Experiment No: 7

Title:Write program to find Optimal Merge Pattern

Theory/Description:

Optimal merge pattern is a pattern that relates to the merging of two or more sorted files in a single sorted file. This type of merging can be done by the two-way merging method.

If we have two sorted files containing n and m records respectively then they could be merged together, to obtain one sorted file in time O (n+m). If the number of sorted files are given, there are many ways to merge them into a single sorted file. This merge can be performed pair wise. Hence, this type of merging is called as 2-way merge patterns.

As, different pairings require different amounts of time, in this strategy we want to determine an optimal way of merging many files together. At each step, two shortest sequences are merged.

Example

Let us consider the given files, f1, f2, f3, f4 and f5 with 20, 30, 10, 5 and 30 number of elements respectively.

If merge operations are performed according to the provided sequence, then

M1 = merge f1 and f2 => 20 + 30 = 50
M2 = merge M1 and f3 => 50 + 10 = 60

M3 = merge M2 and f4 => 60 + 5 = 65

M4 = merge M3 and f5 => 65 + 30 = 95

Hence, the total number of operations is 50 + 60 + 65 + 95 = 270

Now, the question arises is there any better solution?

Sorting the numbers according to their size in an ascending order, we get the following sequence −

f4, f3, f1, f2, f5

Hence, merge operations can be performed on this sequence

M1 = merge f4 and f3 => 5 + 10 = 15

M2 = merge M1 and f1 => 15 + 20 = 35

M3 = merge M2 and f2 => 35 + 30 = 65

M4 = merge M3 and f5 => 65 + 30 = 95

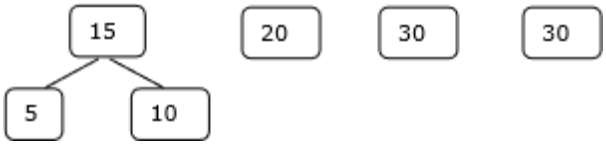
Therefore, the total number of perations is 15 + 35 + 65 + 95 = 210

Obviously, this is better than the previous one.

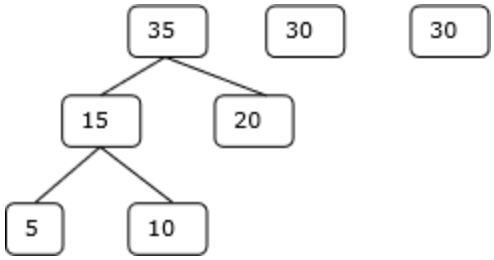
In this context, we are now going to solve the problem using this algorithm.

Initial Set

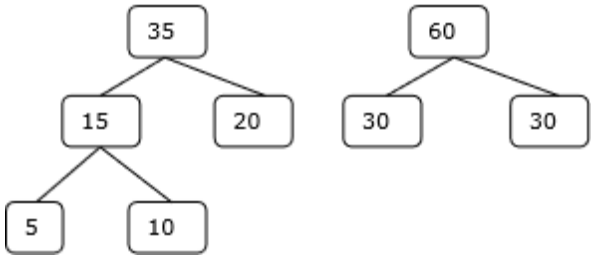
Step-1



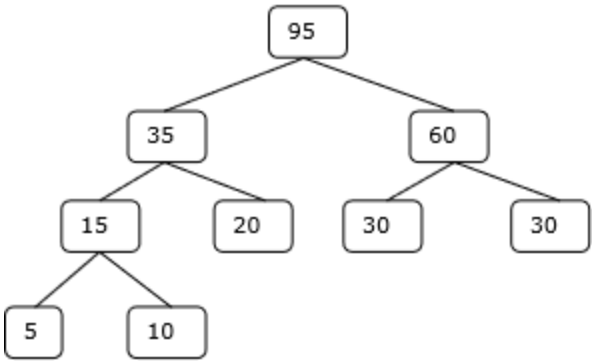
Step-2



Step-3



Step-4



Hence, the solution takes 15 + 35 + 60 + 95 = 205 number of comparisons.

To merge a p-record file and a q-record file requires possibly p + q record moves, the obvious choice being, merge the two smallest files together at each step.

Two-way merge patterns can be represented by binary merge trees. Let us consider a set of n sorted files {f1, f2, f3, …, fn}. Initially, each element of this is considered as a single node binary tree. To find this optimal solution, the following algorithm is used.

```
treenode = record
{
treenode* lchild
treenode* Rchild;
integer weight;
}
```

```
Algorithm: TREE (n)
//list is the list of n single nodes
{
        For i=1 to n-1 do

        {

// get a new tree node

        Pt:= new treenode;

// merge two trees with smallest  length

        (Pt -> lchild) = least(list);

        (Pt -> rchild) = least(list);

        (Pt ->weight) = ((Pt -> lchild)  -> weight) + ((Pt -> rchild) -> weight);

        Insert  (list , Pt);

        }

        // tree left in list

        Return least(list);

        }
```

At the end of this algorithm, the weight of the root node represents the optimal cost.

Self Study: Find the complexity of optimal merge pattern algorithm and justify its compexity