

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

**Інститут ІКНІ
Кафедра ПЗ**

ЗВІТ

До лабораторної роботи № 5

З дисципліни: *“Алгоритми та структури даних”*

На тему: *“Метод сортування злиттям”*

Лектор:

доц. каф. ПЗ
Коротєєва Т.О.

Виконав:

ст. гр. ПЗ – 22
Солтисюк Д.А.

Прийняв:

асист. каф. ПЗ
Франко А.В.

« ____ » _____ 2022 р.

Σ = ____ .

Тема роботи: Метод сортування злиттям.

Мета роботи: Вивчити алгоритм сортування злиттям. Здійснити програмну реалізацію алгоритму сортування злиттям. Дослідити швидкодію алгоритму.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Сортування злиттям (англійською «Merge Sort») — алгоритм сортування, в основі якого лежить принцип «розділяй та володарюй». В основі цього способу сортування лежить злиття двох упорядкованих ділянок масиву в одну впорядковану ділянку іншого масиву.

Підчас сортування в дві допоміжні послідовності з основної поміщаються перші дві відсортовані підпослідовності, які потім зливаються в одну і результат записується в тимчасову послідовність. Потім з основної послідовності вибираються наступні дві відсортовані підпослідовності і так до тих пір доки основна послідовність не стане порожньою. Після цього послідовність з тимчасової переміщається в основну. І знову продовжується сортування злиттям двох відсортованих підпослідовностей. Сортування триватиме до тих пір, поки довжина відсортованої підпослідовності не стане рівною довжині самої послідовності.

Сортування злиттям можна задати рекурсивно: масив поділяється на дві приблизно рівні частини, які після сортування (тим самим способом) зливаються. Коли ж довжина масиву зменшується до 1, відбувається повернення з рекурсії.

Час роботи алгоритму $T(n)$ по впорядкуванню n елементів задовільняє рекурентному співвідношенню: $T(n) = 2 \cdot T(\frac{1}{2}n) + O(n)$, де $T(\frac{1}{2}n)$ — час на впорядкування половини масиву, $O(n)$ — час на злиття цих половинок.

Враховуючи, що $T(1) = O(1)$, розв'язком співвідношення є: $T(n) = O(n \cdot \log(n))$.

Крім того, алгоритм потребує для своєї роботи $E(n)$ додаткової пам'яті.

Алгоритм не міняє порядок розташування однакових елементів, а отже він є стабільним.

Алгоритм MergeSort

Дано $X = x_1, \dots, x_n$, ; $Y = y_1, \dots, y_n$, . i - індекс для множини X , j - індекс для множини Y , k – індекс для множини Z , $i = 1 \dots n$; $j = 1 \dots m$; $k = 1 \dots (n+m)$.

1. Ініціалізація індексів $i=1$, $j=1$, $k=1$.
2. Виконувати Merge3 - Merge4 доки $k < (n+m)$.
3. Якщо $x_i < y_j$, то $z_k = x_i$; $i=i+1$, інакше $z_k = y_j$, $j=j+1$.
4. $k=k+1$.
5. Кінець.

ЗАВДАННЯ

Задано одновимірний масив дійсних чисел. Впорядкувати елементи по зростанню.

