ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH

Kiến trúc máy tính (CO2008)

Bài tập lớn

Merge Sort Số Thực Chính Xác Đơn

GVHD: Nguyễn Xuân Minh

SV thực hiện: Nguyễn Đại Tiến ———— 2114988 - L02
Nguyễn Trương Phước Thọ — 2114913 - L07

TP.HCM, 12/2022

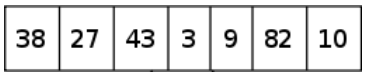
Mục Lục

1 Giải pháp thực hiện

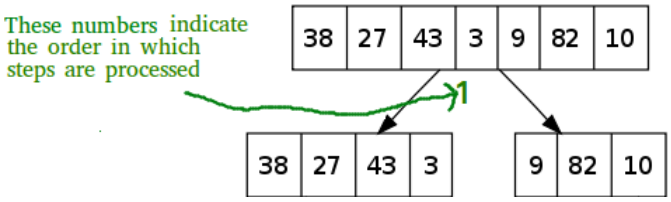
1.1 Cơ sở thuật toán Merge Sort

- Thuật toán **Merge Sort** là thuật toán sắp xếp dựa trên mô hình **chia để trị**. Trong thuật toán này, mảng ban đầu được chia thành 2 nửa bằng nhau rồi từ 2 nửa đó tiếp tục chia nhỏ như vậy cho đến khi không thể chia nhỏ được nữa. Khi đó, 2 nửa được chia ra sẽ trộn lại với nhau theo một thứ tự sắp xếp nhất định (ở bài tập lớn lần này sẽ là thứ tự **tăng dần**) rồi cứ tiếp tục trộn dần lại cho đến khi thu được mảng hoàn chỉnh. Cuối cùng, kết quả của thuật toán Merge Sort sẽ cho ra 1 mảng tuân theo thứ tự sắp xếp tăng dần.

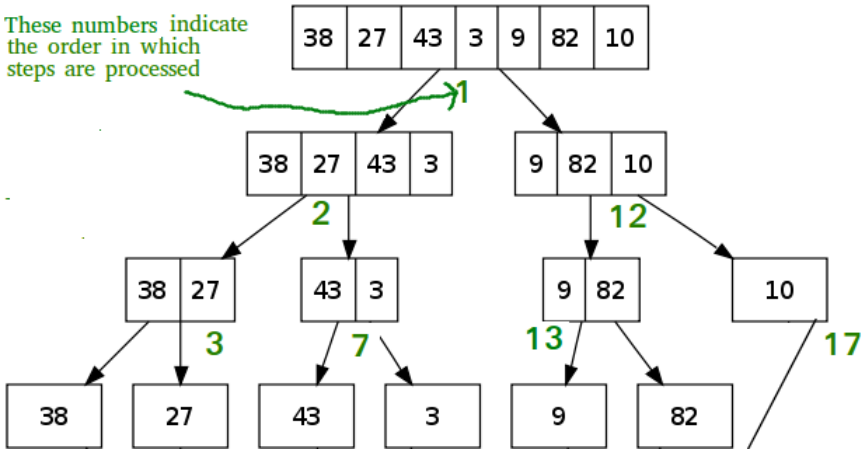
- Mô phỏng thuật toán
  Ví dụ ta có một mảng 7 phần tử cần sắp xếp tăng dần như bên dưới



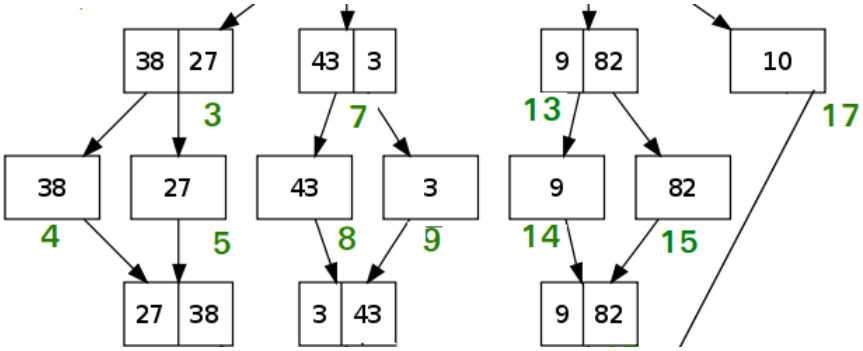
  - Ban đầu giá trị $left = 0, right = 6, mid = \frac{left+right}{2} = 3$, do $left < right$ nên ta bắt đầu chia đôi mảng với phần thứ nhất từ left đến pivot, phần thứ hai từ pivot+1 đến right.



