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 (CO2007)

BÁO CÁO BÀI TẬP LỚN - ĐỀ 6 - MERGE SORT

GVHD: Võ Tấn Phương
Trần Thanh Bình
Lớp: L05 - L09

Tp. Hồ Chí Minh, Tháng 11/2020

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 (CO2007)

BÁO CÁO BÀI TẬP LỚN - ĐỀ 6 - MERGE SORT

GVHD: Võ Tấn Phương
Trần Thanh Bình
Lớp: L05 - L09
Thành viên nhóm: 1912123 - Lê Trần Hoàng Thịnh
1912190 - Nguyễn Mai Thy
1911704 - Nguyễn Thủy Ngọc

Tp. Hồ Chí Minh, Tháng 11/2020

Mục lục

1 Cơ sở lý thuyết

1.1 Khái niệm

- Sắp xếp trộn (Merge sort) là một thuật toán sắp xếp để sắp xếp các danh sách (hoặc bất kỳ cấu trúc dữ liệu nào có thể truy cập tuần tự như luồng tập tin) theo một trật tự nào đó.
- Đây là một thuật toán chia để trị, được phát minh năm 1945 bởi John von Neumann. Năm 1948, Goldstine và Neumann đã đưa ra một bản miêu tả chi tiết của thuật toán này.
- Merge sort được xếp vào dạng thuật toán so sánh.
- Độ phức tạp thuật toán:
  Tốt: $O(Nlog(N))$
  Xấu: $O(Nlog(N))$
  Trung Bình: $O(Nlog(N))$
- Không gian bộ nhớ sử dụng: $O(N)$
- Ứng dụng thực tế: Trong khoa học máy tính, merge sort là một thuật toán hiệu quả và có tính ứng dụng cao. Merge sort hiệu quả trong việc xử lý những dữ liệu theo kiểu tuần tự, thời gian truy cập cao. Thuật toán này thường là lựa chọn số một trong việc sắp xếp linked list. Perl 5.8 dùng merge sort như thuật toán sắp xếp mặc định, Java dùng nó cho phương thức Array.sort(), Linux dùng merge sort cho linked list,..

1.2 Cách thực hiện

Merge sort sử dụng phương pháp chia để trị, chia danh sách thành cách phần nhỏ, sắp xếp các phần nhỏ sau đó trộn các phần nhỏ lại với nhau để được một danh sách đã sắp xếp hoàn chỉnh. Giả sử có hai danh sách đã được sắp xếp lại a[1...m] và b[1...n]. Ta có thể trộn chúng lại thành một danh sách mới c[1...m+n] được sắp xếp theo cách sau:

- So sánh hai phần tử đứng đầu của hai danh sách, lấy phần tử nhỏ hơn cho vào danh sách mới. Tiếp tục như vậy cho đến khi một trong hai danh sách là rỗng.
- Khi một trong hai danh sách là rỗng ta lấy phần còn lại của danh sách kia cho vào cuối danh sách mới.
  Ví dụ: Cho danh sách a (1, 4, 7, 9) và b (3, 5) được sắp xếp theo thứ tự tăng dần, quá trình diễn ra như sau:

Danh sách a	Danh sách b	So sánh	Danh sách c
1, 4, 7, 9	3, 5	$1 < 3$	1
4, 7, 9	3, 5	$4 > 3$	3
4, 7, 9	5	$4 < 5$	4
7, 9	5	$7 > 5$	5
7, 9			1, 3, 4, 5, 7, 9

Sau khi sắp xếp ta được danh sách c(1, 3, 4, 5, 7, 9) được sắp xếp tăng dần.

