# Аналіз і узагальнення результатів

* 1. Перевірка коректності результатів

Головною задачею курсової роботи була реалізація програми для впорядкування масивів наступними методами: швидке сортування, сортування злиттям та інтроспективне сортування.

Критичні ситуації у роботі програми виявлені не були. Під час тестування було виявлено, що більшість помилок виникало тоді, коли користувачем вводилися не числові вхідні дані. Тому всі дані, які вводить користувач, ретельно провіряються на валідність і лише потім подаються на обробку програмі.

Для перевірки та доведення достовірності результатів виконання програмного забезпечення було власноруч перевірено елементи вихідного масиву, записаного у файл:

а) Метод швидкого сортування.

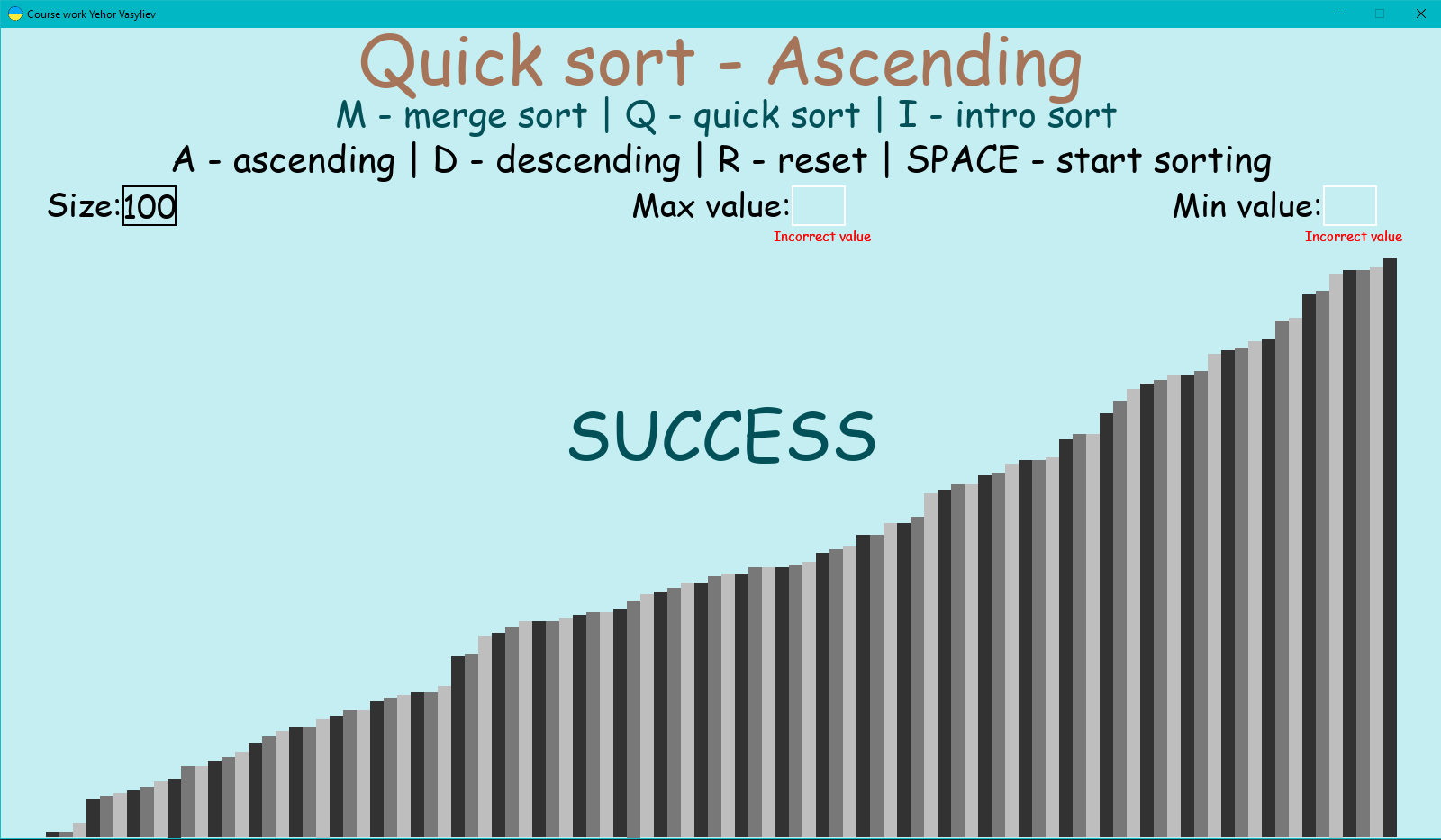
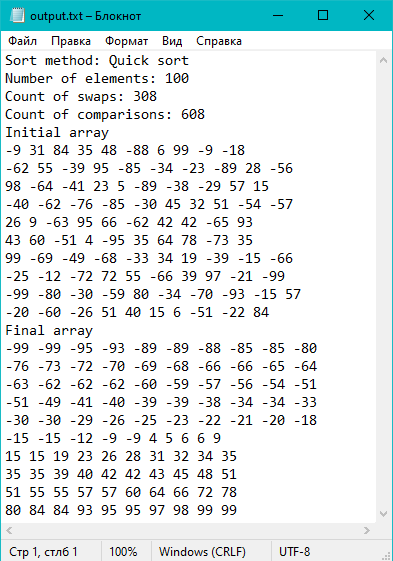
Результат виконання методу швидкого сортування наведено на рисунку 7.1:

Рисунок 7.1 – Результат виконання методу швидкого сортування

Оскільки результуючий масив дійсно відсортований (рисунок 7.2), то даний метод працює вірно.

Рисунок 7.2 – Перевірка впорядкованості масиву після його обробки методом швидкого сортування

б) Метод сортування злиттям.

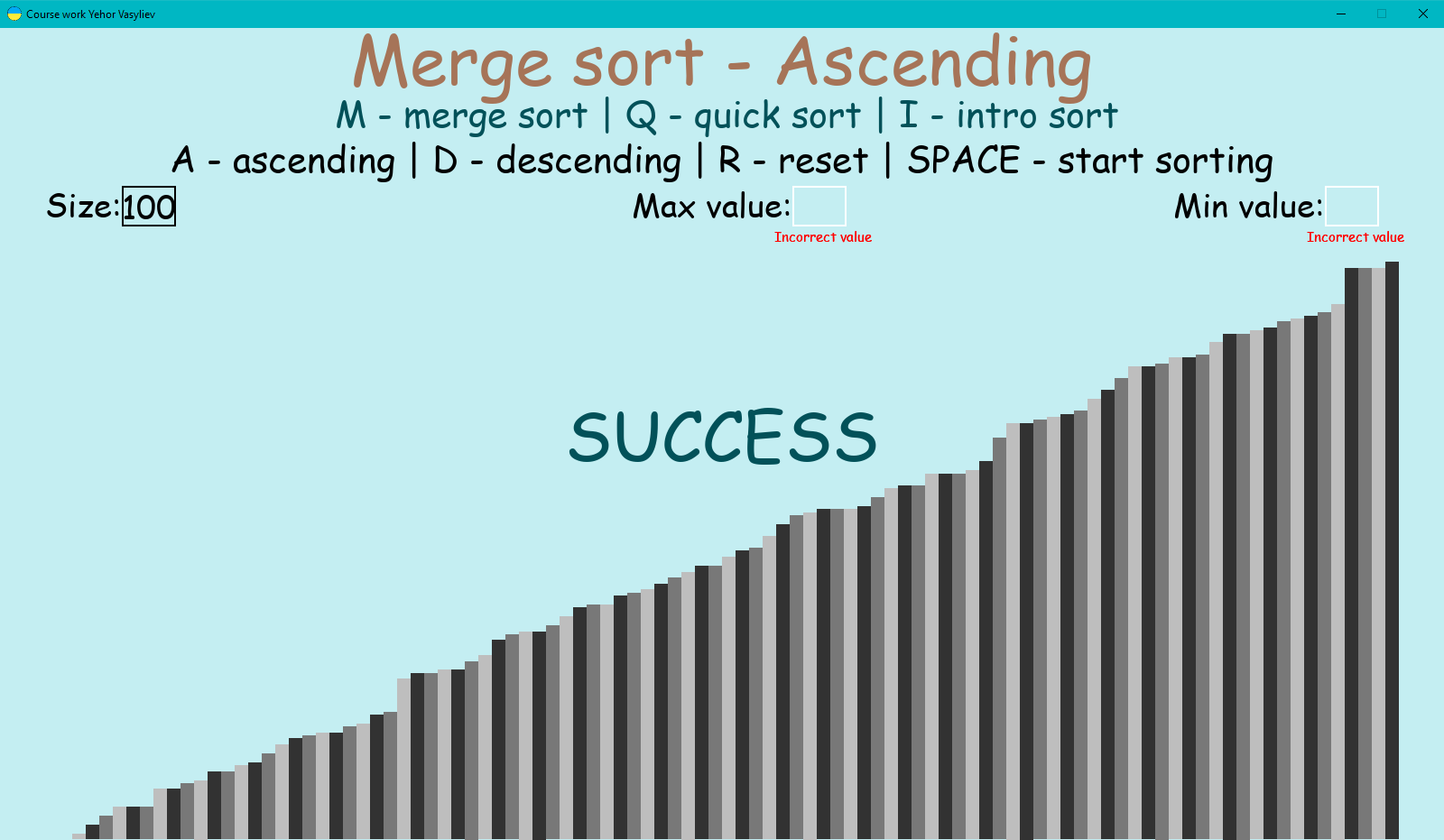
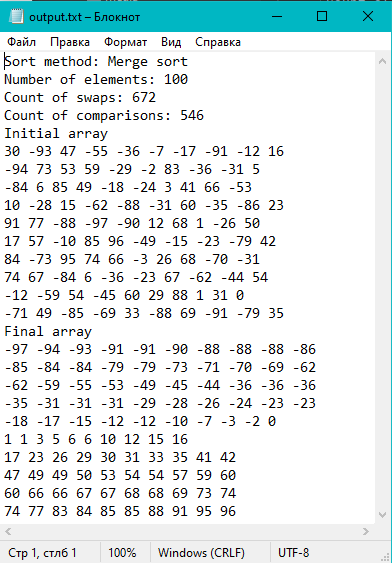
Результат виконання методу сортування злиттям наведено на рисунку 7.3:

Рисунок 7.3 – Результат виконання методу швидкого сортування

Оскільки результуючий масив дійсно відсортований (рисунок 7.4), то даний метод працює вірно.

Рисунок 7.4 – Перевірка впорядкованості масиву після його обробки методом сортування злиттям

в) Метод інтроспективного сортування.

