Experiment No.4

Title : Optimal Merge Pattern by Greedy Method

Reg.no.24141028

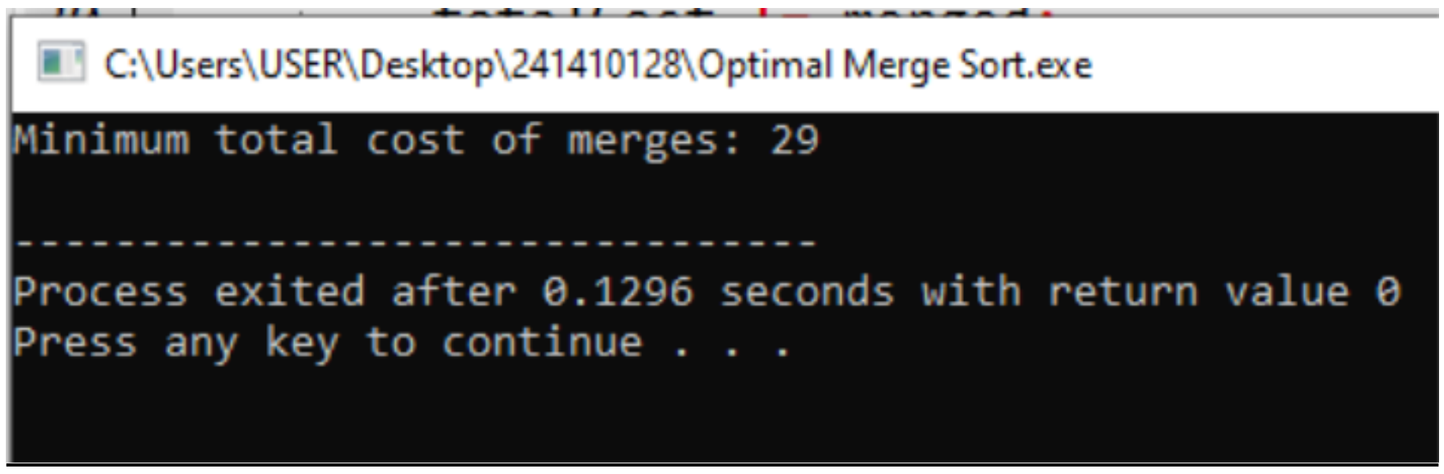
Program :

```
#include <stdio.h>

// Function to sort array in ascending order

void sort(int arr[], int n)

{

    int i; int j;

    for (i = 0; i < n - 1; i++)

    {

        for (j = i + 1; j < n; j++)

        {

            if (arr[i] > arr[j])

            {

                int temp = arr[i];

                arr[i] = arr[j];

                arr[j] = temp;

            }

        }

    }

}

int optimalMerge(int files[], int n)
```

```
{
    int totalCost = 0; int i;
    while (n > 1)
    {
        sort(files, n);
        // Merge two smallest files
        int merged = files[0] + files[1];
        totalCost += merged;
        //Replace first file with merge file
        files[0] = merged;
        for (i = 1; i < n - 1; i++)
        {
            files[i] = files[i + 1];
      }
        n--;
}
return totalCost;
}
// Main function
int main()
{
    int files[] = {4, 3, 2, 6};
    int n = sizeof(files) / sizeof(files[0]);
    int cost = optimalMerge(files, n);
```

```
    printf("Minimum total cost of merges: %d\n", cost);

    return 0;

}
```

Output :