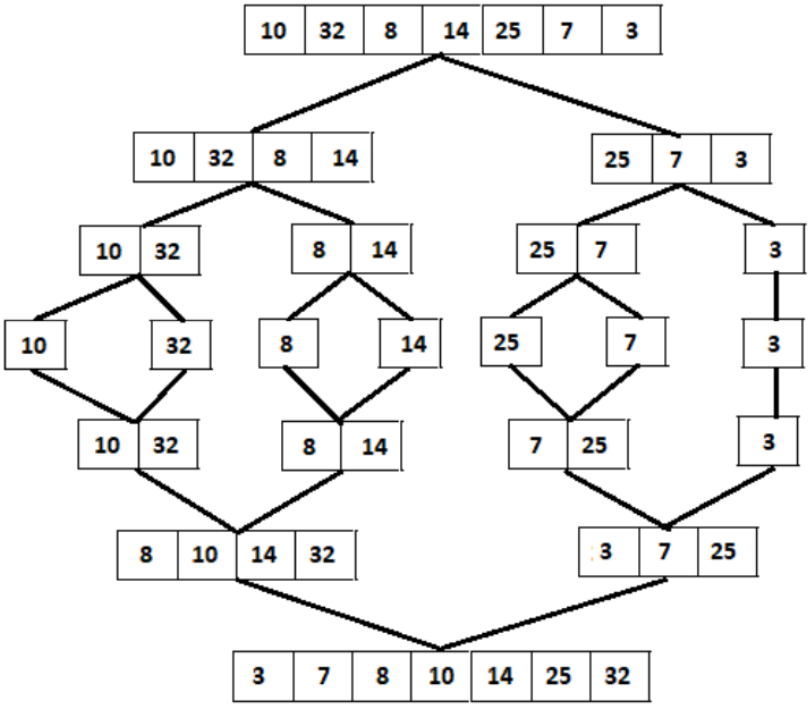
1.3 Recursive Merge Sort

- Trong các giáo trình giải thuật, merge sort thường được hiện thực dưới dạng đệ quy.
- Để sắp xếp trộn đoạn a[k1...k2] của danh sách a[1...a] ta chia đoạn đó thành 2 phần a[k1, k3] và a[k3, k2], trong đó k3 = int((k1+k2)/2) tiến hành sắp xếp với mỗi phần rồi trộn chúng lại. Lời gọi thủ tục sắp xếp trộn với a[1...n] sẽ cho kết quả là sắp toàn bộ danh sách a[1...n].

  Ví dụ: Cho danh sách a [10, 32, 8, 14, 25, 7, 3]
  Giải thuật trộn đệ quy chia a thành hai danh sách con và tiến hành 3 bước:
  Danh sách trái: 10, 32, 8, 14

Danh sách phải: 25, 7, 3
Sắp xếp danh sách trái ta được: 8, 14, 10, 32
Sắp xếp danh sách phải ta được: 3, 7, 25
Trộn 2 danh sách ta được: 3, 7, 8, 10, 14, 25, 32

1.4 Mô phỏng giải thuật



2 Hiện thực chương trình sắp xếp mảng 20 số nguyên

2.1 Chương trình mô phỏng bằng ngôn ngữ C++

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

void merge(int* arr, int l, int m, int r) {
   int n1 = m - l + 1;
   int i, j, k;
   int* L = new int[n1];
   int* R = new int[n2];
   for (i = 0; i < n1; i++) {
      L[i] = arr[l + i];
   }
   for (j = 0; j < n2; j++) {
      R[j] = arr[m + 1 + j];
   }
   i = 0;
   j = 0;
   k = l;
   while (i < n1 && j < n2) {
      if (L[i] >= R[j]) {
         arr[k] = L[i];
         i++;
```

```
      }
      else {
         arr[k] = R[j];
         j++;
      }
      k++;
   }
   while (i < n1) {
      arr[k] = L[i];
      k++;
      i++;
   }
   while (j < n2) {
      arr[k] = R[j];
      k++;
      j++;
   }
}

void MergeSort(int* arr, int l, int r) {
   if (l < r) {
      int m = l + (r - l) / 2;
      MergeSort(arr, l, m);
      MergeSort(arr, m + 1, r);
      merge(arr, l, m, r);
   }
}

void printArray(int* arr, int size) {
   for (int i = 0; i < size; i++) {
      cout << arr[i] << " ";
   }
   cout << endl;
}

