**NHẬP MÔN CÔNG NGHỆ THÔNG TIN 2**

BIÊN BẢN HỌP NHÓM

Tài liệu này ghi nhận lại nội dung và kết quả của một buổi họp nhóm.

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

**Biên bản họp nhóm**

# Thông tin chung

Tên nhóm: Mã nhóm: (Nếu có)

Các thành viên có tham dự:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** | **Email** | **SĐT** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

Các thành viên vắng mặt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ và tên** | **Email** | **SĐT** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |

Mục tiêu cuộc họp nhằm:

1. …
2. …

Địa điểm:

Thời gian bắt đầu: Thời gian kết thúc:

# Nội dung

**B. SORT**

**Selection Sort**

* **Ý tưởng chung:** Giả sử sắp xếp mảng tăng dần. Mảng được chia thành 2 mảng con là mảng đã và chưa được sắp xếp. Selection Sort tìm phần tử nhỏ nhất trong mảng chưa được sắp xếp và đổi chỗ với phần tử ở đầu mảng chưa sắp xếp. Tức là mỗi vòng lặp, phần tử nhỏ nhất của mảng chưa sắp xếp sẽ được chuyển đến mảng đã sắp xếp.
* **Mã giả:**

Function swap(a,b):

input temp = a

a = b

b = temp

Begin

Input a[], n, i, j

For i = 0 to n-1 do

Input min = i

For j = i+1 to n do

If (a[j] < a[min])

min = j

Endif

Endfor

if ( min != i )

swap(a[min], a[j])

Endif

Endfor

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp | Mảng lộn xộn | Mảng bị đảo ngược |
| Độ phức tạp | O(n2) | O(n2) | O(n2) |
| Ví dụ |  |  | A = { 8, 5, 3, 2, 1} cần sắp xếp tăng dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1**: i = 1. Lặp lần thứ 1

Tìm phần tử nhỏ nhất trong a[0….4] : min = a[2] = 1. Đổi chỗ với phần tử đầu tiên

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 5 | 2 | 17 | 24 |

**B2**: i = 2. Lặp lần thứ 2

Tìm phần tử nhỏ nhất trong a[1…4] : min = a[2] = 2. Đổi chỗ với phần tử đầu tiên của a[1…4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B3**: i = 3. Lặp lần thứ 3

Tìm phần tử nhỏ nhất trong a[2..4] : min = a[2] = 5. Giữ nguyên vị trí

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B4**: i = 4. Lặp lần thứ 4

Tìm phần tử nhỏ nhất trong a[3…4]: min = a[4] = 17. Đổi chỗ với phần tử đầu tiên của a[3…4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

Mảng sau khi sắp xếp :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Insertion Sort**

* **Ý tưởng chung**: Giả sử sắp xếp mảng tăng dần. Bắt đầu duyệt từ phần tử thứ 2 trở đi, so sánh phần tử hiện tại với phần tử trước nó. Nếu phần tử hiện tại lớn hơn thì đổi chỗ 2 phần tử với nhau, sau đó tiếp tục so sánh và đổi chỗ như vậy ở các vòng lặp tiếp theo sao cho đảm bảo tính tăng dần của mảng cho đến khi hoàn thành sắp xếp.
* **Mã giả:**

Begin

Input a[], n,i , j, current

For i = 1 to n do

current = a[j]

input j = i – 1

while (j >=0 and a[j] > current) do

a[j+1] = a[j]

j = j -1

endwhile

a[j +1] = current

Endfor

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp, chỉ có vòng lặp bên ngoài chạy n lần | Mảng có thứ tự lộn xộn | Mảng có thứ tự bị đảo ngược |
| Độ phức tạp | O(n) | O(n2) | O(n2) |
| Ví dụ |  |  | A = { 6, 5, 4, 3, 2} cần sắp xếp tăng dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1**: i = 1. Lặp lần thứ 1

Vì 5 lớn hơn 2 nên giữ nguyên vị trí của 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 17 | 24 |

**B2**: i = 2. Lặp lần thứ 2

Vì 1 nhỏ hơn cả 5 và 2 nên 1 sẽ di chuyển về đầu dãy, các phần tử 2 và 5 sẽ di chuyển sang phải 1 vị trí so với hiện tại.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B3**: i = 3. Lặp lần thứ 3

Vì 24 lớn hơn tất cả phần tử từ a[0…2] nên 24 vẫn giữ nguyên vị trí

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B4**: i = 4. Lặp lần thứ 4

Vì 17 lớn hơn 1,2,5 nhưng nhỏ hơn 24 nên 17 sẽ di chuyển về vị trí sau 5, phần tử 24 sẽ di chuyển về sau 1 vị trí