Application Based on Optimal Merge Pattern : Merging Music Tracks

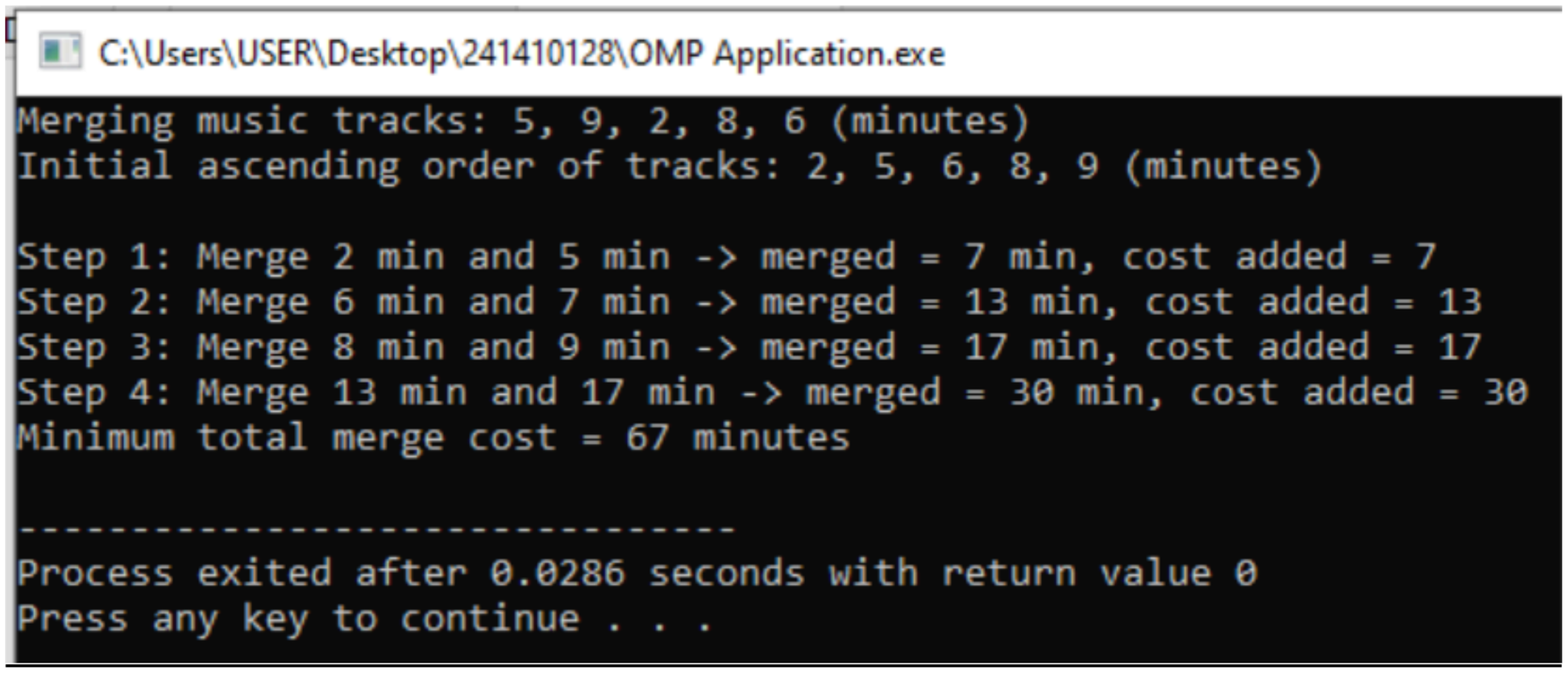
Program :

```
#include <stdio.h>

void sort(int arr[], int n)

{

    int i; int j;

    for(i = 0; i < n-1; i++) {

        for(j = i+1; j < n; j++) {

            if(arr[i] > arr[j]) {

                int temp = arr[i];

                arr[i] = arr[j];

                arr[j] = temp;

            }

        }

    }

}
```

```
// Function to calculate minimum merge cost
int optimalMerge(int tracks[], int n)
{
        int i;
        int totalCost = 0;
        int step = 1;
        while(n > 1)
{
            sort(tracks, n);
          int merged = tracks[0] + tracks[1];
            totalCost += merged;
            printf("Step %d: Merge %d min and %d min -> merged = %d min, cost
added = %d\n",
                        step, tracks[0], tracks[1], merged, merged);
              // Replace first track with merged track
            tracks[0] = merged;
            // Shift remaining tracks
            for(i = 1; i < n-1; i++) {
                tracks[i] = tracks[i+1];
                }
                n--;
                step++;
        }
        return totalCost;
}
int main()
{
```

```
int i;
int tracks[] = {5, 9, 2, 8, 6};      // lengths of music tracks in minutes
int n = sizeof(tracks)/sizeof(tracks[0]);
printf("Merging music tracks: ");
for(i = 0; i < n; i++) {
        printf("%d", tracks[i]);
        if(i < n-1) printf(", ");
}
printf(" (minutes)\n");
// Print initial ascending order
int tracksAsc[n];
for(i = 0; i < n; i++) tracksAsc[i] = tracks[i];
sort(tracksAsc, n);
printf("Initial ascending order of tracks: ");
for(i = 0; i < n; i++) {
        printf("%d", tracksAsc[i]);
        if(i < n-1) printf(", ");
}
printf(" (minutes)\n\n");
        int cost = optimalMerge(tracks, n);
        printf("Minimum total merge cost = %d minutes\n", cost);
        return 0;
}
```

Output :


C:\Users\USER\Desktop\241410128\OMP Application.exe
Merging music tracks: 5, 9, 2, 8, 6 (minutes)
Initial ascending order of tracks: 2, 5, 6, 8, 9 (minutes)
Step 1: Merge 2 min and 5 min -> merged = 7 min, cost added = 7
Step 2: Merge 6 min and 7 min -> merged = 13 min, cost added = 13
Step 3: Merge 8 min and 9 min -> merged = 17 min, cost added = 17
Step 4: Merge 13 min and 17 min -> merged = 30 min, cost added = 30
Minimum total merge cost = 67 minutes

Process exited after 0.0286 seconds with return value 0
Press any key to continue . . .