int main() {
   int arr[20];
   cout<<"Please enter size of array: "<<endl;
   int n;
   while (true){
      cin>>n;
      if (n < 0 || n > 20)
         cout<<"Invalid size please enter another number: "<<endl;
      else break;
   }
   for (int i = 0; i < n; i++) {
      cout << "Please enter an int number: ";
      cin >> arr[ i ];
   }
   MergeSort(arr, 0, n - 1);
   printArray(arr, n;
   system("pause");
   return 0;
}
```

2.2 Kết quả chạy trên MARS:

2.2.1 Trường hợp input size không hợp lệ:

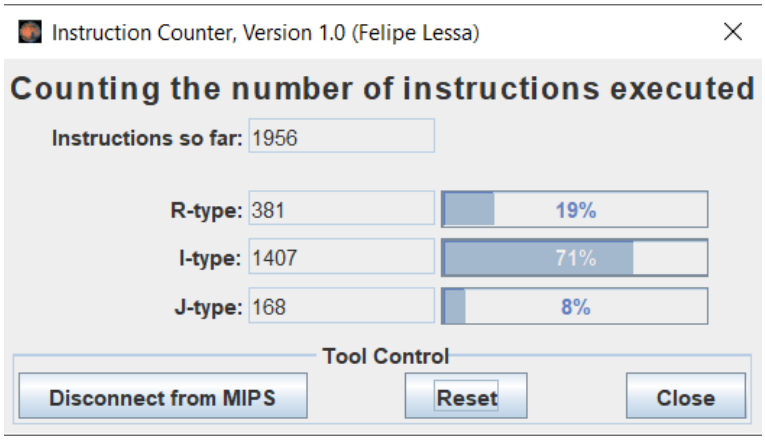
Trong các trường hợp input size không là số nguyên dương hoặc lớn hơn 20, chương trình chạy cho kết quả sau:

```
Please enter size of array: 22
Invalid size please enter another number: -5
Invalid size please enter another number: 23
Invalid size please enter another number: -1
Invalid size please enter another number: 0
Invalid size please enter another number: 15
Please enter an int number:
```

2.2.2 Trường hợp input size hợp lệ:

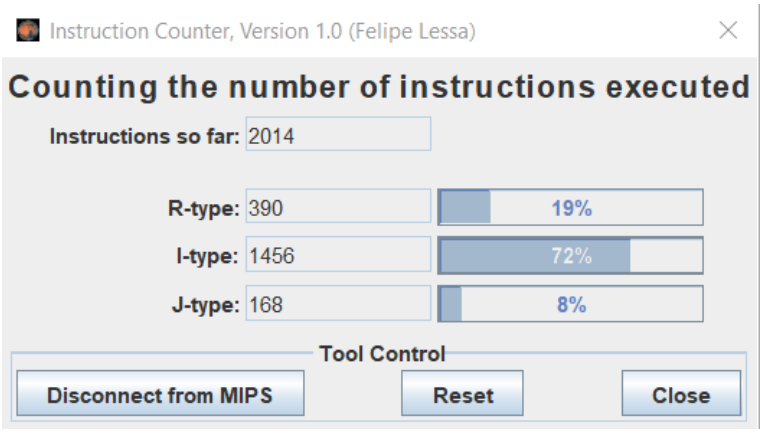
- **Test case 1:**
  + Input:
    Size of array: 12
    Array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
  + Output:

```
2 3
1 2 3
5 6
4 5 6
1 2 3 4 5 6
8 9
7 8 9
11 12
10 11 12
7 8 9 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12
```



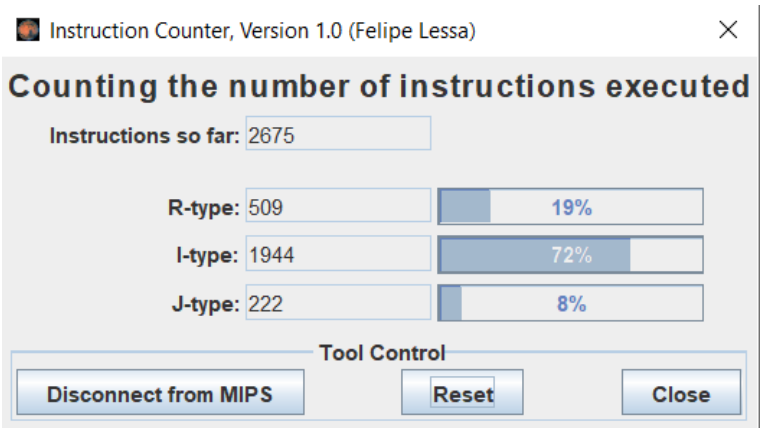
- **Test case 2:**
  + Input:
    Size of array: 12
    Array = {7, 1, 8, 2, 9, 3, 10, 4, 11, 5, 12, 6}
  + Output:

