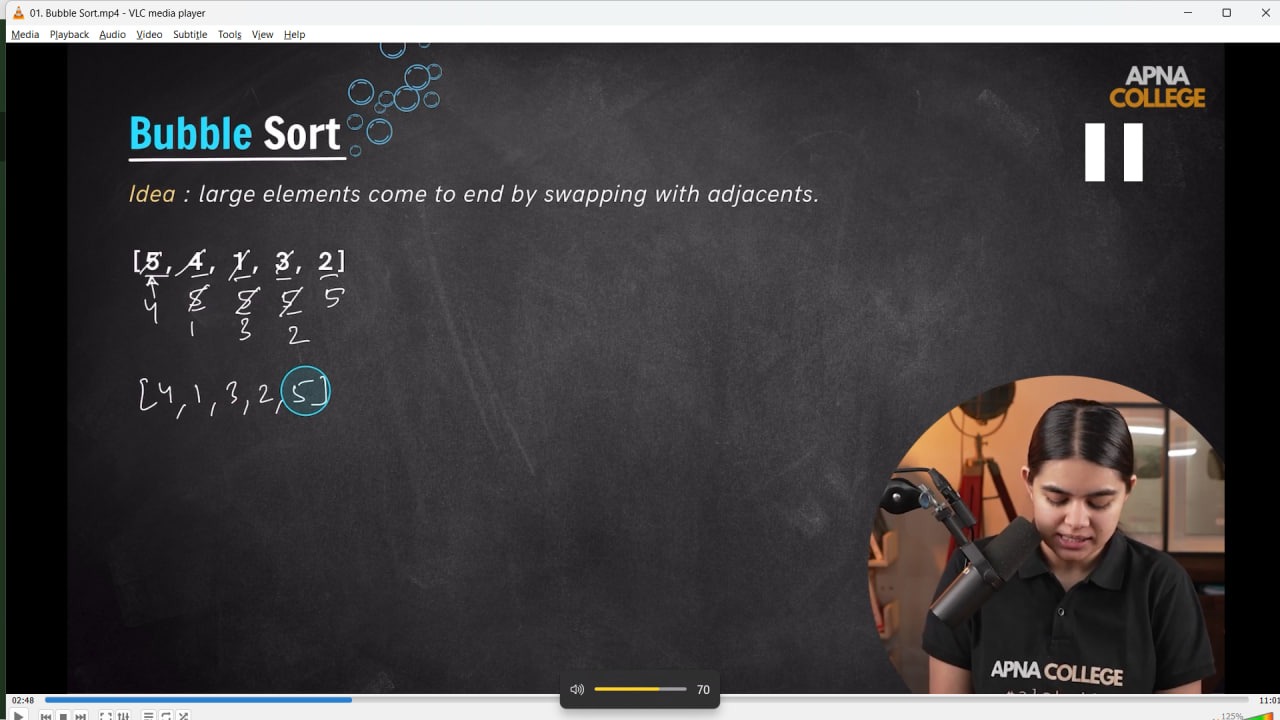
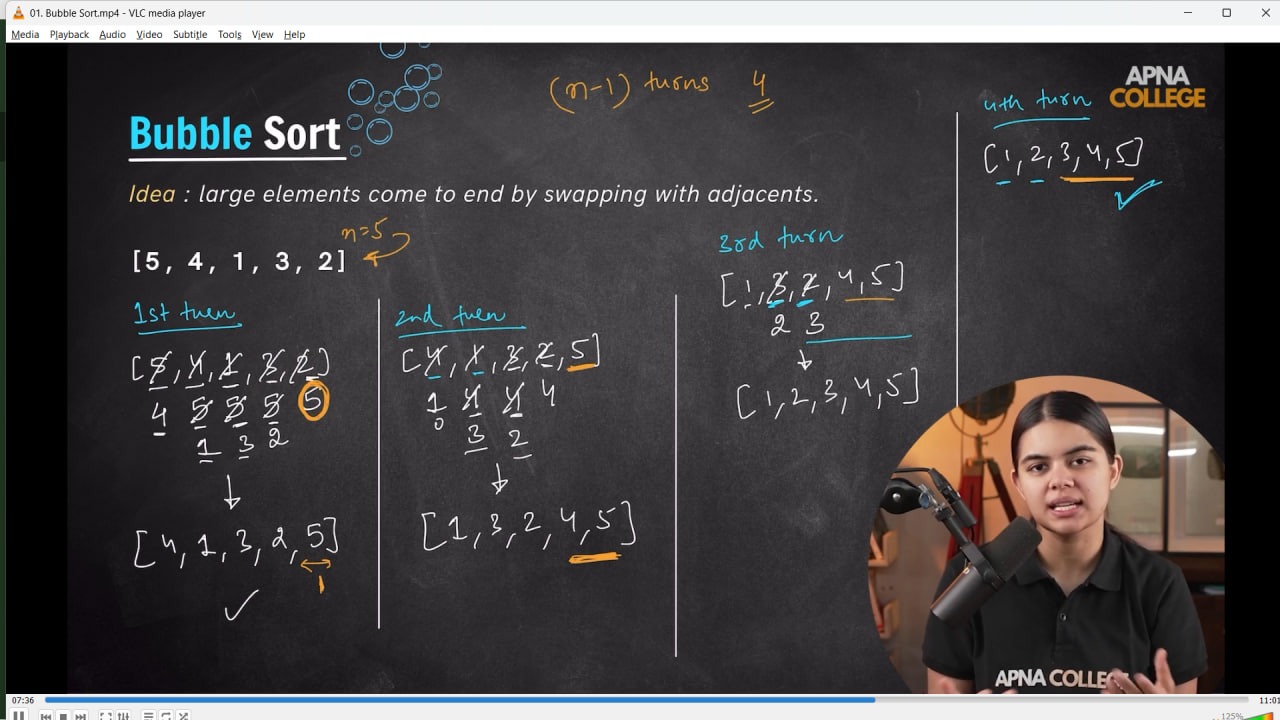