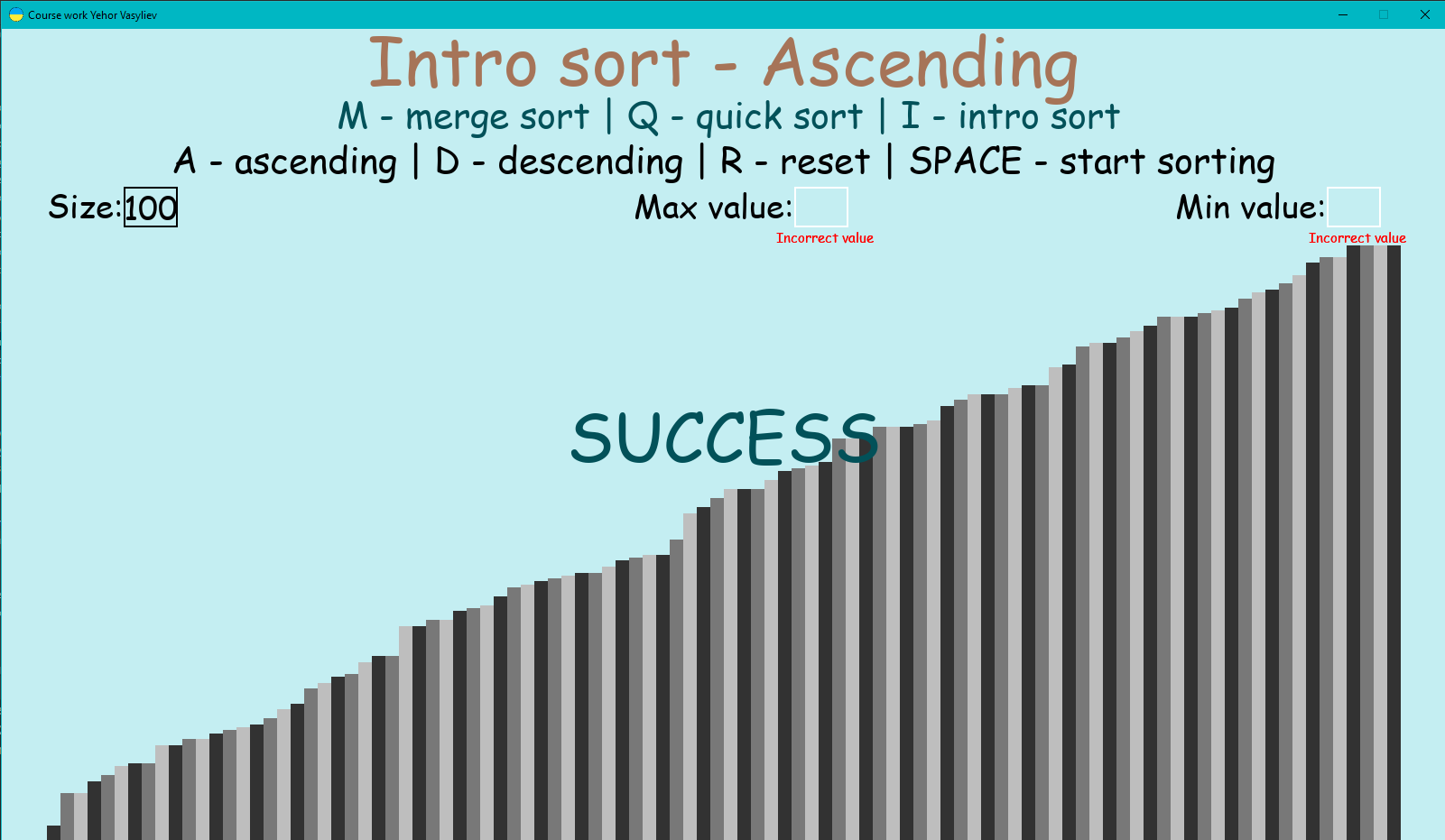
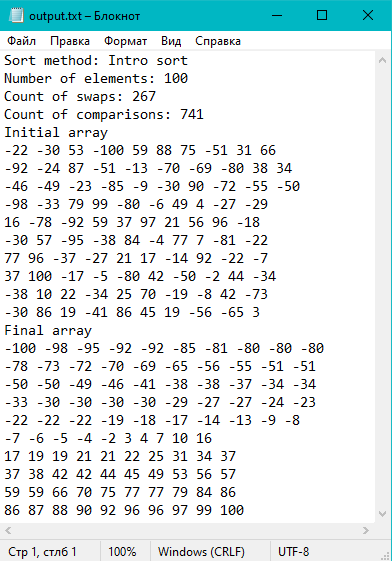
Результат виконання методу інтроспективного сортування наведено на рисунку 7.5: 

Рисунок 7.5 – Результат виконання методу інтроспективного сортування

Оскільки результуючий масив дійсно відсортований (рисунок 7.6), то даний метод працює вірно.

