Câu 1:

+) Mô tả:

X0 x1 x2 x3 x4

Ví dụ: 5 2 1 2 8

X=2 ( Giá trị cần ktra)

5 2 1 2 8 count = 0, x=2

I = 0 5(0) 2 1 2 8 count = 0,x=2

I = 1 5 2(1) 1 2 8 count = 1,x=2

I = 2 5 2 1(0) 2 8 count = 1,x=2

I = 3 5 2 1 2(1) 8 count = 2,x=2

I = 4 5 2 1 2 8(0) count = 2,x=2

+) Thiết kế giải thuật:

int count(int a[],int n,int x){

if(n-1==0){

if(a[n-1]==x){

return 1;

}

else return 0;

}

if(a[n-1]==x){

return 1+count(a,n-1,x);

}

else return count(a,n-1,x);

}

Câu 2:

-3 23 7 46 8 15 34

Sắp xếp nổi bọt :

Duyệt lần 1 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 23 7 46 8 15 34

-3 23 7 46 8 15 34

-3 7 23 46 8 15 34

-3 7 23 46 8 15 34

-3 7 23 8 46 15 34

-3 7 23 8 15 46 34

-3 7 23 8 15 34 46

Duyệt lần 2 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 7 23 8 15 34 46

-3 7 23 8 15 34 46

-3 7 23 8 15 34 46

-3 7 8 23 15 34 46

-3 7 8 15 23 34 46

-3 7 8 15 23 34 46

void BubbleSort(int arr[], int n)

{

for (int i =0; i < n-1; i++)

{

for (int j = n-1; j > i; j--)

{

if (arr[j] > arr[j+1])

Swap(&arr[j], &arr[j+1]);

}

}

}

Sắp xếp lựa chọn :

Duyệt lần 1 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 23 7 46 8 15 34

-3 23 7 46 8 15 34

Duyệt lần 2 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 23 7 46 8 15 34

|---------|

-3 7 23 46 8 15 34

Duyệt lần 3 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 7 23 46 8 15 34

|--------------------|

-3 7 8 46 23 15 34

Duyệt lần 4 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 7 8 46 23 15 34

|--------------------|

-3 7 8 15 23 46 34

Duyệt lần 5 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 7 8 15 23 46 34

-3 7 8 15 23 46 34

Duyệt lần 6 :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 7 8 15 23 46 34

|----------|

-3 7 8 15 23 34 46

void SelectionSort(int arr[], int n)

{

for (int i = 0; i < n; i++)

{

int m =i;

for (int j = i+1; j < n; j++)

{

if (arr[j] < arr[m])

m=j;

}

if (m != i)

{

Swap(&arr[m], &arr[i]);

}

}

}

Sắp xếp chèn :

x[0] x[1] x[2] x[3] x[4] x[5] x[6]

-3 23 7 46 8 15 34

-3 23 7 46 8 15 34

-3 23 7 46 8 15 34

|--------------|

-3 7 23 46 8 15 34

-3 7 23 46 8 15 34

|-------------------------|

-3 7 8 23 46 15 34

|-------------------------|

-3 7 8 15 23 46 34

|--------------|

-3 7 8 15 23 34 46

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| pp chọn | |  |  |  |  |  |  |  |  |
| Lần duyệt | x0 | x1 | x2 | x3 | x4 | x5 | x6 |  |  |
| i=0 | ***-3*** | ***23*** | ***7*** | ***46*** | ***8*** | ***15*** | ***34*** | Xét từ x0 đến x6 có x0 nhỏ nhất --> giữ nguyên chỗ |  |
| i-1 | -3 | ***23*** | ***7*** | ***46*** | ***8*** | ***15*** | ***34*** | Xét từ x1 đến x6 có x2 nhỏ nhất --> đổi chỗ x1 và x2 |  |
| i=2 | -3 | 7 | ***23*** | ***46*** | ***8*** | ***15*** | ***34*** | Xét từ x2 đến x6 có x4 nhỏ nhất --> đổi chỗ x2 và x4 |  |
| i=3 | -3 | 7 | 8 | ***46*** | ***23*** | ***15*** | ***34*** | Xét từ x3 đến x6 có x5 nhỏ nhất --> đổi chỗ x3 và x5 |  |
| i=4 | -3 | 7 | 8 | 15 | ***23*** | ***46*** | ***34*** | Xét từ x4 đến x6 có x4 nhỏ nhất --> giữ nguyên chỗ |  |
| i=5 | -3 | 7 | 8 | 15 | 23 | ***46*** | ***34*** | Xét từ x5 đến x6 có x6 nhỏ nhất --> đổi chỗ x5 và x6 |  |
|  | -3 | 7 | 8 | 15 | 23 | 34 | ***46*** | Hoàn thành |  |
|  |  |  |  |  |  |  |  |  |  |
|  | in đậm, nghiêng: dãy đang xét | | | | |  |  |  |  |
|  | vàng: phần tử nhỏ nhất trong dãy | | | | | |  |  |  |

void insertionSort(int arr[], int n)

{

int tam;

for (int i = 1; i < n; i++)

{

tam = arr[i];

int j = i -1 ;

while(j >=0 && tam < arr[j])

{

arr[j+1] = arr[j];

j = j - 1;

}

arr[j+1] = tam;

}

}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lần duyệt | x0 | x1 | x2 | x3 | x4 | x5 | x6 |  |
| i=1 | -3 | 23 | 7 | 46 | 8 | 15 | 34 | Lấy t = x1 chèn vào dãy đích |
|  | -3 |  | 7 | 46 | 8 | 15 | 34 | Thực hiện chèn |
| i=2 | **-3** | **23** | 7 | 46 | 8 | 15 | 34 | Lấy t=x2 chèn vào dãy đích |
|  | -3 |  | 23 | 46 | 8 | 15 | 34 | Thực hiện chèn |
| i=3 | **-3** | **7** | **23** | 46 | 8 | 15 | 34 | Lấy t=x3 chèn vào dãy đích |
|  | -3 | 7 | 23 |  | 8 | 15 | 34 | Thực hiện chèn |
| i=4 | **-3** | **7** | **23** | **46** | 8 | 15 | 34 | Lấy t=x4 chèn vào dãy đích |
|  | -3 | 7 |  | 23 | 46 | 15 | 34 | Thực hiện chèn |
| i=5 | **-3** | **7** | **8** | **23** | **46** | 15 | 34 | Lấy t=x5 chèn vào dãy đích |
|  | -3 | 6 | 8 |  | 23 | 46 | 34 | Thực hiện chèn |
| i=6 | **-3** | **6** | **8** | **15** | **23** | **46** | 34 | Lấy t=x6 chèn vào dãy đích |
|  | -3 | 6 | 8 | 15 | 23 |  | 46 | Thực hiện chèn |
|  | **-3** | **6** | **8** | **15** | **23** | **34** | 46 | Hoàn thành |
|  | im đậm: phần tử dãy đích | | | |  |  |  |  |
|  | vàng: phần tử đầu dãy nguồn | | | | |  |  |  |
|  | trống: chỗ phụ hợp để chèn | | | | |  |  |  |

Sắp xếp phân đoạn

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | x0 | x1 | x2 | x3 | x4 | x5 | x6 |  |
|  | -3 | 23 | 7 | 46 | 8 | 15 | 34 | i =0, j=6 --> tg = 46 |
|  | -3 | 23 | 7 | **46** | 8 | 15 | **34** | i=3, j=6 có x3 > x6 --> đổi chỗ |
|  | -3 | 23 | 7 | 34 | 8 | 15 | **46** | dừng |
|  | X1 | | | | | | X2 |  |
|  |  |  |  |  |  |  |  |  |
| Xét X1 | x0 | x1 | x2 | x3 | x4 | x5 |  |  |
|  | -3 | 23 | 7 | 34 | 8 | 15 |  | i=0, j=5 --> tg=7 |
|  | -3 | **23** | **7** | 34 | 8 | 15 |  | i=1, j=2 có x1 > x2 --> đổi chỗ |
|  | -3 | 7 | 23 | 34 | 8 | 15 |  | dừng |
|  | X3 | X4 | | | | |  |  |
|  |  |  |  |  |  |  |  |  |
| Xét X4 | x0 | x1 | x2 | x3 | x4 |  |  |  |
|  | 7 | 23 | 34 | 8 | 15 |  |  | i=0, j=4 --> tg =34 |
|  | 7 | 23 | **34** | 8 | **15** |  |  | i=2, j=4 có x2>x4 --> đổi chỗ |
|  | 7 | 23 | 15 | 8 | 34 |  |  | dừng |
|  | X5 | | | | X6 |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Xét X5 | x0 | x1 | x2 | x3 |  |  |  | i=0, j=3 --> tg=23 |
|  | 7 | 23 | 15 | 8 |  |  |  |  |
|  | 7 | **23** | 15 | **8** |  |  |  | i=1, j=3 có x1>x3 --> đổi chỗ |
|  | 7 | 8 | 15 | 23 |  |  |  | Dừng |
|  | X7 | | | X8 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| Xét X7 | x0 | x1 | x2 |  |  |  |  | i=0, j=2 --> tg =8 |
|  | 7 | 8 | 15 |  |  |  |  | Hoàn thành |
|  |  |  |  |  |  |  |  |  |

