ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
BỘ MÔN KIẾN TRÚC MÁY TÍNH

--------*-------

KIẾN TRÚC TẬP LỆNH
Đề tài 4: QuickSort – Sắp xếp nhanh

GV dạy lý thuyết: Đinh Đức Anh Vũ
GV dạy bài tập: Trần Thanh Bình

Lớp : CN01
Nhóm sinh viên thực hiện:

STT	Họ và tên	MSSV
1	Bùi Hoàng Quang Huy	2153372
2	Nguyễn Minh Hiếu	2153343
3	Nguyễn Thành Danh	2052417

Chấm điểm bài tập lớn Kiến trúc máy tính:

File	Điểm nộp và gửi bài đúng yêu cầu (1 điểm)	Điểm hình thức (2 điểm)	Điểm nội dung (2 điểm)	Tổng điểm
File asm				
File word				
Tổng điểm				

Tp. HCM, tháng 12 năm 2022

MỤC LỤC

LỜI NÓI ĐẦU

Đầu tiên, nhóm chúng em xin được gửi lời cảm ơn đến các giảng viên là thầy Trần Thanh Bình và thầy Đinh Đức Anh Vũ đã giúp nhóm thực hiện bài tập lớn này. Nhờ sự giúp đỡ tận tình của quý thầy cô, chúng em đã vượt qua những khúc mắc, khó khăn trong suốt quá trình thực hiện bài tập, từ đó hoàn thành đúng tiến độ của môn học và cho ra sản phẩm chất lượng. Ngoài ra, không thể không nhắc đến sự quan tâm giúp đỡ của các anh chị, các bạn sinh viên trong cộng đồng sinh viên trường Đại học Bách Khoa nói riêng và ĐHQG-HCM nói chung, những đóng góp to lớn của các anh, chị và các bạn đã giúp chúng em nắm chắc hơn cách sử dụng MARS MIPS - là một môi trường phát triển tương tác nhẹ (IDE) để lập trình bằng hợp ngữ MIPS mà nhóm lần đầu tiên được tiếp cận trong năm học đầu tiên theo học ở môi trường Đại học. Cuối cùng, nhóm chúng em xin gửi lời cảm ơn một lần nữa đến các tập thể, cá nhân đã giúp đỡ và truyền cảm hứng cho nhóm trong suốt quá trình thực hiện dự án bài tập lớn này.

Đề tài thực hiện:

Đề 4: (3sv) Sắp xếp chuỗi.

Cho một chuỗi số nguyên 50 phần tử. Sử dụng hợp ngữ assembly MIPS, viết thủ tục sắp xếp chuỗi đó theo thứ tự tăng dần theo giải thuật quick sort. Yêu cầu xuất ra từng bước trong quá trình demo.

Yêu cầu đề tài:

Yêu cầu

- Sử dụng tập lệnh MIPS để hiện thực các thủ tục bên dưới.
- Thống kê số lệnh, loại lệnh (instruction type) của mỗi chương trình.
- Tính và trình bày cách tính thời gian chạy của chương trình trên máy tính kiến trúc MIPS có tần số 3.4 GHz.
- Code
  - Code style phải rõ ràng, có comment, phân hoạch công việc theo từng hàm, CHỈ DÙNG LỆNH MIPS CHUẨN.
  - Truyền nhận và trả kết quả khi gọi hàm theo quy ước của thanh ghi (thanh ghi $a argument, thanh ghi $v giá trị trả về khi gọi hàm).
  - Xuất kết quả để kiểm tra.
- Báo cáo ngắn
  - Báo cáo ngắn gồm các phần sau:
    - Trình bày giải pháp hiện thực.
    - Giải thuật (nếu có).
    - Ghi rõ các trường hợp test (test-cases ít nhất 30 cases) và kết quả các test cases.
    - Mỗi testcase thống kê số lệnh chạy được (R, I, J) và thời gian thực thi của mỗi testcase (CPI=1 cho 1 lệnh MIPS chuẩn, nên trình bày ở dạng bảng).
  - File báo cáo đặt tên theo dạng KTMT_assignment_DE_x_Nhom_x.
  - Submit bài tập lớn: file báo cáo (file pdf) và source code (mỗi nhóm đại diện 1 thành viên nộp bài).

1.Giải pháp hiện thực

Khái niệm: Thuật toán Quick Sort là một thuật toán sắp xếp, còn được gọi là sắp xếp kiểu phân chia (Part Sort). Là một thuật toán hiệu quả dựa trên việc phân chia mảng dữ liệu thành các nhóm phần tử nhỏ hơn.

```
Phân loại: Giải thuật sắp xếp
Phức tạp thời gian: Trung bình O(n log n)
Xấu nhất: O(n^2)
Phức tạp dữ liệu: Khác nhau tùy vào cách hiện thực
Tối ưu: Thỉnh thoảng
```

Mô tả hoạt động: Giải thuật sắp xếp nhanh chia mảng thành hai phần bằng cách so sánh từng phần tử của mảng với một phần tử được gọi là **phần tử chốt**. Một mảng bao gồm các phần tử nhỏ hơn hoặc bằng phần tử chốt và một mảng gồm các phần tử lớn hơn phần tử chốt.

Quá trình phân chia này diễn ra cho đến khi độ dài của các mảng con đều bằng 1. Với phương pháp đệ quy ta có thể sắp xếp nhanh các mảng con sau khi kết thúc chương trình ta được một mảng đã sắp xếp hoàn chỉnh.

- ## Kỹ thuật chọn phần tử chốt

Kỹ thuật chọn phần tử chốt ảnh hưởng khá nhiều đến khả năng rơi vào các vòng lặp vô hạn đối với các trường hợp đặc biệt. Tốt nhất chọn phần tử chốt nằm ở trung vị của danh sách. Khi đó, sau $\log_2(n)$ lần chia ta đạt được kích thước mảng con bằng 1.

Dưới đây là một số cách chọn phần tử chốt:

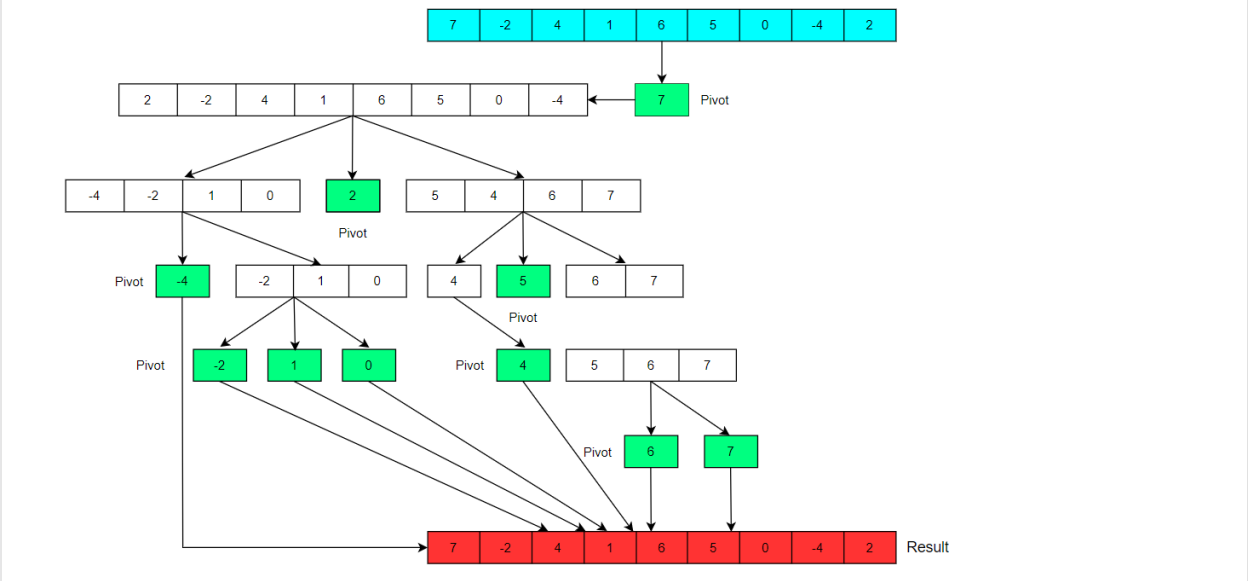
- Chọn phần tử đứng đầu hoặc đứng cuối làm phần tử chốt.
- Chọn phần tử đứng giữa danh sách làm phần tử chốt.
- Chọn phần tử trung vị trong ba phần tử đứng đầu, đứng giữa và đứng cuối làm phần tử chốt.
- Chọn phần tử ngẫu nhiên làm phần tử chốt. Tuy nhiên cách này rất dễ dẫn đến khả năng rơi vào các trường hợp đặc biệt.

- **Ý tưởng thuật toán Quick Sort – demo:**

1. Chọn phần tử chốt.
2. Khai báo 2 biến con trỏ để trỏ để duyệt 2 phía của phần tử chốt.
3. Biến bên trái trỏ đến từng phần tử mảng con bên trái của phần tử chốt.
4. Biến bên phải trỏ đến từng phần tử mảng con bên phải của phần tử chốt.
5. Khi biến bên trái nhỏ hơn phần tử chốt thì di chuyển sang phải.
6. Khi biến bên phải nhỏ hơn phần tử chốt thì di chuyển sang trái.
7. Nếu không xảy ra trường hợp 5 và 6 thì tráo đổi giá trị 2 biến trái và phải.
8. Nếu trái lớn hơn phải thì đây là giá trị chốt mới.

⇨ **Kỹ thuật chọn phần tử chốt được sử dụng:** Chọn phần tử đầu tiên trong dãy cần sắp xếp làm phần tử chốt.

`Minh họa giải thuật:`



2. Giải thuật Quick sort

- ***Giải thuật Quick sort hiện thực bằng ngôn ngữ C++:***

```
#include <iostream>
#include <random>
#include <functional>
#include <cstdlib>
#include <algorithm>
#include <vector>

using namespace std;

void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

int partition(int A[], int head, int tail) {
    int pivot = A[head];
    int i = head;
    int j;
    for (j = head + 1; j < tail; j++) {
        if (A[j] <= pivot) {
            i++;
            swap(&A[i], &A[j]);
        }
    }
    swap(&A[i], &A[head]);
    return i;
}

void quickSort(int A[], int head, int tail) {
    int mid;
    if (head < tail) {
        mid = partition(A, head, tail);
        quickSort(A, head, mid);
        quickSort(A, mid + 1, tail);
    }
}
int main() {
    return 0;
}
```