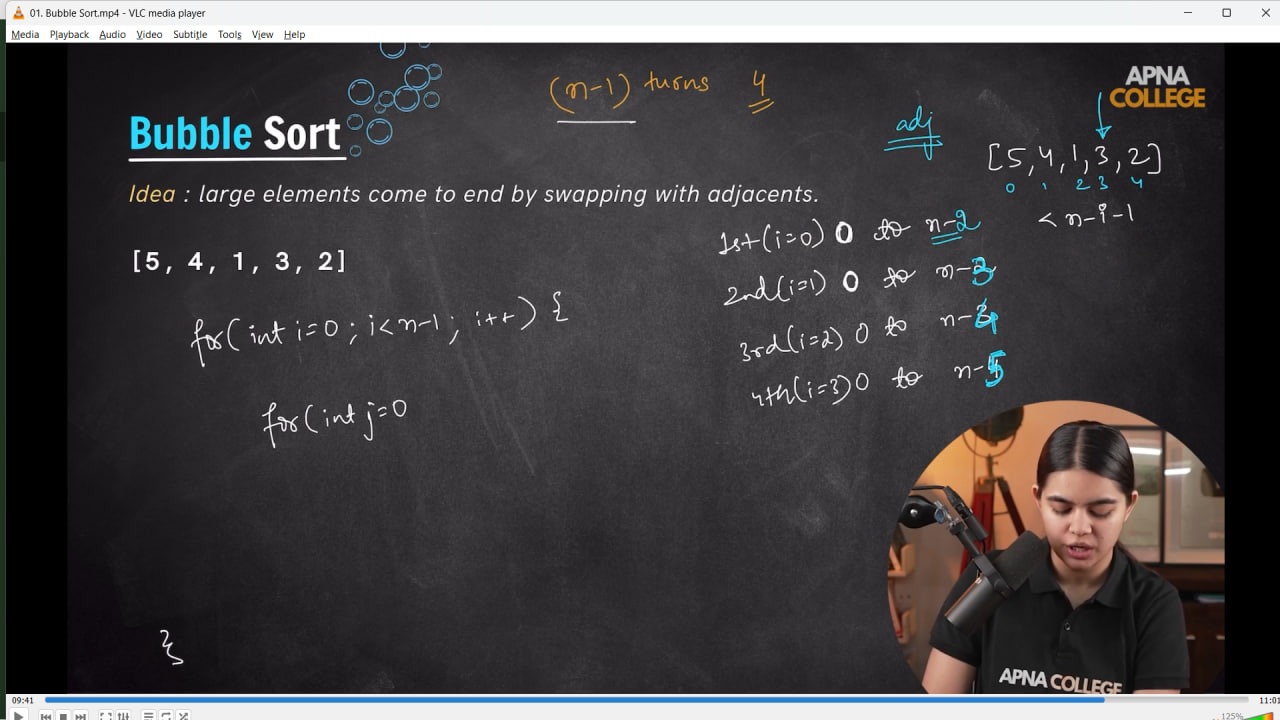
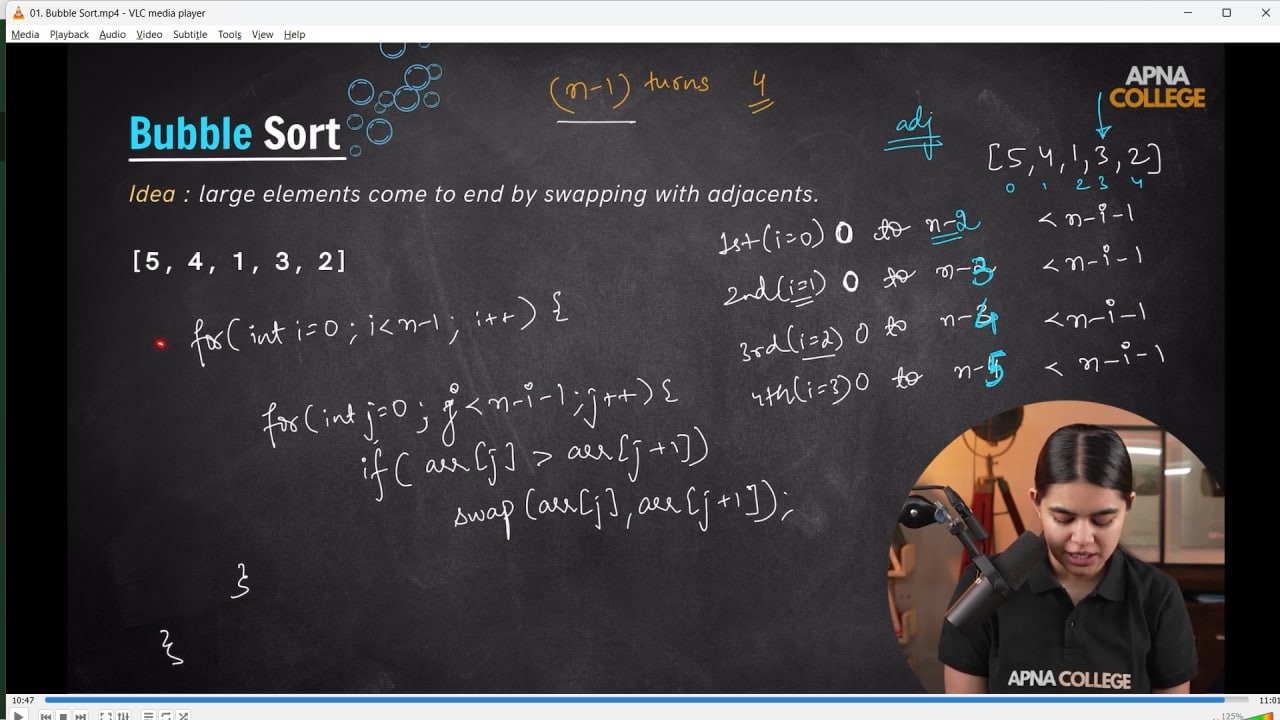
**3 Sorting Algorithms –**

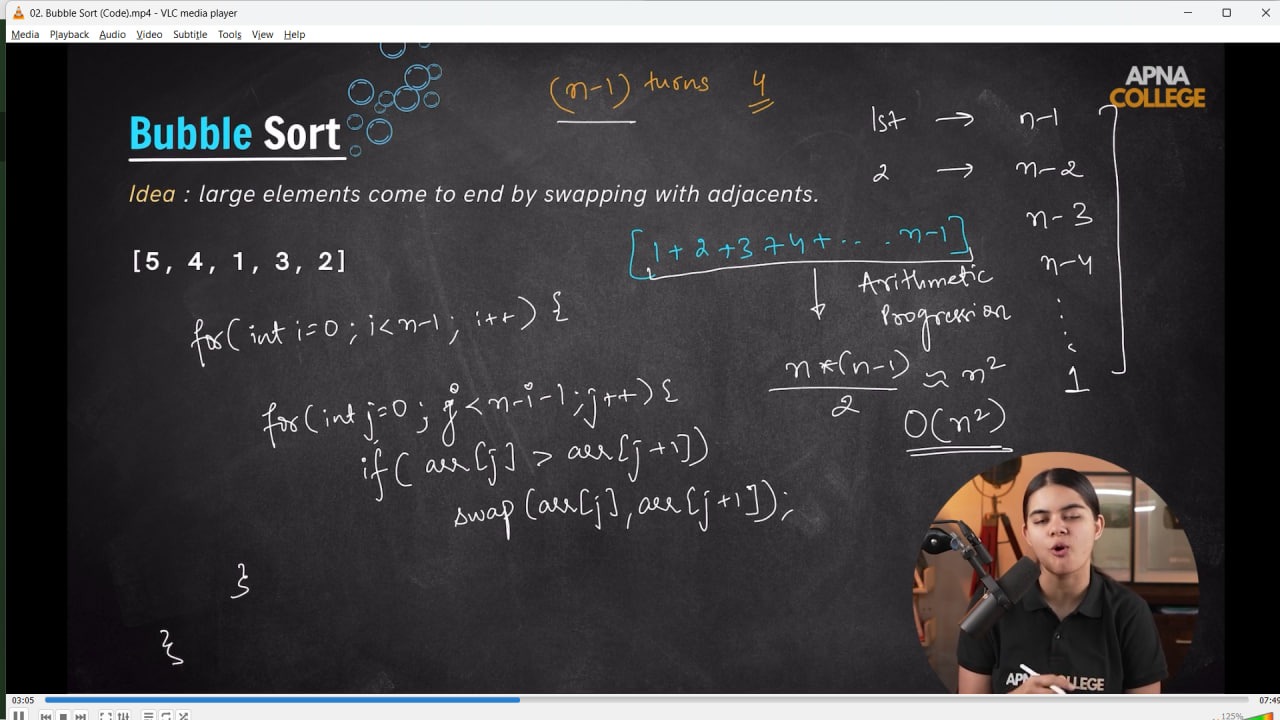
**// 1) Bubble Sort ALgo -**











void printArray(int arr[], int n)

{

    cout << "The sorted array after operation is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

}

void bubbleSort(int arr[], int n)

{

    for (int i = 0; i < n - 1; i++)

    {

        bool isSwap = false; // for the cases where array is already sorted, so for avoiding TC issues

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

        {

            if (arr[j] > arr[j + 1])

            {

                swap(arr[j], arr[j + 1]);

                isSwap = true;

            }

        }

        if (!isSwap)

        {

            // means array is already sorted, so return it from here

            return;

        }

    }

    printArray(arr, n);

}

int main()

{

    int n;

    cout << "value of array size" << endl;

    cin >> n;

    int arr[n];

    cout << "Write down the array elemenets - " << endl;

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    cout << "So, the inserted array is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

    bubbleSort(arr, n);

    return 0;

    /\*

    value of array size

    8

    Write down the array elemenets -

    5 4 3 2 1 9 8 6

    So, the inserted array is -

    5 4 3 2 1 9 8 6

    The sorted array is -

    1 2 3 4 5 6 8 9

    T.C  - O(n^2)

     \*/

}

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// For descending Ordrt -**

// void printArray(int arr[], int n)

// {

//     cout << "The sorted array after operation is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// void bubbleSort(int arr[], int n)

// {

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

//     {

//         bool isSwap = false; // for the cases where array is already sorted, so for avoiding TC issues

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

//         {

//             if (arr[j] < arr[j + 1])

//             {

//                 swap(arr[j], arr[j + 1]);

//                 isSwap = true;

//             }

//         }

//         if (!isSwap)

//         {

//             // means array is already sorted, so return it from here

//             return;

//         }

//     }

//     printArray(arr, n);

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     bubbleSort(arr, n);

//     return 0;