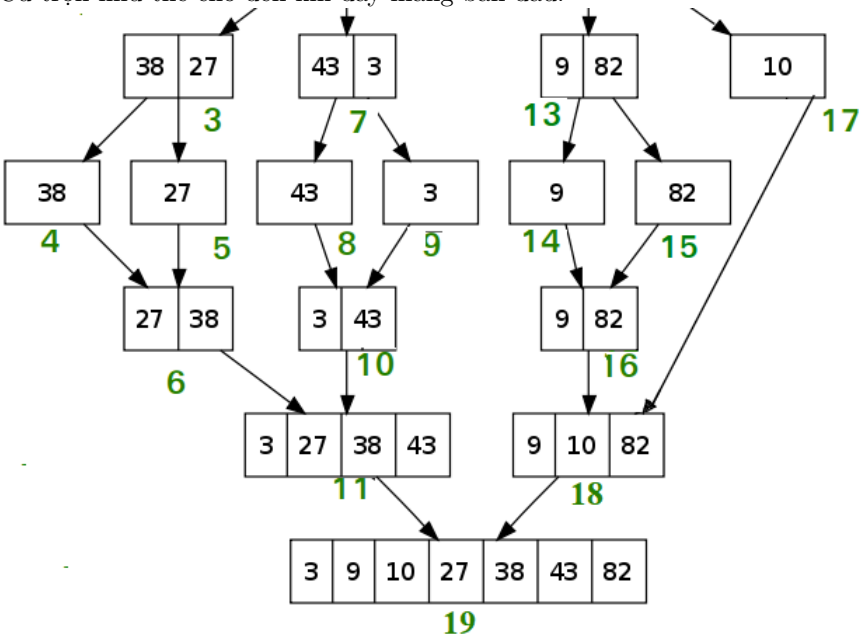
  - Ta tiếp tục chia nhỏ các phần cho đến khi không thể chia được nữa khi đó $left \geq right$



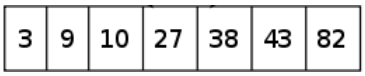
  - Sau khi đã chia hoàn toàn ra các phần nhỏ, ta sẽ bắt đầu trộn các phần tử lại với nhau theo thứ tự sắp xếp nhất định (ở ví dụ này, sắp xếp theo thứ tự tăng dần). Ta so sánh từng phần tử của mỗi phần được chia nhỏ xong thêm chúng vào mảng.



