ФГБОУ ВО “Чувашский государственный университет им.

И.Н. Ульянова”

Кафедра вычислительной техники

Лабораторная работа №5

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 28.

Выполнил:

Студент гр. ИВТ-41-22

Ярдыков Эдуард Евгеньевич

Проверил:

Павлов Леонид Александрович

Чебоксары, 2024

Изучить основные методы сортировки, получить практические навыки программирования задач сортировки, получить навыки экспериментальных исследований алгоритмов.

Подготовка к работе:

**Реализуемые в моем варианте алгоритмы:**

1. Сортировка вставками в связанный список.

2. Сортировка Шелла.

3. Быстрая сортировка (рекурсивный вариант).

4. Быстрая сортировка (итерационный вариант).

5. Цифровая обменная сортировка.

6. Пирамидальная сортировка.

7. Сортировка подсчетом (перечислением).

8. Сортировка простым двухпутевым слиянием.

9. Сортировка слиянием списков для простого двухпутевого слияния.

Алгоритмы на псевдокоде для каждого метода упорядочивания:

**1. Сортировка вставками в связанный список.**

unction insertionSortLinkedList(head):

if head is null or head.next is null:

return head

sorted = null

current = head

while current is not null:

nextNode = current.next

if sorted is null or sorted.data >= current.data:

current.next = sorted

sorted = current

else:

traverse = sorted

while traverse.next is not null and traverse.next.data < current.data:

traverse = traverse.next

current.next = traverse.next

traverse.next = current

current = nextNode

return sorted

**2.** Сортировка Шелла.

function shellSort(array):

n = length(array)

gap = n / 2

while gap > 0:

for i from gap to n-1:

temp = array[i]

j = i

while j >= gap and array[j - gap] > temp:

array[j] = array[j - gap]

j = j - gap

array[j] = temp

gap = gap / 2

**3,4.** Быстрая сортировка

function quickSortRecursive(array, low, high):

if low < high:

pivot = partition(array, low, high)

quickSortRecursive(array, low, pivot - 1)

quickSortRecursive(array, pivot + 1, high)

function quickSortIterative(array, low, high):

// Создаем стек для хранения пар low и high

stack = createStack()

push(stack, low)

push(stack, high)

// Пока стек не пуст

while !isEmpty(stack):

high = pop(stack)

low = pop(stack)

pivot = partition(array, low, high)

// Если есть элементы слева от опорного

if pivot - 1 > low:

push(stack, low)

push(stack, pivot - 1)

// Если есть элементы справа от опорного

if pivot + 1 < high:

push(stack, pivot + 1)

push(stack, high)

5. Цифровая обменная сортировка.

function radixSort(array):

maxNumber = getMax(array)

exp = 1

n = length(array)

while maxNumber / exp > 0:

countingSort(array, n, exp)

exp = exp \* 10

function countingSort(array, n, exp):

output = new array[n]

count = new array[10]

for i from 0 to 9:

count[i] = 0

for i from 0 to n-1:

index = (array[i] / exp) % 10

count[index] = count[index] + 1

for i from 1 to 9:

count[i] = count[i] + count[i - 1]

for i from n-1 down to 0:

index = (array[i] / exp) % 10

output[count[index] - 1] = array[i]

count[index] = count[index] - 1

for i from 0 to n-1:

array[i] = output[i]

6. Пирамидальная сортировка.

function heapSort(array):

n = length(array)

// Строим кучу (heapify)

for i from n/2 - 1 down to 0:

heapify(array, n, i)

// Поочередно извлекаем элементы из кучи

for i from n-1 down to 0:

swap(array[0], array[i])

heapify(array, i, 0)

function heapify(array, n, i):

largest = i

left = 2\*i + 1

right = 2\*i + 2

if left < n and array[left] > array[largest]:

largest = left

if right < n and array[right] > array[largest]:

largest = right

if largest != i:

swap(array[i], array[largest])

heapify(array, n, largest)

7. Сортировка подсчетом (перечислением).

function countingSort(array):

n = length(array)

maxVal = getMaxValue(array)

count = new array[maxVal + 1]

output = new array[n]

for i from 0 to maxVal:

count[i] = 0

for i from 0 to n-1:

count[array[i]] = count[array[i]] + 1

for i from 1 to maxVal:

count[i] = count[i] + count[i - 1]

for i from n-1 down to 0:

output[count[array[i]] - 1] = array[i]

count[array[i]] = count[array[i]] - 1

for i from 0 to n-1:

array[i] = output[i]

8. Сортировка простым двухпутевым слиянием.

function simpleTwoWayMergeSort(array):

if length(array) <= 1:

return array

else:

mid = length(array) / 2

left = array[0...mid-1]

right = array[mid...end]

// Рекурсивно сортируем и объединяем две половины

left = simpleTwoWayMergeSort(left)

right = simpleTwoWayMergeSort(right)

return merge(left, right)

9. Сортировка слиянием списков для простого двухпутевого слияния.

function merge(left, right):

result = []

while left is not empty and right is not empty:

if left[0] <= right[0]:

result.push(left[0])

left = left[1...end]

else:

result.push(right[0])

right = right[1...end]

while left is not empty:

result.push(left[0])

left = left[1...end]

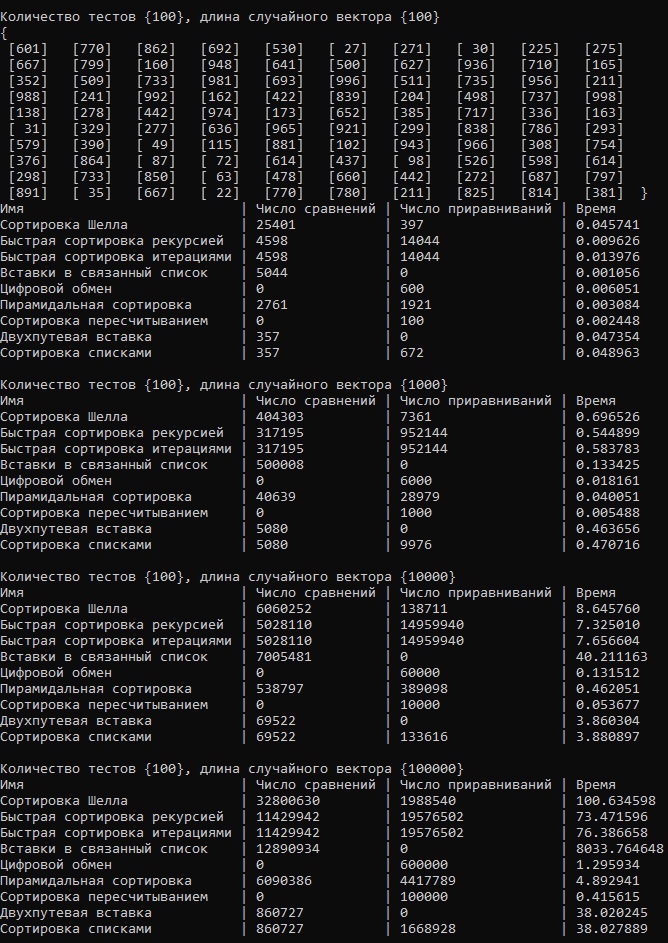
while right is not empty:

result.push(right[0])

right = right[1...end]

return result

Скриншоты результатов выполнения программы:



Вывод: Изучил основные методы сортировки, получил практические навыки программирования задач сортировки, получил навыки экспериментальных исследований алгоритмов.