//     /\*

//     value of array size

//     5

//     Write down the array elemenets -

//     9 5 3 1 4

//     So, the inserted array is -

//     9 5 3 1 4

//     The sorted array is -

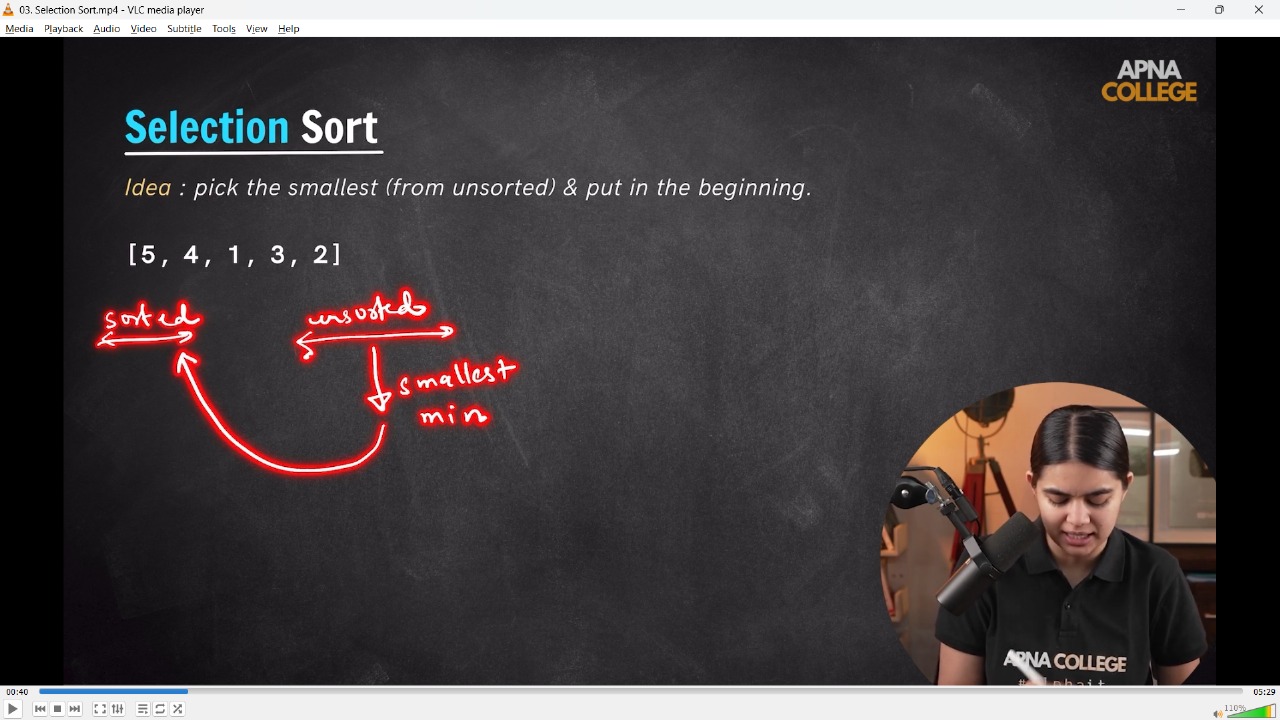
//     9 5 4 3 1

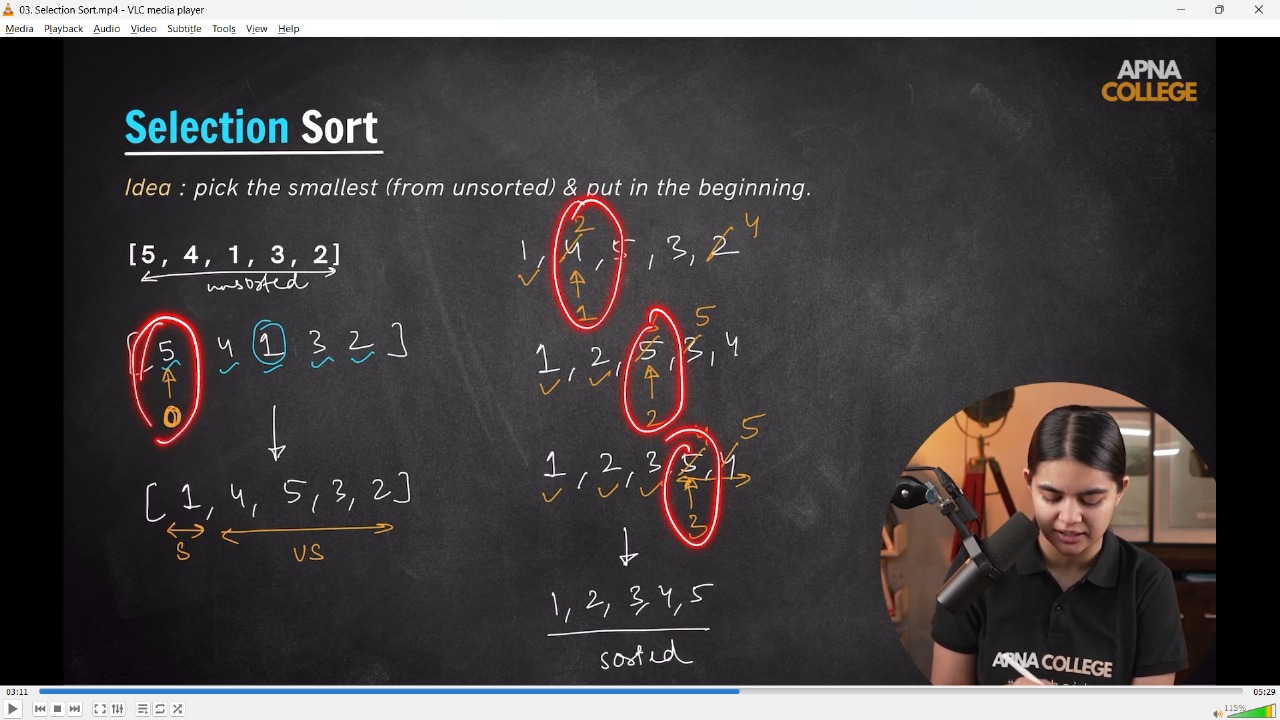
//      \*/

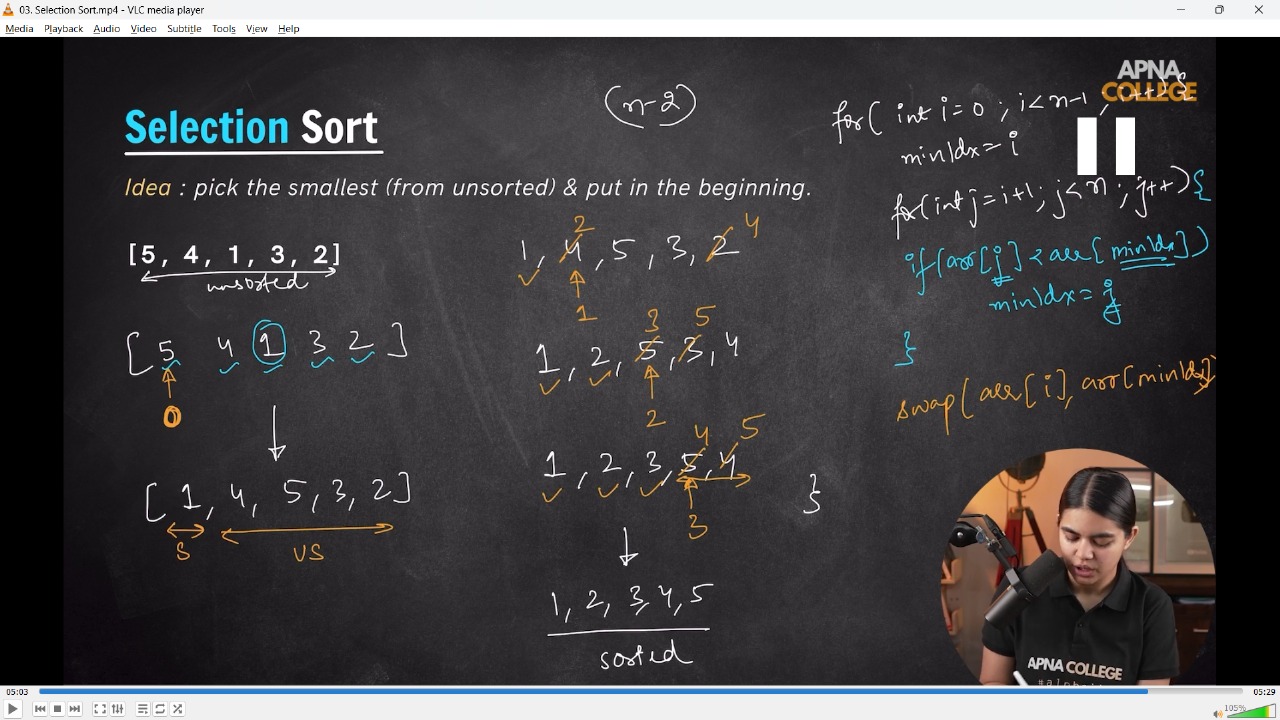
// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**//2) Selection Sort Implementation –**







// void printArray(int arr[], int n)

// {

//     cout << "Array after sorting is -  " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// void selectionSort(int arr[], int n)

// {

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

//     {

//         int minIndex = i;

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

//         {

//             if (arr[j] < arr[minIndex])

//             {

//                 minIndex = j;

//             }

//         }

//         swap(arr[i], arr[minIndex]);

//     }

//     printArray(arr, n);

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     selectionSort(arr, n);

//     return 0;

/\*

value of array size

5

Write down the array elemenets -

5 4 1 3 2

So, the inserted array is -

5 4 1 3 2

Array after sorting is -

1 2 3 4 5

 \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

//2.1) Selection Sort Implementation for decreasing Order -

void printArray(int arr[], int n)

{

    cout << "Array after sorting is -  " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

}

void selectionSort(int arr[], int n)

{

    for (int i = 0; i < n - 1; i++)

    {

        int minIndex = i;

        for (int j = i; j < n; j++)

        {

            if (arr[j] > arr[minIndex])

            {

                minIndex = j;

            }

        }

        swap(arr[i], arr[minIndex]);

    }

    printArray(arr, n);

}

int main()

{

    int n;

    cout << "value of array size" << endl;

    cin >> n;

    int arr[n];

    cout << "Write down the array elemenets - " << endl;

    for (int i = 0; i < n; i++)

    {

        cin >> arr[i];

    }

    cout << "So, the inserted array is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << " ";

    }

    cout << endl;

    selectionSort(arr, n);

    return 0;

/\*

value of array size

5

Write down the array elemenets -

1 6 2 4 3 5

So, the inserted array is -

1 6 2 4 3

Array after sorting is -

6 4 3 2 1

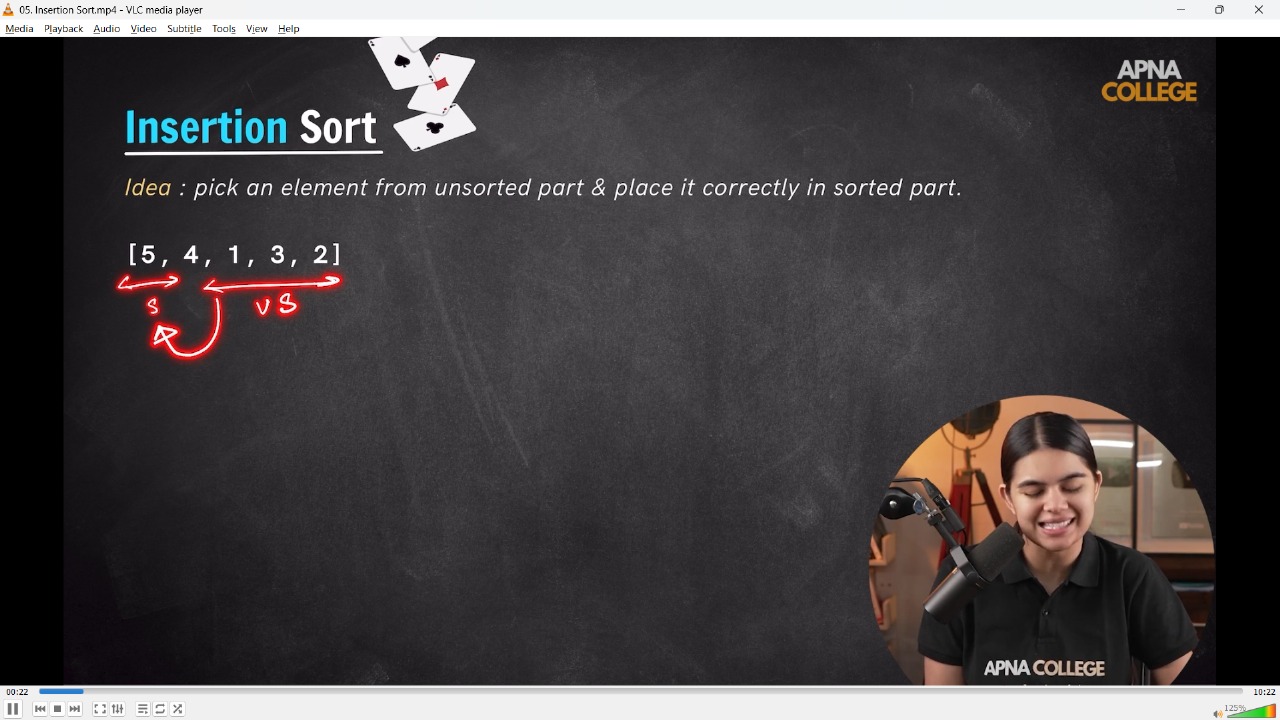
T.C - O(n\*logn) is better than O(n^2).