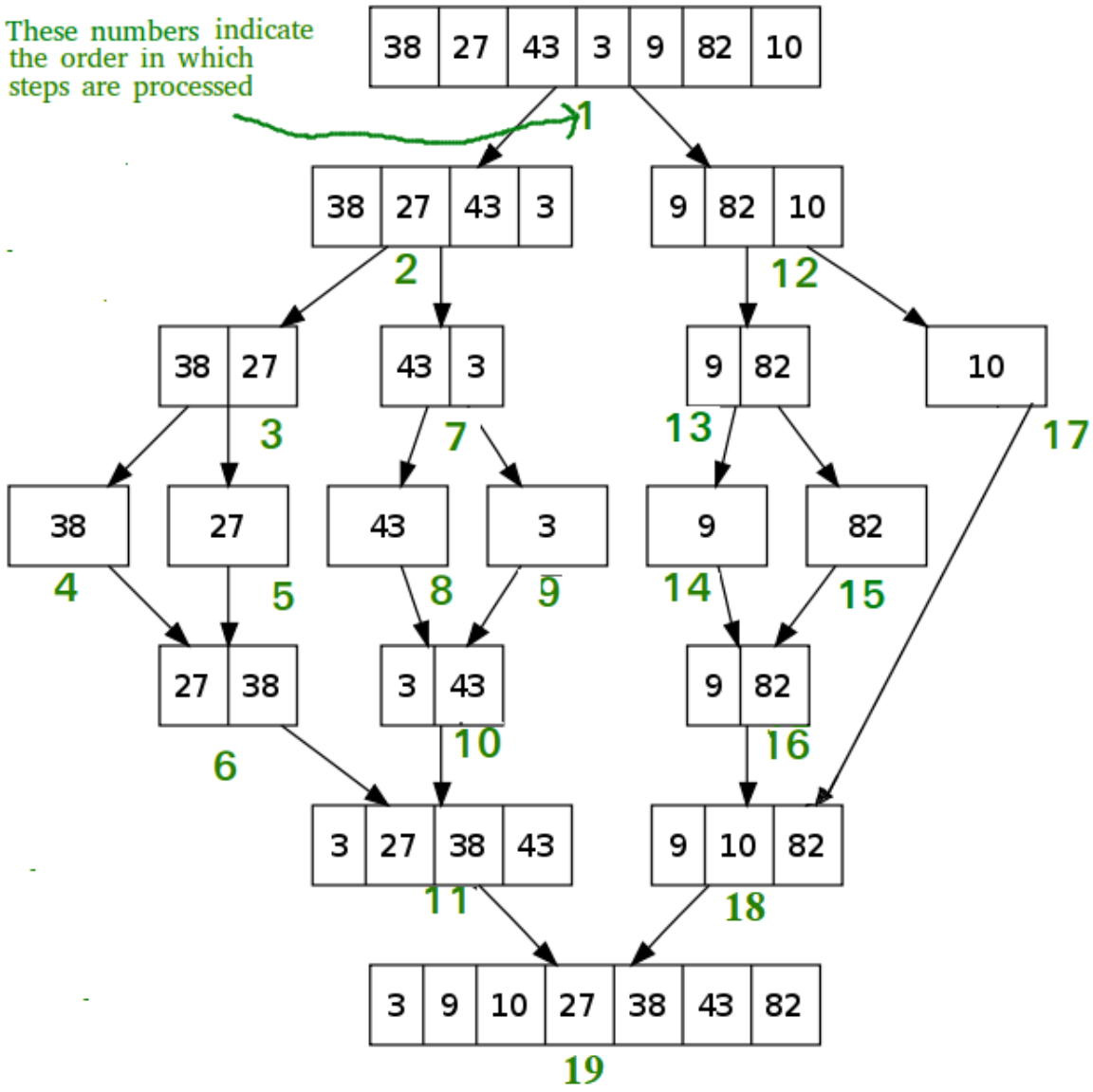
- Cứ trộn như thế cho đến khi đầy mảng ban đầu.



- Ta sẽ thu được một mảng đã sắp xếp theo thứ tự tăng dần theo thuật toán Merge Sort



- Ta cùng nhìn lại tổng thể của phương pháp
  Với thứ tự các bước thực hiện được biểu diễn bằng hàng số màu xanh



1.2 Thuật toán Merge Sort

- Ta cần thực thi hàm phụ trợ $merge(arr, left, mid, right)$ để trộn 2 phần tử theo thứ tự tăng dần trước khi thực thi hàm MergeSort.
- Các bước thực hiện của hàm MergeSort
  $MergeSort(arr[], left, right)$
  $If left < right$:
  - Tính toán phần tử ở giữa mid để chia đôi mảng thành 2 phần bằng nhau :
    $mid = \frac{left+right}{2}$

- Gọi hàm MergeSort nửa trái
  $mergeSort(arr, left, mid)$
- Gọi hàm MergeSort nửa phải
  $mergeSort(arr, mid + 1, right)$
- Trộn 2 mảng đã được sắp xếp ở 2 bước trên
  $merge(arr, left, mid, right)$

Chi tiết thuật toán được trình bày ở **phần 2**.

1.3 Merge sort số thực chính xác đơn

Áp dụng phương pháp lí thuyết được trình bày ở trên, ta tiến hành xây dựng chương trình Merge sort số thực chính xác đơn trên Mars Mips.
Các bước thực hiện :