- ***Giải thuật Quick sort hiện thực bằng hợp ngữ MIPS:***

```
.globl main

.text

main:
  jal   input
  addi  $sp, $sp, -4 # 1 space
  sw    $v0, 0($sp)  # size
  li    $a1, 0       # a1 = head
  move  $a2, $v0     # a2 = tail = size
  jal   quicksort

  jal   output

  li    $v0, 10      # exit
  syscall

input:
  li  $t0, 0            # offset = 0
  li  $t1, 0            # size = 0
input_loop:
  li  $v0, 4            # print string syscall code = 4
  la  $a0, input_msg
  syscall

  li  $v0, 5            # read int syscall code = 5
  syscall

  sw    $v0, nums($t0)  # num[offset/4] = input
  addi  $t0, $t0, 4     # offset += 4
  beq   $v0, 10000, exit_input # if index == 10000, break -
-> choose a number to be a condition to break input loop
  addi  $t1, $t1, 1     # size += 1
  j     input_loop
exit_input:
  move  $v0, $t1        # return size
  jr    $ra             # return to caller
```

```
# void quickSort(int A[], int head, int tail) {
#   int mid;
#   if(head < tail) {
#     mid = partition(A, head, tail);
#     quickSort(A, head, mid);
#     quickSort(A, mid + 1, tail);
#   }
# }
quicksort:
  addi  $sp, $sp, -12 # 3 spaces
  sw    $a1, 0($sp)   # head
  sw    $a2, 4($sp)   # tail
  sw    $ra, 8($sp)   # return address
  bge   $a1, $a2, exit_quicksort # if head >= tail, exit
  jal   partition     # mid = partition(A, head, tail)
  move  $s0, $v0      # mid = $v0, from partition()
  lw    $a1, 0($sp)   # restore head
  move  $a2, $s0      # tail = mid
  jal   quicksort     # quickSort(A, head, mid)
  addi  $a1, $s0, 1   # head = mid + 1
  lw    $a2, 4($sp)   # restore tail
  jal   quicksort     # quickSort(A, mid + 1, tail)
exit_quicksort:
  lw    $a1, 0($sp)     # restore head
  lw    $a2, 4($sp)     # restore tail
  lw    $ra, 8($sp)     # restore return address
  addi  $sp, $sp, 12    # restore the stack
  jr    $ra             # return to caller
# int partition(int A[], int head, int tail) {
#   int pivot = A[head];
#   int i = head;
#   int j;
#
#   for(j = head + 1; j < tail; j++) {
#     if(A[j] <= pivot) {
#       i++;
#       swap(&A[i], &A[j]);
#     }
#   }
#   swap(&A[i], &A[head]);
#   return i;
# }
```

```
partition:
  addi  $sp, $sp, -4    # 1 space
  sw    $ra, 0($sp)     # return address
  move  $s1, $a1        # s1 = head
  move  $s2, $a2        # s2 = tail
  sll   $s3, $a1, 2     # index *= 4
  lw    $s3, nums($s3)  # pivot
  move  $s4, $a1        # i = head
  addi  $s5, $a1, 1     # j = head + 1
part_loop:
  bge   $s5, $s2, exit_loop # if j >= tail, break
  sll   $t1, $s4, 2         # index *= 4
  sll   $t2, $s5, 2         # index *= 4
  lw    $t1, nums($t1)
  lw    $t2, nums($t2)
  bgt   $t2, $s3, endif     # A[j] > pivot, endif
  addi  $s4, $s4, 1         # i++
  move  $a1, $s4
  move  $a2, $s5
  jal   swap                # swap(i, j)
  addi  $s5, $s5, 1         # j++
  j     part_loop

endif:
  addi  $s5, $s5, 1         # j++
  j     part_loop

exit_loop:
  move  $a1, $s4        # a1 = i
  move  $a2, $s1        # a2 = head
  jal   swap            # swap(i, head)
  move  $v0, $s4        # return i
  lw    $ra, 0($sp)
  addi  $sp, $sp, 4   # restore the stack
  jr    $ra           # return to caller
swap:
  sll  $a1, $a1, 2    # index1 *= 4
  sll  $a2, $a2, 2    # index2 *= 4
  lw   $t1, nums($a1)  # t1 = nums[index1]
  lw   $t2, nums($a2)  # t2 = nums[index2]
  sw   $t2, nums($a1)  # nums[index1] = t2
  sw   $t1, nums($a2)  # nums[index2] = t1
  jr   $ra             # return to caller
```

```
output:
  li   $t0, 0                    # offset
  li   $t1, 0                    # index
  lw   $t2, 0($sp)               # size
output_loop:
  li   $v0, 1                    # print int syscall code = 1
  lw   $a0, nums($t0)            # output = nums[offset/4]
  syscall

  li   $a0, 32                   # space ASCII = 32
  li   $v0, 11                   # print char syscall code =
11
  syscall

  addi  $t0, $t0, 4              # offset += 4
  addi  $t1, $t1, 1              # index  += 1
  bne   $t1, $t2, output_loop # index != size, continue
  li   $a0, 10                   # \n ASCII = 10
  li   $v0, 11                   # print char syscall code =
11
  syscall

  addi  $sp, $sp, 4              # restore the stack
  jr    $ra                      # return to caller
.data

nums:       .space  1000
input_msg:  .asciiz "Please enter a number, input the
number 10000 to stop: "
```

⇨ **Chương trình có 37 lệnh R, 65 lệnh I, 15 lệnh J.**

- ***Minh họa – phần edit trong MIPS:***

Edit Execute

mips12.asm*

```
.globl main

.text

main:
  jal   input #J-format                       #J
  addi  $sp, $sp, -4 # 1 space                 #I
  sw    $v0, 0($sp)  # size                   #I
  li    $a1, 0       # a1 = head              #I

  move  $a2, $v0     # a2 = tail = size       #R
  jal   quicksort                             #J

  jal   output                                #J

  li    $v0, 10      # exit                    #I
  syscall                                     #R

input:
  li  $t0, 0           # offset = 0           #I
  li  $t1, 0           # size = 0             #I
input_loop:
  li  $v0, 4           # print string syscall code = 4       #I
  la  $a0, input_msg                                          #2 lenh I
  syscall                                                     #R

  li  $v0, 5           # read int syscall code = 5           #I
  syscall                                                     #R

  sw    $v0, nums($t0)  # num[offset/4] = input               #2 lenh I, 1 lenh R
  addi  $t0, $t0, 4     # offset +=                           #I
  beq   $v0, 10000, exit_input # if index == 10000, break --> choose a number to be a condition to break input loop    #2 lenh I
  addi  $t1, $t1, 1     # size += 1                                                                                     #I
  j     input_loop                                                                                                    #J

exit_input:
```

Line: 152 Column: 78 Show Line Numbers

Edit Execute

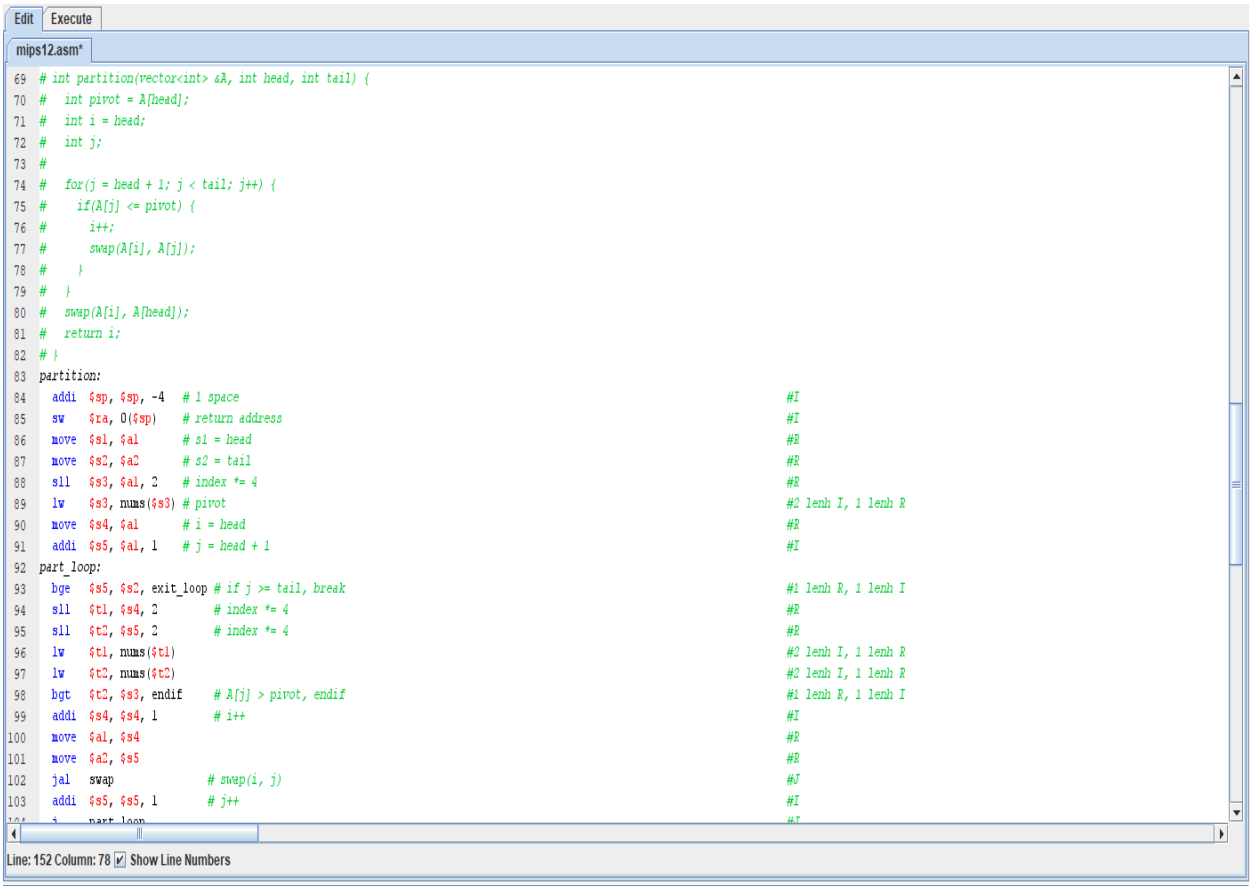
mips12.asm*

```

exit_input:
  add   $s7,$s7,$zero   #save size to use later
  move  $v0, $t1        # return size                                                 #R

  jr    $ra             # return to caller                                          #R
# void quickSort(vector<int> &A, int head, int tail) {
#   int mid;
#   if(head < tail) {
#     mid = partition(A, head, tail);
#     quickSort(A, head, mid);
#     quickSort(A, mid + 1, tail);
#   }
# }
quicksort:
  addi  $sp, $sp, -12 # 3 spaces                                                    #I
  sw    $a1, 0($sp)   # head                                                        #I
  sw    $a2, 4($sp)   # tail                                                        #I
  sw    $ra, 8($sp)   # return address                                            #I
  bge   $a1, $a2, exit_quicksort # if head >= tail, exit                            #1 lenh R, 1 lenh I
  jal   partition     # mid = partition(A, head, tail)                              #J
  move  $s0, $v0      # mid = $r0, from partition()                                 #R
  lw    $a1, 0($sp)   # restore head                                                #I
  move  $a2, $s0      # tail = mid                                                  #R
  jal   quicksort     # quickSort(A, head, mid)                                     #J
  addi  $a1, $s0, 1   # head = mid + 1                                             #I
  lw    $a2, 4($sp)   # restore tail                                               #I
  jal   quicksort     # quickSort(A, mid + 1, tail)                                 #J
exit_quicksort:
  lw    $a1, 0($sp)     # restore head                                              #I
  lw    $a2, 4($sp)     # restore tail                                              #I
  lw    $ra, 8($sp)     # restore return address                                    #I
  addi  $sp, $sp, 12    # restore the stack                                         #I
  jr    $ra             # return to caller                                          #R
# int partition(vector<int> &A, int head, int tail) {
#   int pivot = A[head];
```