Xếp các đoạn với nhau ta được dãy: -3 , 7,8,15,23,34,46

int partition (int arr[], int low, int high)

{

    int pivot = arr[high]; // pivot

    int i = (low - 1);

    for (int j = low; j <= high - 1; j++)

    {

        if (arr[j] < pivot)

        {

            i++;

            swap(&arr[i], &arr[j]);

        }

    }

    swap(&arr[i + 1], &arr[high]);

    return (i + 1);

}

void quickSort(int arr[], int low, int high)

{

    if (low < high)

    {

        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);

        quickSort(arr, pi + 1, high);

    }

}

Sắp xếp vụn đống:

-3(0)

/ \

23(1) 7(2)

/ \ / \

46(3) 8(4) 15(5) 34(6)

-3(0)

/ \

46(1) 7(2)

/ \ / \

23(3) 8(4) 15(5) 34(6)

-3 46 7 23 8 15 34

-3(0)

/ \

46(1) 34(2)

/ \ / \

23(3) 8(4) 15(5) 7(6)

-3 46 34 23 8 15 7

46(0)

/ \

-3(1) 34(2)

/ \ / \

23(3) 8(4) 15(5) 7(6)

46 -3 34 23 8 15 7

46(0)

/ \

23(1) 34(2)

/ \ / \

-3(3) 8(4) 15(5) 7(6)

46 23 34 -3 8 15 7

7(0)

/ \

23(1) 34(2)

/ \ / \

-3(3) 8(4) 15(5) 46(6)

7 23 34 -3 8 15 46

=> 7 23 34 -3 8 15 46

34(0)

/ \

23(1) 7(2)

/ \ /

-3(3) 8(4) 15(5)

34 23 7 -3 8 15 46

34(0)

/ \

23(1) 15(2)

/ \ /

-3(3) 8(4) 7(5)

34 23 15 -3 8 7 46

7(0)

/ \

23(1) 15(2)

/ \ /

-3(3) 8(4) 34(5)

7 23 15 -3 8 34 46

23(0)

/ \

7(1) 15(2)

/ \

-3(3) 8(4)

23 7 15 -3 8 34 46

23(0)

/ \

7(1) 15(2)

/ \

-3(3) 8(4)

23 7 15 -3 8 34 46

23(0)

/ \

7(1) 15(2)

/ \

-3(3) 8(4)

8 7 15 -3 23 34 46

8(0)

/ \

7(1) 15(2)

/

-3(3)

8 7 15 -3 23 34 46

15(0)

/ \

7(1) 8(2)

/

-3(3)

15 7 8 -3 23 34 46

-3(0)

/ \

7(1) 8(2)

/

15(3)

-3 7 8 15 23 34 46

-3(0)

/ \

7(1) 8(2)

-3 7 8 15 23 34 46

-3(0)

/ \

7(1) 8(2)

Z: -3 7 8 15 23 34 46

void heapify(int arr[], int n, int i)

{

    int largest = i; // Initialize largest as root

    int l = 2 \* i + 1; // left = 2\*i + 1

    int r = 2 \* i + 2; // right = 2\*i + 2

    if (l < n && arr[l] > arr[largest])

        largest = l;

    if (r < n && arr[r] > arr[largest])

        largest = r;

    if (largest != i) {

        swap(arr[i], arr[largest]);

        heapify(arr, n, largest);

    }

}

 void heapSort(int arr[], int n)

{

    for (int i = n / 2 - 1; i >= 0; i--)

        heapify(arr, n, i);

    for (int i = n - 1; i > 0; i--) {

        // Move current root to end

        swap(arr[0], arr[i]);

         heapify(arr, i, 0);

    }

}

Sắp xếp trộn:

Z: x[0] x[1] x[2] x[3] x[4] x[5] x[6]

Z: -3 23 7 46 8 15 34

Z: [-3] [23] [7] [46] [8] [15] [34]

Z: [-3 23] [7 46] [8 15] [34]

Z: [-3 7 23 46] [8 15 34]

Z: [-3 7 8 15 23 34 46]

void merge(int arr[], int l, int m, int r)

{

    int n1 = m - l + 1;

    int n2 = r - m;

    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)

        L[i] = arr[l + i];

    for (int j = 0; j < n2; j++)

        R[j] = arr[m + 1 + j];

    int i = 0;

    int j = 0;

    int 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];

        i++;

        k++;

    }

    while (j < n2) {

        arr[k] = R[j];

        j++;

        k++;

    }

}

void mergeSort(int arr[],int l,int r){

    if(l>=r){

        return;//returns recursively

    }

    int m =l+ (r-l)/2;

    mergeSort(arr,l,m);

    mergeSort(arr,m+1,r);

    merge(arr,l,m,r);

}