array G={4.5, 4.5, 5.5, 6.5, 6.5, 6.5, 8.0, 8.0, 9.5, 9.5}

Output => Array A={4.5, 8.0}

[5 Marks]

Answer/Solution:



2,11}

Q.2

```
// Algorithm to findout common duplicate
// elements set between array B with m elemts
// & G with n elements. The result is stored
// in arry A. B is sorted in descendg & G
// is sorted in ascendg order.
COMMON-DUPLICATE-SET(B, m, G, n, A)
{
    // finding duplicate element set of arry B &
    // store in B1
    DUPLICATE-SET(B, m, B1)
    // finding duplicate element set of arry G &
    // store into arry G1
    DUPLICATE-SET(G, n, G1)
    // Finding common elements betn B1 & G1
    // This is the final common duplicate set
    // intersection of B1 & G1
    i ← 1
    j ← 1
    k ← 1
    m1 ← B.length
    n1 ← G.length
    while(i ≤ m1 && j ≤ n1)
    {
        if (B1[i] == G1[j])
        {
            A[k] ← B1[i]
            i ← i+1
            j ← j+1
            k ← k+1
        }
        else if (B1[i] < G1[j])
        {
            i ← i+1
        }
        else j ← j+1
    }
}
```

Scheme:

- **Correct algorithm by any method with time complexity explanation: 5 mark**
- **Partial Correct: step marks (0.5 to 3 marks)**
- **Wrong answer with no proper approach: 0 mark**

3. a) Write a recursive algorithm named as FIND-ARRAY-MAX(A, n) to compute the maximum element in an array of n integers by assuming the existence of a function “max(x, y)” that returns the maximum of two integers x and y. [2.5 Marks]

b) What is the exact comparison complexity of FIND-ARRAY-MAX(A, n) algorithm? Derive a recurrence relation and solve it to justify your answer. [2.5 Marks]

Answer/Solution:

a) Recursive algorithm to compute the maximum element in an array of n integers

```
FIND-ARRAY-MAX(A, n)
{
if (n = 1) then
    return(A[1])
else
    return(max(A[n], FIND-ARRAY-MAX (A, n − 1) ))
end if
```

Scheme:

- **Correct algorithm by any method : 2.5 mark**
- **Partial Correct: step marks (0.5 to 2 marks)**
- **Wrong answer with no proper approach: 0 mark**

b) Exact comparison complexity of FIND-ARRAY-MAX(A, n) algorithm

The function max(x, y) uses exactly one comparison. Thus, the comparison complexity of

FIND-ARRAY-MAX(A, n) can be described the recurrence relation:

$T(1) = 0$, if $n=1$

$T(n) = T(n - 1) + 1$, if $n > 1$

This recurrence can be expanded as $T(n) = 1 + 1 + \ldots +1$ $(n - 1)$ times to give $T(n) = n - 1$

$T(n)= \theta(n)$

Scheme:

- **Correct recursive equation with solution : 2.5 mark**
- **Partial Correct: step marks (0.5 to 2 marks)**
- **Wrong answer with no proper approach: 0 mark**

4. Write an algorithm MAX-HEAP-DELETE(A, n, i) to delete an element at index i, in a n-element max heap A, rooted is at index 1, by assuming the existence of two algorithms HEAP-INCREASE-KEY(A, n, i, key) and MAX-HEAPIFY(A, n, i), where,

