Сортировки одномерных и двухмерных массивов в C# можно выполнить с помощью различных алгоритмов, таких как пузырьковая, вставочная, выбором и двоичный поиск.

Пузырьковая сортировка использует алгоритм прохода по массиву несколько раз, каждый раз меняя местами соседние элементы, если они находятся в неправильном порядке. Проходы повторяются до тех пор, пока все элементы массива не будут отсортированы. Пример кода сортировки одномерного массива пузырьком:

```csharp

int[] arr = new int[] { 4, 2, 7, 1, 3 };

for (int i = 0; i < arr.Length - 1; i++)

{

for (int j = 0; j < arr.Length - i - 1; j++)

{

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

{

int temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

```

Данный код проходит по массиву `arr` и меняет местами элементы, если они находятся в неправильном порядке. Количество проходов определяется длиной массива (`arr.Length`). В итоге массив будет отсортирован по возрастанию.

Вставочная сортировка использует алгоритм вставки каждого элемента массива на свое место, чтобы получить отсортированный массив. Каждый следующий элемент вставляется в массив уже отсортированных элементов. Пример кода сортировки одномерного массива вставками:

```csharp

int[] arr = new int[] { 4, 2, 7, 1, 3 };

for (int i = 1; i < arr.Length; i++)

{

int j = i;

while (j > 0 && arr[j - 1] > arr[j])

{

int temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

j--;

}

}

```

Данный код проходит по массиву `arr` и вставляет каждый элемент на свое место, чтобы получить отсортированный массив. Количество проходов также равно длине массива (`arr.Length`).

Сортировка выбором также основана на выборе наименьшего элемента из неотсортированной части массива и его вставке в начало отсортированной части массива. Пример кода сортировки одномерного массива выбором:

```csharp

int[] arr = new int[] { 4, 2, 7, 1, 3 };

for (int i = 0; i < arr.Length - 1; i++)

{

int minIndex = i;

for (int j = i + 1; j < arr.Length; j++)

{

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

{

minIndex = j;

}

}

int temp = arr[i];

arr[i] = arr[minIndex];

arr[minIndex] = temp;

}

```

Данный код проходит по массиву `arr` и выбирает наименьший элемент из неотсортированной части массива, затем вставляет его в начало уже отсортированной части массива.

Двоичный поиск используется для поиска элемента в уже отсортированном массиве. Алгоритм основан на разделении массива на две части и поиске элемента в правой или левой части в зависимости от его значения. Пример кода двоичного поиска в одномерном массиве:

```csharp

int[] arr = new int[] { 1, 2, 3, 4, 7 };

int target = 3;

int left = 0;

int right = arr.Length - 1;

int mid = 0;

while (left <= right)

{

mid = (left + right) / 2;

if (arr[mid] == target)

{

Console.WriteLine("Element found at index " + mid);

break;

}

else if (arr[mid] < target)

{

left = mid + 1;

}

else

{

right = mid - 1;

}

}

if (left > right)

{

Console.WriteLine("Element not found");

}

```

Данный код осуществляет поиск элемента `target` в отсортированном массиве `arr`. Если элемент найден, выводится его индекс, иначе выводится сообщение о том, что элемент не найден. Алгоритм повторяется до тех пор, пока левая граница массива `left` не станет больше правой границы массива `right`.