# Опис алгоритмів

Перелік всіх основних змінних та їхнє призначення наведено в таблиці 3.1.

Таблиця 3.1 – Основні змінні та їхні призначення

|  |  |
| --- | --- |
| Змінна | Призначення |
| n | Розмір масиву (за умовчанням 150 елементів) |
| min\_v | Нижня межа генерації випадкових чисел (за умовчанням -100) |
| max\_v | Верхня межа генерації випадкових чисел (за умовчанням 100) |
| start | Індекс елемента масиву починаючи від якого масив буде сортуватися |
| end | Індекс елемента масиву до якого масив буде сортуватися |
| lst | Ім’я масиву |
| ascending | Тип сортування True – за зростання, False – за спаданням (за умовчанням True) |
| count\_of\_swaps | Кількість перестановок |
| count\_of\_comparisons | Кількість порівнянь |
| max\_depth\_curr | Поточна глибина рекурсії для інтроспективного сортування (за умовчанням ) |

## Загальний алгоритм

1. ПОЧАТОК
2. Зчитати розмір масиву n
   1. Якщо n некоректний, вивести повідомлення та залишити значення за замовченням
3. Зчитати діапазон генерації випадкових чисел min\_v та max\_v
   1. Якщо min\_v або max\_v некоректні, вивести повідомлення та залишити значення за замовченням
4. Зчитати метод сортування.
5. ЯКЩО обраний метод сортування злиттям, ТО обробити дані згідно алгоритму методу сортування злиттям (пункт 3.3)
6. ЯКЩО обраний метод швидкого сортування, ТО обробити дані згідно алгоритму методу швидкого сортування (пункт 3.4)
7. ЯКЩО обраний метод інтроспективного сортування, ТО обробити дані згідно алгоритму методу інтроспективного сортування (пункт 3.5).
8. ЯКЩО розмір масиву менше 200 елементів, ТО створити візуалізацію сортування
9. Вивести повідомлення про успішне сортування
10. Записати назву використаного алгоритму, кількість елементів у ньому, кількість порівнянь й перестановок що відбулись при сортуванні та відсортований масив у файл.
11. КІНЕЦЬ

## 

## Алгоритм сортування методом злиття

1. ПОЧАТОК
2. ЯКЩО end – start > 1
   1. middle = (start + end) // 2
   2. обробити дані згідно з методом сортування злиттям (пункт 3.3) для start = start, end = middle
   3. обробити дані згідно з методом сортування злиттям (пункт 3.3) для start = middle, end = end
   4. left = lst[start, ... middle]
   5. right = lst[middle, … end]
   6. a = 0
   7. b = 0
   8. c = start
   9. ПОКИ a < length(left) та b < length(right)
      1. count\_of\_comparisons += 1
      2. ЯКЩО left[a] < right [b] та ascending == True або left[a] > right[b] та ascending == False:
         1. count\_of\_swaps += 1
         2. намалювати поточний стан масиву
         3. lst[c] = lst[a]
         4. a += 1
      3. ІНАКШЕ:
         1. намалювати поточний стан масиву
         2. count\_of\_swaps += 1
         3. lst[c] = lst[b]
         4. b += 1
   10. ПОКИ a < length(left):
       1. count\_of\_swaps += 1
       2. намалювати поточний стан масиву
       3. lst[c] = lst[a]
       4. a += 1
       5. c +=1
   11. ПОКИ b < length(left):
       1. count\_of\_swaps += 1
       2. намалювати поточний стан масиву
       3. lst[c] = lst[b]
       4. b += 1
       5. c +=1
   12. повернути lst
3. КІНЕЦЬ

## Алгоритм швидкого сортування

## ПОЧАТОК

1. ЯКЩО end == length(lst):
   1. end -= 1
2. ЯКЩО start ≥ end:
   1. повернути
3. x = lst[start]
4. ДЛЯ і від start + 1 до end + 1 з кроком 1:
   1. count\_of\_comparisons += 1
   2. ЯКЩО lst[i] ≤ x та ascending == True або lst[i] ≥ x ascending == False:
      1. j += 1
      2. ЯКЩО j != i:
         1. count\_of\_swaps += 1
         2. lst[j] та lst[j] поміняти місцями
      3. намалювати поточний стан масиву
5. ЯКЩО start != j:
   1. count\_of\_swaps += 1
   2. lst[start] та lst[j] поміняти місцями
6. намалювати поточний стан масиву
7. обробити дані згідно з методом швидкого сортування (пункт 3.4) для start = start, end = j - 1
8. обробити дані згідно з методом швидкого сортування (пункт 3.4) для start = j + 1, end = end
9. КІНЕЦЬ