Line: 152 Column: 78 Show Line Numbers



Edit Execute

mips12.asm*

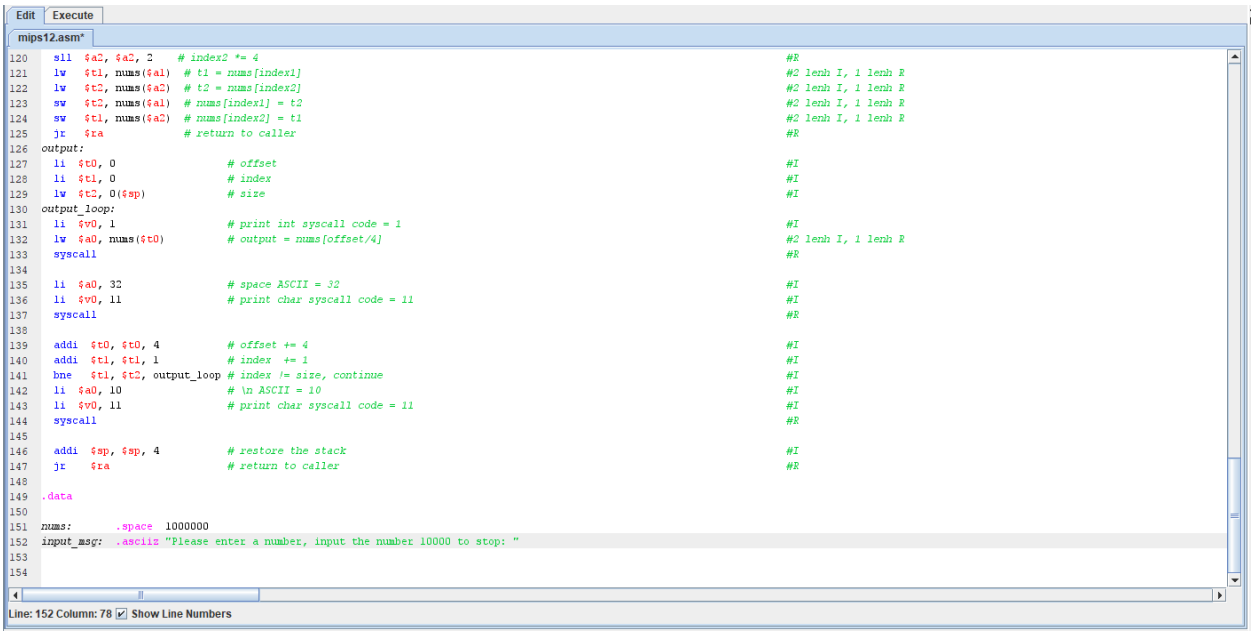
```
# int partition(vector<int> &A, int head, int tail) {
#   int pivot = A[head];
#   int i = head;
#   int j;
#
#   for(j = head + 1; j < tail; j++) {
#     if(A[j] <= pivot) {
#       i++;
#       swap(A[i], A[j]);
#     }
#   }
#   swap(A[i], A[head]);
#   return i;
# }
partition:
  addi  $sp, $sp, -4   # 1 space                                  #I
  sw    $ra, 0($sp)    # return address                           #I
  move  $s1, $a1       # s1 = head                                #R
  move  $s2, $a2       # s2 = tail                                #R
  sll   $s3, $a1, 2    # index *= 4                               #R
  lw    $s3, nums($s3) # pivot                                   #2 lenh I, 1 lenh R
  move  $s4, $a1       # i = head                                 #R
  addi  $s5, $a1, 1    # j = head + 1                             #I
part_loop:
  bge   $s5, $s2, exit_loop # if j >= tail, break                 #1 lenh R, 1 lenh I
  sll   $t1, $s4, 2         # index *= 4                          #R
  sll   $t2, $s5, 2         # index *= 4                          #R
  lw    $t1, nums($t1)                                           #2 lenh I, 1 lenh R
  lw    $t2, nums($t2)                                           #2 lenh I, 1 lenh R
  bgt   $t2, $s3, endif     # A[j] > pivot, endif                 #1 lenh R, 1 lenh I
  addi  $s4, $s4, 1         # i++                                 #I
  move  $a1, $s4                                                 #R
  move  $a2, $s5                                                 #R
  jal   swap               # swap(i, j)                          #J
  addi  $s5, $s5, 1         # j++                                 #I
  j     part_loop                                                #J

endif:
  addi  $s5, $s5, 1         # j++                                 #I
  j     part_loop                                                #J

exit_loop:
  move  $a1, $s4       # a1 = i                                   #R
  move  $a2, $s1       # a2 = head                                #R
  jal   swap           # swap(i, head)                            #J
  move  $v0, $s4       # return i                                 #R
  lw    $ra, 0($sp)                                              #I
  addi  $sp, $sp, 4    # restore the stack                        #I
  jr    $ra            # return to caller                         #R
swap:
  sll  $a1, $a1, 2     # index1 *= 4                              #R
  sll  $a2, $a2, 2     # index2 *= 4                              #R
  lw   $t1, nums($a1)  # t1 = nums[index1]                       #2 lenh I, 1 lenh R
  lw   $t2, nums($a2)  # t2 = nums[index2]                       #2 lenh I, 1 lenh R
  sw   $t2, nums($a1)  # nums[index1] = t2                       #2 lenh I, 1 lenh R
  sw   $t1, nums($a2)  # nums[index2] = t1                       #2 lenh I, 1 lenh R
  jr   $ra             # return to caller                        #R
output:
  li  $t0, 0                # offset                             #I
  li  $t1, 0                # index                              #I
  lw  $t2, 0($sp)           # size                               #I
output_loop:
  li  $v0, 1                # print int syscall code = 1         #I
  lw  $a0, nums($t0)        # output = nums[offset/4]            #2 lenh I, 1 lenh R
  syscall                                                        #R

  li  $a0, 32               # space ASCII = 32                   #I
  li  $v0, 11               # print char syscall code = 11       #I
  syscall                                                        #R
```

Line: 152 Column: 78 Show Line Numbers


Edit
Execute
mips12.asm*
sll $a2, $a2, 2 # index2 *= 4 #R
lw $t1, nums($a1) # t1 = nums[index1] #2 lenh I, 1 lenh R
lw $t2, nums($a2) # t2 = nums[index2] #2 lenh I, 1 lenh R
sw $t2, nums($a1) # nums[index1] = t2 #2 lenh I, 1 lenh R
sw $t1, nums($a2) # nums[index2] = t1 #2 lenh I, 1 lenh R
jr $ra # return to caller #R
output:
li $t0, 0 # offset #I
li $t1, 0 # index #I
lw $t2, 0($sp) # size #I
output_loop:
li $v0, 1 # print int syscall code = 1 #I
lw $a0, nums($t0) # output = nums[offset/4] #2 lenh I, 1 lenh R
syscall #R
li $a0, 32 # space ASCII = 32 #I
li $v0, 11 # print char syscall code = 11 #I
syscall #R
addi $t0, $t0, 4 # offset += 4 #I
addi $t1, $t1, 1 # index += 1 #I
bne $t1, $t2, output_loop # index != size, continue #I
li $a0, 10 # \n ASCII = 10 #I
li $v0, 11 # print char syscall code = 11 #I
syscall #R
addi $sp, $sp, 4 # restore the stack #I
jr $ra # return to caller #R
.data
nums: .space 1000000
input_msg: .asciiz "Please enter a number, input the number 10000 to stop: "
Line: 152 Column: 78
Show Line Numbers

- **Minh họa chi tiết các hàm:**
- **Void swap: chuyển đổi vị trí của 2 phần tử trong dãy được nhập**

- C++:

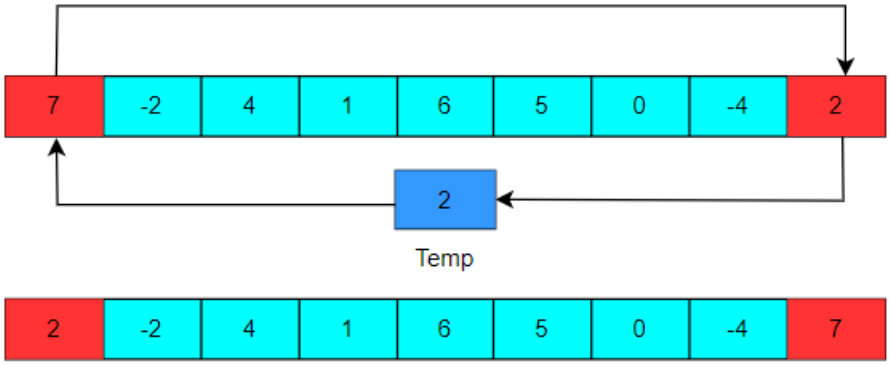
```
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}
```

- MIPS:

```
swap:
    sll  $a1, $a1, 2     # index1 *= 4
    sll  $a2, $a2, 2     # index2 *= 4
    lw   $t1, nums($a1)   # t1 = nums[index1]
    lw   $t2, nums($a2)   # t2 = nums[index2]
    sw   $t2, nums($a1)   # nums[index1] = t2
    sw   $t1, nums($a2)   # nums[index2] = t1

    jr   $ra              # return to caller
```

- **Minh họa hàm swap:**
- **Cho 2 giá trị a và b cần đổi(màu đỏ). Lưu giá trị b(hoặc a) vào biến tạm thời temp(màu xanh đậm)rồi cho a=b, b=temp => giá trị tại 2 vị trí trên mảng đã được thay đổi:**



- **Void partition: chọn phần tử trục và tiến hành đổi chỗ các phần tử khi so sánh với trục**

- C++:

```
int partition(int A[], int head, int tail) {
    int pivot = A[head];
    int i = head;
    int j;
    for (j = head + 1; j < tail; j++) {
        if (A[j] <= pivot) {
            i++;
            swap(&A[i], &A[j]);
        }
    }
    swap(&A[i], &A[head]);
    return i;
}
```

- MIPS:

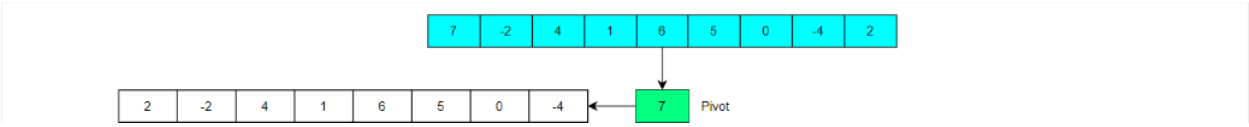
```
partition:
  addi  $sp, $sp, -4    # 1 space
  sw    $ra, 0($sp)     # return address
  move  $s1, $a1        # s1 = head
  move  $s2, $a2        # s2 = tail
  sll   $s3, $a1, 2     # index *= 4
  lw    $s3, nums($s3)  # pivot
  move  $s4, $a1        # i = head
  addi  $s5, $a1, 1     # j = head + 1
part_loop:
  bge   $s5, $s2, exit_loop # if j >= tail,
break
  sll   $t1, $s4, 2          # index *= 4
  sll   $t2, $s5, 2          # index *= 4
  lw    $t1, nums($t1)
  lw    $t2, nums($t2)
  bgt   $t2, $s3, endif      # A[j] > pivot,
endif
  addi  $s4, $s4, 1          # i++
  move  $a1, $s4
  move  $a2, $s5
  jal   swap                 # swap(i, j)
  addi  $s5, $s5, 1          # j++
  j     part_loop
```

- MIPS:

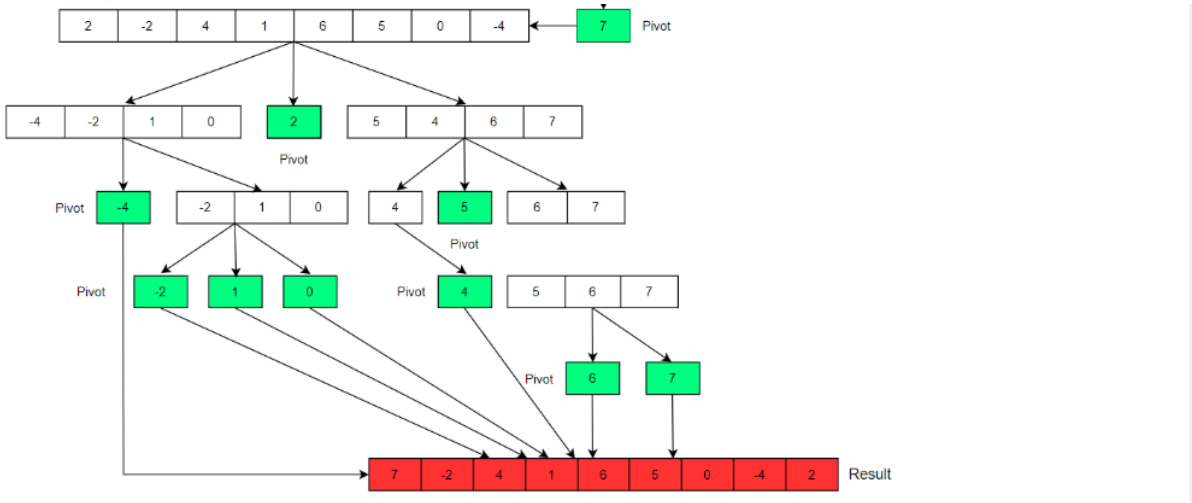
```
endif:
  addi  $s5, $s5, 1          # j++
  j     part_loop

exit_loop:
  move  $a1, $s4          # a1 = i
  move  $a2, $s1          # a2 = head
  jal   swap              # swap(i, head)
  move  $v0, $s4          # return i
  lw    $ra, 0($sp)
  addi  $sp, $sp, 4     # restore the stack
  jr    $ra             # return to caller
```

- **Hình minh họa: Dãy nhập là dòng màu xanh dương**
- **Quy ước: dãy nằm bên trái là các phần tử nhỏ hơn hoặc bằng phần tử trục còn dãy nằm bên phải là các phần tử lớn hơn phần tử trục – phần tử trục có màu xanh lá:**
- **Luôn lấy phần tử đầu làm phần tử trục:**



- **Quá trình chọn trục và thay đổi vị trí các phần tử theo quy ước:**



- **Kết quả là dãy đã sắp xếp ở dòng màu đỏ.**

- **Void quickSort: Dùng đệ quy lặp lại quá trình chọn phần tử trục (đã được minh họa tại hàm partition)**

- C++:

```
void quickSort(int A[], int head, int tail) {
    int mid;
    if (head < tail) {
        mid = partition(A, head, tail);
        quickSort(A, head, mid);
        quickSort(A, mid + 1, tail);
    }
}
```

- MIPS:

```
quicksort:
  addi  $sp, $sp, -12 # 3 spaces
  sw    $a1, 0($sp)   # head
  sw    $a2, 4($sp)   # tail
  sw    $ra, 8($sp)   # return address
  bge   $a1, $a2, exit_quicksort # if head >=
tail, exit
  jal   partition     # mid = partition(A, head,
tail)
  move  $s0, $v0      # mid = $v0, from
partition()
  lw    $a1, 0($sp)   # restore head
  move  $a2, $s0      # tail = mid
  jal   quicksort     # quickSort(A, head, mid)
  addi  $a1, $s0, 1   # head = mid + 1
  lw    $a2, 4($sp)   # restore tail
  jal   quicksort     # quickSort(A, mid + 1,
tail)
exit_quicksort:
  lw    $a1, 0($sp)     # restore head
  lw    $a2, 4($sp)     # restore tail
  lw    $ra, 8($sp)     # restore return address
  addi  $sp, $sp, 12    # restore the stack
  jr    $ra             # return to caller
```

3. Trường hợp test giải thuật (gồm 30 test cases), kết quả và thời gian thực thi của mỗi test case:

TC ID	SIZE	TEST DATA	EXPECTED	GOT	PASS/FAIL	R-type	I-type	J-type	TIME EXCUTE
1	50	-989 -965 -909 -865 -861 -739 -614 -510 -508 -405 -391 -321 -305 -241 -220 -17 -15 10 11 41 43 67 100 100 150 154 177 233 277 346 361 415 423 436 494 495 548 566 582 588 638 642 776 783 845 874 901 938 986 997	-989 -965 -909 -865 -861 -739 -614 -510 -508 -405 -391 -321 -305 -241 -220 -17 -15 10 11 41 43 67 100 100 150 154 177 233 277 346 361 415 423 436 494 495 548 566 582 588 638 642 776 783 845 874 901 938 986 997	-989 -965 -909 -865 -861 -739 -614 -510 -508 -405 -391 -321 -305 -241 -220 -17 -15 10 11 41 43 67 100 100 150 154 177 233 277 346 361 415 423 436 494 495 548 566 582 588 638 642 776 783 845 874 901 938 986 997	PASS	8843	11401	1483	21727
2	50	-924 -849 -801 -772 -739 -611 -504 -427 -404 -342 -331 -319 -272 -244 -240 -169 -150 -141 -63 37 49 70 117 183 298 305 308 357 396 433 445 473 510 541 556 607 618 627 747 756 803 817 836 859 862 866 899 948 957 962	-924 -849 -801 -772 -739 -611 -504 -427 -404 -342 -331 -319 -272 -244 -240 -169 -150 -141 -63 37 49 70 117 183 298 305 308 357 396 433 445 473 510 541 556 607 618 627 747 756 803 817 836 859 862 866 899 948 957 962	-924 -849 -801 -772 -739 -611 -504 -427 -404 -342 -331 -319 -272 -244 -240 -169 -150 -141 -63 37 49 70 117 183 298 305 308 357 396 433 445 473 510 541 556 607 618 627 747 756 803 817 836 859 862 866 899 948 957 962	PASS	8811	11355	1478	21644
3	50	-996 -895 -811 -797 -759 -725 -724 -642 -616 -557 -390 -311 -307 -150 -65 -63 17 28 28 48 59 94 94 100 147 207 210 214 292 423 446 456 574 607 637 654 659 670 699 707 708 732 747 762 763 766 876 958 965 975	-996 -895 -811 -797 -759 -725 -724 -642 -616 -557 -390 -311 -307 -150 -65 -63 17 28 28 48 59 94 94 100 147 207 210 214 292 423 446 456 574 607 637 654 659 670 699 707 708 732 747 762 763 766 876 958 965 975	-996 -895 -811 -797 -759 -725 -724 -642 -616 -557 -390 -311 -307 -150 -65 -63 17 28 28 48 59 94 94 100 147 207 210 214 292 423 446 456 574 607 637 654 659 670 699 707 708 732 747 762 763 766 876 958 965 975	PASS	8875	11447	1488	21810
4	50	-983 -983 -977 -953 -835 -795 -773 -595 -590 -520 -421 -401 -331 -302 -295 17 18 33 51 135 148 161 171 173 189 302 307 329 347 437 448 451 541 553 567 582 604 619 642 652 707 732 760 769 855 893 895 895 923 975	-983 -983 -977 -953 -835 -795 -773 -595 -590 -520 -421 -401 -331 -302 -295 17 18 33 51 135 148 161 171 173 189 302 307 329 347 437 448 451 541 553 567 582 604 619 642 652 707 732 760 769 855 893 895 895 923 975	-983 -983 -977 -953 -835 -795 -773 -595 -590 -520 -421 -401 -331 -302 -295 17 18 33 51 135 148 161 171 173 189 302 307 329 347 437 448 451 541 553 567 582 604 619 642 652 707 732 760 769 855 893 895 895 923 975	PASS	8875	11447	1488	21810
5	50	-952 -949 -913 -870 -748 -592 -583 -578 -536 -367 -358 -327 -267 -257 -242 -192 -151 -122 -103 -70 -69 29 31 34 90 155 216 234 242 247 300 342 375 377 388 468 473 477 491 504 742 765 786 795 828 828 854 865 867 880	-952 -949 -913 -870 -748 -592 -583 -578 -536 -367 -358 -327 -267 -257 -242 -192 -151 -122 -103 -70 -69 29 31 34 90 155 216 234 242 247 300 342 375 377 388 468 473 477 491 504 742 765 786 795 828 828 854 865 867 880	-952 -949 -913 -870 -748 -592 -583 -578 -536 -367 -358 -327 -267 -257 -242 -192 -151 -122 -103 -70 -69 29 31 34 90 155 216 234 242 247 300 342 375 377 388 468 473 477 491 504 742 765 786 795 828 828 854 865 867 880	PASS	8843	11401	1483	21727

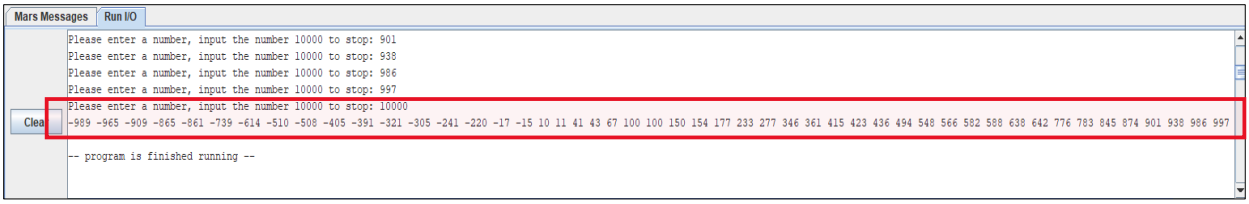
6	50	484 -306 3 83 400 576 -799 832 -748 -140 785 165 -892 -719 523 276 77 -928 523 870 233 -854 -951 749 -174 760 492 484 173 -767 52 20 601 341 861 893 649 318 497 936 -730 -935 854 985 570 317 799 186 786 536	-951 -935 -928 -892 -854 -799 -767 -748 -730 -719 -306 -174 -140 3 20 52 77 83 165 173 186 233 276 317 318 341 400 484 484 492 497 523 523 536 570 576 601 649 749 760 785 786 799 832 854 861 870 893 936 985	-951 -935 -928 -892 -854 -799 -767 -748 -730 -719 -306 -174 -140 3 20 52 77 83 165 173 186 233 276 317 318 341 400 484 484 492 497 523 523 536 570 576 601 649 749 760 785 786 799 832 854 861 870 893 936 985	PASS	5120	7091	806	13017
7	50	291 -994 975 977 824 384 447 552 915 386 988 401 -796 501 383 698 895 -58 -959 -278 784 379 21 505 672 750 -215 -175 437 153 347 -196 -654 697 792 30 404 856 288 203 -758 21 -732 961 637 449 -363 -347 35 177	-994 -959 -796 -758 -732 -654 -363 -347 -278 -215 -196 -175 -58 21 21 30 35 153 177 203 288 291 347 379 383 384 386 401 404 437 447 449 501 505 552 637 672 697 698 750 784 792 824 856 895 915 961 975 977 988	-994 -959 -796 -758 -732 -654 -363 -347 -278 -215 -196 -175 -58 21 21 30 35 153 177 203 288 291 347 379 383 384 386 401 404 437 447 449 501 505 552 637 672 697 698 750 784 792 824 856 895 915 961 975 977 988	PASS	5341	7361	834	13536
8	50	461 62 -444 870 -982 -965 356 974 427 457 112 574 150 739 740 558 172 -266 10 884 810 -432 196 -844 698 735 20 767 799 977 725 53 682 94 719 110 307 350 237 810 499 -580 395 23 -949 722 -254 59 732 50	-982 -965 -949 -844 -580 -444 -432 -266 -254 10 20 23 50 53 59 62 94 110 112 150 172 196 237 307 350 356 395 427 457 461 499 558 574 682 698 719 722 725 732 735 739 740 767 799 810 810 870 884 974 977	-982 -965 -949 -844 -580 -444 -432 -266 -254 10 20 23 50 53 59 62 94 110 112 150 172 196 237 307 350 356 395 427 457 461 499 558 574 682 698 719 722 725 732 735 739 740 767 799 810 810 870 884 974 977	PASS	4964	6956	786	12706
9	50	703 -77 714 -77 -514 -806 -681 -916 -313 -49 -984 -640 -401 -210 114 -663 -645 -956 -334 67 -896 -74 -618 652 -395 143 -3 433 -147 -975 157 555 651 274 673 -691 -661 964 -660 561 -733 -377 -993 -717 408 -219 872 626 236 -133	-993 -984 -975 -956 -916 -896 -806 -733 -717 -691 -681 -663 -661 -660 -645 -640 -618 -514 -401 -395 -377 -334 -313 -219 -210 -147 -133 -77 -77 -74 -49 -3 67 114 143 157 236 274 408 433 555 561 626 651 652 673 703 714 872 964	-993 -984 -975 -956 -916 -896 -806 -733 -717 -691 -681 -663 -661 -660 -645 -640 -618 -514 -401 -395 -377 -334 -313 -219 -210 -147 -133 -77 -77 -74 -49 -3 67 114 143 157 236 274 408 433 555 561 626 651 652 673 703 714 872 964	PASS	5508	7478	852	13838
10	50	-26 -670 -324 266 -22 -543 565 -934 113 -623 -409 36 447 -685 199 145 -130 -56 824 -903 904 -618 -219 -65 203 252 928 -192 -762 240 809 -456 -151 998 835 -512 153 -958 -804 931 632 -899 483 -454 -494 -900 -509 377 4 -681	-958 -934 -903 -900 -899 -804 -762 -685 -681 -670 -623 -618 -543 -512 -509 -494 -456 -454 -409 -324 -219 -192 -151 -130 -65 -56 -26 -22 4 36 113 145 153 199 203 240 252 266 377 447 483 565 632 809 824 835 904 928 931 998	-958 -934 -903 -900 -899 -804 -762 -685 -681 -670 -623 -618 -543 -512 -509 -494 -456 -454 -409 -324 -219 -192 -151 -130 -65 -56 -26 -22 4 36 113 145 153 199 203 240 252 266 377 447 483 565 632 809 824 835 904 928 931 998	PASS	4930	6896	780	12606

11	50	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	PASS	20963	24193	2899	48055
12	50	87 -608 833 139 -303 -746 -350 238 -484 -550 389 -748 -315 -528 338 903 224 -765 201 410 916 -497 -17 583 -403 710 -951 820 34 -307 948 593 990 927 258 236 -356 71 899 98 -404 -814 - 542 -713 854 -68 315 235 588 -420	-951 -814 -765 -748 -746 -713 -608 -550 - 542 -528 -497 -484 -420 -404 -403 -356 - 350 -315 -307 -303 -68 -17 34 71 87 98 139 201 224 235 236 238 258 315 338 389 410 583 588 593 710 820 833 854 899 903 916 927 948 990	-951 -814 -765 -748 -746 -713 -608 -550 - 542 -528 -497 -484 -420 -404 -403 -356 - 350 -315 -307 -303 -68 -17 34 71 87 98 139 201 224 235 236 238 258 315 338 389 410 583 588 593 710 820 833 854 899 903 916 927 948 990	PASS	4846	6792	764	12402
13	50	636 -576 292 -348 753 159 -740 -670 -979 -798 612 -434 -494 -69 926 287 422 -736 -428 -379 395 628 760 -203 122 850 -623 -316 209 98 457 -972 - 393 577 189 911 377 177 320 9 -299 -444 -549 6 - 652 533 -947 -244 805 865	-979 -972 -947 -798 -740 -736 -670 -652 - 623 -576 -549 -494 -444 -434 -428 -393 - 379 -348 -316 -299 -244 -203 -69 6 9 98 122 159 177 189 209 287 292 320 377 395 422 457 533 577 612 628 636 753 760 805 850 865 911 926	-979 -972 -947 -798 -740 -736 -670 -652 - 623 -576 -549 -494 -444 -434 -428 -393 - 379 -348 -316 -299 -244 -203 -69 6 9 98 122 159 177 189 209 287 292 320 377 395 422 457 533 577 612 628 636 753 760 805 850 865 911 926	PASS	5132	7114	811	13057
14	50	86 -5 71 20 -23 -29 -5 34 -35 -39 47 -71 -10 -4 72 31 23 -48 1 7 76 -4 -4 74 -77 25 -22 64 75 -66 27 89 48 34 11 69 -75 56 24 4 - 33 -62 -68 -33 43 -46 31 84 46 -73	-77 -75 -73 -71 -68 -66 -62 -48 -46 -39 -35 -33 -33 -29 -23 -22 -10 -5 -5 -4 -4 -4 1 4 7 11 20 23 24 25 27 31 31 34 34 43 46 47 48 56 64 69 71 72 74 75 76 84 86 89	-77 -75 -73 -71 -68 -66 -62 -48 -46 -39 -35 -33 -33 -29 -23 -22 -10 -5 -5 -4 -4 -4 1 4 7 11 20 23 24 25 27 31 31 34 34 43 46 47 48 56 64 69 71 72 74 75 76 84 86 89	PASS	5870	7994	921	14785
15	50	4 26 -85 -75 79 -77 14 62 -30 9 16 -43 -48 -71 - 72 95 2 28 46 57 -48 -92 -95 81 9 80 86 -45 -97 -16 18 46 -20 -74 -67 -70 73 40 93 - 45 -10 -70 -80 -60 -88 -19 15 -54 92 49	-97 -95 -92 -88 -85 -80 -77 -75 -74 -72 -71 -70 -70 -67 -60 -54 -48 -48 -45 -45 -43 -30 -20 -19 -16 -10 2 4 9 9 14 15 16 18 26 28 40 46 46 49 57 62 73 79 80 81 86 92 93 95	-97 -95 -92 -88 -85 -80 -77 -75 -74 -72 -71 -70 -70 -67 -60 -54 -48 -48 -45 -45 -43 -30 -20 -19 -16 -10 2 4 9 9 14 15 16 18 26 28 40 46 46 49 57 62 73 79 80 81 86 92 93 95	PASS	4906	6888	773	12567
16	50	72 -83 64 68 -18 -26 -90 90 -12 -65 45 -73 -42 - 95 28 31 60 -2 22 56 88 -5 -9 52 -17 63 -53 1 97 -86 45 37 52 50 52 8 -39 43 86 37 -36 -89 0 -72 90 -27 45 88 63 -13	-95 -90 -89 -86 -83 -73 -72 -65 -53 -42 -39 -36 -27 -26 -18 -17 -13 -12 -9 -5 -2 0 1 8 22 28 31 37 37 43 45 45 45 50 52 52 52 56 60 63 63 64 68 72 86 88 88 90 90 97	-95 -90 -89 -86 -83 -73 -72 -65 -53 -42 -39 -36 -27 -26 -18 -17 -13 -12 -9 -5 -2 0 1 8 22 28 31 37 37 43 45 45 45 50 52 52 52 56 60 63 63 64 68 72 86 88 88 90 90 97	PASS	5171	7163	809	13143
17	50	71 -75 11 -63 66 40 -96 -28 -2 -10 85 -38 -24 -12 18 55 38 -10 -83 -7 72 8 86 -62 91 0 -96 -85 82 90 91 -16 -2 5 24 74 5 92 67 16 - 73 -36 -29 78 -52 29 -2 -91 94 77	-96 -96 -91 -85 -83 -75 -73 -63 -62 -52 -38 -36 -29 -28 -24 -16 -12 -10 -10 -7 -2 -2 -2 0 5 5 8 11 16 18 24 29 38 40 55 66 67 71 72 74 77 78 82 85 86 90 91 91 92 94	-96 -96 -91 -85 -83 -75 -73 -63 -62 -52 -38 -36 -29 -28 -24 -16 -12 -10 -10 -7 -2 -2 -2 0 5 5 8 11 16 18 24 29 38 40 55 66 67 71 72 74 77 78 82 85 86 90 91 91 92 94	PASS	5705	7824	901	14430
18	50	-57 -53 -5 80 -72 -27 38 -21 85 4 25 38 33 -54 14 31 -59 -12 95 61 13 -28 -48 -18 36 84 44 12 50 36 52 94 -39 87 62 98 66 -1 46 22 5 -93 26 -52 -60 -8 -92 67 84 -68	-93 -92 -72 -68 -60 -59 -57 -54 -53 -52 -48 -39 -28 -27 -21 -18 -12 -8 -5 -1 4 5 12 13 14 22 25 26 31 33 36 36 38 38 44 46 50 52 61 62 66 67 80 84 84 85 87 94 95 98	-93 -92 -72 -68 -60 -59 -57 -54 -53 -52 -48 -39 -28 -27 -21 -18 -12 -8 -5 -1 4 5 12 13 14 22 25 26 31 33 36 36 38 38 44 46 50 52 61 62 66 67 80 84 84 85 87 94 95 98	PASS	5347	7423	863	13633
19	50	593 -311 572 482 -729 -374 -717 782 -31 -267 45 -957 -921 -524 966 61 578 -545 882 643 176 -87 - 116 496 -865 872 -886 232 676 -566 538 560 172 563 74 118 -537 740 403 683 -95 -581 -700 -503 542 - 758 66 80 953 -357	-957 -921 -886 -865 -758 -729 -717 -700 - 581 -566 -545 -537 -524 -503 -374 -357 - 311 -267 -116 -95 -87 -31 45 61 66 74 80 118 172 176 232 403 482 496 538 542 560 563 572 578 593 643 676 683 740 782 872 882 953 966	-957 -921 -886 -865 -758 -729 -717 -700 - 581 -566 -545 -537 -524 -503 -374 -357 - 311 -267 -116 -95 -87 -31 45 61 66 74 80 118 172 176 232 403 482 496 538 542 560 563 572 578 593 643 676 683 740 782 872 882 953 966	PASS	5310	7282	821	13413
20	50	-920 -702 -270 548 -500 -968 384 -67 164 727 545 138 643 -945 25 894 -634 -633 408 887 986 - 582 -898 -67 63 987 25 333 431 702 596 605 -166 559 901 451 279 -225 444 360 719 -385 213 -437 -807 -18 - 746 554 790 -190	-968 -945 -920 -898 -807 -746 -702 -634 - 633 -582 -500 -437 -385 -270 -225 -190 - 166 -67 -67 -18 25 25 63 138 164 213 279 333 360 384 408 431 444 451 545 548 554 559 596 605 643 702 719 727 790 887 894 901 986 987	-968 -945 -920 -898 -807 -746 -702 -634 - 633 -582 -500 -437 -385 -270 -225 -190 - 166 -67 -67 -18 25 25 63 138 164 213 279 333 360 384 408 431 444 451 545 548 554 559 596 605 643 702 719 727 790 887 894 901 986 987	PASS	6001	8119	940	15060
21	50	-202 -196 -615 552 -813 -972 618 -78 572 430 103 135 430 -457 306 820 -748 -302 203 280 356 -376 -845 -171 683 734 369 194 7 14 101 726 -25 638 590 229 495 -103 237 501 956 -974 323 -102 -698 - 906 -52 460 144 -200	-974 -972 -906 -845 -813 -748 -698 -615 - 457 -376 -302 -202 -200 -196 -171 -103 - 102 -78 -52 -25 7 14 101 103 135 144 194 203 229 237 280 306 323 356 369 430 430 460 495 501 552 572 590 618 638 683 726 734 820 956	-974 -972 -906 -845 -813 -748 -698 -615 - 457 -376 -302 -202 -200 -196 -171 -103 - 102 -78 -52 -25 7 14 101 103 135 144 194 203 229 237 280 306 323 356 369 430 430 460 495 501 552 572 590 618 638 683 726 734 820 956	PASS	5271	7298	835	13404
22	50	-2997 -6216 -4889 51 -3899 -5137 4576 -4483 5208 1923 6109 9646 1676 -1117 3237 3892 - 9812 -6795 58 9596 6080 -7578 -8715 -4409 1105 5473 9056 400 5454 715 4622 9694 - 4121 190 3011 1721 3805 -4514 1400 8039 4324 - 8020 9800 -3831 -9983 -7322 -7216 6740 7360 - 7391	-9983 -9812 -8715 -8020 -7578 -7391 - 7322 -7216 -6795 -6216 -5137 -4889 - 4514 -4483 -4409 -4121 -3899 -3831 - 2997 -1117 51 58 190 400 715 1105 1400 1676 1721 1923 3011 3237 3805 3892 4324 4576 4622 5208 5454 5473 6080 6109 6740 7360 8039 9056 9596 9646 9694 9800	-9983 -9812 -8715 -8020 -7578 -7391 - 7322 -7216 -6795 -6216 -5137 -4889 -4514 -4483 -4409 -4121 -3899 -3831 -2997 - 1117 51 58 190 400 715 1105 1400 1676 1721 1923 3011 3237 3805 3892 4324 4576 4622 5208 5454 5473 6080 6109 6740 7360 8039 9056 9596 9646 9694 9800	PASS	5361	7426	848	13635
23	50	-7774 -4347 -5672 875 -2032 -1577 9568 -5905 9688 4455 4873 4601 5952 -7578 5670 36 -4099 - 9834 6374 4155 139 -7177 -9669 -219 3210 6810 1747 9129 2628 6812 3253 1870 - 9066 7826 6211 4756 7746 -4681 9446 5098 485 - 2113 9826 -1054 -5956 -4554 -4685 5245 5013 - 7554	-9834 -9669 -9066 -7774 -7578 -7554 - 7177 -5956 -5905 -5672 -4685 -4681 - 4554 -4347 -4099 -2113 -2032 -1577 - 1054 -219 36 139 485 875 1747 1870 2628 3210 3253 4155 4455 4601 4756 4873 5013 5098 5245 5670 5952 6211 6374 6810 6812 7746 7826 9129 9446 9568 9688 9826	-9834 -9669 -9066 -7774 -7578 -7554 - 7177 -5956 -5905 -5672 -4685 -4681 -4554 -4347 -4099 -2113 -2032 -1577 -1054 -219 36 139 485 875 1747 1870 2628 3210 3253 4155 4455 4601 4756 4873 5013 5098 5245 5670 5952 6211 6374 6810 6812 7746 7826 9129 9446 9568 9688 9826	PASS	5982	8102	945	15029
24	50	736 -9 811 270 -807 514 -669 825 -182 -671 697 - 54 -781 -932 43 651 89 -68 160 380 706 -709 - 869 614 -938 562 138 231 749 -544 284 4 908 165 951 309 -150 742 628 271 -376 -623 -356 291 930 698 950 402 672 -497	-938 -932 -869 -807 -781 -709 -671 -669 - 623 -544 -497 -376 -356 -182 -150 -68 -54 -9 4 43 89 138 160 165 231 270 271 284 291 309 380 402 514 562 614 628 651 672 697 698 706 736 742 749 811 825 908 930 950 951	-938 -932 -869 -807 -781 -709 -671 -669 - 623 -544 -497 -376 -356 -182 -150 -68 -54 - 9 4 43 89 138 160 165 231 270 271 284 291 309 380 402 514 562 614 628 651 672 697 698 706 736 742 749 811 825 908 930 950 951	PASS	5181	7167	818	13166
25	50	561 -568 327 -880 4 222 666 -354 -439 -757 173 - 341 -706 -747 298 104 615 -755 -620 -584 903 54 721 330 381 14 -607 -230 669 359 89 -916 -56 579 92 837 125 256 347 755 -163 -933 -923 302 399 545 -387 -475 969 847	-933 -923 -916 -880 -757 -755 -747 -706 - 620 -607 -584 -568 -475 -439 -387 -354 - 341 -230 -163 -56 4 14 54 89 92 104 125 173 222 256 298 302 327 330 347 359 381 399 545 561 579 615 666 669 721 755 837 847 903 969	-933 -923 -916 -880 -757 -755 -747 -706 - 620 -607 -584 -568 -475 -439 -387 -354 - 341 -230 -163 -56 4 14 54 89 92 104 125 173 222 256 298 302 327 330 347 359 381 399 545 561 579 615 666 669 721 755 837 847 903 969	PASS	5176	7186	819	13181
26	50	-302 -52 -670 445 -836 -824 898 919 136 975 847 596 252 -875 265 57 -999 -186 809 146 290 - 367 -701 -245 507 453 162 842 638 897 130 229 -373 288 996 603 763 -573 824 679 691 -394 900 -880 -950 797 -719 512 924 -899	-999 -950 -899 -880 -875 -836 -824 -719 - 701 -670 -573 -394 -373 -367 -302 -245 - 186 -52 57 130 136 146 162 229 252 265 288 290 445 453 507 512 596 603 638 679 691 763 797 809 824 842 847 897 898 900 919 924 975 996	-999 -950 -899 -880 -875 -836 -824 -719 - 701 -670 -573 -394 -373 -367 -302 -245 - 186 -52 57 130 136 146 162 229 252 265 288 290 445 453 507 512 596 603 638 679 691 763 797 809 824 842 847 897 898 900 919 924 975 996	PASS	5871	8057	942	14870

