BÁO CÁO THỰC HÀNH

Họ tên	Trương Thanh Minh	Lớp: KHTN2021
MSSV	21520064	STT: 6
Bài Thực Hành	LAB05	
CBHD	**Trương Văn Cương**	

1 Yêu cầu thực hành

Điểm buổi thực hành

Chuyên cần (20%)		
Trình bày (20%)		
Nội dung thực hành (60%)		
Câu 1: **Câu 2:** **Câu 3:**		
Tổng (100%)		

Bài tập thực hành:

Cho mảng a có n phần tử, với $1 \leq n \leq 100$. Phần tử thứ i mang giá trị nguyên dương a_i.

Yêu cầu: Với ngôn ngữ lập trình Assemply, hãy nhập mảng, kiểm tra tính hợp lệ về dữ liệu của mảng và sắp xếp lại mảng.

1. Sắp xếp tăng dần theo thuật toán Bubble Sort.
2. Sắp xếp tăng dần theo thuật toán Selection Sort.
3. Tìm phần tử trung vị.

2 Cơ sở lý thuyết

1. Thuật toán Bubble Sort

Bubble sort là thuật toán đơn giản nhất, hoạt động theo cách hoán đổi nhiều lần các phần tử liền kề nếu chúng không đúng thứ tự. Thuật toán này không phù hợp

với các tập dữ liệu lớn vì độ phức tạp thời gian trung bình và trường hợp xấu nhất của nó khá cao.

- **Ý tưởng**

  Ý tưởng của thuật toán này là đẩy phần tử lớn nhất xuống cuối dãy, đồng thời các phần tử có giá trị nhỏ hơn sẽ dịch chuyển về đầu dãy. Tựa như sự nổi bọt vậy, những phần tử nhẹ hơn sẽ nổi lên trên và ngược lại.

- **Mã giả**

```
void bubbleSort(int a[], int n){
    int i, j;
    for (i = (n-1); i >= 0; i--) {
        for (j = 1; j <= i; j++){
            if (a[j-1] > a[j])
            swap(a[j-1],a[j]);
        }
    }
}
```

2. **Thuật toán Selection Sort**

- **Ý tưởng**

  Ý tưởng của thuật toán này là tìm từng phần tử cho mỗi vị trí của mảng.

- **Mã giả**

```
void selectionSort(int a[], int n){
    int i, j, min, temp;
    for (i = 0; i < n-1; i++) {
        min = i;
        for (j = i+1; j < n; j++){
            if (a[j] < a[min]) min = j;
        }
        swap(a[i], a[min]);
    }
}
```

3 Thực hành

1. **Sắp xếp mảng**

Vì chương trình có hai thuật toán sắp xếp, do đó, ban đầu, ta in ra câu lệnh để người dùng lựa chọn.

```
li $v0, 5
syscall
move $t7, $v0 # t7 là lựa chọn

addi $t8, $zero, 1 # t8 = 1
beq $t7, $t8, BubbleSort # bubbleSort
addi $t8, $t8, 1 # t8 = 2
beq $t7, $t8, SelectionSort # selection sort
j Gohome


Gohome:
```

Figure 1: Đoạn code để người dùng lựa chọn thuật toán thực thi

a. Sắp xếp tăng dần theo thuật toán Bubble Sort

Trước khi thực hiện việc sắp xếp, ta tiến hành các khởi tạo các thanh ghi cần thiết.

```
# thuat toan sap xep
add $t1, $zero, $zero # bien i là biến để kiểm tra xem nó đã xét đủ các phần tử trong mảng chưa

la $s0, arr # s0 la base cua mang khi xet thang i
```

Figure 2: Khởi tạo các thanh ghi cần thiết

- Trong phương pháp này, ta cần hai vòng for để thực hiện, vòng for thứ nhất ($loop\_for\_sort1$) ta tiến hành for biến i để tìm đến vị trí cần xét tiếp theo. Sau đó, ta lưu lại giá trị biến a[i] vào thanh ghi t3.

```
loop_for_sort1:
        beq $t1, $t0, ExitSort

        lw $t3, ($s0) # t3 = a[i]
```

Figure 3: Nội dung vòng for 1

- Tiếp theo, ta tiến hành dòng for thứ 2 ($loop\_for\_sort2$), nhưng trước khi vào vòng for này, ta khởi tạo lại biến $j = i + 1$, đồng thời khởi tạo lại giá trị base cho vòng for này.

```
la $s1, arr # s1 la base cua mang khi xet thang j
la $s3, arr
addi $s1, $s0, 4
addi $t2, $t1, 1 # t2 là giá trị j để kiểm soát nó có vượt quá giới hạn mảng hay không
```

Figure 4: Tiền chuẩn bị cho vòng for tiếp theo

- Sau khi khởi tạo xong tất cả cần thiết, ta đi vào vòng for thứ 2. Trong vòng for này ta thực hiện thao tác so sánh $t3$ (a[i]) và $t4$ (a[j]). Nếu $a[i] < a[j]$, thì ta thực hiện thao tác đổi chỗ hai giá trị đó với nhau. Sau đó, ta thực hiện cộng $t2$ lên 1 và giá trị base lên 4 để phục vụ cho việc xét tiếp giá trị của vị trí tiếp theo.

```
loop_for_sort2:
        beq $t2, $t0, j_to_add_t1
        lw $t4, ($s1) # t4 = a[j]

        slt $t5, $t3, $t4 # t5 = (t3 < t4) ? 1 : 0
        beq $t5, $zero, DoiCho

        addi $s1, $s1, 4 # tăng giá trị base cho vòng for j lên 4
        addi $t2, $t2, 1 # tăng giá trị j lên 1
        j loop_for_sort2
```

Figure 5: Nội dung vòng for thứ 2

- Theo như vòng for thứ 2, nếu thỏa $t3 \geq t4$, ta gọi đến hàm đổi chỗ để thực hiện giống như tư tưởng thuật toán sắp xếp bubble sort quen thuộc

```
DoiCho:
        # thay đổi giá trị tại mảng arr
        sw, $t3, ($s1)
        sw, $t4, ($s0)

        # swap giá trị t3 và t4 cho nhau
        move $t5, $t3
        move $t3, $t4
        move $t4, $t5

        addi $s1, $s1, 4
        addi $t2, $t2, 1
        j loop_for_sort2
```

Figure 6: Đoạn code thể hiện việc đổi chỗ cho 2 phần tử

- Nếu như giá trị j đã duyệt qua hết mảng, ta gọi đến hàm $j\_to\_add\_t1$ để thoát vòng for thứ 2 và gọi lại vòng for thứ nhất.

```
j_to_add_t1:
        # phục vụ cho việc xét tiếp vị trí tiếp theo cho i
        addi $t1, $t1, 1 # tăng i lên 1
        addi $s0, $s0, 4
        j loop_for_sort1
```

Figure 7: Đoạn code thể hiện thao tác quay lại vòng for thứ nhất

- Cuối cùng, nếu như i đã xét qua hết tất cả các phần tử của mảng, ta gọi đến hàm $ExitSort$. Ở hàm này, ta sẽ để trống và không thực hiện thêm câu lệnh nào hết.

b. Sắp xếp tăng dần theo thuật toán Selection Sort

Trước khi thực hiện sắp xếp, ta tiến hành khởi tạo các thanh ghi cần thiết.

```
la $s0, arr # base cho vị trí i
la $s2, arr # base tạm được dùng để khi đổi vị trí
add $t1, $zero, $zero # vị trí i
```

Figure 8: Khởi tạo giá trị thanh ghi cần thiết

- Sau đó, ta lần lượt tiến hành cố định một vị trí để tìm ra phần tử phù hợp với vị trí đó. Chương trình sẽ dừng khi xét đủ các phần tử trong mảng thông qua hàm $ExitSelectionSort$.

```
beq $t1, $t0, ExitSelectionSort

lw $t3, ($s0) # t3 = a[i]
```

Figure 9: Khởi tạo thanh ghi

- Để bước vào vòng for tiếp theo, ta phải khởi tạo các giá trị cần thiết cho nó, ta sẽ gán $j = i + 1$, bởi vì các phần tử trước đó đã được cố định vị trí nên ta không cần xét nữa.

```
la $s1, arr # s1 la base cua mang khi xet thang j
addi $s1, $s0, 4
addi $t2, $t1, 1 # t2 là giá trị j để kiểm soát nó có vượt quá giới hạn mảng hay không
```

Figure 10: Khởi tạo các thanh ghi cần thiết

- Sau đó, ta tiếp tục duyệt thêm 1 vòng for nữa để xét lần lượt các phần tử ở vị trí sau i để tìm phần tử phù hợp cho vị trí i.

```
loop_for_ss2:
        beq $t2, $t0, DoiChoSS
        lw $t4, ($s1) # t4 = a[j]
        lw $t7, ($s2) # t7 = a[min]

        slt $t5, $t7, $t4 # a[min] < a[j] -> khong thay doi
        beq $t5, $zero, UpdateValueMin

        addi $s1, $s1, 4 # tăng giá trị base cho vòng for j lên 4
        addi $t2, $t2, 1 # tăng giá trị j lên 1
        j loop_for_sort2

UpdateValueMin: # t7 = min, t4 = j
        add $s2, $s2, $s1 # cap nhat lai min = j
        addi $s1, $s1, 4 # tăng giá trị base cho vòng for j lên 4
        addi $t2, $t2, 1 # tăng giá trị j lên 1
        j loop_for_sort2

DoiChoSS:
        # thay đổi giá trị tại mảng arr
        sw $t3, ($s2)
        sw $t7, ($s0)

        # swap giá trị t3 và t4 cho nhau
        move $t5, $t3
        move $t3, $t7
        move $t7, $t5

        addi $s1, $s1, 4
        addi $t2, $t2, 1

        # phục vụ cho việc xét tiếp vị trí tiếp theo cho i
        addi $t1, $t1, 1 # tăng i lên 1
        addi $s0, $s0, 4
        addi $s2, $s2, 4
        j loop_for_sort1
```

Figure 11: Cập nhật phần tử cho từng vị trí i

- **Kết quả thực thi chương trình trên**

```
Nhap so phan tu mang (be hon 100): 5
Nhap phan tu thu 0: 11
Nhap phan tu thu 1: 4
Nhap phan tu thu 2: 6
Nhap phan tu thu 3: 5
Nhap phan tu thu 4: 22
 Mang ban vua nhap la:
11 4 6 5 22
Tong cua mang la: 48
So so chan cua mang: 3
So so le cua mang: 2
So lon nhat trong mang: 22
So be nhat trong mang: 4
Chon thuat toan ban muon sap xep (vui long chi chon 1 trong 2 lua chon o duoi)
1. Bubble Sort
2. Selection Sort
2
5 4 6 11 22
Gia tri trung vi cua mang la: 6
```

Figure 12: Kết quả thực thi đoạn code

2. Tìm phần tử trung vị

- Ý tưởng: lấy phần tử thứ $\frac{n+1}{2}$ trong mảng đã sắp xếp và in nó ra.
- Để lấy phần tử thứ $\frac{n+1}{2}$ trong mảng, đầu tiên ta cộng thanh ghi $t0$ (thanh ghi lưu lại giá trị phần tử của mảng), sau đó dùng phép srl để chia giá trị đó cho 2.

```
addi $t0, $t0, 1
srl $t0, $t0, 1 # phan tu trung vi
```

Figure 13: Đoạn code tìm ra vị trí của phần tử trung vị

- Tiếp theo, để tìm giá trị phần tử trung vị, giá duyệt từng phần tử mảng, khi nào đến vị trí đó thì dừng và gọi đến hàm xuất giá trị trung vị.

```
loopFindMid:
        beq $t1, $t0, xuatTrungVi # i == t0 -> den vi tri trung vi
        lw $t2, ($s0) # t2 = a[i]

        # xét tiếp vị trí tiếp theo
        addi $t1, $t1, 1 # tăng i lên 1 đơn vị
        addi $s0, $s0, 4
        j loopFindMid
xuatTrungVi:
        li $v0, 1
        la $a0, ($t2)
        syscall
```

Figure 14: Đoạn code mô tả cách tìm phần tử trung vị

- **Kết quả demo đoạn code trên**

```
Nhap so phan tu mang (be hon 100): 5
Nhap phan tu thu 0: 1
Nhap phan tu thu 1: 6
Nhap phan tu thu 2: 4
Nhap phan tu thu 3: 9
Nhap phan tu thu 4: 5
 Mang ban vua nhap la:
1 6 4 9 5
Tong cua mang la: 25
So so chan cua mang: 2
So so le cua mang: 3
So lon nhat trong mang: 9
So be nhat trong mang: 1
Chon thuat toan ban muon sap xep (vui long chi chon 1 trong 2 lua chon o duoi)
1. Bubble Sort
2. Selection Sort
1
Mang sau khi sap xep tang dan la: Gia tri trung vi cua mang la: 1 4 5 6 9
Gia tri trung vi cua mang la: 5
```

Figure 15: Kết quả thực hiện chương trình

!Lưu ý: Toàn bộ code xem chi tiết ở: [trthminh/T-ch-c-c-u-tr-c-m-y-t-nh-2 (github.com)]