**PSEUDO CODE \_ 6 SORT + Counting Sort**

**Selection Sort**

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.

**Insertion Sort**

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.

**Binary Insertion Sort**

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.

**Bubble Sort**

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.

**Radix Sort**

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

**Flash Sort**

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.

**Counting Sort**

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.

**EXAMPLES \_ 6 SORT + Counting Sort**

**a[5] = {2, 5, 1, 24, 17} – Giả sử sắp xếp mảng tăng dần**

**Selection Sort**

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

a = 1, 5, 2, 24, 17

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]

a = 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

a = 1, 2, 5, 24, 17

B4: i = 4. Lặp lần thứ 4

Tìm phần tử nhỏ nhất trong a[3…4]: min = a[4] = 6. Đổi chỗ với phần tử đầu tiên của a[3…4]

a = 1, 2, 5, 17, 24

Mảng sau khi sắp xếp : 1, 2, 5, 17, 24

**Insertion Sort**

Lặp từ phần tử thứ 2 (i=1) đến phần tử cuối cùng của mảng (i=4)

B1: i = 1. Lặp lần thứ 1

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

a = 2, 5, 1, 24, 17

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.

a = 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

a = 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

a = 1, 2, 5, 17, 24

Mảng sau khi sắp xếp : 1, 2, 5, 17, 24

**Binary Insertion Sort a[5] = {2, 5, 1, 24, 17}**

**Bubble Sort**

B1: i = 1. Lặp lần thứ 1

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

a = 2, 5, 1, 24, 17

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

a = 2, 1, 5, 24, 17

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

a = 2, 1, 5, 24, 17

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

a = 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

a = 1, 2, 5, 17, 24

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

a = 1, 2, 5, 17, 24

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

a = 1, 2, 5, 17, 24

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

a = 1, 2, 5, 17, 24

Mảng sau khi sắp xếp : 1, 2, 5, 17, 24

**Radix Sort**

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

a1 = 2, 5, 1, 4, 7

Đếm số lần xuất hiện của các chữ số mảng a1

count = 0, 1, 1, 0, 1, 1, 0, 1

Phần tử sau bằng phần tử liền trước cộng chính nó

count = 0, 1, 2, 2, 3, 4, 4, 5

Mảng kết quả :

A = 1, 2, 4, 5, 7

a = 1, 2, 24, 5, 17

B3 : i = 2. Lặp lần thứ 2

Dùng counting sort để sắp xếp các chữ số hàng chục

a2 = 0, 0, 2, 0, 1

Đếm số lần xuất hiện các chữ số mảng a2

count = 3, 1, 1

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

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**

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]:

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]

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]

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]

a = 1, 5, 2, 17, 24

flash = 1

Mảng A[1]-- : A[1 4] 🡪 A[1 3]

B4: nmove =4

Đổi chỗ a[0] và a[1]

a = 5, 1, 2, 17, 24

flash = 5

nmove = 4 = n – 1 🡪 Stop

B5: Phân lớp gồm lớp 0[ 5 1 2], lớp 1[ 17 24]

Dùng insertion sort để sắp xếp lại mảng này.

**Counting Sort**

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