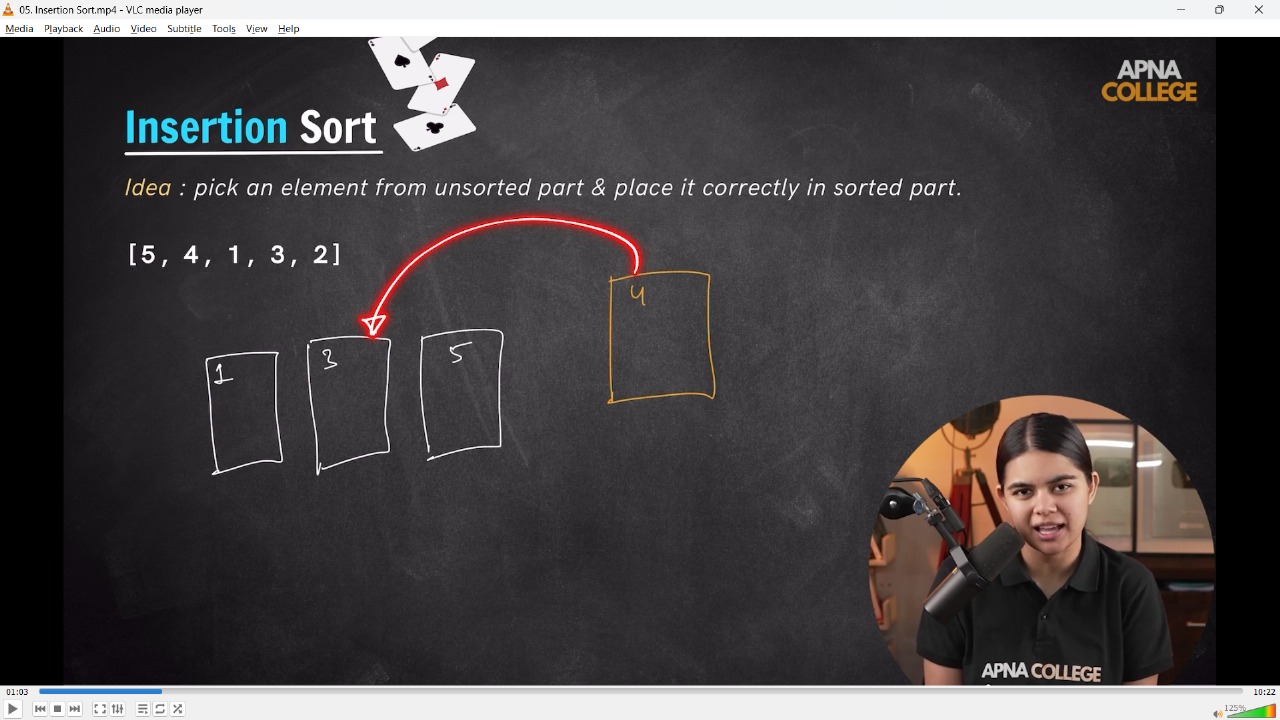
\*/

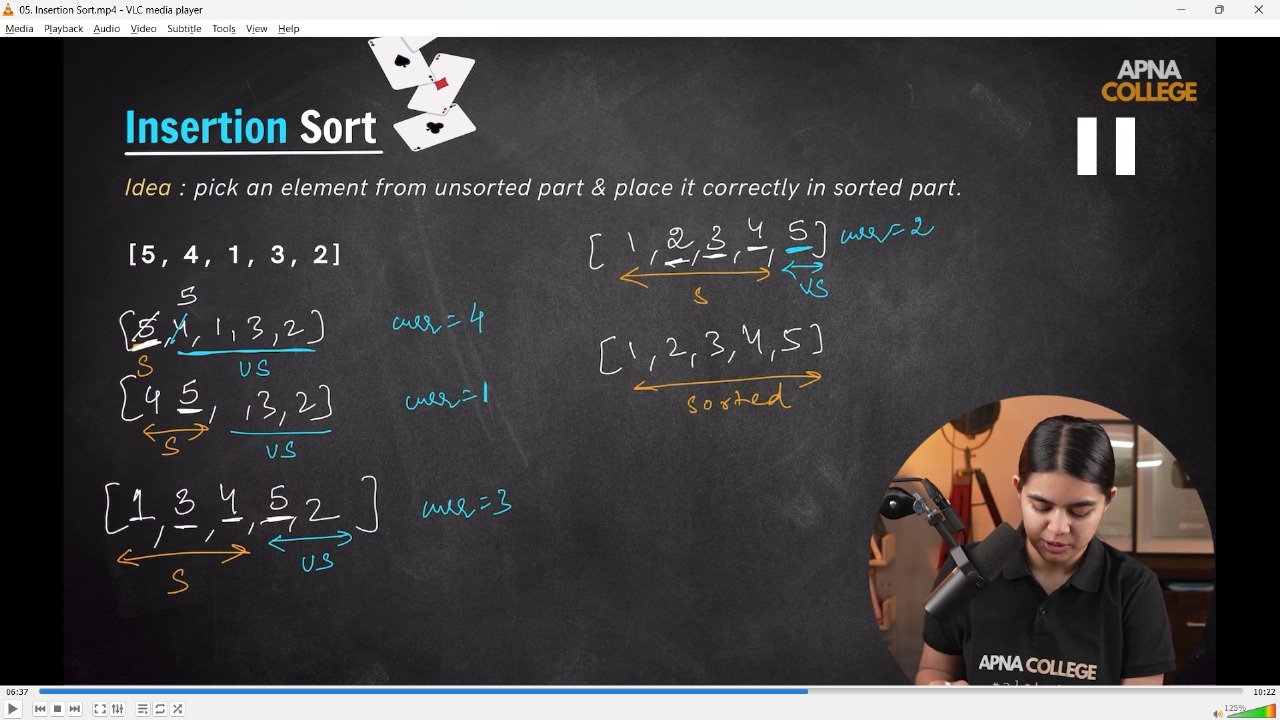
}

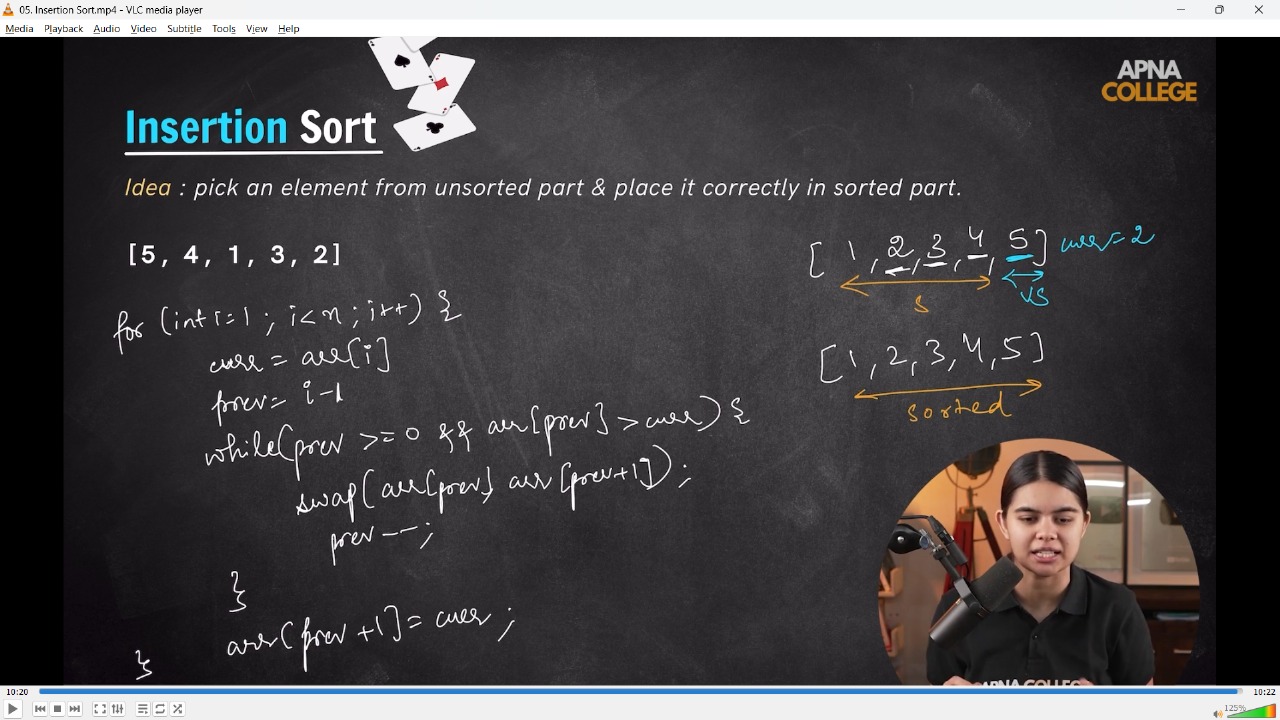
// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 3) Insertion Sort Algorithm -**









// void printArray(int arr[], int n)

// {

//     cout << "After Sorting, Array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// void inseertionSort(int arr[], int n)

// {

//     for (int i = 1; i < n; i++)

//     {

//         int curr = arr[i];

//         int prev = i - 1;

//         while (prev >= 0 && arr[prev] > curr)

//         {

//             swap(arr[prev], arr[prev + 1]);

//             prev--;

//         }

//         arr[prev + 1] = curr;

//     }

//     printArray(arr, n);

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     inseertionSort(arr, n);

// /\*

// value of array size

// 5

// Write down the array elemenets -

// 5 4 1 3 2

// So, the inserted array is -

// 5 4 1 3 2

// After Sorting, Array is -

// 1 2 3 4 5

//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

// 3.1) Insertion Sort Algorithm for descending order -