ХІД РОБОТИ

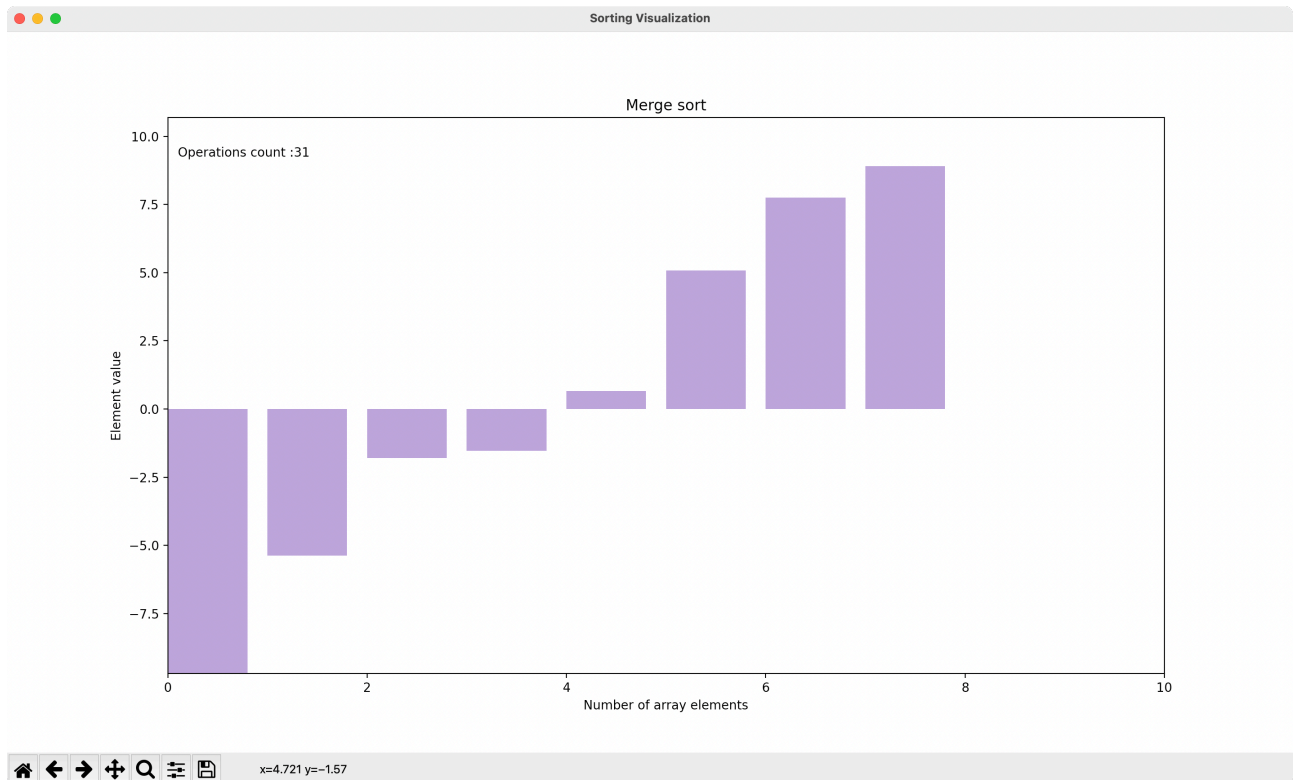
Код функції сортування:

```
def merge_sort(arr, comparator):
    return _merge_sort(arr, 0, len(arr) - 1, comparator)

def _merge_sort(arr, lb, ub, comparator):
    if ub <= lb:
        return
    elif lb < ub:
        mid = (lb + ub) // 2
        yield from _merge_sort(arr, lb, mid, comparator)
        yield from _merge_sort(arr, mid + 1, ub, comparator)
        yield from _merge(arr, lb, mid, ub, comparator)
        yield arr

def _merge(arr, lb, mid, ub, comparator):
    new = []
    i = lb
    j = mid + 1
    while i <= mid and j <= ub:
        if not comparator(arr[i], arr[j]):
            new.append(arr[i])
            i += 1
        else:
            new.append(arr[j])
            j += 1
    if i > mid:
        while j <= ub:
            new.append(arr[j])
            j += 1
    else:
        while i <= mid:
            new.append(arr[i])
            i += 1
    for i, val in enumerate(new):
        arr[lb + i] = val
    yield arr
```

РЕЗУЛЬТАТИ



ВИСНОВКИ

На даній лабораторній роботі я розглядав реалізацію алгоритму сортування злиттям. В результаті виконання лабораторної роботи я навчився використовувати даний алгоритм для сортування масивів. Варто зазначити, що даний алгоритми є стабільним і оптимально ефективним у всіх випадках, але витрачає додаткову пам'ять. Даний алгоритм є основою для багатьох мов програмування.