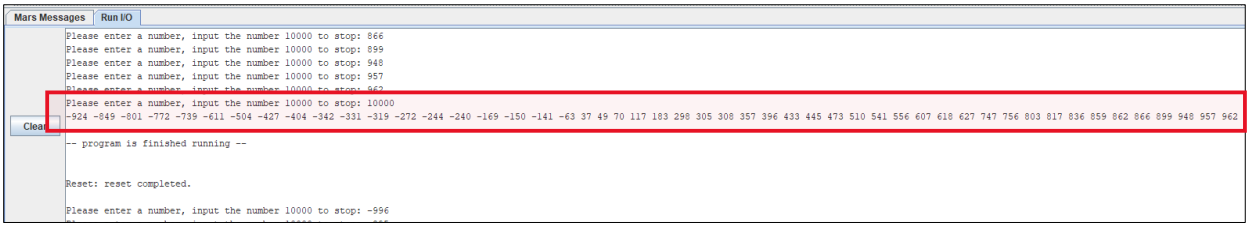
27	50	460 745 -480 -150 353 -728 142 604 -579 687 957 -532 -171 -26 -867 105 824 976 693 388 70 -697 -417 734 85 624 227 -394 -622 -391 551 294 -282 -749 -832 -339 351 75 620 -467 -702 -600 -875 282 -589 -888 394 -409 643 115	-888 -875 -867 -832 -749 -728 -702 -697 -622 -600 -589 -579 -532 -480 -467 -417 -409 -394 -391 -339 -282 -171 -150 -26 70 75 85 105 115 142 227 282 294 351 353 388 394 460 551 604 620 624 643 687 693 734 745 824 957 976	-888 -875 -867 -832 -749 -728 -702 -697 -622 -600 -589 -579 -532 -480 -467 -417 -409 -394 -391 -339 -282 -171 -150 -26 70 75 85 105 115 142 227 282 294 351 353 388 394 460 551 604 620 624 643 687 693 734 745 824 957 976	PASS	5174	7133	805	13112
28	50	-666 105 322 611 989 732 737 371 -695 -250 376 961 923 181 280 -60 -946 294 241 -424 -483 937 501 -586 199 324 -786 662 471 -517 -566 -683 274 -565 527 126 838 -88 -598 -900 573 -18 -779 134 -279 -153 -992 -617 -7 -640	-992 -946 -900 -786 -779 -695 -683 -666 -640 -617 -598 -586 -566 -565 -517 -483 -424 -279 -250 -153 -88 -60 -18 -7 105 126 134 181 199 241 274 280 294 322 324 371 376 471 501 527 573 611 662 732 737 838 923 937 961 989	-992 -946 -900 -786 -779 -695 -683 -666 -640 -617 -598 -586 -566 -565 -517 -483 -424 -279 -250 -153 -88 -60 -18 -7 105 126 134 181 199 241 274 280 294 322 324 371 376 471 501 527 573 611 662 732 737 838 923 937 961 989	PASS	5067	7099	811	12977
29	50	802 -305 932 -125 -955 -868 -111 -198 -315 -123 -574 -435 -809 -471 251 -156 -593 -322 -132 989 -101 -765 -830 536 -272 797 -275 -654 -396 -323 486 -26 748 642 396 -367 -419 437 -288 793 -218 -25 -90 -812 327 -15 655 741 126 -408	-955 -868 -830 -812 -809 -765 -654 -593 -574 -471 -435 -419 -408 -396 -367 -323 -322 -315 -305 -288 -275 -272 -218 -198 -156 -132 -125 -123 -111 -101 -90 -26 -25 -15 126 251 327 396 437 486 536 642 655 741 748 793 797 802 932 989	-955 -868 -830 -812 -809 -765 -654 -593 -574 -471 -435 -419 -408 -396 -367 -323 -322 -315 -305 -288 -275 -272 -218 -198 -156 -132 -125 -123 -111 -101 -90 -26 -25 -15 126 251 327 396 437 486 536 642 655 741 748 793 797 802 932 989	PASS	6091	8170	934	15195
30	50	579 -821 588 -696 -992 126 -122 -899 -538 -564 995 -610 -312 -240 -393 -310 468 -955 -836 -981 113 -587 321 -695 -309 -249 -260 -537 -669 -730 485 -846 -441 27 681 311 -551 -797 -784 -875 -768 -854 -704 210 -753 626 -641 -295 132 -761	-992 -981 -955 -899 -875 -854 -846 -836 -821 -797 -784 -768 -761 -753 -730 -704 -696 -695 -669 -641 -610 -587 -564 -551 -538 -537 -441 -393 -312 -310 -309 -295 -260 -249 -240 -122 27 113 126 132 210 311 321 468 485 579 588 626 681 995	-992 -981 -955 -899 -875 -854 -846 -836 -821 -797 -784 -768 -761 -753 -730 -704 -696 -695 -669 -641 -610 -587 -564 -551 -538 -537 -441 -393 -312 -310 -309 -295 -260 -249 -240 -122 27 113 126 132 210 311 321 468 485 579 588 626 681 995	PASS	5166	7159	812	13137

- # Minh họa kết quả test case chạy trên MIPS:

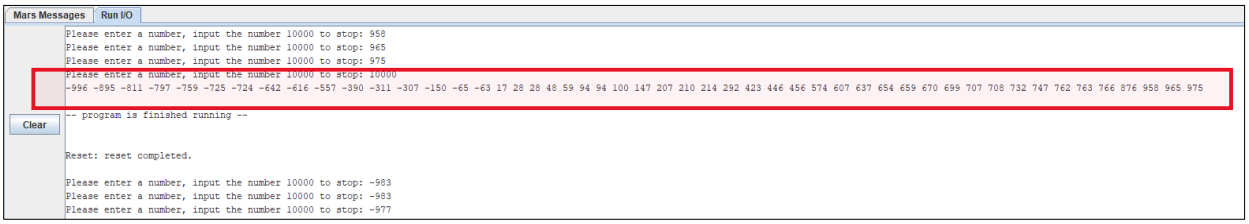
- **Test Case 1:**



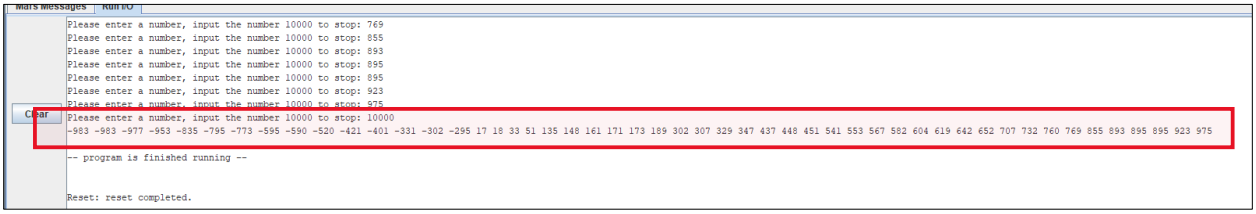
- **Test Case 2:**



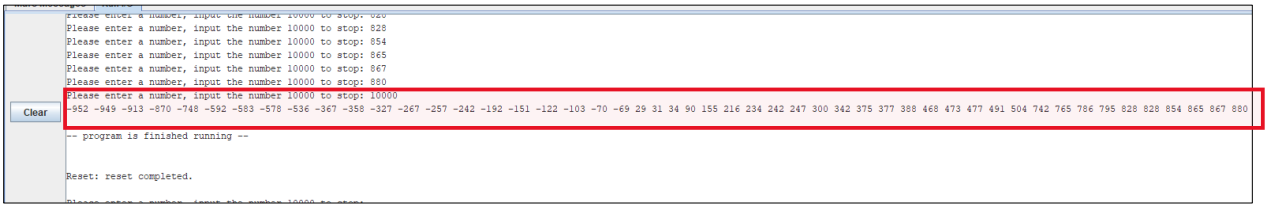
- **Test Case 3:**



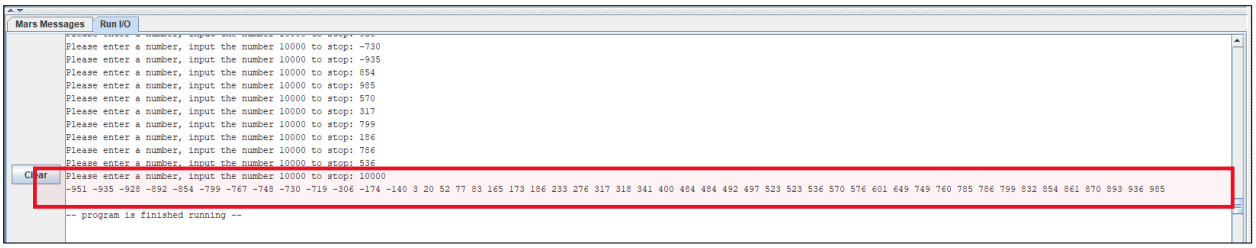
- **Test Case 4:**



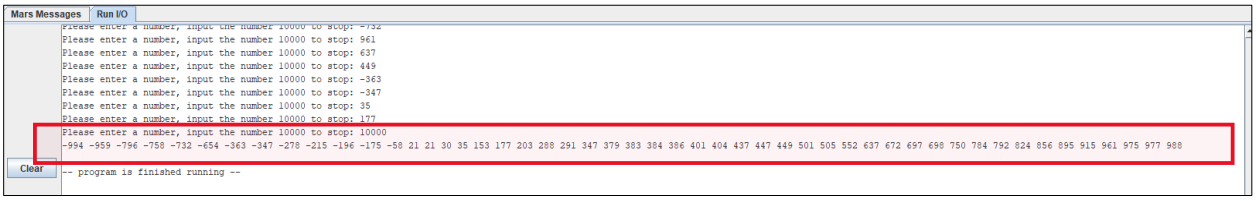
- **Test Case 5:**



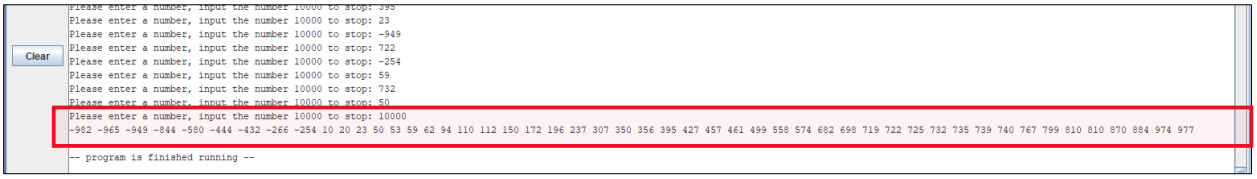
- **Test Case 6:**



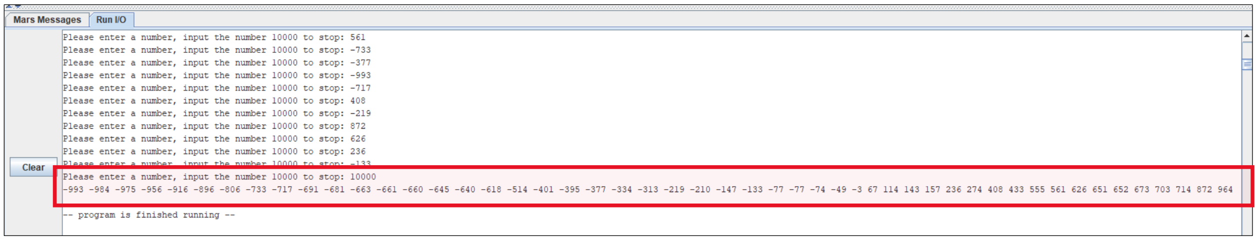
- **Test Case 7:**



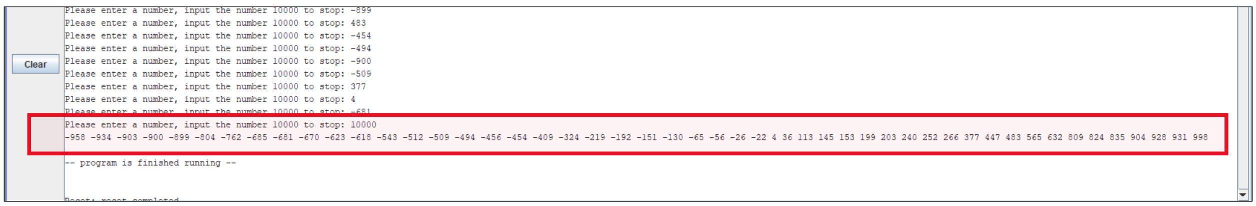
- **Test Case 8:**



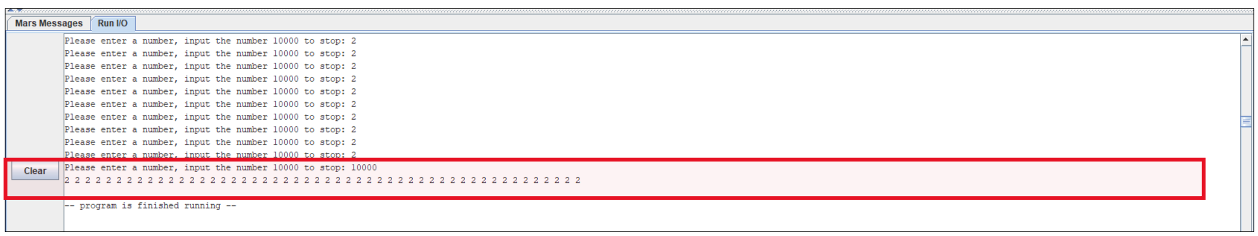
- **Test Case 9:**



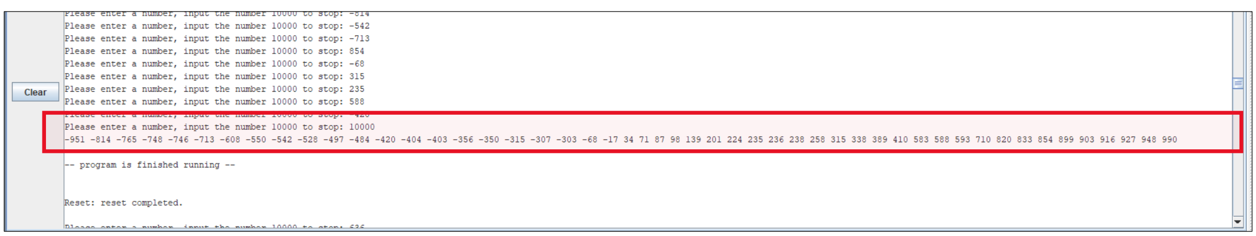
- **Test Case 10:**



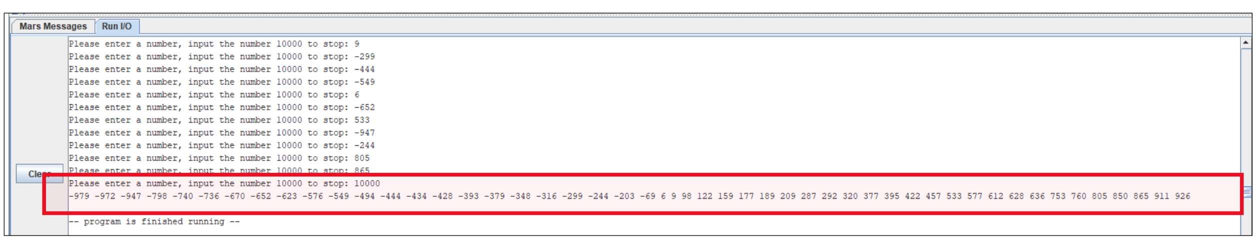
- **Test Case 11:**



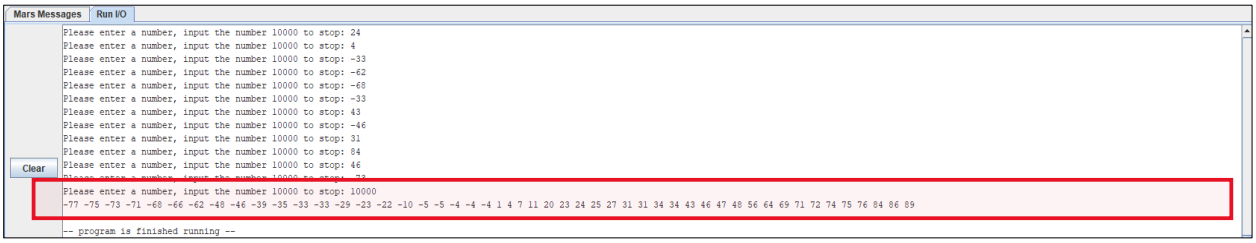
- **Test Case 12:**



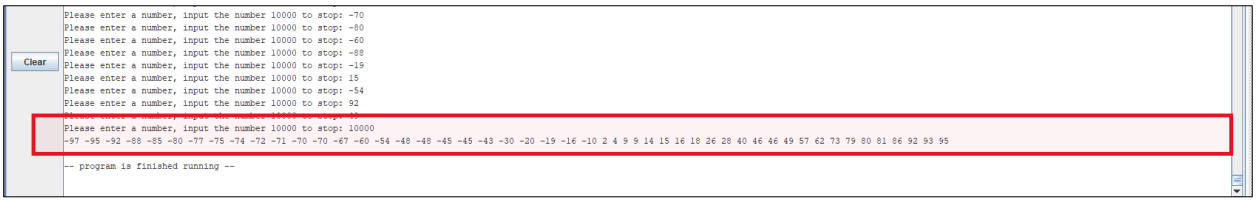
- **Test Case 13:**



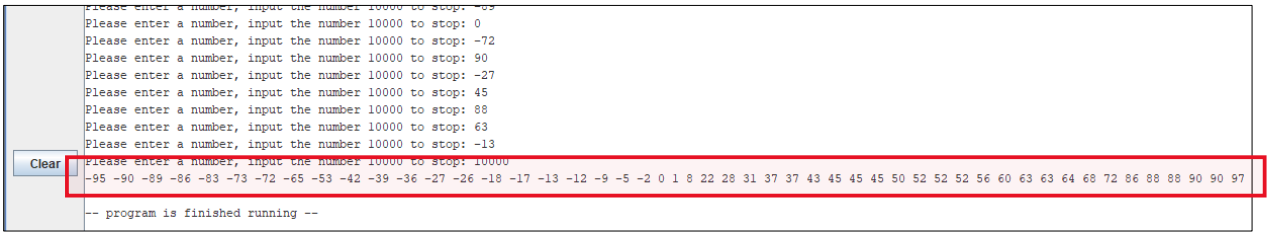
- **Test Case 14:**



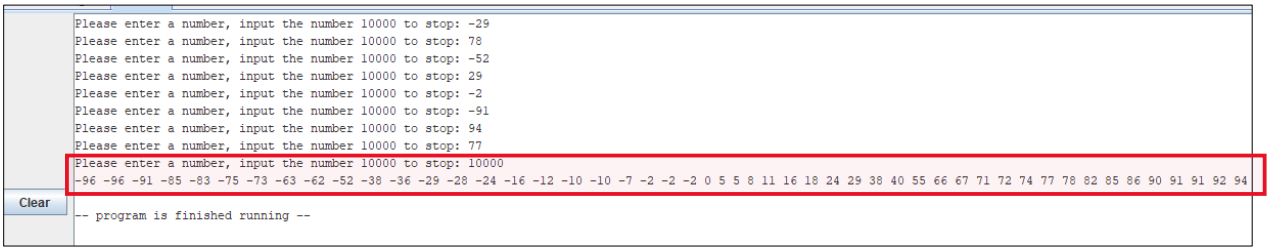
- **Test Case 15:**



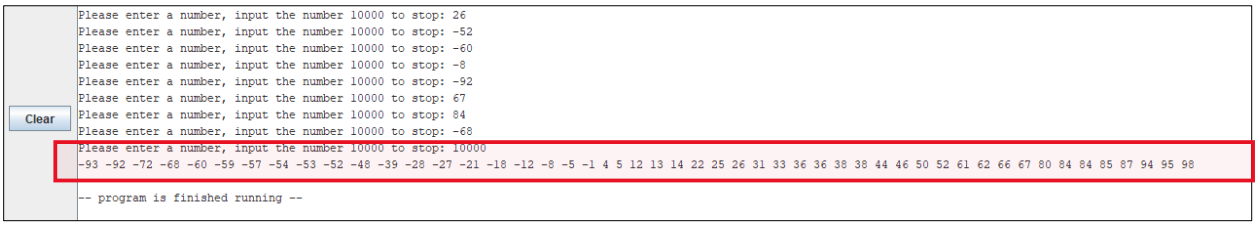
- **Test Case 16:**