// void printArray(int arr[], int n)

// {

//     cout << "After Sorting, Array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// void inseertionSort(int arr[], int n)

// {

//     for (int i = 1; i < n; i++)

//     {

//         int curr = arr[i];

//         int prev = i - 1;

//         while (prev >= 0 && arr[prev] > curr)

//         {

//             swap(arr[prev], arr[prev + 1]);

//             prev--;

//         }

//         arr[prev + 1] = curr;

//     }

//     printArray(arr, n);

// }

// int main()

// {

//     int n;

//     cout << "value of array size" << endl;

//     cin >> n;

//     int arr[n];

//     cout << "Write down the array elemenets - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cin >> arr[i];

//     }

//     cout << "So, the inserted array is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

//     inseertionSort(arr, n);

// /\*

// value of array size

// 5

// Write down the array elemenets -

// 2 1 4 3 5

// So, the inserted array is -

// 2 1 4 3 5

// After Sorting, Array is -

// 1 2 3 4 5

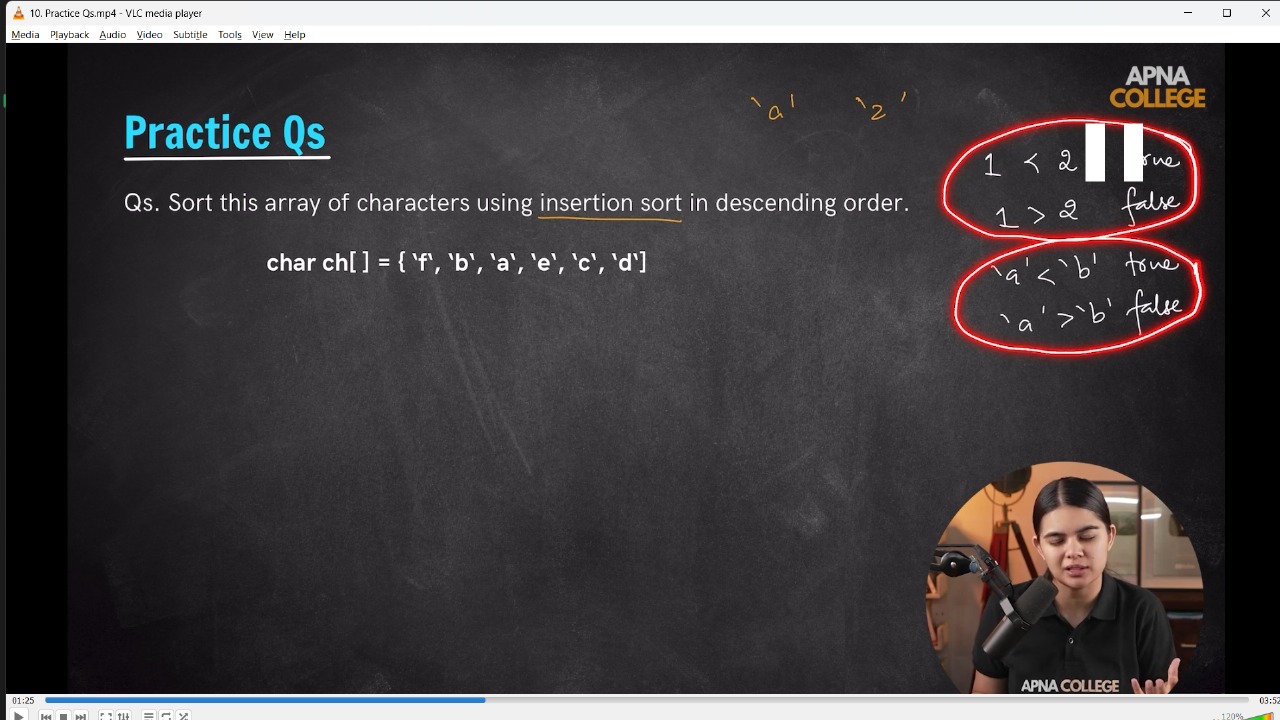
// T.C - O(n^2)

