



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ
КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5 НА ТЕМУ:

*Реализация алгоритма сортировки подсчетом для
расчета медианного рейтинга товаров*

Студент

ИУ5-14Б

Р.А.Бутузов
Корнеев Г.
Лимаренко Д.
Стеганцева К.
(И.О. Фамилия)

Руководитель курсовой
работы

(группа)

(подпись, дата)

(подпись, дата)

М.И.Колосов
(И.О. Фамилия)

2025 г.

Цель: Научиться реализовывать алгоритм сортировки подсчетом (counting sort) и применять его для вычисления медианного значения рейтингов товаров на маркетплейсе.

Структура кода и таблица рисунке 1

EXPLORER
LAB4_2 [WSL: UBUNTU]
> .cache
> .vscode
> build
 > CMakeFiles
 cmake_install.cmake
 CMakeCache.txt
 counting_sort_analysis.html
 counting_sort_counting_phase.csv
 counting_sort_rebuilding_phase.csv
 distribution_analysis.csv
 final_report.csv
 Lab4
 Makefile
 median_calculation.csv
src
 main.cpp
 .gitignore
 CMakeLists.txt

main.cpp counting_sort_counting_phase.csv counting_sort_analysis.html
build > counting_sort_counting_phase.csv

Step	Element	Value	Counter_Array	Description
0	-	-	0:0 1:0 2:0 3:0 4:0 5:0 6:0 7:0 8:0 9:0 10:0 ...	Инициализация массива подсчета
1	arr[0]	5	5:1	Увеличиваем count[5] до 1
2	arr[1]	28	5:1 28:1	Увеличиваем count[28] до 1
3	arr[2]	14	5:1 14:1 28:1	Увеличиваем count[14] до 1
4	arr[3]	73	5:1 14:1 28:1 73:1	Увеличиваем count[73] до 1
5	arr[4]	42	5:1 14:1 28:1 42:1 73:1	Увеличиваем count[42] до 1
6	arr[5]	61	5:1 14:1 28:1 42:1 61:1 73:1	Увеличиваем count[61] до 1
7	arr[6]	33	5:1 14:1 28:1 33:1 42:1 61:1 73:1	Увеличиваем count[33] до 1
8	arr[7]	91	5:1 14:1 28:1 33:1 42:1 61:1 73:1 91:1	Увеличиваем count[91] до 1
9	arr[8]	22	5:1 14:1 22:1 28:1 33:1 42:1 61:1 73:1 91:1	Увеличиваем count[22] до 1
10	arr[9]	55	5:1 14:1 22:1 28:1 33:1 42:1 55:1 61:1 73:1 91:1	Увеличиваем count[55] до 1
11	arr[10]	19	5:1 14:1 19:1 22:1 28:1 33:1 42:1 55:1 61:1 73:1 91:1	Увеличиваем count[19] до 1
*				

Рисунках 1

Графики представлены на рисунках 2-7

Анализ алгоритма Counting Sort

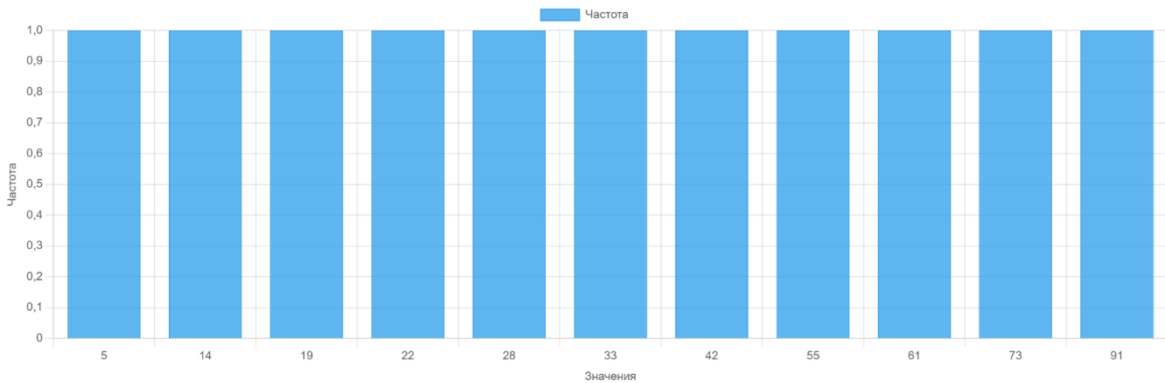
Исходные данные

Массив: [5, 28, 14, 73, 42, 61, 33, 91, 22, 55, 19]