Рисунок 7.6 – Перевірка впорядкованості масиву після його обробки методом інтроспективного сортування

* 1. Тестування ефективності алгоритмів

Для проведення тестування ефективності програми було зроблено 7 сесій сортування масивів різних розмірностей кожним методом з записом необхідних статистичних даних

Результати тестування ефективності алгоритмів сортування масивів наведено в таблиці 7.1:

Таблиця 7.1 – Тестування ефективності методів сортування

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Розмірність масиву | Параметри тестування | Метод сортування | | |
| Швидке | Злиттям | Інтроспек-тивне |
| 100 | Кількість порівнянь | 582 | 542 | 822 |
| Кількість перестановок | 237 | - | 281 |
| 1000 | Кількість порівнянь | 10949 | 8719 | 14530 |
| Кількість перестановок | 5265 | - | 5052 |
| 2500 | Кількість порівнянь | 38300 | 25112 | 33856 |
| Кількість перестановок | 13949 | - | 12078 |
| 5000 | Кількість порівнянь | 110842 | 55102 | 57177 |
| Кількість перестановок | 29747 | - | 22270 |
| 10000 | Кількість порівнянь | 338066 | 120341 | 101136 |
| Кількість перестановок | 49268 | - | 44197 |
| 15000 | Кількість порівнянь | 696842 | 189136 | 171169 |
| Кількість перестановок | 85046 | - | 68812 |

Оскільки сортування злиттям фактично ставить на місце елемента масиву елемент з іншого масиву (його лівої чи правої відсортованої частини) і воно не має перестановок як таких, а цифри у вихідному файлі, що відповідають за відповідне поле позначають скільки разів елемент масиву замінювався іншим, то ці дані не відображають справжню ефективність алгоритму і у таблицю занесені не були.

Візуалізація результатів таблиці 7.1 наведено на рисунку 7.7 та 7.8:

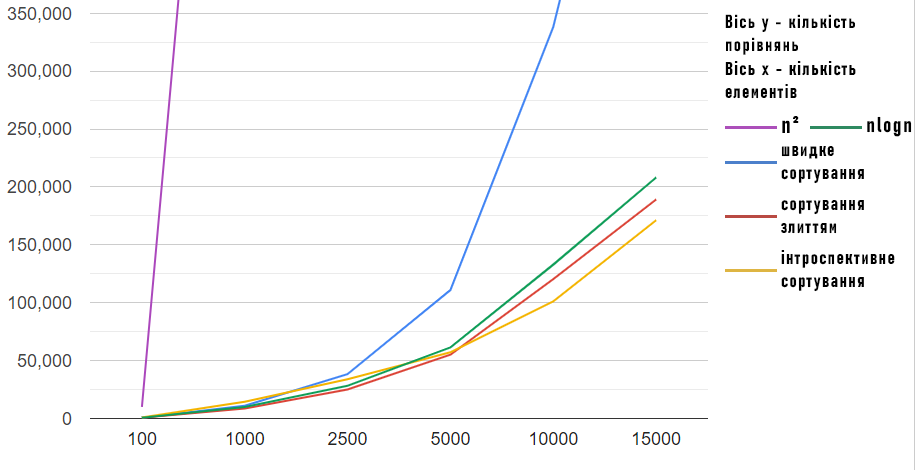
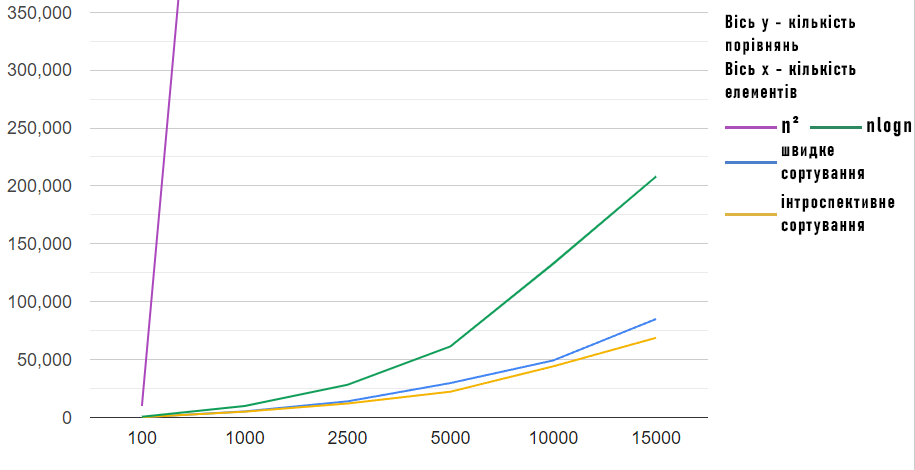
Рисунок 7.7 – Графік залежності кількості порівнянь при сортуванні від розміру   
масиву

