# **Алгоритм сортування вибірки**

# **(Selection Sort Algorithm)**

**Сортування вибором(Selection Sort Algorithm)** - це простий і ефективний алгоритм сортування, який працює шляхом багаторазового вибору найменшого (або найбільшого) елемента з невідсортованої частини списку і переміщення його до відсортованої частини списку. Алгоритм багаторазово вибирає найменший (або найбільший) елемент з невідсортованої частини списку і міняє його місцями з першим елементом невідсортованої частини. Цей процес повторюється для решти не відсортованої частини списку, поки весь список не буде відсортовано. Одна з варіацій сортування вибором називається "Двонаправлене сортування вибором", яке проходить через список елементів, чергуючи найменший і найбільший елементи, таким чином алгоритм може бути швидким у деяких випадках.

Алгоритм підтримує два підмасиви у заданому масиві.

* Підмасив, який вже відсортовано.
* Підмасив, що залишився невідсортованим.

На кожній ітерації сортування вибором вибирається мінімальний елемент (за зростанням) з не відсортованого підмасиву і переміщується на початок не відсортованого підмасиву. Після кожної ітерації розмір відсортованого підмасиву збільшується на одиницю, а розмір не відсортованого підмасиву зменшується на одиницю.

***Як працює вибіркове сортування?***

Розглянемо наступний масив як приклад: arr[] = {64, 25, 12, 22, 11}

***Перший прохід:***

Для першої позиції у відсортованому масиві послідовно обходиться весь масив від індексу 0 до 4. У першій позиції, де зараз зберігається 64, після обходу всього масиву стає зрозуміло, що 11 є найменшим значенням.

| **64** | 25 | 12 | 22 | 11 |
| --- | --- | --- | --- | --- |

Таким чином, замініть 64 на 11. Після однієї ітерації 11, яке є найменшим значенням у масиві, як правило, з'являється на першій позиції відсортованого списку.

| **11** | 25 | 12 | 22 | 64 |
| --- | --- | --- | --- | --- |

***Другий прохід:***

Для другої позиції, де присутнє 25, знову послідовно пройдіться по решті масиву.

| 11 | **25** | 12 | 22 | 64 |
| --- | --- | --- | --- | --- |

Після обходу ми виявили, що 12 є другим найменшим значенням у масиві, і воно повинно з'явитися на другому місці у масиві, тому поміняємо ці значення місцями.

| 11 | **12** | 25 | 22 | 64 |
| --- | --- | --- | --- | --- |

***Третій прохід:***

Тепер, для третього місця, де присутнє 25, знову пройдіться по решті масиву і знайдіть третє найменше значення, присутнє в масиві.

| 11 | 12 | **25** | 22 | 64 |
| --- | --- | --- | --- | --- |

Під час обходу 22 виявилося третім найменшим значенням, і воно повинно бути на третьому місці в масиві, тому поміняємо 22 з елементом, що знаходиться на третій позиції.

| 11 | 12 | **22** | 25 | 64 |
| --- | --- | --- | --- | --- |

***Четвертий прохід:***

Аналогічно, для четвертої позиції пройдіться рештою масиву і знайдіть четвертий найменший елемент у масиві. Оскільки 25 є 4-м найменшим значенням, отже, його буде розміщено на четвертій позиції.

| 11 | 12 | 22 | **25** | 64 |
| --- | --- | --- | --- | --- |

***П'ятий прохід:***

Нарешті, найбільше значення, присутнє в масиві, автоматично потрапляє на останню позицію в масиві. Отриманий масив є відсортованим масивом.

| **11** | **12** | **22** | **25** | **64** |
| --- | --- | --- | --- | --- |

**Щоб розв'язати задачу, виконайте наступні кроки:**

* І***ніціалізуйте мінімальне значення (min\_idx) у позиції 0.***
* ***Пройдіться по масиву, щоб знайти мінімальний елемент у масиві.***
* ***Якщо під час обходу буде знайдено елемент, менший за min\_idx, то поміняйте місцями обидва значення.***
* ***Потім збільште min\_idx, щоб вказати на наступний елемент.***
* ***Повторюйте до тих пір, поки масив не буде відсортовано.***

using System;

class GFG

{

static void sort(int []arr)

{

int n = arr.Length;

// One by one move boundary of unsorted subarray

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

{

// Find the minimum element in unsorted array

int min\_idx = i;

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

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element with the first

// element

int temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

// Prints the array

static void printArray(int []arr)

{

int n = arr.Length;

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

Console.Write(arr[i]+" ");

Console.WriteLine();

}

// Driver code

public static void Main()

{

int []arr = {64,25,12,22,11};

sort(arr);

Console.WriteLine("Sorted array");

printArray(arr);

}

}