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

Mảng sau khi sắp xếp :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Binary Insertion Sort**

* **Ý tưởng chung:** Giả sử sắp xếp mảng tăng dần. Mảng được chia thành 2 mảng con đã và chưa được sắp xếp. Duyệt từ phần tử thứ 2 đến phần tử cuối cùng, phần tử hiện tại là key. Tiếp đó dùng Binary Search vào mảng đã được sắp xếp để tìm vị trí “pos” có giá trị lớn hơn key. Dịch chuyển toàn bộ phần tử từ “pos” về 1 sang phải để tạo vị trí trống cho key.
* **Mã giả:**

Function binarySearch(a, left,right,key)

If ( right >= left)

Input mid =(left + right) / 2

If (a[mid] == key)

mid

Endif

If (a[mid] > key)

binarySearch(a, left, mid-1, key)

Endif

binarySearch(a, mid+1, right, key)

Endif

Begin

Input a[], n

Input vt, i, j, key

For i =1 to n do

j = i – 1

key = a[i]

vt = binarySearch(a, 0, j, key)

while ( j >= vt) do

a[j +1] = a[j]

j = j – 1

endwhile

a[j+1] = key

Endfor

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp, chỉ có vòng lặp bên ngoài chạy n lần | Mảng có thứ tự lộn xộn | Mảng có thứ tự bị đảo ngược |
| Độ phức tạp | O(n) | O(n2) | O(n2) |
| Ví dụ |  |  | A = { 1, 4, 10, 45, 48} cần sắp xếp giảm dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1:** Giả sử phần tử đầu tiên đã được sắp xếp. Key = a[1] = 5

Dùng binary search : 2 < 5 nên giữ nguyên vị trí

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B2**: Key = a[2] = 1

Vì 2 > 1 nên dịch chuyển 2 và 5 sang phải một vị trí.

a[0] = 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B3**: Key = a[3] = 24

Dùng binary search : 24 > 5 nên giữ nguyên vị trí

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 24 | 17 |

**B4**: Key = a[4] = 17

Dùng binary search tìm được phần tử lớn hơn 17 trong a[0…3] là 24

Dịch chuyển 24 sang phải một vị trí, lúc này a[3] = 17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

Mảng kết quả:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Bubble Sort**

* **Ý tưởng chung:** Giả sử sắp xếp mảng tăng dần. Ở mỗi vòng lặp, so sánh 2 phần tử liền kề và đổi chỗ nếu chúng xếp sai thứ tự tăng dần. Lặp lại cho đến khi hoàn thành sắp xếp mảng.
* **Mã giả:**

Begin

Input a[], n

Input i, j

For i = 0 to n-1 do

For j = 0 to n – i – 1 do

If (a[j] > a[j+1])

swap(a[j], a[j+1])

Endif

Endfor

Endfor

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp | Mảng có thứ tự lộn xộn | Mảng có thứ tự bị đảo ngược |
| Độ phức tạp | O(n) | O(n2) | O(n2) |
| Ví dụ |  |  | A = { 5, 10, 15, 20, 25 } cần sắp xếp giảm dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1**: i = 1. Lặp lần thứ 1

So sánh a[0] và a[1]. Giữ nguyên vị trí hai phần tử do 5 > 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

So sánh a[1] và a[2]. Đổi chỗ 2 phần tử do 5 > 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 5 | 24 | 17 |

So sánh a[2] và a[3]. Giữ nguyên vị trí 2 phần tử do 5 < 24

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 5 | 24 | 17 |

So sánh a[3] và a[4]. Đổi chỗ hai phần tử do 24 > 17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 1 | 5 | 17 | 24 |

**B2** : i = 2. Lặp lần thứ 2

So sánh a[0] và a[1]. Đổi chỗ 2 phần tử do 2 > 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

So sánh a[1] và a[2]. Giữ nguyên vị trí do 2 < 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

So sánh a[2] và a[3]. Giữ nguyên vị trí do 5 < 17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

So sánh a[3] và a[4]. Giữ nguyên vị trí do 17 < 24

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

Mảng sau khi sắp xếp :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Radix Sort**

* **Ý tưởng chung:** : Giả sử sắp xếp mảng tăng dần. Radix Sort so sánh từng chữ số với nhau từ chữ số bé nhất (hàng đơn vị) đến chữ số lớn nhất.
* **Mã giả:**

Function findMax(a[], n)