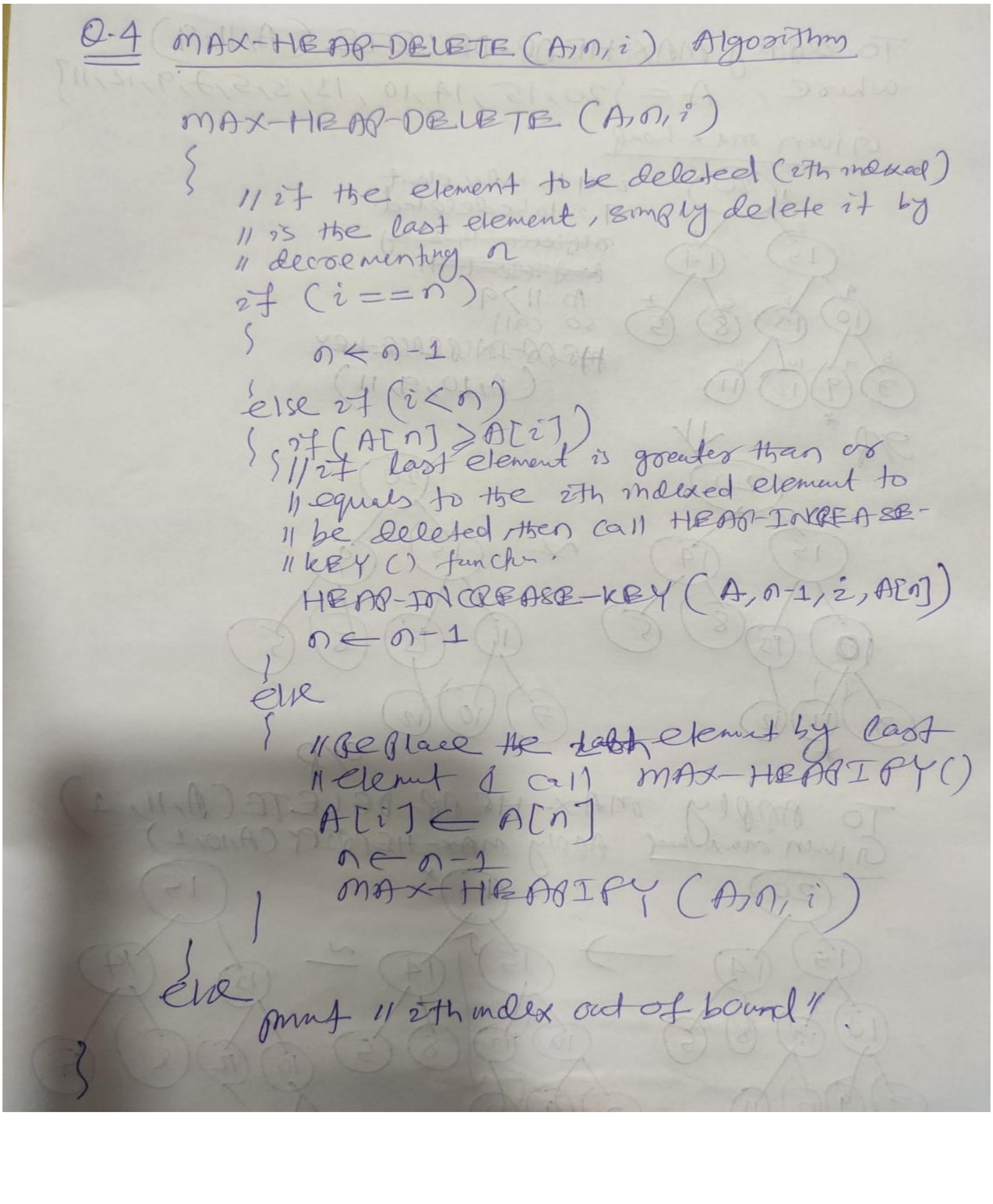
HEAP-INCREASE-KEY(A, n, i, key) is an algorithm that rebuild the n-element max-heap A if value at index i increases to the new value key, which is assumed to be at least as large as i's current key value, else display appropriate error message.

MAX-HEAPIFY(A, n, i) is an algorithm where, the value at A[i] may "float down" in the n-elemented max-heap A so that the subtree rooted at index i obeys the max-heap value property.

Apply MAX-HEAP-DELETE(A, 11, 9) and MAX-HEAP-DELETE(A, 11, 1) separately to the following max-heap array A.

A={20, 15, 14, 10, 13, 8, 5, 7, 9, 12, 11} [5 Marks]

Answer/Solution:



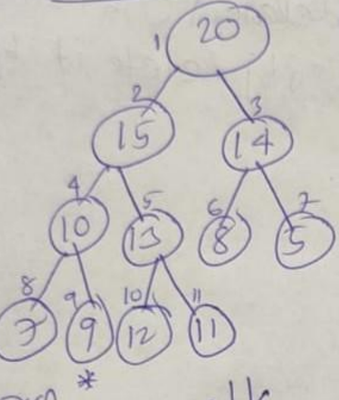
Q-4 MAX-HEAP-DELETE (A, n, i) Algorithm

```
MAX-HEAP-DELETE (A, n, i)
{
    // if the element to be deleted (ith indexed)
    // is the last element, simply delete it by
    // decrementing n
    if (i == n)
    {
        n ← n - 1
    }
    else if (i < n)
    {  if (A[n] ≥ A[i])
       {  // if last element is greater than or
          // equals to the ith indexed element to
          // be deleted, then call HEAP-INCREASE-
          // KEY () function.
          HEAP-INCREASE-KEY (A, n-1, i, A[n])
          n ← n - 1
       }
       else
       {
          // Replace the ~~last~~ element by last
          // element & call MAX-HEAPIFY()
          A[i] ← A[n]
           n ← n - 1
           MAX-HEAPIFY (A, n, i)
       }
    }
    else
       print " ith index out of bound ".
}
```

Q-9 continue..