- **Bước 1** : Tạo file lưu trữ dạng nhị phân trên đĩa FLOAT15.BIN (15 phần tử x 4 bytes = 60 bytes) thông qua đoạn chương trình tạo file trên Mars Mips
  - Mảng 15 phần tử số thực chính xác đơn được chọn trong bài tập lớn lần này là : 5.6, 0.8, 1.3, 1.0, 25.8, 3.7, 4.9, 0.0, 11.5 , 2.6, -3.0, 4.5, 7.2, -9.4, 10.0
- **Bước 2** : Xây dựng các thao tác trong hàm main
  - Tiến hành đọc dữ liệu trên file dữ liệu FLOAT15.BIN vừa tạo : mở file dữ liệu, lưu vào mảng số thực 15 phần tử trong file dữ liệu, đóng file dữ liệu.
  - Nhập dữ liệu : quy định thanh ghi, tiến hành lưu các giá trị vào các thanh ghi quy định.
  - Xuất mảng ban đầu trước khi sắp xếp.
  - Gọi đến hàm Merge Sort (được xây dựng ở các bước sau).
  - Xuất mảng sau khi đã sắp xếp.
- **Bước 3** : Xây dựng hàm phụ trợ display để xuất mảng
  - Truyền vào địa chỉ mảng, số phần tử của mảng.
  - Quy định thanh ghi.
  - Tiến hành gán địa chỉ mảng, số phần tử mảng vào các thanh ghi quy định.
  - Chạy vòng lặp xuất từng phần tử mảng.
- **Bước 4** : Xây dựng hàm phụ trợ Merge
  - Truyền vào địa chỉ mảng, vị trí phần tử bên trái, vị trí phần tử bên phải, vị trí ở giữa.
  - Quy định thanh ghi.
  - Khởi tạo một mảng tạm, lưu mảng ban đầu vào mảng tạm.
  - Tiến hành chạy vòng lặp so sánh từng phần tử ở nửa trái và nửa phải sau đó cho vào mảng ban đầu.
- **Bước 5** : Xây dựng hàm Merge Sort
  - Truyền vào địa chỉ mảng, vị trí phần tử đầu left, vị trí phần tử cuối right.

- Quy định thanh ghi.
- Tiến hành bảo vệ các thanh ghi quy định, thanh ghi $ra
- Xây dựng hàm Merge Sort theo cơ sở lí thuyết được trình bày ở phần trước : tính toán vị trí ở giữa, chia mảng thành 2 phần bằng nhau ngăn cách bởi vị trí ở giữa, gọi đến hàm Merge Sort nửa đầu, gọi đến hàm Merge Sort nửa sau, sau đó gọi đến hàm phụ trợ Merge để trộn 2 nửa lại.
- Gọi đến hàm phụ trợ display để xuất mảng qua từng bước.
- Cuối cùng, trả về giá trị của các thanh ghi được bảo vệ.

2 Giải thuật

2.1 Mã giả

Algorithm 1: Merge

Data: $arr[], \quad left, \quad right, \quad mid$
Result: Merge 2 sorted array
allocate tempArr[];
$i = 0$;
while $idx \leq right - left + 1$ **do**
  $arr[i] = tempArr[i + left]$;
end
$i = 0$;
$j = mid - left + 1$;
$k = left$;
while $i \leq mid - left$ *and* $j \leq right - left$ **do**
  if $tempArr[i] \leq tempArr[j]$ **then**
    $arr[k] = tempArr[i]$;
    $i++$;
  else
    $arr[k] = tempArr[j]$;
    $j++$;
  end
  $k++$;
end
while $i \leq mid - left$ **do**
  $arr[k] = tempArr[i]$;
  $i++$;
  $k++$;
end
while $j \leq right - left$ **do**
  $arr[k] = tempArr[j]$;
  $j++$;
  $k++$;
end
free tempArr[]

Algorithm 2: MergeSort

Data: $arr[], \quad left, \quad right$
Result: Sorted array
if $left < right$ **then**
  $mid = \frac{left+right}{2}$;
  $MergeSort(arr, left, mid)$;
  $MergeSort(arr, mid + 1, right)$;
  $Merge(arr, left, right, mid)$;
end