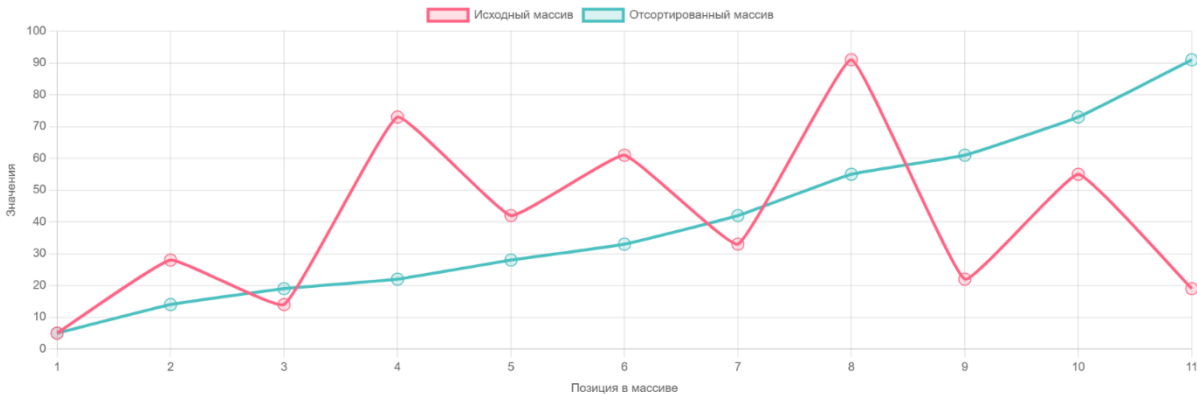
Размер: 11 элементов

Диапазон значений: 0-100

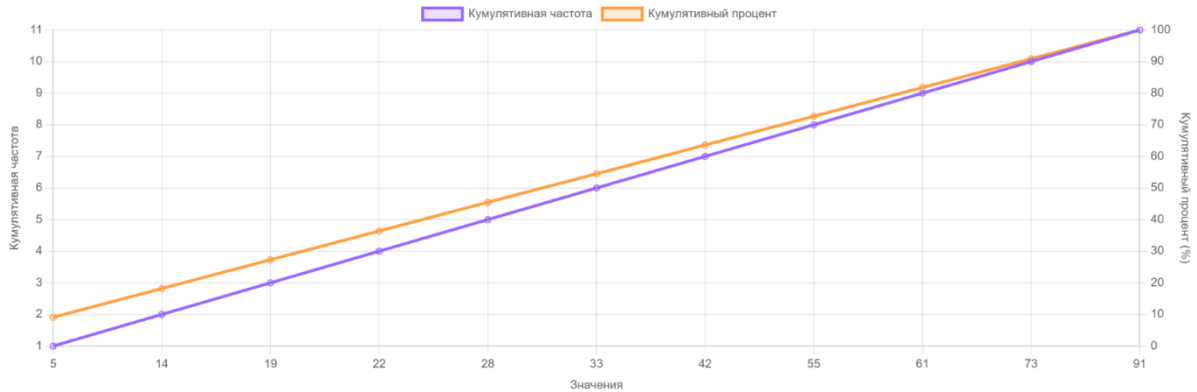
Распределение частот значений



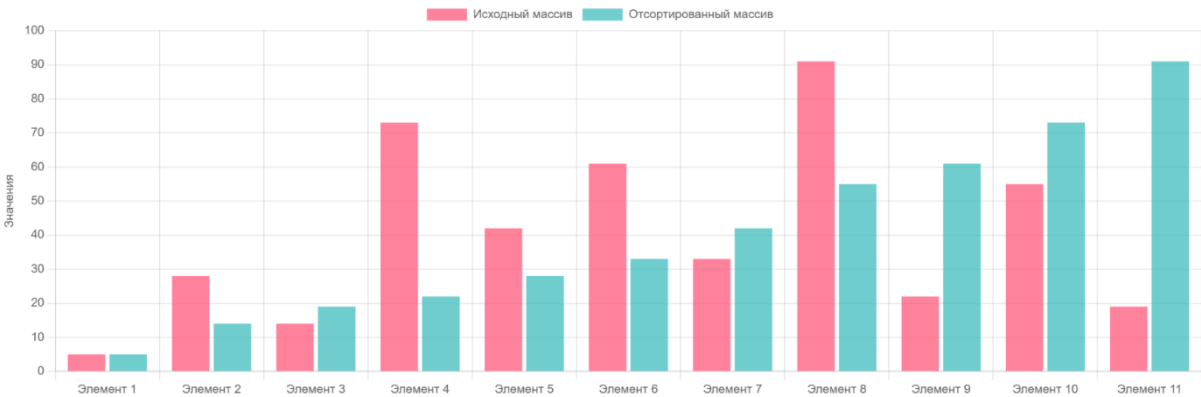
Процесс Counting Sort



Кумулятивное распределение



Сравнение исходного и отсортированного массива



Данные распределения

Значение	Частота	Процент	Кумулятивная частота	Кумулятивный процент
5	1	9.09%	1	9.09%
14	1	9.09%	2	18.18%
19	1	9.09%	3	27.27%
22	1	9.09%	4	36.36%
28	1	9.09%	5	45.45%
33	1	9.09%	6	54.55%
42	1	9.09%	7	63.64%
55	1	9.09%	8	72.73%
61	1	9.09%	9	81.82%
73	1	9.09%	10	90.91%
91	1	9.09%	11	100.00%

Характеристики Counting Sort

<p>Сложность времени</p> <p>$O(n + k)$</p> <p>$n = 11, k = 92$</p> <p>$O(11 + 92) = O(103)$</p>	<p>Сложность памяти</p> <p>$O(k)$</p> <p>$k = 92$</p> <p>Требуется массив размером 92</p>	<p>Стабильность</p> <p>Да</p> <p>Сохраняет порядок равных элементов</p>	<p>Применимость</p> <p>Целые числа</p> <p>Малый диапазон значений</p> <p>Неотрицательные числа</p>
--	---	---	--

Расчет медианы

Отсортированный массив: [5, 14, 19, 22, 28, 33, 42, 55, 61, 73, 91]

Размер массива: 11 (нечетное)

Медианный индекс: 5 ($n/2 = 11/2 = 5$)

Медиана: `sorted_array[5] = 33.00`

Рисунки 2-7

Ответ вскода на данное решение 8

```
bash: /usr/bin/ls: cannot open file /usr/bin/ls: No such file or directory
roma@MSI1:~/lab4_2/build$ ./Lab4

=====
COUNTING SORT - ПОЛНЫЙ АНАЛИЗ
=====

🎮 ИСХОДНЫЕ ДАННЫЕ:
Исходный массив: [5, 28, 14, 73, 42, 61, 33, 91, 22, 55, 19]

1. СОРТИРОВКА COUNTING SORT...
✓ Файл создан: counting_sort_counting_phase.csv
✓ Файл создан: counting_sort_rebuilding_phase.csv