## Алгоритм інтроспективного сортування

1. ПОЧАТОК
2. ЯКЩО end – start <= 1:
   1. повернути
3. ІНАКШЕ ЯКЩО max\_depth\_curr == 0:
   1. обробити дані згідно з методом пірамідального сортування (пункт 3.7) для start = start, end = end
4. ІНАКШЕ p = partition (start, end) (пункт 3.6)
5. обробити дані згідно з методом інтроспективного сортування (пункт 3.5) для start = start, end = p + 1, max\_depth\_curr = max\_depth\_curr -1
6. обробити дані згідно з методом інтроспективного сортування (пункт 3.5) для start = p + 1, end = end, max\_depth\_curr = max\_depth\_curr -1
7. КІНЕЦЬ

## Алгоритм поділу масиву на частини та їх сортування (частина альтернативного алгоритму швидкого сортування)

1. ПОЧАТОК
2. pivot = lst[start]
3. left = start – 1
4. right = end
5. ПОКИ True:
   1. left += 1
   2. ЯКЩО ascending == True:
      1. ПОКИ lst[left] < pivot:
         1. count\_of\_comparisons += 1
         2. left +=1
   3. ІНАКШЕ:
      1. ПОКИ lst[left] > pivot:
      2. count\_of\_comparisons += 1
      3. left +=1
   4. right -=1
   5. ЯКЩО ascending == True:
      1. ПОКИ lst[right] > pivot:
         1. count\_of\_comparisons += 1
         2. right -= 1
   6. ІНАКШЕ:
      1. ПОКИ lst[right] < pivot:
      2. count\_of\_comparisons += 1
      3. right -=1
   7. ЯКЩО left ≥ right:
      1. повернути right
   8. count\_of\_swaps += 1
   9. намалювати поточний стан масиву
   10. lst[left] та lst[right] поміняти місцями
6. КІНЕЦЬ

## Алгоритм пірамідального сортування

1. ПОЧАТОК
2. обробити дані згідно з алгоритмом створення купи для start = start, end == end (пункт 3.8)
3. ДЛЯ i від end – 1 до start з кроком -1:
   1. count\_of\_swaps += 1
   2. намалювати поточний стан масиву
   3. lst[start] та lst[i] поміняти місцями
   4. ЯКЩО ascending == True:
      1. обробити дані згідно з алгоритмом створення максимальної купи (пункт 3.9) для i = 0, start = start, end = i
   5. ІНАКШЕ:
      1. обробити дані згідно з алгоритмом створення мінімальної купи (пункт 3.10) для i = 0, start = start, end = i
4. КІНЕЦЬ

## Алгоритм створення купи

1. ПОЧАТОК
2. length = end - start
3. index = ((length – 1) – 1) // 2
4. ПОКИ index ≥ 0
   1. ЯКЩО ascending == True:
      1. обробити дані згідно з алгоритмом створення максимальної купи (пункт 3.9) для i = index, start = start, end = end
   2. ІНАКШЕ:
      1. обробити дані згідно з алгоритмом створення мінімальної купи (пункт 3.10) для i = index, start = start, end = end
   3. index -= 1
5. КІНЕЦЬ

## Алгоритм створення максимальної купи

1. ПОЧАТОК
2. size = end - start
3. l = 2 \* i + 1
4. r = 2 \* i + 2
5. largest = i
6. count\_of\_comparisons += 2
7. ЯКЩО l < size і lst[start + l] > lst [start + i]:
   1. largest = l
8. ЯКЩО r < size і lst[start + r ] > lst [start + largest]:
   1. largest = r
9. ЯКЩО largest != i:
   1. count\_of\_swaps += 1
   2. lst[start + largest] та lst[start + i] поміняти місцями
   3. намалювати поточний стан масиву
   4. обробити дані згідно з алгоритмом створення максимальної купи (пункт 3.9) для i = largest, start = start, end = end
10. КІНЕЦЬ

## Алгоритм створення мінімальної купи

1. ПОЧАТОК
2. size = end - start
3. l = 2 \* i + 1
4. r = 2 \* i + 2
5. smallest = i
6. count\_of\_comparisons += 2
7. ЯКЩО l < size і lst[start + l] > lst [start + i]:
   1. smallest = l
8. ЯКЩО r < size і lst[start + r] > lst [start + smallest]:
   1. smallest = r
9. ЯКЩО smallest != i:
   1. count\_of\_swaps += 1
   2. намалювати поточний стан масиву
   3. lst[start + largest] та lst[start + i] поміняти місцями
   4. обробити дані згідно з алгоритмом створення мінімальної купи (пункт 3.9) для i = smallest, start = start, end = end
10. КІНЕЦЬ