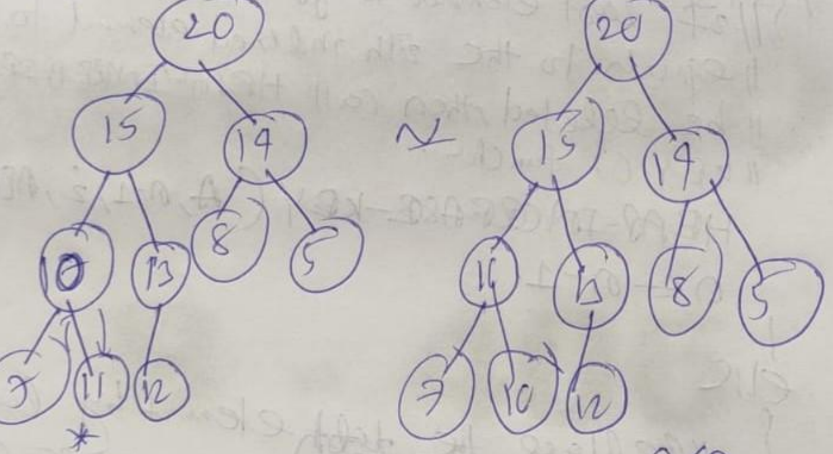
To apply MAX-HEAP-DELETE (A, 11, 9)
where, A = {20, 15, 14, 10, 13, 8, 5, 7, 9, 12, 11}

Given max-heap



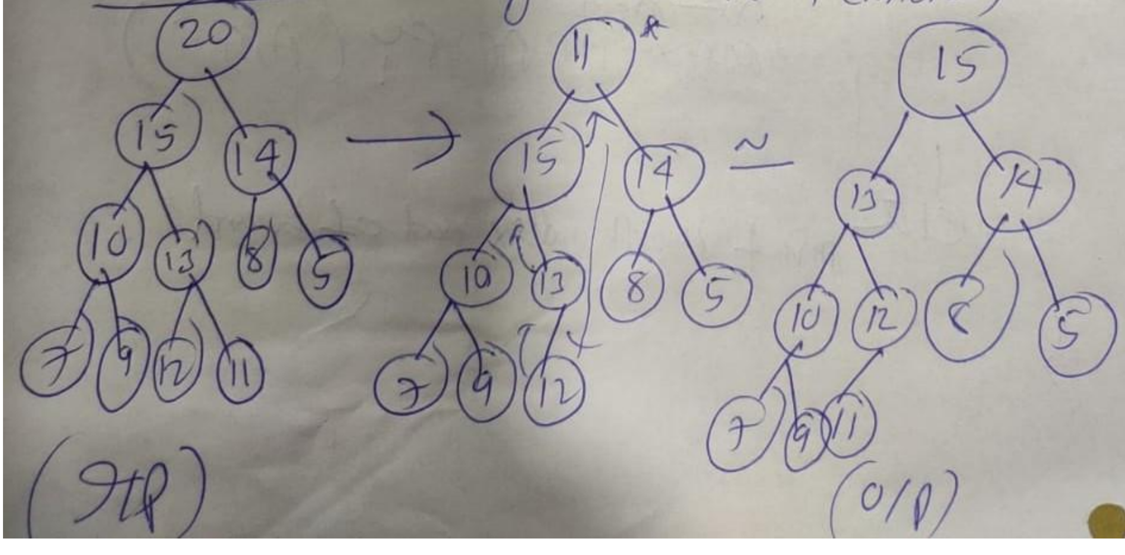
i=9 th index elemt
9 is to be deleted.
~~replace this by~~
~~last elemt 11~~
As 11>9
so call
HEAP-INCREASE-KEY
(A, 10, 9, 11)

I/P



O/P

To Apply MAX-HEAP-DELETE(A, 11, 1)
Given max-heap Apply MAX-HEAPIFY (A, 10, 1)



(I/P)

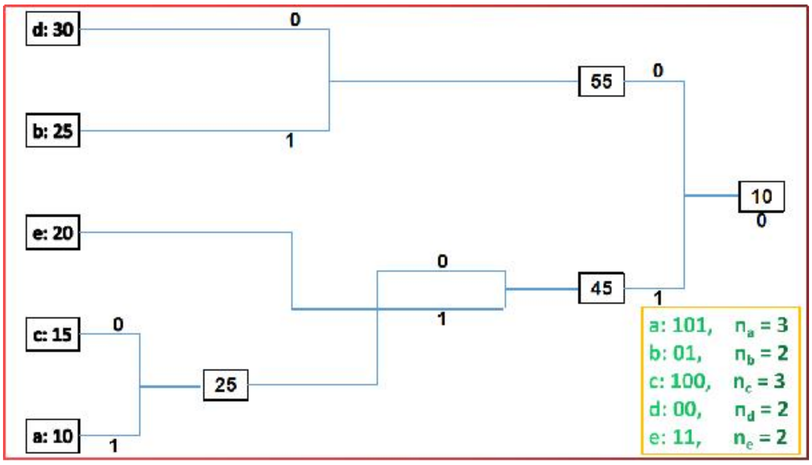
(O/P)

Scheme:

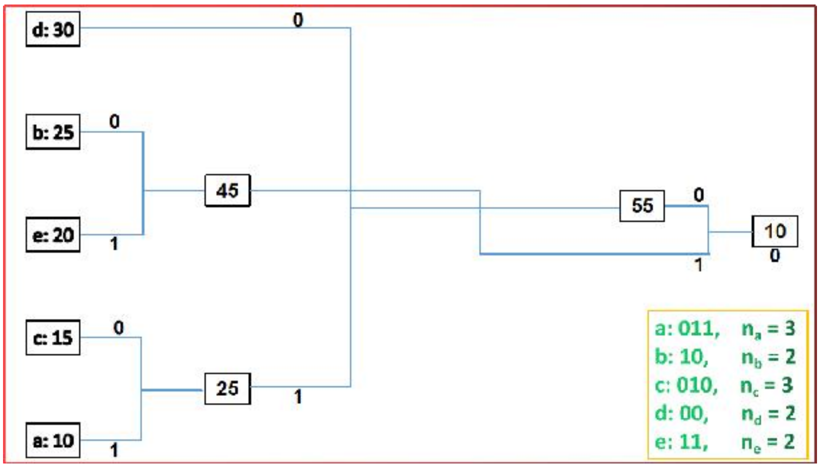
- **Correct Algorithm --2 marks**
- **MAX-HEAP-DELETE(A, 11, 9) and MAX-HEAP-DELETE(A, 11, 1) solutions - 3 Marks**
- **Partial Correct: step marks (0.5 to 3 marks)**
- **Wrong answer with no proper approach: 0 mark**

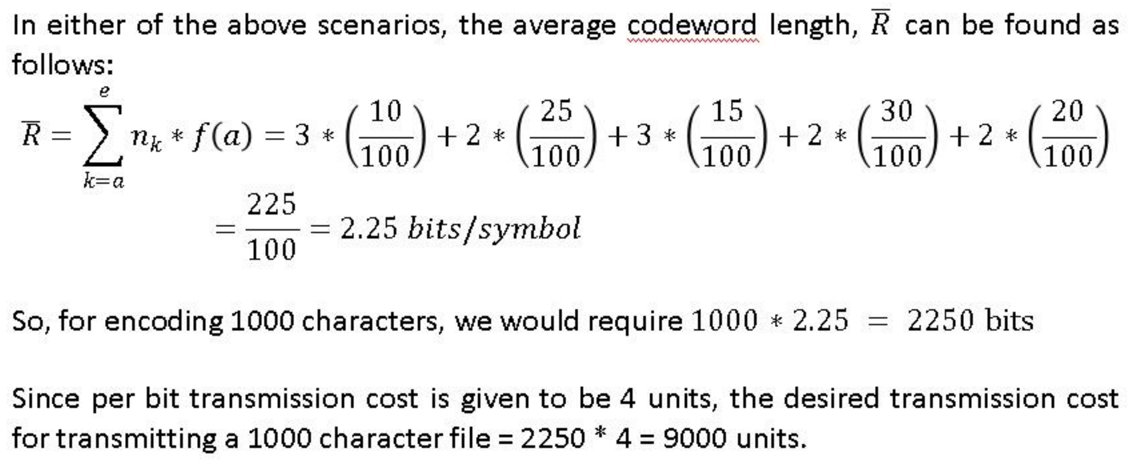
5. Suppose a file to be transferred through the network contains the following characters with their number of occurrences as < a: 10, b: 25, c: 15, d: 30, e: 20 >. Determine an efficient strategy that can minimize the total cost of transferring that file of 1000 characters. Find out the total cost of transfer if transferring cost for 1-bit of data is 4 units. [5 Marks]

Solution Possibility 1:



Solution Possibility 2:





In either of the above scenarios, the average codeword length, $\overline{R}$ can be found as follows:

$$\overline{R} = \sum_{k=a}^{e} n_k * f(a) = 3 * \left(\frac{10}{100}\right) + 2 * \left(\frac{25}{100}\right) + 3 * \left(\frac{15}{100}\right) + 2 * \left(\frac{30}{100}\right) + 2 * \left(\frac{20}{100}\right)$$

$$= \frac{225}{100} = 2.25\ bits/symbol$$

So, for encoding 1000 characters, we would require $1000 * 2.25 = 2250$ bits

Since per bit transmission cost is given to be 4 units, the desired transmission cost for transmitting a 1000 character file = 2250 * 4 = 9000 units.

Scheme:

- **Correct possible solutions with huffman code and total cost: 5 mark**
- **Partial Correct: step marks (0.5 to 3 marks)**
- **Wrong answer with no proper approach: 0 mar**k