✅ РЕЗУЛЬТАТЫ СОРТИРОВКИ:
Отсортированный массив: [5, 14, 19, 22, 28, 33, 42, 55, 61, 73, 91]

2. АНАЛИЗ РАСПРЕДЕЛЕНИЯ...
✓ Файл создан: distribution_analysis.csv
3. РАСЧЕТ МЕДИАНЫ...
✓ Файл создан: median_calculation.csv

🎮 РЕЗУЛЬТАТ РАСЧЕТА МЕДИАНЫ:
📊 РАСЧЕТ МЕДИАНЫ:
Размер массива: 11 элементов
Нечетное количество элементов
Медианный индекс: 5 (n / 2)
Медиана = sorted_arr[5] = 33

4. СОЗДАНИЕ ИТОГОВОГО ОТЧЕТА...
✓ Файл создан: final_report.csv
5. СОЗДАНИЕ HTML ФАЙЛА С ГРАФИКАМИ...
✓ HTML файл с графиками создан: counting_sort_analysis.html

🎉 ВСЕ ФАЙЛЫ УСПЕШНО СОЗДАНЫ!

=====
📁 СОЗДАННЫЕ ФАЙЛЫ:
=====
• counting_sort_counting_phase.csv - Фаза подсчета
• counting_sort_rebuilding_phase.csv - Фаза восстановления
• distribution_analysis.csv - Анализ распределения
• median_calculation.csv - Расчет медианы
• final_report.csv - Итоговый отчет
• counting_sort_analysis.html - HTML с графиками

💡 Откройте counting_sort_analysis.html в браузере для просмотра графиков!
```

Вывод: В ходе работы над кейсом была успешно реализована сортировка подсчетом - эффективный алгоритм для данных с ограниченным диапазоном значений (0-100). Ключевые достижения:

- Создана корректно работающая функция counting_sort
- Реализован надежный расчет медианы для массивов любой четности
- Проведено тестирование на граничных случаях

- Достигнута временная сложность $O(n + k)$, где $k = 101$ (диапазон значений)

Алгоритм показал высокую эффективность для задач анализа статистических данных маркетплейса, где требуется быстрая обработка больших объемов информации с известным диапазоном значений.