Рисунок 7.8 – Графік залежності кількості перестановок при сортуванні від розміру масиву

* 1. Аналіз часової складності

1. Алгоритм швидкого сортування

Кількість операцій при швидкому сортуванні в загальному випадку можна описати наступним чином: T(n) = T(k) + T(n-k-1) + Ꝋ(n), де T(k) відповідає за перший рекурсивний виклик з k елементами, менших за опорний, T(n-k-1) – другий рекурсивний виклик з n-k-1 елементами, більших за опорний. Розберемо три випадки:

* 1. Найгірший випадок відбувається, коли процес розділення завжди обирає найбільший або найменший елемент як опорний. У моїй стратегії, коли перший елемент завжди вибирається як опорний, найгірша ситуація буде, коли масив вже відсортований у порядку зростання або спадання. В такому випадку кількість операцій можна описати як T(n) = T(0) + T(n-1) + Ꝋ(n) = T(n-1) + Ꝋ(n), що еквівалентно часовій складності О() (для кожного і-го рекурсивного виклику виконується константний час і + 1 разів)
  2. Найкращий випадок відбувається, коли процес розділення завжди обирає середній елемент як опорний. В такому випадку часову кількість операцій можна оцінити як T(n) = 2T(n/2) + Ꝋ(n), що еквівалентно часовій складності О() (за Майстер – методом, випадок 2, пункт 2.2)
  3. Щоб провести аналіз середнього випадку необхідно розглянути всі можливі перестановки масиву та обчислити час, затрачений на кожну перестановку, але уявлення про середній випадок ми можемо отримати розглянувши ситуацію, коли масив кожного разу розбивається на О() та О() елементів відповідно, тоді кількість операцій можна оцінити як T(n) = T() + T() + Ꝋ(n), що також еквівалентно часовій складності О()

1. Алгоритм сортування злиттям

Алгоритм сортування злиттям завжди ділить масив на дві половини та займає лінійний час на їх об’єднання, тому кількість операцій можна описати наступним чином: T(n) = 2T(n/2) + Ꝋ(n). З цього випливає, що його часова складність у всіх трьох випадках (найгірший, найкращий, середній) завжди становить О() (за Майстер – методом, випадок 2, пункт 2.2).

1. Алгоритм інтроспективного сортування

Алгоритм інтроспективного сортування по суті представляє собою покращену версію швидкого сортування, яка перемикається на пірамідальне сортування при занадто великій глибині рекурсії (найгірший випадок швидкого сортування). Враховуючи що пірамідальне сортування має постійну часову складність О() (О) для перебудови купи та O(n) для її створення), а середній та найкращий випадок швидкого сортування також мають часові складності О(), то і інтроспективне сортування у всіх трьох випадках має часову складність О().

* 1. Висновки

За результатами тестування можна зробити такі висновки:

а) Всі розглянуті методи дозволяють коректно сортувати великі і над великі масиви.

б) В середньому випадку усі методи мають логарифмічну складність (О)).

в) З розглянутих методів найоптимальнішим для практичного використання є метод інтроспективного сортування, оскільки він має постійну складність О), тому кращий за швидке сортування, а на практиці до того ж кращий за сортування злиттям.

г) Жоден з алгоритмів на практиці (при ініціалізації випадковими числами) не перевищує свою верхню межу часової складності, що видно на графіках (рисунки 7.7, 7.8) однак швидке сортування виявилось найменш ефективним серед розглянутих алгоритмів через неоптимальний вибір опорного елементу.

д) Незважаючи на найкращу ефективність інтроспективного сортування, питання про доцільність його використання для не надвеликих масивів залишається відкритим, оскільки маючи відносно невелику перевагу над сортуванням методом злиття, його реалізація виявляється набагато складнішою.