```
1 8
1 7 8
3 9
2 3 9
1 2 3 7 8 9
4 11
4 10 11
6 12
5 6 12
4 5 6 10 11 12
1 2 3 4 5 6 7 8 9 10 11 12
```



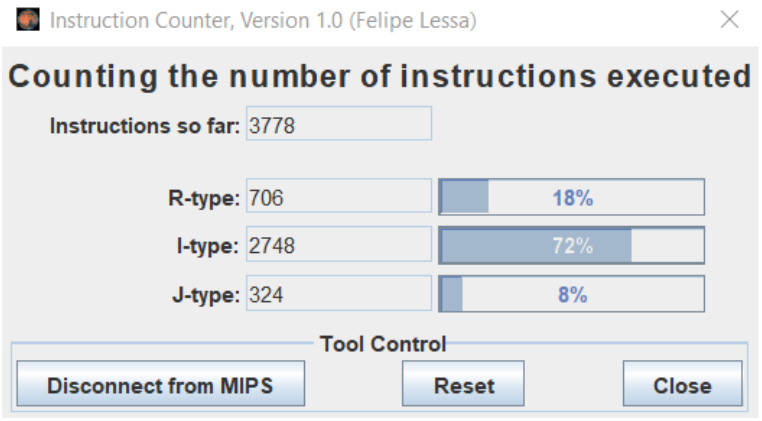
- **Test case 3:**
  + Input:
    Size of array: 15
    Array = {31, 247, 9, 99, 19, 8, 25, -37, -33, 24, 66, 63, 13, 38, -59}
  + Output:

```
9 247
9 31 247
19 99
8 25
8 19 25 99
8 9 19 25 31 99 247
-37 -33
24 66
-37 -33 24 66
13 63
-59 38
-59 13 38 63
-59 -37 -33 13 24 38 63 66
-59 -37 -33 8 9 13 19 24 25 31 38 63 66 99 247
```



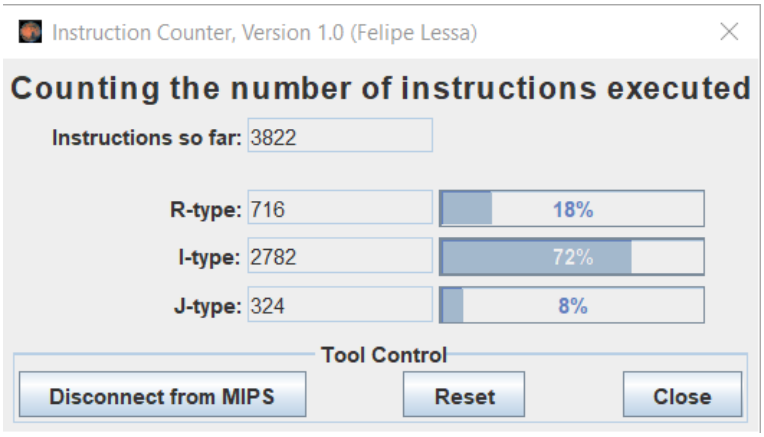
- **Test case 4:**
  + Input:
    Size of array: 20
    Array = {21, -51, 47, 56, 98, 12, 11, 3, 6, -38, 49, -59, -247, 294, 105, 6547, 2510, 3587, 21, 199}
  + Output:

```
-51 21
56 98
47 56 98
-51 21 47 56 98
11 12
-38 6
-38 3 6
-38 3 6 11 12
-51 -38 3 6 11 12 21 47 56 98
-59 49
105 294
-247 105 294
-247 -59 49 105 294
2510 6547
21 199
21 199 3587
21 199 2510 3587 6547
-247 -59 21 49 105 199 294 2510 3587 6547
-247 -59 -51 -38 3 6 11 12 21 21 47 49 56 98 105 199 294 2510 3587 6547
```



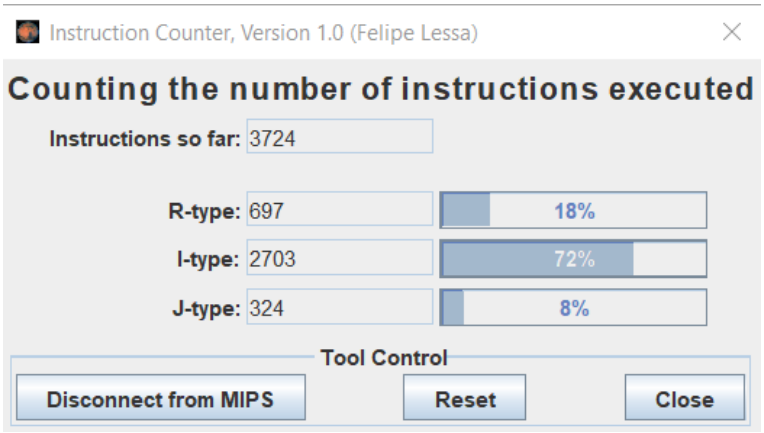
- **Test case 5:**
  + Input:
    Size of array: 20
    Array = {45, 21, 36, -85, 64, -81, -35, 211, 34, 256, -1024, 16, 8, 2, 32, 49, 5, 25, 7, 63}
  + Output:

```
21 45
-85 64
-85 36 64
-85 21 36 45 64
-81 -35
34 256
34 211 256
-81 -35 34 211 256
-85 -81 -35 21 34 36 45 64 211 256
-1024 16
2 32
2 8 32
-1024 2 8 16 32
5 49
7 63
7 25 63
5 7 25 49 63
-1024 2 5 7 8 16 25 32 49 63
-1024 -85 -81 -35 2 5 7 8 16 21 25 32 34 36 45 49 63 64 211 256
```



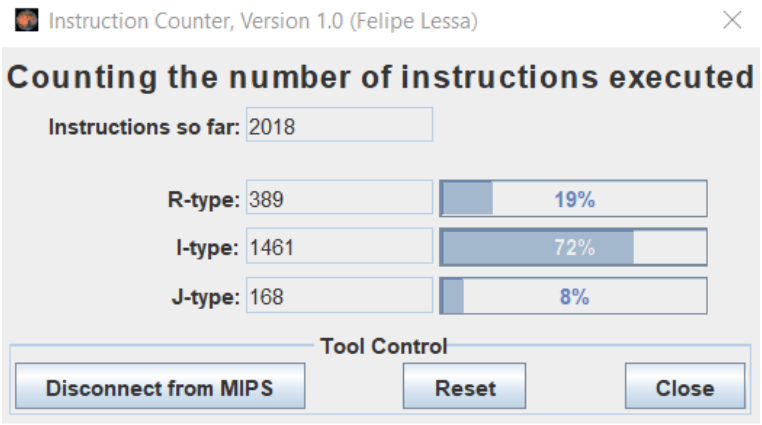
- **Test case 6:**
  + Input:
    Size of array: 20
    Array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11}
  + Output:

```
1 2
4 5
3 4 5
1 2 3 4 5
6 7
9 10
8 9 10
6 7 8 9 10
1 2 3 4 5 6 7 8 9 10
19 20
16 17
16 17 18
16 17 18 19 20
14 15
11 12
11 12 13
11 12 13 14 15
11 12 13 14 15 16 17 18 19 20
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
```



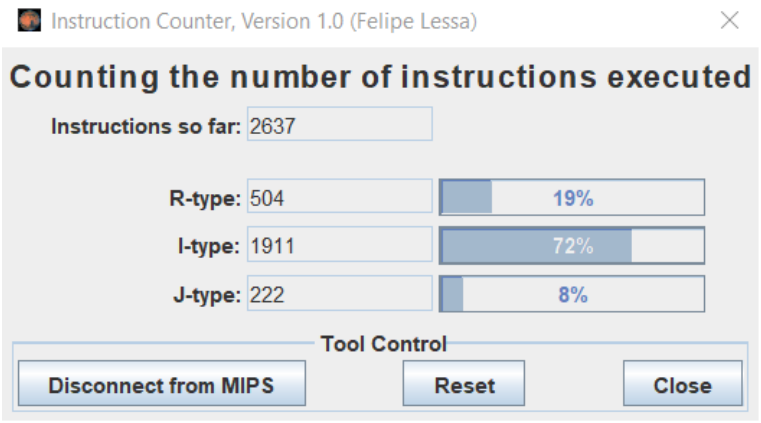
- **Test case 7:**
  + Input:
    Size of array: 12
    Array = {65, 36, 45, 42, 64, 56, 24, 16, -8, 257, 149, 99}
  + Output:

```
36 45
36 45 65
56 64
42 56 64
36 42 45 56 64 65
-8 16
-8 16 24
99 149
99 149 257
-8 16 24 99 149 257
-8 16 24 36 42 45 56 64 65 99 149 257
```



- **Test case 8:**
  + Input:
    Size of array: 15
    Array = {-3, 27, 36, 94, 54, -112, -153, 27, 33, 987, -932, -120, 348, 620, 741}
  + Output:

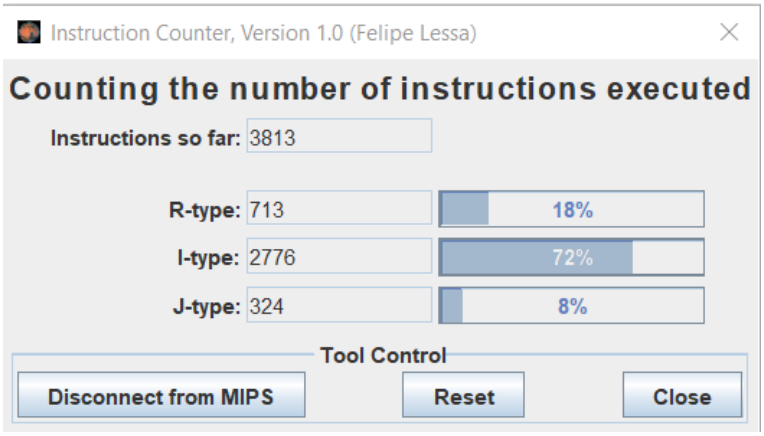
```
27 36
-3 27 36
54 94
-153 -112
-153 -112 54 94
-153 -112 -3 27 36 54 94
27 33
-932 984
-932 27 33 984
-120 348
620 741
-120 348 620 741
-932 -120 27 33 348 620 741 984
-932 -153 -120 -112 -3 27 27 33 36 54 94 348 620 741 984
```



– **Test case 9:**

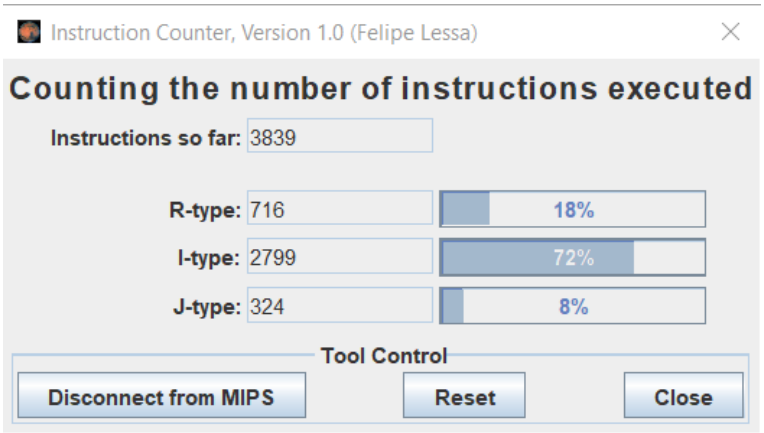
+ Input:
  Size of array: 20
  Array = {31, -41, 65, 2, 1, -74, 56, 24, 96.2, 2348, 23, 24, -561, 157, 94, -207, 213, 566, 444, 30}
+ Output:

```
-41 31
1 2
1 2 65
-41 1 2 31 65
-74 56
2348 9601
24 2348 9601
-74 24 56 2348 9601
-74 -41 1 2 24 31 56 65 2348 9601
23 24
94 157
-561 94 157
-561 23 24 94 157
-207 213
30 444
30 444 566
-207 30 213 444 566
-561 -207 23 24 30 94 157 213 444 566
-561 -207 -74 -41 1 2 23 24 24 30 31 56 65 94 157 213 444 566 2348 9601
```



- **Test case 10:**
  + Input:
    Size of array: 20
    Array = {-94214, 21, 95354, 5433, 48, 54, 1, 68, 0, -852, 487, 651, 69, -5449, 48, 741, 247, 301, 500, 647}
  + Output:

```
-94214 21
48 5433
48 5433 95354
-94214 21 48 5433 95354
1 54
-852 0
-852 0 68
-852 0 1 54 68
-94214 -852 0 1 21 48 54 68 5433 95354
487 651
-5449 48
-5449 48 69
-5449 48 69 487 651
248 741
500 647
301 500 647
248 301 500 647 741
-5449 48 69 248 301 487 500 647 651 741
-94214 -5449 -852 0 1 21 48 48 54 68 69 248 301 487 500 647 651 741 5433 95354
```



3 Thực thi chương trình

3.1 Tổng số lệnh thực hiện chương trình

Tổng số lệnh từng loại, IC và CPU time cho 10 test case trên như sau:

Testcase	Số phần tử	Trường hợp	R	I	J	IC	CPU time(ns)
1	12	best	381	1407	168	1956	978
2	12	worst	390	1456	168	2014	1007
3	15	normal	509	1944	222	2675	1337.5
4	20	normal	706	2748	324	3778	1889
5	20	normal	716	2782	324	3822	1911
6	20	normal	697	2703	324	3724	1862
7	12	normal	389	1461	168	2018	1014
8	15	normal	504	1911	222	2637	1318.5
9	20	normal	713	2776	324	3813	1906.5
10	20	normal	716	2799	324	3839	1919.5

3.2 Thời gian thực thi chương trình

Sau đây là cách tính thời gian chạy của chương trình trên máy tính kiến trúc MIPS có tần số 2 GHz.
Công thức tính thời gian thực thi:

$$CPUtime = \frac{Instructioncount * CPI}{Clockrate}$$

Trong đó:

- CPU time: thời gian thực thi.
- Instruction count (IC): tổng số lệnh thực thi.
- CPI: thời gian thực thi 1 lệnh (chu kỳ) = 1.
- Tần số máy tính (Clock rate) = 2 GHz.

Ví dụ:

Với test case 4 là chuỗi số 21, -51, 47, 56, 98, 12, 11, 3, 6, -38, 49, -59, -247, 294, 105, 6547, 2510, 3587, 21, 199. Ta có: $IC = 3778$.
Thời gian thực hiện trên máy tính có tần số 2 GHz là:

$$CPUtime = \frac{Instructioncount * CPI}{Clockrate} = \frac{3778 * 1}{2 * 10^9} = 1889(ns)$$

4 Tài liệu tham khảo

(1) Nguyễn Văn Hiếu. Merge Sort – Sắp xếp trộn. Link: https://nguyenvanhieu.vn/thuat-toan-sap-xep-merge-sort/. Truy cập lần cuối: 12/12/2020.

(2) Merge Sort. Link: https://www.geeksforgeeks.org/merge-sort/. Truy cập lần cuối: 12/12/2020.