1. Merge Sort

Merupakan salah satu jenis algoritma yang pengurutannya menggunakan *divide conquer*, yaitu dengan cara memecah lalu menyelesaikan setiap bagian-bagian pencahayanya tadi lalu digabungkan kembali. Setelah digabungkan kembali, maka akan dibandingkan blok yang sama. Apakah data pertama lebih besar daripada data ke-tengah +1. Jika ya, maka data ke-tengah +1 dipindah menjadi data pertama. Lalu data pertama digeser sampai ke-tengah, dan begitu seterusnya. Sehingga, metode merge sort ini merupakan metode yang membutuhkan fungsi rekursi untuk penyelesaiannya. Proses rekursi akan berhenti jika telah mencapai elemen dasar. Hal ini terjadi bilamana bagian yang akan diurutkan menyisakan tempat satu elemen. Sisa pengurutan satu elemen tersebut menandakan bahwa bagian tersebut telah berurutan sesuai dengan rangkaian.

Merge Sort.cpp

```cpp
#include <iostream>
using namespace std;

void merge(int arr[], int l, int m, int r) {
  int i = l;
  int j = m + 1;
  int k = l;

  /* membuat temp array */
  int temp[5];

  while (i <= m && j <= r) {
    if (arr[i] <= arr[j]) {
      temp[k] = arr[i];
      i++;
      k++;
    } else {
      temp[k] = arr[j];
      j++;
      k++;
    }

  }

  /* menyalin elemen yang tersisa di babak kedua jika ada  */
  while (i <= m) {
    temp[k] = arr[i];
    i++;
    k++;

  }

  /* menyalin elemen yang tersisa di babak kedua jika ada */
  while (j <= r) {
    temp[k] = arr[j];
    j++;
```

Source code di atas menampilkan perintah untuk membuat temp array. Dimana data yang dimasukkan nantinya jumlahnya ada 5. Lalu dibawahnya, menampilkan perintah untuk menyalin elemen di babak kedua jika ada elemen yang tersisa.

```cpp
  /* menyalin elemen yang tersisa di babak kedua jika ada */
  while (j <= r) {
    temp[k] = arr[j];
    j++;
    k++;
  }

  /* menyalin temp array ke array asli */
  for (int p = l; p <= r; p++) {
    arr[p] = temp[p];
  }
}

/* l untuk indeks sebelah kiri and r untuk indeks sebelah kanan di subarray dari array untuk disortir  */
void mergeSort(int arr[], int l, int r) {
  if (l < r) {
    // mencari titik tengah
    int m = (l + r) / 2;

    // Rekrusif mergesort bagian pertama dan kedua
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);

    // merge
    merge(arr, l, m, r);
  }
}

int main() {
  int myarray[5];
  //int arr_size = sizeof(myarray)/sizeof(myarray[0]);
  int arr_size = 5;

  cout << "Enter 5 integers in any order: " << endl;
```

Source code juga menampilkan perintah untuk menyalin elemen di babak kedua terdapat elemen yang tersisa. Selanjutnya terdapat perintah untuk menyalin temp array ke bentuk array yang asli. Lalu, terdapat komponen L untuk ditempatkan di indeks sebelah kiri dan komponen L untuk ditempatkan di indeks sebelah kanan sub array dari array yang asli untuk disortir. Didalamnya juga terdapat perintah untuk mencari titik tengah, rekursif merge sort bagian pertama dan kedua, dan merge itu sendiri.

```
int main() {
  int myarray[5];
  //int arr_size = sizeof(myarray)/sizeof(myarray[0]);
  int arr_size = 5;

  cout << "Enter 5 integers in any order: " << endl;
  for (int i = 0; i < 5; i++) {
    cin >> myarray[i];
  }
  cout << "Before Sorting" << endl;
  for (int i = 0; i < 5; i++) {
    cout << myarray[i] << " ";
  }
  cout << endl;
  mergeSort(myarray, 0, (arr_size - 1)); // memanggil mergesort(array,kanan,kiri)

  cout << "After Sorting" << endl;
  for (int i = 0; i < 5; i++) {
    cout << myarray[i] << " ";
  }

  return 0;
}
```

Source code di atas menampilkan perintah dimana user dapat memasukkan data yang ada, untuk di sorting. Didalamnya juga terdapat data awal yang belum disorting. Setelah dimasukkan maka akan menampilkan data yang telah di sorting. Waktu yang dibutuhkan untuk sorting source code dari data yang dimasukkan diatas adalah 15, 48 detik.

- Kelebihan Merge Sort

  1. Merge sort termasuk algoritma yang sangat efisien dalam penggunaannya dibanding dengan algoritma lain. Sebab setiap list selalu dibagi menjadi list yang lebih kecil, lalu digabungkan lagi sehingga tidak perlu melakukan banyak perbandingan.

  2. Cocok untuk sorting akses data yang lambat. Contoh tape drive atau hard disk.

  3. Cocok untuk sorting data yang biasanya diakses secara *sequentially* (berurutan).

- Kekurangan Merge Sort

  1. Terlalu banyak menggunakan ruang pada memori.

  2. Merge sort membutuhkan banyak ruang daripada jenis sorting lainnya.