EXPERIMENT NO. 04

Roll No.:24141001
Batch:I1

Title: Greedy method to solve problems of Optimal Merge Pattern

Objective:

- Understand the concept of Optimal Merge Pattern in greedy algorithms.
- Learn how to minimize the total cost of merging multiple files.
- Implement the greedy strategy that repeatedly merges the two smallest files first.
- Analyze time and space complexity of the merging process.
- Apply the greedy technique to real-world problems such as file merging, data compression, and optimal tree construction.

Theory:

The Optimal Merge Pattern problem arises when we need to merge multiple sorted files into one final file.

Each merge operation has a cost equal to the sum of sizes of the files being merged.

Greedy Strategy:

Always merge the two smallest files first.

This ensures that large files are merged later, reducing total cost.
This is exactly the principle used in Huffman coding and many file-compression algorithms.

To efficiently pick the smallest two elements each time, a min-heap (priority queue) is used.

Applications:

1. File Merging

Reducing cost of merging multiple sorted files.

2. Data Compression (Huffman Coding)

Optimal merge pattern is the basis for Huffman trees.

3. Compiler Design

Merging intermediate code files efficiently.

4. External Sorting

Used in merge sort when dataset is too large.

5. Database Systems

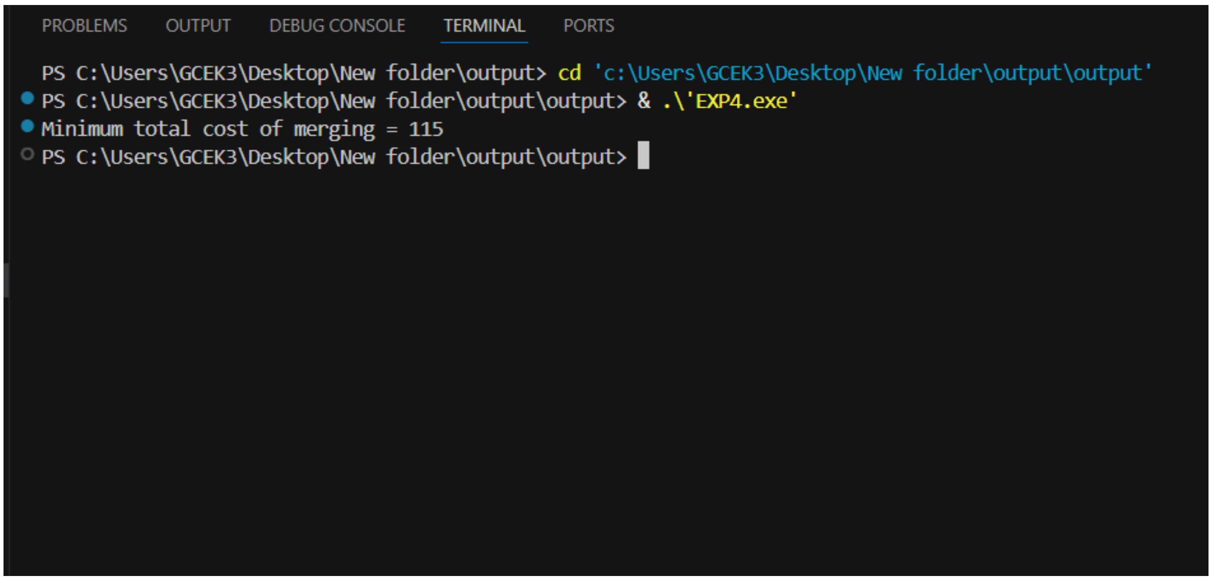
Merging sorted database segments.

Program:

```
#include <stdio.h>
#include <limits.h>

// Function to find index of smallest element in array
int findMinIndex(int arr[], int n, int skip) {
   int min = INT_MAX, index = -1;
   for (int i = 0; i < n; i++) {
      if (arr[i] != -1 && arr[i] < min && i != skip) {
         min = arr[i];
         index = i;
      }
   }
   return index;
}

int optimalMerge(int files[], int n) {
   int totalCost = 0;

   for (int step = 1; step < n; step++) {
      // Find two smallest files
      int first = findMinIndex(files, n, -1);
      int second = findMinIndex(files, n, first);

      int cost = files[first] + files[second];
      totalCost += cost;

      // Merge them → replace one with new file size, mark other as used
      files[first] = cost;
      files[second] = -1;
   }
   return totalCost;
}
```

```c
int main() {
    int files[] = {20, 30, 10, 5};
    int n = sizeof(files) / sizeof(files[0]);

    int result = optimalMerge(files, n);
    printf("Minimum total cost of merging = %d\n", result);

    return 0;
}
```

OUTPUT:



```
PS C:\Users\GCEK3\Desktop\New folder\output> cd 'c:\Users\GCEK3\Desktop\New folder\output\output'
PS C:\Users\GCEK3\Desktop\New folder\output\output> & .\'EXP4.exe'
Minimum total cost of merging = 115
PS C:\Users\GCEK3\Desktop\New folder\output\output> 
```

Conclusion:
The Optimal Merge Pattern demonstrates the power of the greedy method by always merging the smallest two files first. This strategy ensures minimal merge cost and forms the basis of important algorithms such as Huffman Coding. The use of a min-heap makes the approach efficient for large datasets, achieving O(n log n) time complexity.