**Линейный поиск**

Лучший случай (элемент на 0 месте)

private static int LineSearch(IReadOnlyList<int> list, int searchElement)  
{  
 for (var i = 0; i < list.Count; i++) // O(1)  
 {  
 if (list[i] == searchElement) // O(1)  
 {  
 return i;  
 }  
 }  
   
 return -1;  
}

Tл(n) = O(1) + O(1) => 2O(1) => O(1)

Худший случай (элемента нет в массиве)

private static int LineSearch(IReadOnlyList<int> list, int searchElement)  
{  
 for (var i = 0; i < list.Count; i++) // O(n)  
 {  
 if (list[i] == searchElement) // O(n)  
 {  
 return i;  
 }  
 }  
   
 return -1;  
}

Tх(n) = O(n) + O(n) => 2O(n) => O(n)

**Линейный поиск с барьером**

Лучший случай (элемент на 0 месте)

private static int LineSearchBarrier(IList<int> list, int searchElement)  
{  
 list.Add(searchElement); // O(1)  
   
 var i = 0;  
 while (list[i] != searchElement) // O(1)  
 {  
 i++;  
 }  
  
 return i < list.Count - 1 ? i : -1;  
}

Tл(n) = O(1) + O(1) => 2O(1) => O(1)

Худший случай (элемента нет в массиве)

private static int LineSearchBarrier(IList<int> list, int searchElement)  
{  
 list.Add(searchElement); // O(1)  
   
 var i = 0;  
 while (list[i] != searchElement) // O(n)  
 {  
 i++;   
 }  
  
 return i < list.Count - 1 ? i : -1;  
}

Tх(n) = O(1) + O(n) => O(n)

**Сортировка методом простого обмена**

Лучший случай (массив отсортирован)

private static List<int> BubbleSort(IEnumerable<int> list)  
{  
 var res = list.ToList();  
  
 for (var i = 0; i < res.Count - 1; i++) // O(n – 1)   
 {  
 for (var j = 0; j < res.Count - 1; j++) // O(n – 1)  
 {  
 if (res[j] > res[j + 1]) // O(n – 1)  
 {  
 (res[j], res[j + 1]) = (res[j + 1], res[j]); // O(0)  
 }  
 }  
 }  
  
 return res;  
}

Tл(n) = O(n - 1) \* (O(n - 1) + O(n - 1) + O(0)) = O(n - 1) \* 2O(n - 1) = 2O((n - 1)2)=> 2O(n2) => O(n2)

Худший случай (массив отсортирован в обратном порядке)

private static List<int> BubbleSort(IEnumerable<int> list)  
{  
 var res = list.ToList();  
  
 for (var i = 0; i < res.Count - 1; i++) // O(n - 1)  
 {  
 for (var j = 0; j < res.Count - 1; j++) // O(n – 1)  
 {  
 if (res[j] > res[j + 1]) // O(n – 1)  
 {  
 (res[j], res[j + 1]) = (res[j + 1], res[j]); // O(n - 1)  
 }  
 }  
 }  
  
 return res;  
}

Tх(n) = O(n - 1) \* (O(n - 1) + O(n - 1) + O(n - 1)) = O(n - 1) \* 3O(n - 1) = 3O((n - 1)2)=> 3O(n2) => O(n2)

**Сортировка методом простого выбора**

Лучший случай (массив отсортирован)

private static List<int> SelectionSort(IEnumerable<int> list)  
{  
 var res = list.ToList();  
  
 for (var i = 0; i < res.Count - 1; i++) // O(n – 1)  
 {  
 var min = i;  
   
 for (var j = i + 1; j < res.Count; j++) // O(n - 1)  
 {  
 if (res[j] < res[min]) // O(n - 1)  
 {  
 min = j;  
 }  
 }  
  
 if (min != i) // O(n - 1)  
 {  
 (res[i], res[min]) = (res[min], res[i]);  
 }  
 }  
  
 return res;  
}

Tл(n) = O(n - 1) \* (O(n - 1) + O(n - 1)) + O(n - 1) = O(n - 1) \* 2O(n - 1) + O(n – 1) = 2O((n – 1)2) + O(n – 1) => O(n2) + O(n) => O(n2)

Худший случай (массив отсортирован в обратном порядке)

private static List<int> SelectionSort(IEnumerable<int> list)  
{  
 var res = list.ToList();  
  
 for (var i = 0; i < res.Count - 1; i++) // O(n – 1)  
 {  
 var min = i;  
   
 for (var j = i + 1; j < res.Count; j++) // O(n - 1)  
 {  
 if (res[j] < res[min]) // O(n - 1)  
 {  
 min = j;  
 }  
 }  
  
 if (min != i) // O(n - 1)  
 {  
 (res[i], res[min]) = (res[min], res[i]); // O(n - 1)  
 }  
 }  
  
 return res;  
}

Tх(n) = O(n - 1) \* (O(n - 1) + O(n - 1)) + 2O(n - 1) = O(n - 1) \* 2O(n - 1) + 2O(n – 1) = 2O((n – 1)2) + 2O(n – 1) => O(n2) + O(n) => O(n2)