Отчет о выполнении задания 1  
«Алгоритмы сортировки»

**Чуклин Алексей Сергеевич**

Группа ИВ-521

cas98@bk.ru

В задании 1 требовалось реализовать и исследовать эффективность трех алгоритмов сортировки: Counting Sort, Bubble Sort и Heap Sort. Все алгоритмы реализованы на языке C под операционной системой GNU/Linux.

# Алгоритм Counting Sort

Алгоритм Counting Sort (сортировка подсчетом) — алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов.

В работе реализована устойчивая версия алгоритма с вычислительной сложностью в худшем случае *O*(*n* + *k*), где *n* — это количество элементов, а *k* — это .... Сложность по памяти O(n + k).

Псевдокод устойчивой версии алгоритма CountingSort:

function CountingSort(A, n, k)

for i = 0 to k — 1 do

C[i] = 0;

for i = 0 to n — 1 do

C[A[i]] = C[A[i]] + 1;

for j = 1 to k — 1 do

C[j] = C[j] + C[j - 1];

for i = n - 1 to 0 do

C[A[i]] = C[A[i]] - 1;

B[C[A[i]]] = A[i];

end function

# Алгоритм Bubble Sort

Алгоритм Bubble Sort (сортировка пузырьком) – алгоритм сортировки, который состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно и, если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются N-1 раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает — массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива .

Вычислительная сложность в худшем случае O(), где n – это количество элементов. Сложность по памяти O(1).

Псевдокод алгоритма:

function BubbleSort(v[0:n - 1], n)

swapped = True

while swapped do

swapped = False

for i = 1 to n - 1 do

if v[i - 1] > v[i] then

swap(v[i - 1], v[i])

swapped = True

end if

end for

end while

end function

# Алгоритм Heap Sort

Алгоритм Heap Sort (пирамидальная сортировка) -  [алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), работающий в худшем, в среднем и в лучшем случае (то есть гарантированно) за [*Θ*](https://ru.wikipedia.org/wiki/%D0%9E-%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F)*(n* log *n)* операций при сортировке *n* элементов. Количество применяемой служебной памяти не зависит от размера массива (то есть, O(1)).

Псевдокод алгоритма:

function heapsort(a, count)

heapify(a, count)

end := count – 1

while end > 0 do

swap (a[end], a[0])

end := end – 1

siftDown(a, 0, end)

function heapify(a, count)

start := (count - 2) / 2

while start ≥ 0 do

siftDown(a, start, count-1)

start := start - 1

function siftDown(a, start, end)

root := start

while root \* 2 + 1 ≤ end do

child := root \* 2 + 1

if child + 1 ≤ end and a[child] < a[child + 1] then

child := child + 1

if a[root] < a[child] then

swap(a[root], a[child])

root := child

else

return

## Организация экспериментов

## Оборудование

CPU: Intel Core i5 1.7 GHz

RAM: 4Gb DDR3 L

OS: Linux Mint 17.3

GCC: 4.8.4

Параметры компилятора

-O