2.2 Đoạn code trên ngôn ngữ C++

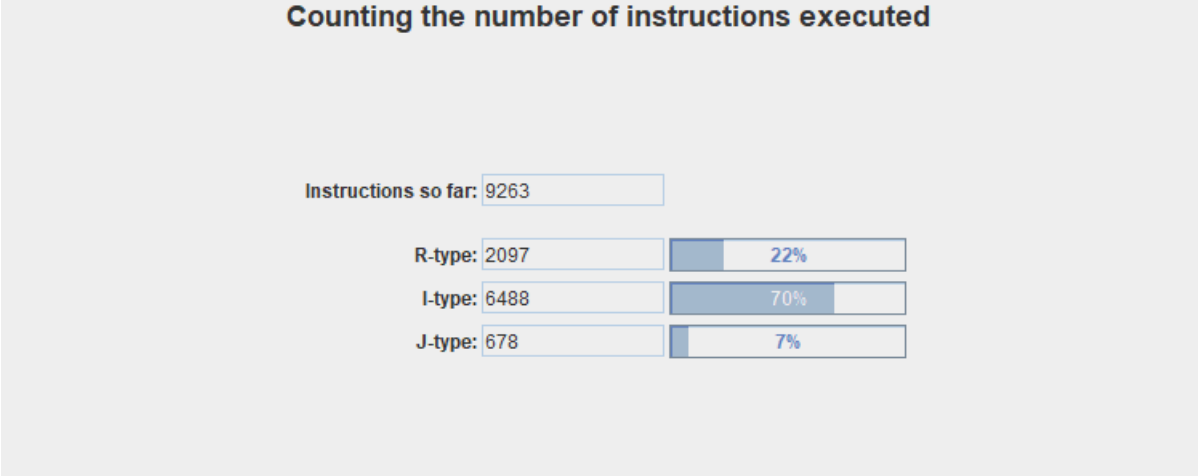
```
void merge(float *arr, int left, int right, int mid)
{
    float *tempArr = new float[right - left + 1];
    for (int i = 0; i < right - left + 1; i++)
    {
        tempArr[i] = arr[left + i];
    }
    int i = 0;
    int j = mid - left + 1;
    int k = left;
    while (i <= mid - left && j <= right - left)
    {
        if (tempArr[i] <= tempArr[j])
        {
            arr[k++] = tempArr[i++];
        }
        else
        {
            arr[k++] = tempArr[j++];
        }
    }
    while (i <= mid - left)
    {
        arr[k++] = tempArr[i++];
    }
    while (j <= right - left)
    {
        arr[k++] = tempArr[j++];
    }
    delete tempArr;
}

void mergeSort(float *arr, int left, int right)
{
    int mid = (right + left) / 2;
    if (left < right)
    {
        mergeSort(arr, left, mid);
```

```
39            mergeSort(arr, mid + 1, right);
40            merge(arr, left, right, mid);
41        }
42    }
```

3 Thống kê lệnh

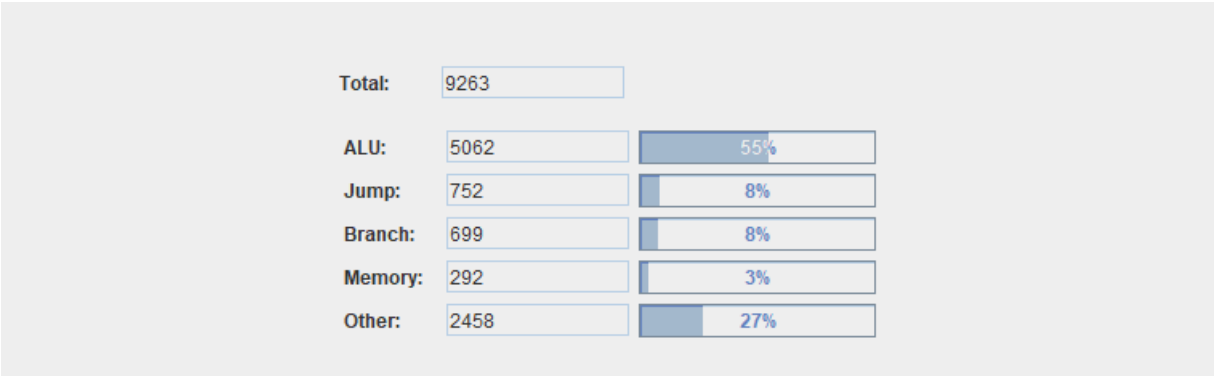
- Thống kê các kiểu lệnh :



Tổng số : 9263 lệnh

  - Lệnh kiểu R : 2097 lệnh, chiếm 22% tổng số lệnh
  - Lệnh kiểu I : 6488 lệnh, chiếm 70% tổng số lệnh
  - Lệnh kiểu J : 678 lệnh, chiếm 7% tổng số lệnh

- Thống kê số lượng lệnh :



Tổng số : 9963 lệnh

  - Lệnh ALU : 5062 lệnh, chiếm 55% tổng số lệnh
  - Lệnh Jump : 752 lệnh, chiếm 8% tổng số lệnh
  - Lệnh Branch : 699 lệnh, chiếm 8% tổng số lệnh
  - Lệnh Memory : 292 lệnh, chiếm 3% tổng số lệnh
  - Các lệnh khác : 2458 lệnh, chiếm 27% tổng số lệnh