//  \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 3.2) Insertion Soet Practice Qun -**



void printArray(char arr[], int n)

{

    cout << "After Sorting, Array is - " << endl;

    for (int i = 0; i < n; i++)

    {

        cout << arr[i] << ",";

    }

    cout << endl;

}

void sortChar(char arr[], int n)

{

    for (int i = 1; i < n; i++)

    {

        int curr = arr[i];

        int prev = i - 1;

        while (prev >= 0 && arr[prev] < curr)

        {

            swap(arr[prev], arr[prev + 1]);

            prev--;

        }

        arr[prev + 1] = curr;

    }

    printArray(arr, n);

}

int main()

{

    char ch[6] = {'f', 'b', 'a', 'e', 'c', 'd'};

    sortChar(ch, 6);

    return 0;

    /\*

    After Sorting, Array is -

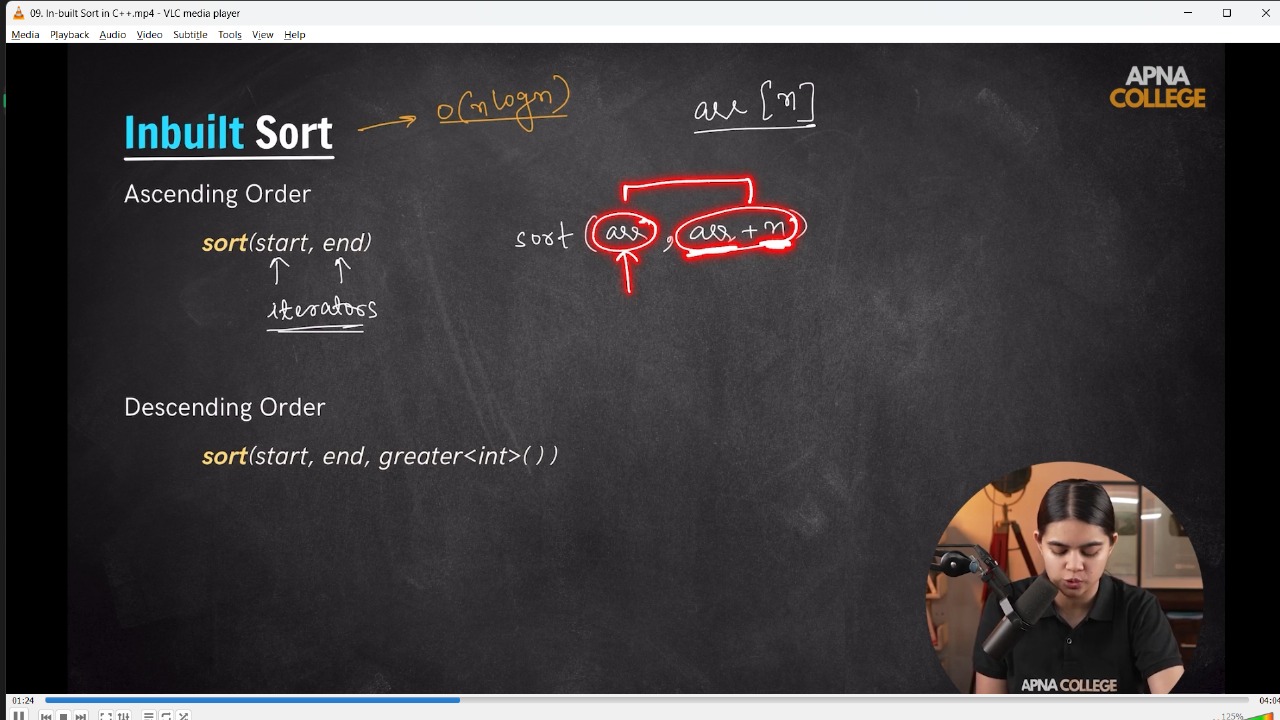
    f,e,d,c,b,a,

    \*/

}

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

**// 4) in-built sort functionality –**



// void printArray(int arr[], int n)

// {

//     cout << "Array after sorting is - " << endl;

//     for (int i = 0; i < n; i++)

//     {

//         cout << arr[i] << " ";

//     }

//     cout << endl;

// }

// int main()

// {

//     /\* Sort syntax -

//     sort(start, end)

//      \*/

//     int arr[8] = {1, 4, 6, 5, 8, 3, 2, 7};

//     // sort(arr, arr + 8);

//     // printArray(arr,8);

//     /\*

//     Array after sorting is -

//     1 2 3 4 5 6 7 8

//      \*/

//     sort(arr + 2, arr + 5);

//     printArray(arr, 8);

//     /\*

//     Array after sorting is -

//     1 2 3 4 5 6 7 8

//      \*/

//     //  For sorting in decreasing order -

//     // sort(start,end,greater<int>())

//     sort(arr, arr + 8, greater<int>());

//     printArray(arr,8);

//     /\*

//     Array after sorting is -

//     8 7 6 5 4 3 2 1

//      \*/

// }

// \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_