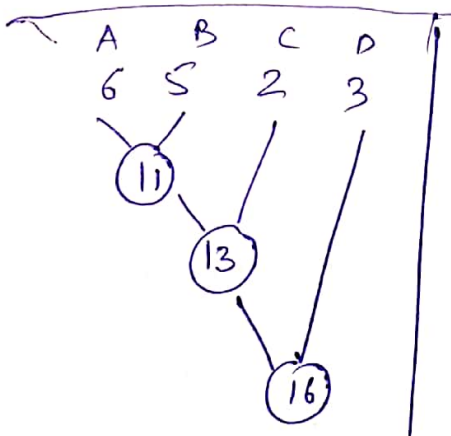
OPTICAL MERGE PATTERN

It relates to the merging of two or more sorted files in a single sorted file. This type of merging can be done by Two-way Merging method.

There are multiple ways to perform pairwise merge to get single sorted file, we have to select one which require minimum no. of comparisons.

List :- A B C D

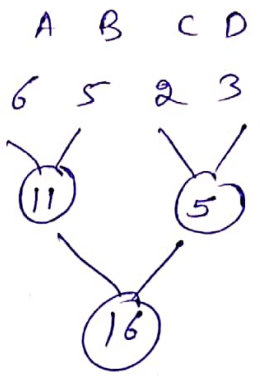
Sizes :- 6 5 2 3



11 + 13 + 16

Total Cost ⇒ = 40

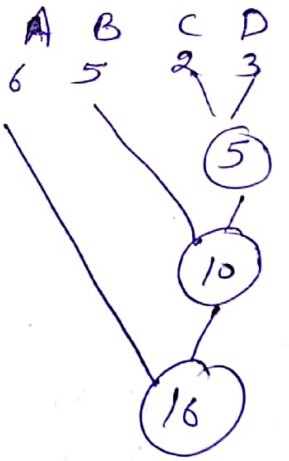
Pattern I



11 + 5 + 16

Total Cost ⇒ = 32

Pattern II



16 + 10 + 5

Total Cost = 31

Pattern III

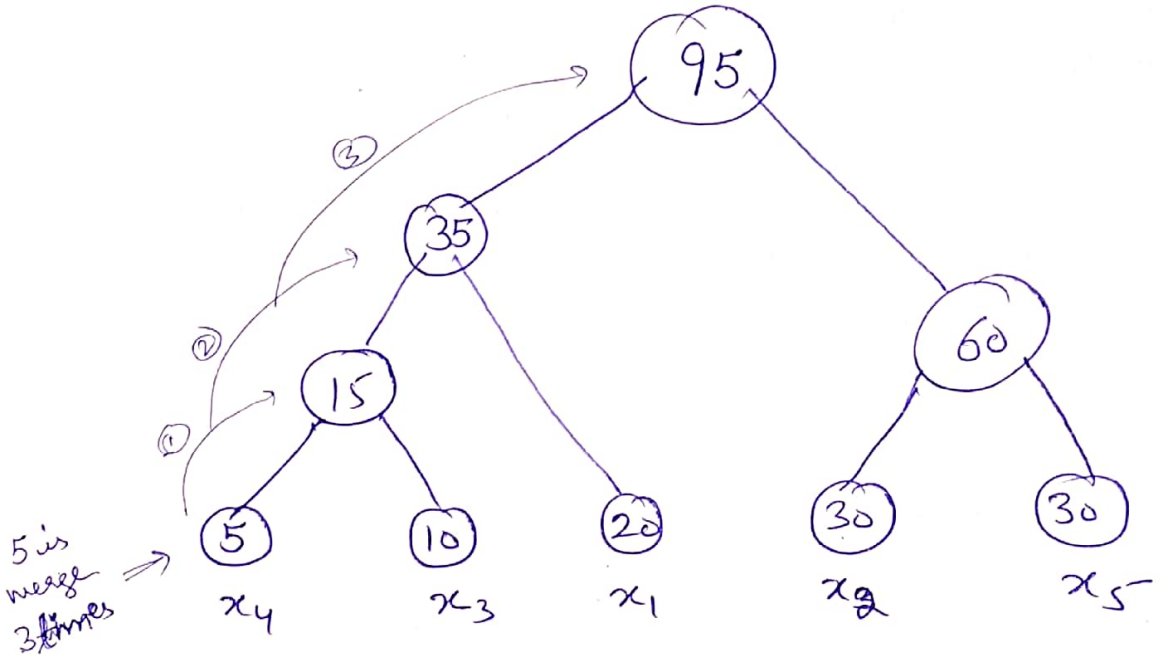
In this we select two smaller list for Merging that Give Best Result i.e. Minimum result (total cost)

Example :-1)

x_1	x_2	x_3	x_4	x_5
20	30	10	5	30

~~Select~~ Convert into increasing order →

x_4	x_3	x_1	x_2	x_5
5	10	20	30	30



Total Cost ⇒ 15 + 35 + 60 + 95 = 205
(Internal Node Size)

OR

Total Cost ⇒ How many time the No. of file being Merged that depends on the distance.

5 →	is merge	3 times
10 →	" "	3 "
20 →	" "	2 "
30 ⇒	" "	2 "
30 →	" "	2 "

⇒ 3x5 + 3x10 + 2x20 + 2x30 + 2x30

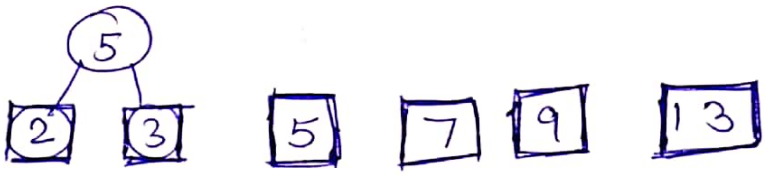
= 205

Example :-

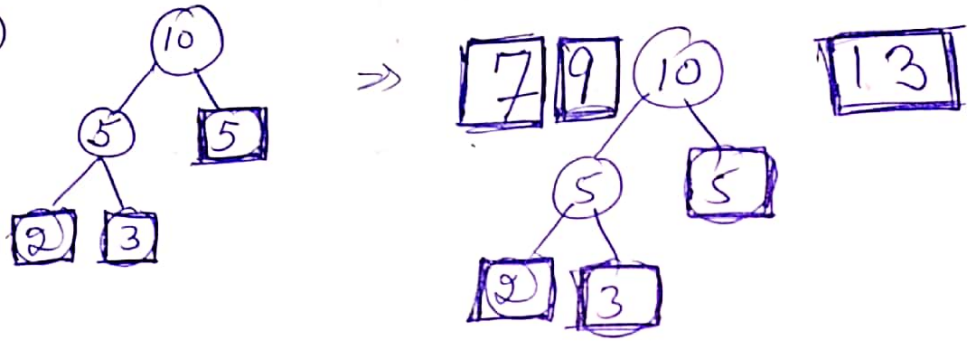
2) Files → x_1 x_2 x_3 x_4 x_5 x_6

Sizes → 2 3 5 7 9 13 (Sorted if not first arrange in increasing order)

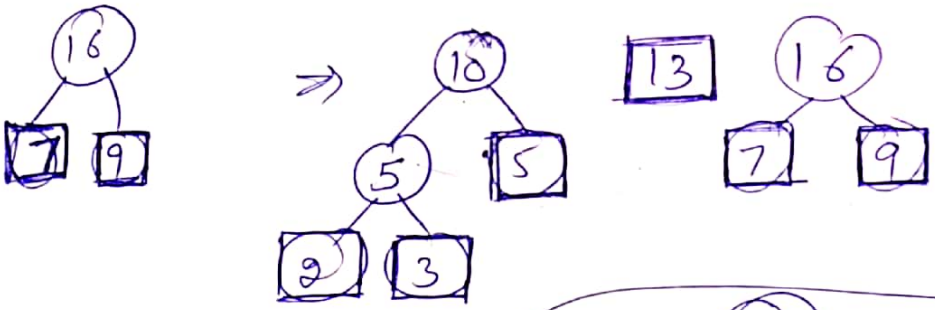
(i)



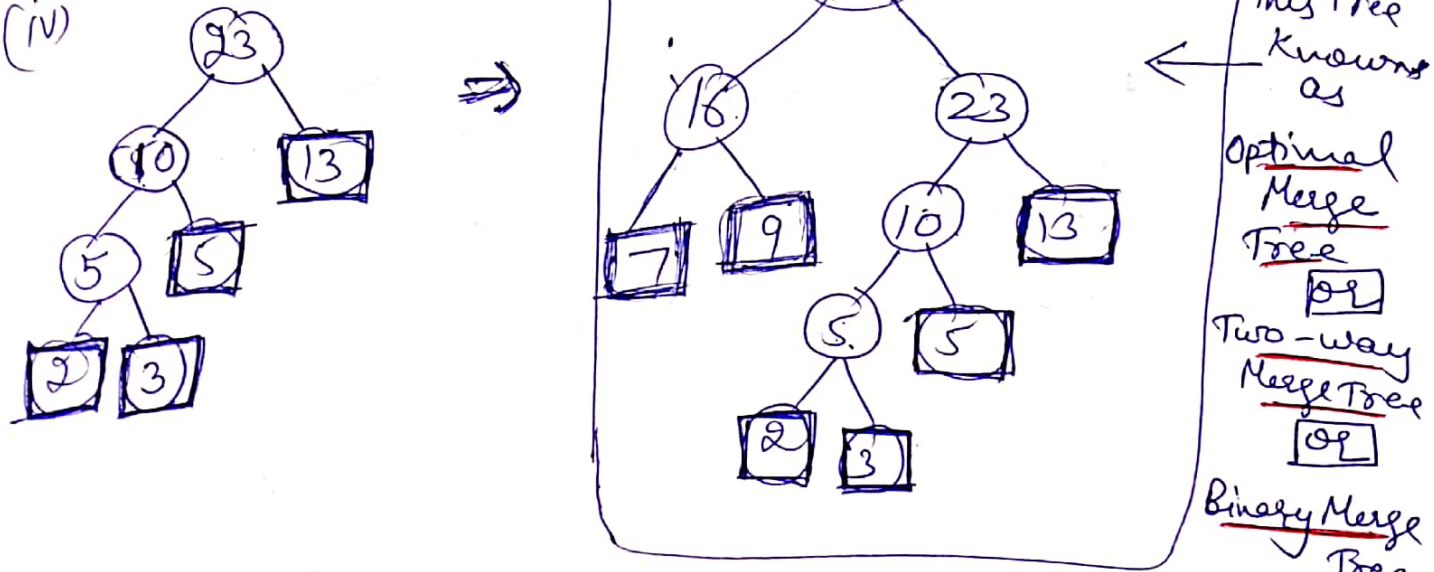
(ii)



(iii)



(iv)



This Tree knowns as Optimal Merge Tree OR Two-way Merge Tree OR Binary Merge Tree

Total Cost = 5 + 10 + 23 + 16 + 39

Optimal = 93

OR

Total cost of Merging $\Rightarrow$ $\boxed{\sum x_i * d_i}$

$= 4 \times 2 + 3 \times 4 + 5 \times 3 + 7 \times 2 + 9 \times 2 + 13 \times 2$

$= 8 + 12 + 15 + 14 + 13 + 26$

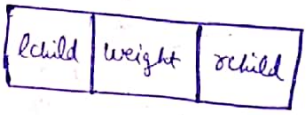
$= \textcircled{93}$ Ans.

Algorithm for optimal Merge Pattern

Algorithm:- MergeTree(n) ← n single node

```
{
// list is a global list of
   n single nodes.
for i = 1 to n-1 do
{
   p = new treenode; // get a new treenode
   (p→lchild) = least(list); // least func finds node with least length of a list.
   (p→rchild) = least(list); //
   (p→weight) = ((p→lchild)→weight) + ((p→rchild)→weight);
                // Merge two trees with smallest weight.
   Insert(list, p); // Insert function insert p into list.
}
return list; // Tree left in list is the Merge Tree
}
```

```
Struct treenode
{
   treenode *lchild;
   treenode *rchild;
   int weight;
};
```



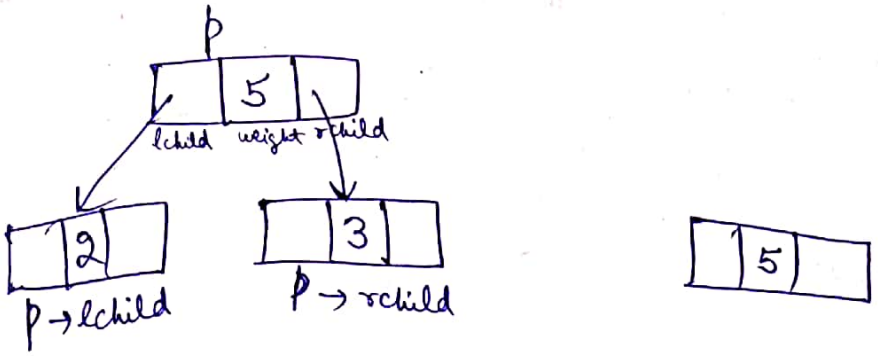
Example:- list → x_1 x_2 x_3

weight → 2 3 5

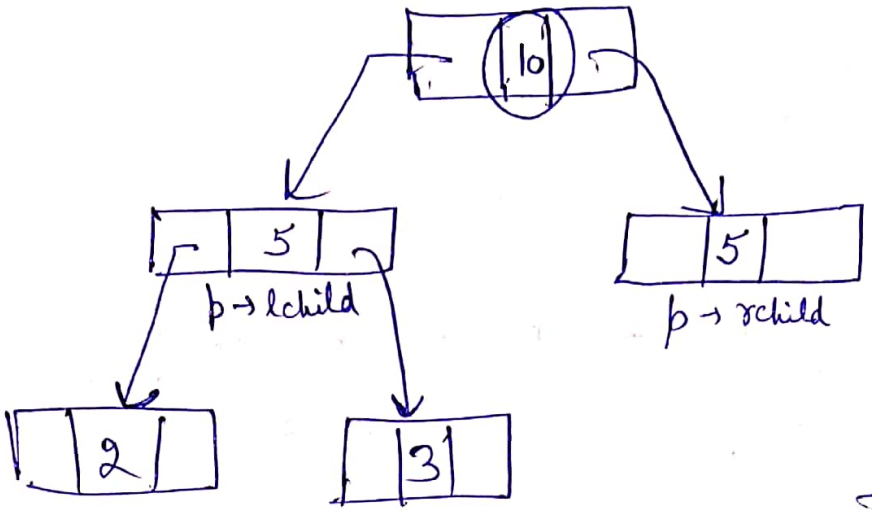
$n = 3$

$i = 1$ to 2

$i = 1$



i=2



Optimal Merge Tree