4 Thời gian chạy của chương trình

4.1 Multiple Clock Cycle

Ta có bảng CPI của các loại lệnh như bên dưới :

Multiple clock cycle

Instruction	#cycles					
Load	5	IF	ID	EXE	MEM	WB
Store	4	IF	ID	EXE	MEM	
Branch	3	IF	ID	EXE		
Arithmetic/logical	4	IF	ID	EXE	WB	
Jump	2	IF	ID			

- Ta có CPI (Cycles per instruction) của các lệnh là:
  - Lệnh Memory có CPI = 5
  - Lệnh Jump có CPI = 2
  - Lệnh Branch có CPI = 3
  - Lệnh ALU có CPI = 4
  - Lệnh Logical(Others) có CPI = 4
- Tổng số lượng lệnh là: 9263
  - Lệnh Memory có 292 lệnh
  - Lệnh Jump có 752 lệnh
  - Lệnh Branch có 699 lệnh
  - Lệnh ALU có 5062 lệnh
  - Lệnh Logical(khác) có 2458 lệnh
- Ta có CPI trung bình : $CPI_{tb} = \frac{292\times5+752\times2+699\times3+5062\times4+2458\times4}{9263} = 3,79$
- Theo đề ta có CR (Clock rate) = 1GHz Vậy tổng thời gian thực thi của đoạn chương trình là:

$$CPU_{times} = \frac{IC\times CPI}{CR} = \frac{9263\times3,79}{10^9} = 35,1(\mu s)$$

4.2 PipeLine

- Ta có tổng số lệnh là IC = 9263 lệnh, có số bước pipeline k = 5
- Thời gian thực hiện 1 chu kỳ :$T = \frac{1}{10^9(Hz)} = 1(ns)$
- Tổng thời gian thực thi của đoạn chương trình là
$CPU_{times} = (IC - 1 + k) * T = (9263 - 1 + 5) \times 1 = 9,267(\mu s)$

5 Kết quả

Tóm tắt đề bài : Sắp xếp theo thứ tự tăng dần mảng số thực chính xác đơn 15 phần tử : 5.6, 0.8, 1.3, 1.0, 25.8, 3.7, 4.9, 0, 11.5 , 2.6, -3.0, 4.5, 7.2, -9.4, 10 bằng thuật toán Merge Sort. In mảng ở từng bước sắp xếp

- Kết quả sau khi chạy code trên Mars Mips

```
mang ban dau : 5.6 0.8 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
5.6 0.8 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
5.6 0.8 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 5.6 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 5.6 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 5.6 1.3 1.0 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 5.6 1.0 1.3 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 25.8 3.7 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 3.7 25.8 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 3.7 25.8 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 3.7 25.8 4.9 0.0 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 3.7 25.8 0.0 4.9 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.8 1.0 1.3 5.6 0.0 3.7 4.9 25.8 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 11.5 2.6 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 2.6 11.5 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 2.6 11.5 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 2.6 11.5 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 2.6 11.5 -3.0 4.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 7.2 -9.4 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 -9.4 7.2 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 -9.4 7.2 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -3.0 2.6 4.5 11.5 -9.4 7.2 10.0
0.0 0.8 1.0 1.3 3.7 4.9 5.6 25.8 -9.4 -3.0 2.6 4.5 7.2 10.0 11.5
-9.4 -3.0 0.0 0.8 1.0 1.3 2.6 3.7 4.5 4.9 5.6 7.2 10.0 11.5 25.8
mang da sap xep : -9.4 -3.0 0.0 0.8 1.0 1.3 2.6 3.7 4.5 4.9 5.6 7.2 10.0 11.5 25.8

-- program is finished running --
```

Hình 1: Kết quả sau khi chạy đoạn chương trình

- Mảng thu được sau khi sắp xếp : -9.4, -3.0, 0.0, 0.8, 1.0, 1.3, 2.6, 3.7, 4.5, 4.9, 5.6, 7.2, 10.0, 11.5, 25.8

References

[1] Bài giảng môn Kiến trúc máy tính - Đại học Bách Khoa Tp.HCM

[2] Bài giảng Merge Sort - môn Cấu trúc dữ liệu và giải thuật - Đại học Bách Khoa Tp.Hcm

[3] Merge Sort Algorithm - https://www.geeksforgeeks.org/merge-sort/