Input max = a[0]

Input i

For i = 1 to n do

If (a[i] > max)

max = a[i]

Endif

Endfor

max

Function countingSort( a[], n, digit)

Input A[10]

Input count[10]

Input i

For i = 0 to max do

count[i] = 0

Endfor

For i = 0 to n do

count[(a[i] / digit) % 10 ] = count[(a[i] / digit) % 10 ] +1

Endfor

For i = 1 to max do

count[i] = count[i] + count[i-1]

Endfor

For i = n – 1 to 0 do

A[count[(a[i] / digit) % 10] – 1 ] = a[i]

count[(a[i] / digit) % 10] = count[(a[i] / digit) % 10 ] – 1

Endfor

For i = 0 to n do

a[i] = A[i]

Endfor

Begin

Input a[], n

Input digit

Input max = findMax(a,n)

For digit = 1 and max/digit > 0 and digit = digit \*10

countingSort(a, n, digit)

Endfor

End

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Cơ số b nhỏ nhất và bằng 0 | K là phần tứ lớn nhất có thể có, b là cơ số | Giá trị b lớn và b = n |
| Độ phức tạp | O(nlogbn) | O((n+b) \* logb(k)) | O(n) |
| Ví dụ |  |  | A = { 10, 11, 12, …, 19}  B = n = 10 |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1**: Tìm phần tử lớn nhất trong mảng : a[3] = 24 có 2 chữ số. VÌ vậy sẽ cần dùng 2 vòng lặp để sắp xếp.

**B2:** i = 1. Lặp lần thứ 1

Dùng counting sort để sắp xếp các chữ số hàng đơn vị của từng phần tử

Mảng a1 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 4 | 7 |

Đếm số lần xuất hiện của các chữ số mảng a1

Mảng count

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |

Phần tử sau bằng phần tử liền trước cộng chính nó

Mảng count

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 2 | 3 | 4 | 4 | 5 |

Mảng kết quả A :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 5 | 7 |

Mảng a

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 24 | 5 | 17 |

**B3** : i = 2. Lặp lần thứ 2

Dùng counting sort để sắp xếp các chữ số hàng chục

Mảng a2 :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 2 | 0 | 1 |

Đếm số lần xuất hiện các chữ số mảng a2 :

Mảng count:

|  |  |  |
| --- | --- | --- |
| 3 | 1 | 1 |

Phần tử liền sau bằng phần tử liền trước cộng chính nó, đồng thời giảm count

Mảng count:

|  |  |  |
| --- | --- | --- |
| 3 | 4 | 5 |

Mảng kết quả A :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

Mảng sau khi sắp xếp :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Flash Sort**

* **Ý tưởng chung:** Giả sử sắp xếp mảng tăng dần. Gồm có 3 bước. Bước 1 là Phân lớp dữ liệu dựa trên giả thiết, bước 2 là Hoán vị toàn cục ( đưa các phần tử về lớp của chúng), bước 3 là Sắp xếp bố cục ( sắp xếp trong phạm vi từng lớp).
* **Mã giả:**

Begin

Input a[], n

Input min = a[0], max = 0

Input m = 0.45\*n

Input A[m]

For i = 0 to m do

A[i] = 0

Endfor

For i = 1 to n do

If (a[i] > min)

min = a[i]

Endif

If (a[i] > a[max])

max = i

Endif

Endfor

If ( a[max] == min)

Endif

Input c = (m-1)/(a[max] – min)

For i = 0 to n do

Input k = c\*(a[i] – min)

A[k] = A[k] + 1

Endfor

For i = 1 to m do

A[i] = A[i] + A[i-1]

Endfor

swap(a[max], a[0])

Input nmove = 0

Input j = 0, t = m – 1, k = 0

Input flash

While (nmove < n – 1) do

While ( j > A[k] – 1) do

j = j + 1

k = c\*(a[j] – a[min])

Endwhile

flash = a[j]

If ( k < 0 ) Endif

While ( j != A[k]) do

k = c\*(flash – min)

Input hold = a[t = (A[k] = A[k] – 1)]

a[t] = flash

flash = hold

nmove = nmove + 1

Endwhile

Endwhile

insertionSort(a,n)

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp | Mảng có thứ tự lộn xộn | Mảng có thứ tự đảo ngược |
| Độ phức tạp | O(n) | O(n) | O(n) |
| Ví dụ |  |  | A = { 50, 23, 14, 6, 4} cần sắp xếp tăng dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

m = (int) 0.45 x 5 = 2

min = 1

max = 24

c = (m-1)/ (a[max] – min) =1/23

**B1**: Chia làm 2 lớp, lớp 0 gồm 3 phần tử [2, 5, 1], lớp 1 gồm [24, 17]

A = [3 2] 🡪 A = [3 5]

