Optimal merge patterns

Introduction:

- Merge two files each has n & m elements, respectively: takes O (n+m).
- Given n files

  What's the minimum time needed to merge all n files?

Example:

(F1, F2, F3, F4, F5)= (20, 30, 10, 5, 30).

M1= F1 & F2 Þ 20+30 = 50

M2= M1 & F3 Þ 50+10 = 60

M3= M2 & F4 Þ 60+5 = 65

M4= M3 & F5 Þ 65+30 = 95

270

Optimal merge pattern: Greedy method.

Sort the list of files:

(5,10, 20, 30, 30)= (F4, F3, F1, F2, F5)

Merge file using **2-way merge**

1. **Merge two file at a time.**
2. **Add merge file into the list of files in sorted order.**

Merge first two files

(5,10, 20, 30, 30)=> (15, 20, 30, 30)

Merge next two files

(15, 20, 30, 30)=> (30, 30, 35)

Merge next two files

(30, 30, 35) => (35, 60)

Merge next two files

(35, 60)=> (95)

Total time=15+35+60+95=205.

Problem:

- Input: Given n sorted files
- Output: Merge n files in a minimum amount of time.

Working of algorithm with example:

Consider a pool of file L which contain n files.

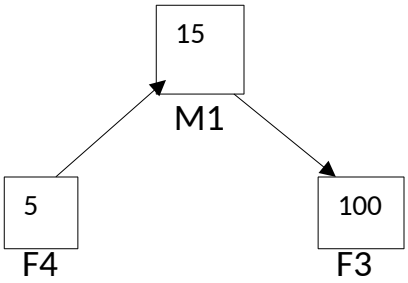
Step 1: find first two min file .

Step -2: merge them & add new merge file to the pool L.
Step-3 : repeat step-1 & 2 until no file remain in the pool L for merging.
e.g.

L (F1, F2, F3, F4, F5)= (20, 30, 10, 5, 30).

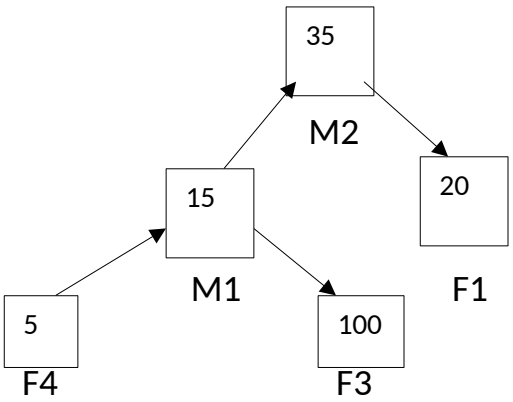
Find first two min files which are F4 and F3

Next merge them & add new file M1 to pool L



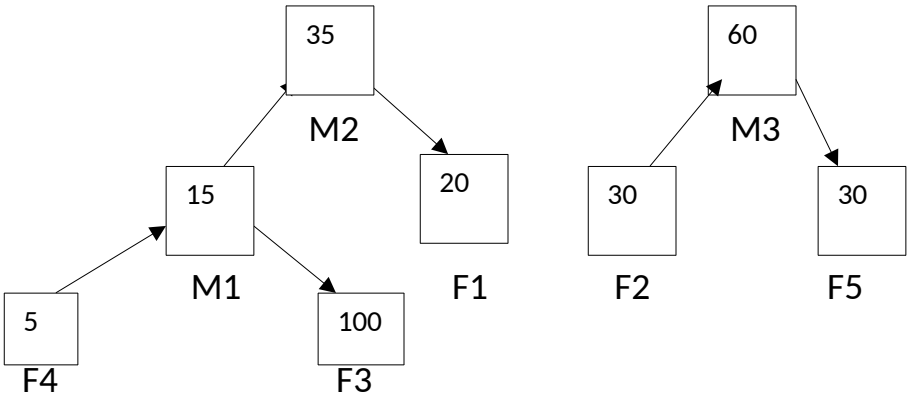
L (F1, F2, M1, F5)= (20, 30, 15, 30).

Now find next two min files(which are F1 and M1) , merge them and add new merge file M2 to L.

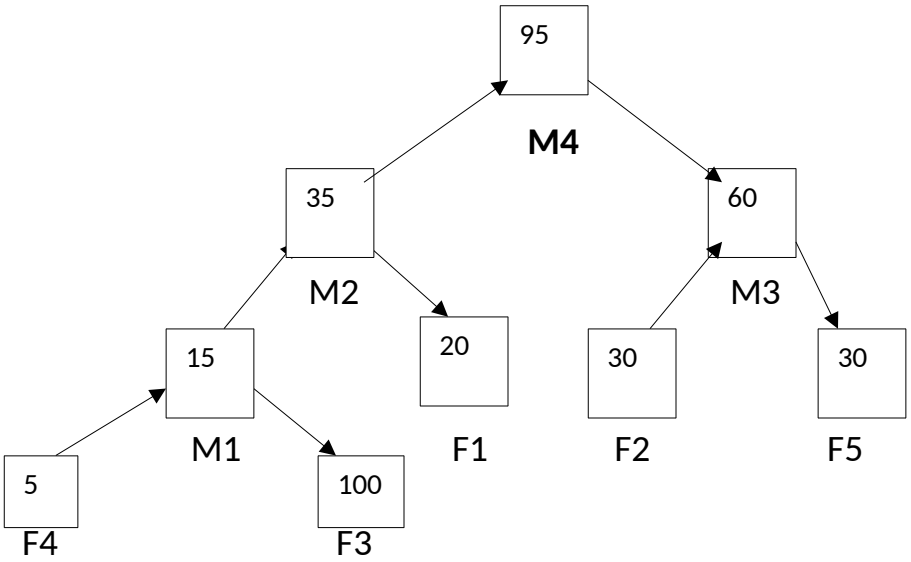


L (F2, M2, F5)= (30, 35, 30).

Now find next two min files(which are F2 and F5) , merge them and add new merge file M3 to L. L (M2, M3)= (35, 60).



Now merge next two min file. L(M4)={95}



Analysis:

T= O (n-1) * max (O (find minimum), O (cost for insertion in L)).

1. Case 1 L is not sorted.
   O (find minimum)= O (n).---we can use first 2 pass of selection sort will give first two minimum which require 2n cost.(O(n))
   O (cost for insertion in L)= O (1).-----L is not sorted so we can insert in last in array.
   Time complexity :T= **O (n^2)**
2. Case 2 L is sorted.
   Case 2.1
   O (find minimum)= O (1) --- because L is sorted we get 2 minimum in fist two place.
   O (cost for insertion in L)= O (n) ----L is sorted so after merge new file will be added to specific location so that the L remain sorted.
   Time complexity: T= **O (n^2)**
   Case 2.2
   L is represented as a min-heap. Value in the root is less then or equal to the values of its children.
   O (**find minimum**)= O (log n) ---- cost for **deletion** of minimum in min heap
   O (**cost for insertion in L**)= O (log n)--cost for **insertion** of minimum in min heap
   Time complexity: T= **O (n log n).**