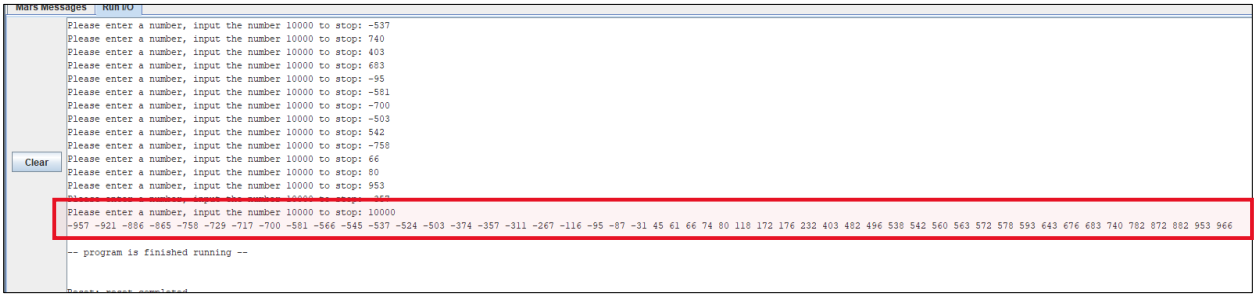
- **Test Case 17:**



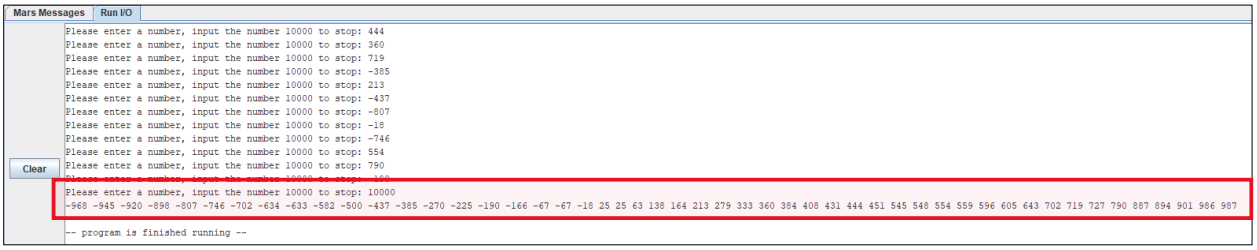
- **Test Case 18:**



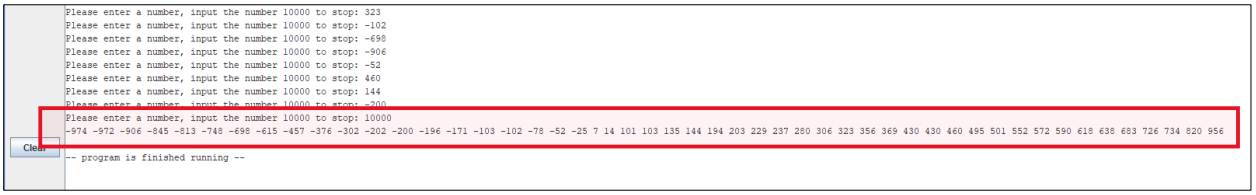
- **Test Case 19:**



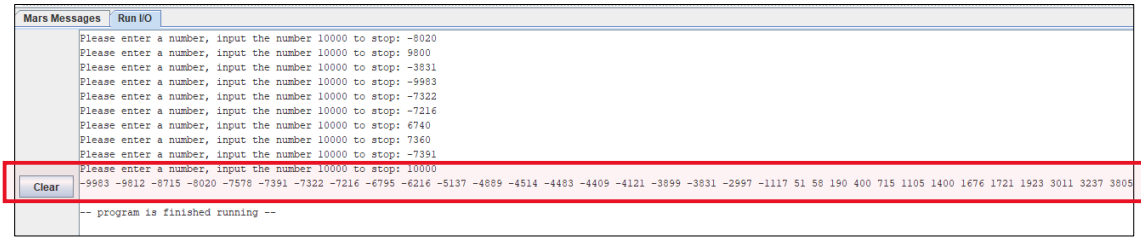
- **Test Case 20:**

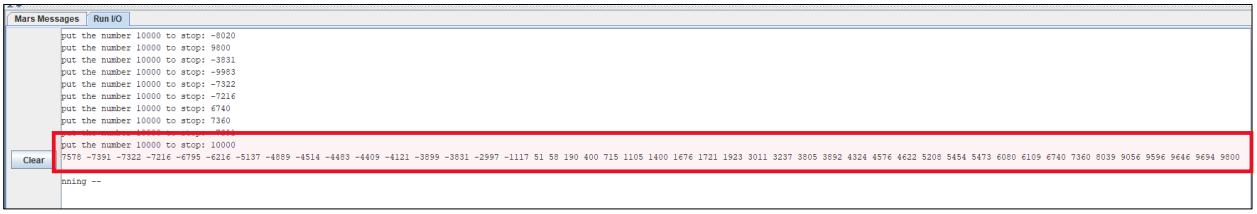


- **Test Case 21:**



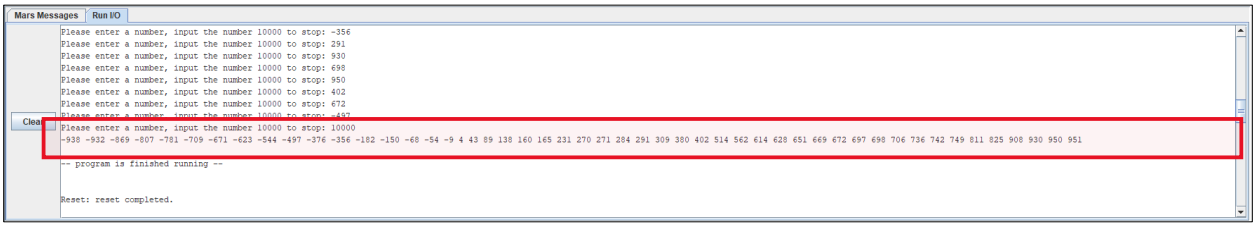
- **Test Case 22:**



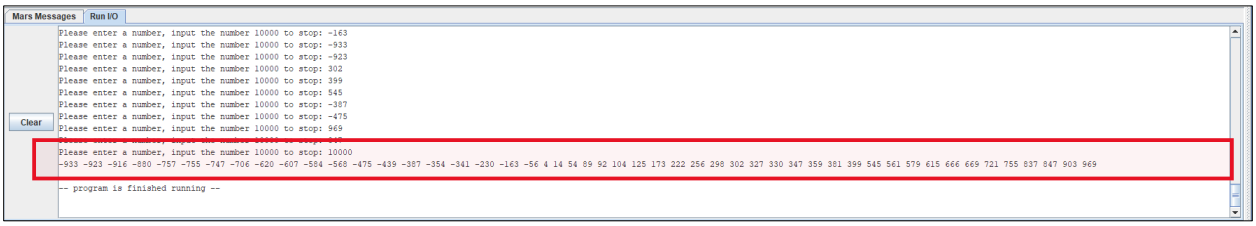


- **Test Case 23:**

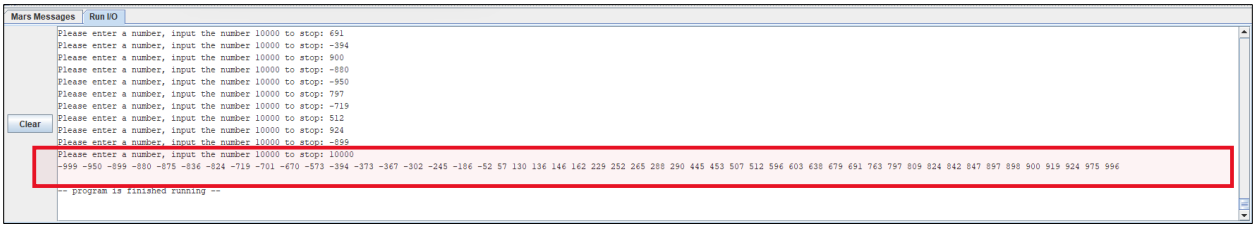
- **Test Case 24:**



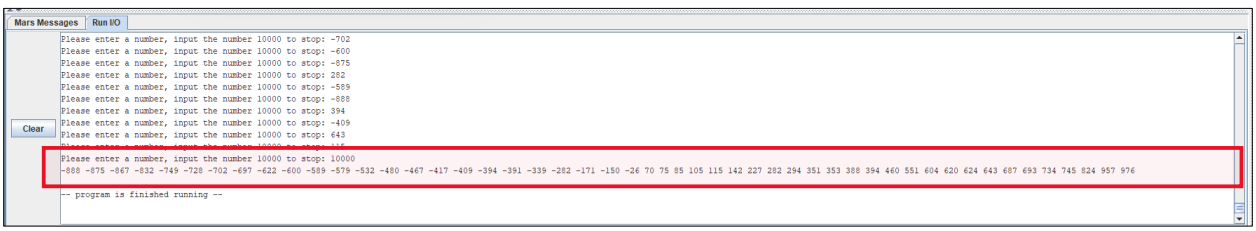
- **Test Case 25:**



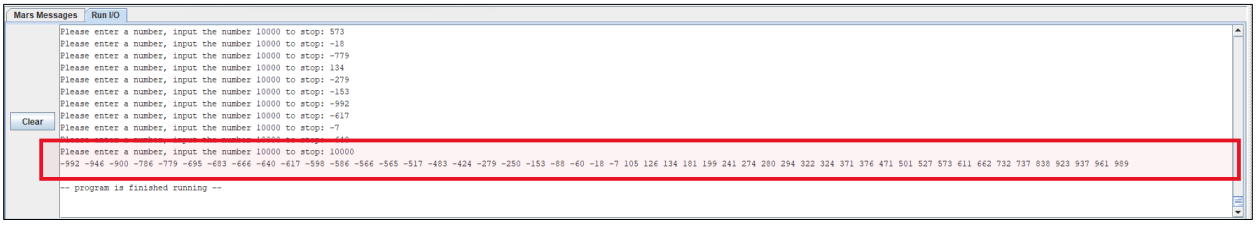
- **Test Case 26:**



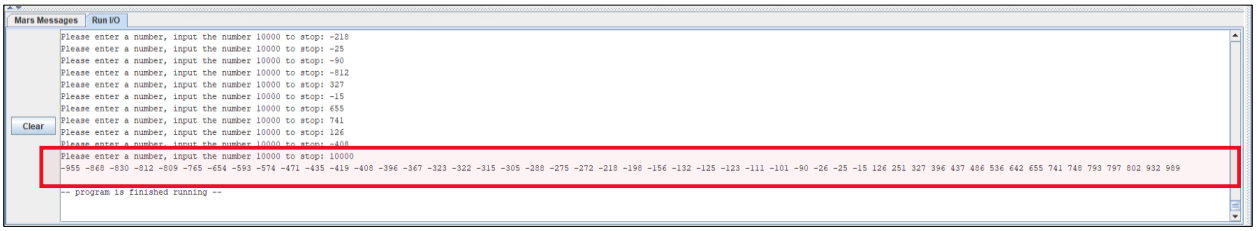
- **Test Case 27:**



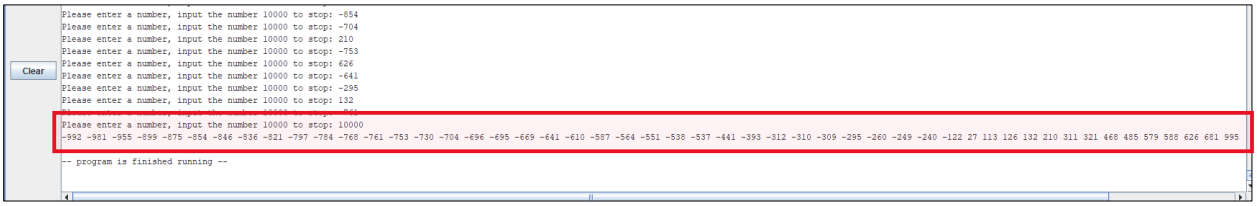
- **Test Case 28:**



- **Test Case 29:**



- **Test Case 30:**



4. Nguồn tham khảo:

1. https://www.geeksforgeeks.org/quick-sort/
2. https://www.geeksforgeeks.org/cpp-program-for-quicksort/
3. https://topdev.vn/blog/thuat-toan-quick-sort-la-gi/