Đổi a[max] và a[0]:

Mảng a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 24 | 5 | 1 | 2 | 17 |

flash = 24

Mảng A[1]-- : A[3 5] 🡪 A[3 4]

**B2**: nmove = 1

Đổi chỗ a[0] và a[4]

Mảng a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17 | 5 | 1 | 2 | 24 |

flash = 17

Mảng A[0]--: A[3 4] 🡪 A[2 4]

**B3**: nmove = 2

Đổi chỗ a[0] và a[3]

Mảng a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 17 | 24 |

flash = 2

Mảng A[0]-- : A[2 4] 🡪 A[1 4]

**B4**: nmove = 3

Đổi chỗ a[0] và a[2]

Mảng a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 5 | 2 | 17 | 24 |

flash = 1

Mảng A[1]-- : A[1 4] 🡪 A[1 3]

**B5**: nmove =4

Đổi chỗ a[0] và a[1]

Mảng a:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 1 | 2 | 17 | 24 |

flash = 5

nmove = 4 = n – 1 🡪 Stop

**B6**: Phân lớp gồm lớp 0[ 5 1 2], lớp 1[ 17 24]

Dùng insertion sort để sắp xếp lại mảng này.

Mảng kết quả :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

**Counting Sort**

* **Ý tưởng chung:** Giả sử sắp xếp mảng tăng dần. Đầu tiên đếm số lần xuất hiện của các phần tử trong mảng A và lưu vào mảng C, sau đó thay đổi chỉ số giới hạn mảng C. Cuối cùng duyệt từng phần tử của A vào mảng B chứa kết quả sắp xếp thông qua mảng C.
* **Mã giả:**

Begin

Input a[], n

Input output[10]

Input max = a[0], min = a[0]

For i = 1 to n do

If (a[i] > max)

max = a[i]

Else If (a[i] < min]

min = a[i]

Endif

Endfor

Input k = max – min + 1

Input count[k]

Fill\_in(count, k, 0)

For i = 0 to k do

count[i] = 0

Endfor

For i = 0 to n do

count[a[i] - min] = count[a[i] – min] + 1

Endfor

For i = 1 to k do

count[i] = count[i] + count[i + 1]

Endfor

For i = 0 to n do

output[count[a[i] – min] – 1] = a[i]

count[a[i] - min] = count[a[i] – min] – 1

Endfor

For i = 0 to n do

a[i] = output[i]

Endfor

End.

* **Nhận xét:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Trường hợp tốt nhất | Trường hợp trung bình | Trường hợp xấu nhất |
| Trường hợp | Mảng đã được sắp xếp | Mảng có thứ tự ngẫu nhiên | Mảng bị đảo ngược thứ tự |
| Độ phức tạp | O(n) | O(n) | O(n) |
| Ví dụ |  |  | A = {1, 7, 13, 24, 30} cần sắp xếp giảm dần |

* **VD đơn giản với mảng 5 phần tử:**

a[0] a[1] a[2] a[3] a[4]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 5 | 1 | 24 | 17 |

**B1**: Đếm số lần xuất hiện của từng phần tử trong mảng a. Kết quả lưu vào mảng count

Max = 24

Min = 1

K = max – min + 1 = 24

Mảng count :

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chỉ số | 0 | 1 | 2 | 3 | 4 | 5 | … | 17 | …. | 23 | 24 |
| P.tử | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |

**B2**: Sửa đổi giá trị mảng count thể hiện giới hạn của chỉ số phần tử sau khi sắp xếp

Mảng count

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chỉ số | 0 | 1 | 2 | 3 | 4 | 5 | … | 17 | …. | 23 | 24 |
| P.tử | 0 | 1 | 2 | 2 | 2 | 3 | 3 | 4 | 4 | 4 | 5 |

**B3**: Duyệt từng phần tử của mảng a và đặt nó vào đúng vị trí trong mảng output dựa vào mảng count

Mảng output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Chỉ số | 0 | 1 | 2 | 3 | 4 |
| Phần tử | 1 | 2 | 5 | 17 | 24 |

Mảng sau khi sắp xếp :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 5 | 17 | 24 |

# Bảng phân công công việc

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Người phụ trách** | **Mô tả nội dung công việc** | **Bắt đầu** | **Kết thúc** | **Kết quả mong đợi** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |