ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Отчет по лабораторной работе №1

По дисциплине

Операционные системы

Выполнила: Султанова Эвелина Марселевна

Группа: P3307

Преподаватель:

г. Санкт-Петербург

2025

Задание

Реализовать собственную оболочку командной строки – shell для операционной системы Windows. Shell должен предоставлять пользователю возможность запускать программы на компьютере с переданными аргументами командной строки и после завершения программы показывать реальное время ее работы (подсчитать самостоятельно как «время завершения» – «время запуска»).

В оболочке поддерживаются команды:

- Exit
- Cd
- Dedup
- Search
- DedupAndSearch
- Запуск исполняемых файлов

Разработать комплекс программ-нагрузчиков по варианту, заданному преподавателем. Каждый нагрузчик должен, как минимум, принимать параметр, который определяет количество повторений для алгоритма, указанного в задании. Программы должны нагружать вычислительную систему, дисковую подсистему или обе подсистемы сразу. Необходимо скомпилировать их без опций оптимизации компилятора.

Программы – нагрузчики:

- Search-name – поиск файла в файловой системе по имени
- Dedup – дедупликация элементов в массиве

Программа (комплекс программ) должна быть реализован на языке C, C++. Дочерние процессы должны быть созданы через заданные системные вызовы выбранной операционной системы, с обеспечением корректного запуска и завершения процессов. Запрещено использовать высокоуровневые абстракции над системными вызовами.

1. Оценка времени работы нагрузчиков

Для команды dedup:

Команда обходит файл размером 321,8 Мб и записывает уникальные элементы в отдельный файл. Количество элементов в файле - 2^{28}. Частота процессора – 2,1 ГГц, среднее количество тактов на выполнение одной операции – 50. Скорость чтения/записи – 200Мб/с. Размер выходного файла – 1 Кб.

$2{,}1 \cdot 10^9$ – количество тактов в секунду

$\frac{50}{2{,}1 \cdot 10^9}$ - время выполнения одной операции

$2^{28} \cdot \frac{50}{2{,}1 \cdot 10^9}$= 6,39132 с– время выполнения операций обработки данных

$\frac{321{,}8\text{Мб}}{200\text{Мб}/c} = 1{,}609\text{с}$ – время чтения

$\frac{1\text{Кб}}{200\text{Мб}/c} = 0{,}00000488\text{с}$ – время чтения

Получается, примерное время выполнения 1 итерации – 8с.

Для команды searchFile C:\\Users\\79124\\Documents\\ElinaComputer Project.gsproj:

Количество файлов в корневой директории: ~72.

Время доступа: 0,2 мс.

Время сравнения названий файлов: 0,001 мс.

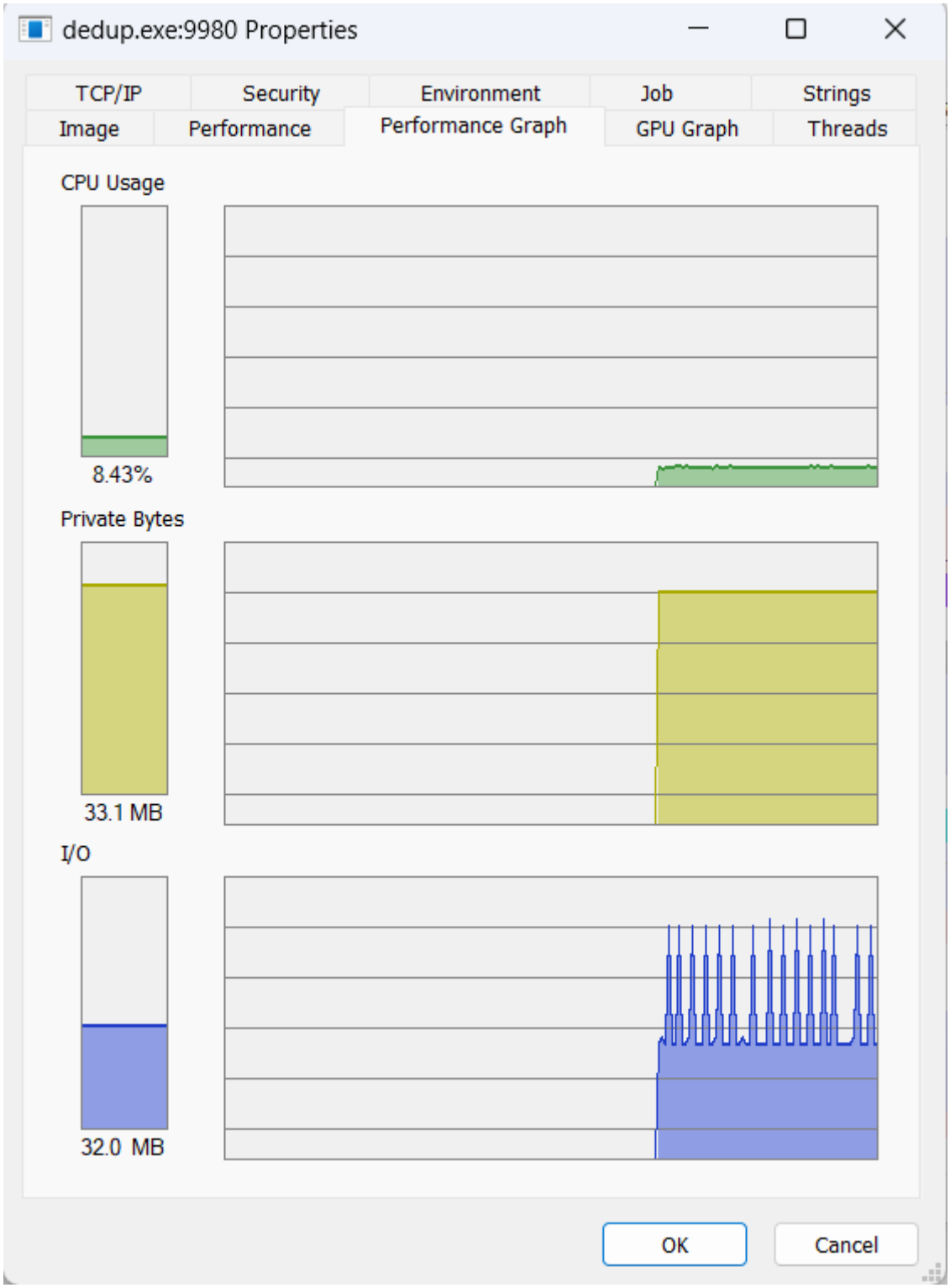
$72 \cdot 0{,}2\text{ мс} + 72 \cdot 0{,}001\text{мс} = 14{,}472\text{ мс}$

2. Профилирование

Dedup

10 повторений в 1 потоке

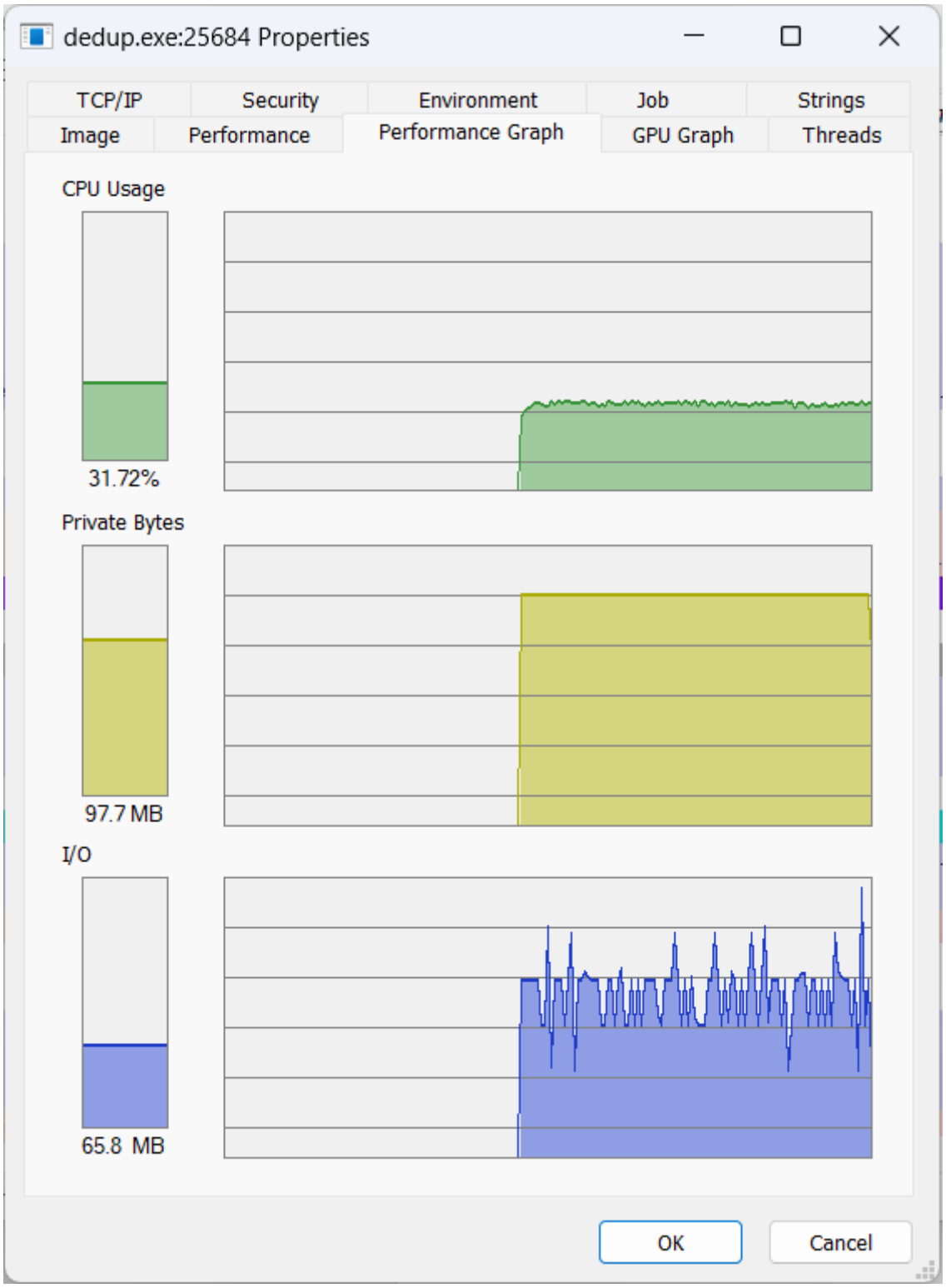
```
.>dedup.exe random_numbers256.txt dedup_numbers.txt 1 1
Дедупликация завершена (итерация 1)
Общее время выполнения программы: 10753 мс
Среднее время выполнения итерации: 10753 мс
Process was active for 11103 мс
.>dedup.exe random_numbers256.txt dedup_numbers.txt 10 1
Дедупликация завершена (итерация 1)
Дедупликация завершена (итерация 2)
Дедупликация завершена (итерация 3)
Дедупликация завершена (итерация 4)
Дедупликация завершена (итерация 5)
Дедупликация завершена (итерация 6)
Дедупликация завершена (итерация 7)
Дедупликация завершена (итерация 8)
Дедупликация завершена (итерация 9)
Дедупликация завершена (итерация 10)
Общее время выполнения программы: 88684 мс
Среднее время выполнения итерации: 8868 мс
Process was active for 88719 мс
.>
```



Программа использует 32Мб данных из SSD, так как читает данные из файла блоками по 32Мб, как и должно быть. Данные находятся в оперативной памяти. Программа постоянно использует около 5,91% ресурсов процессора. Время выполнение программы близко к предсказанном.

10 повторений 4 потока

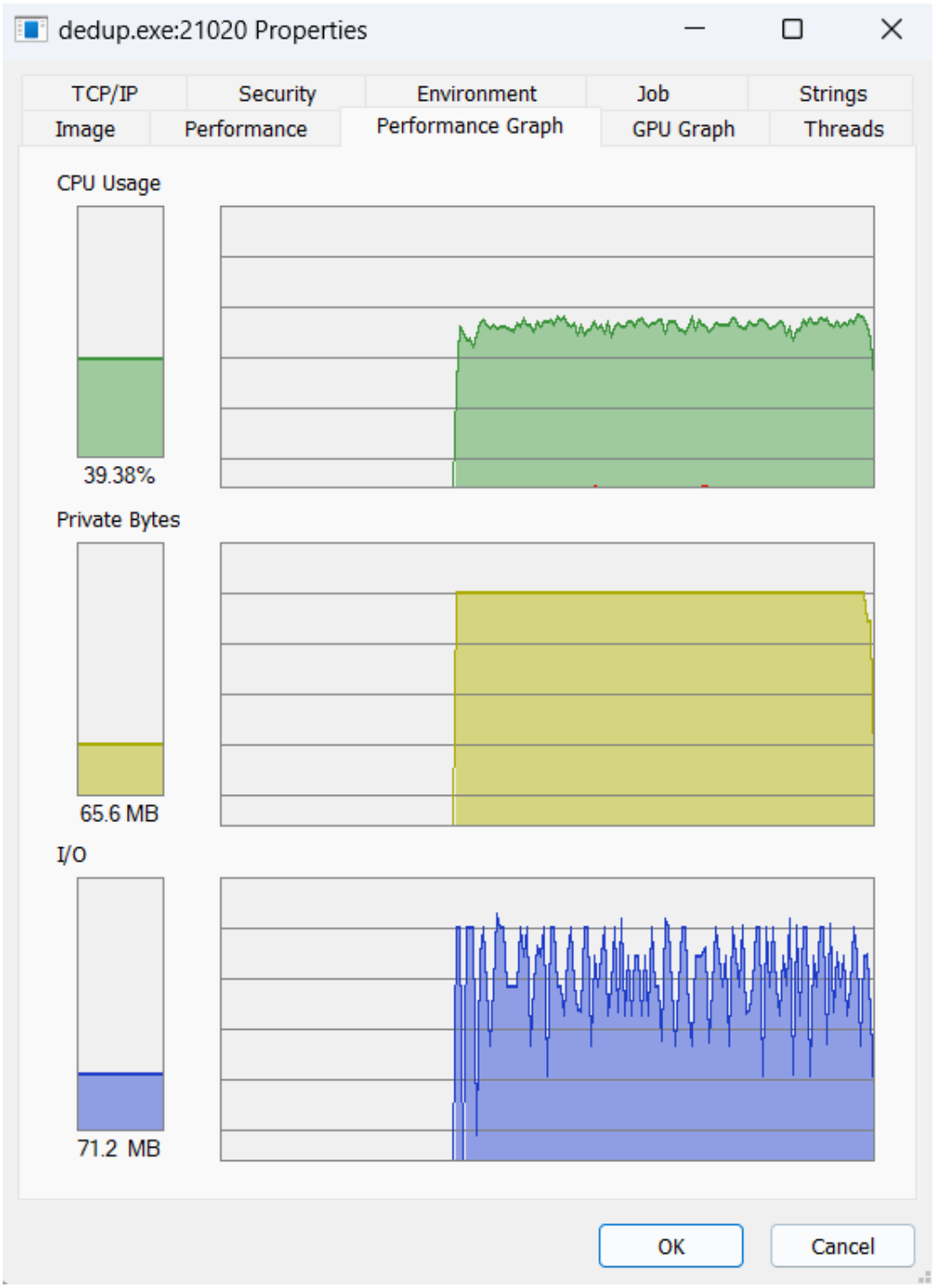
```
.>dedup.exe random_numbers256.txt dedup_numbers.txt 10 4
Общее время выполнения программы: 114689 мс
Среднее время выполнения итерации: 11395 мс
Process was active for 114736 мс
.>
```



Как мы видим, программа использует в два раза больше памяти для чтения блоков, поскольку одновременно выполняется в нескольких потоках.

10 повторений 8 потоков

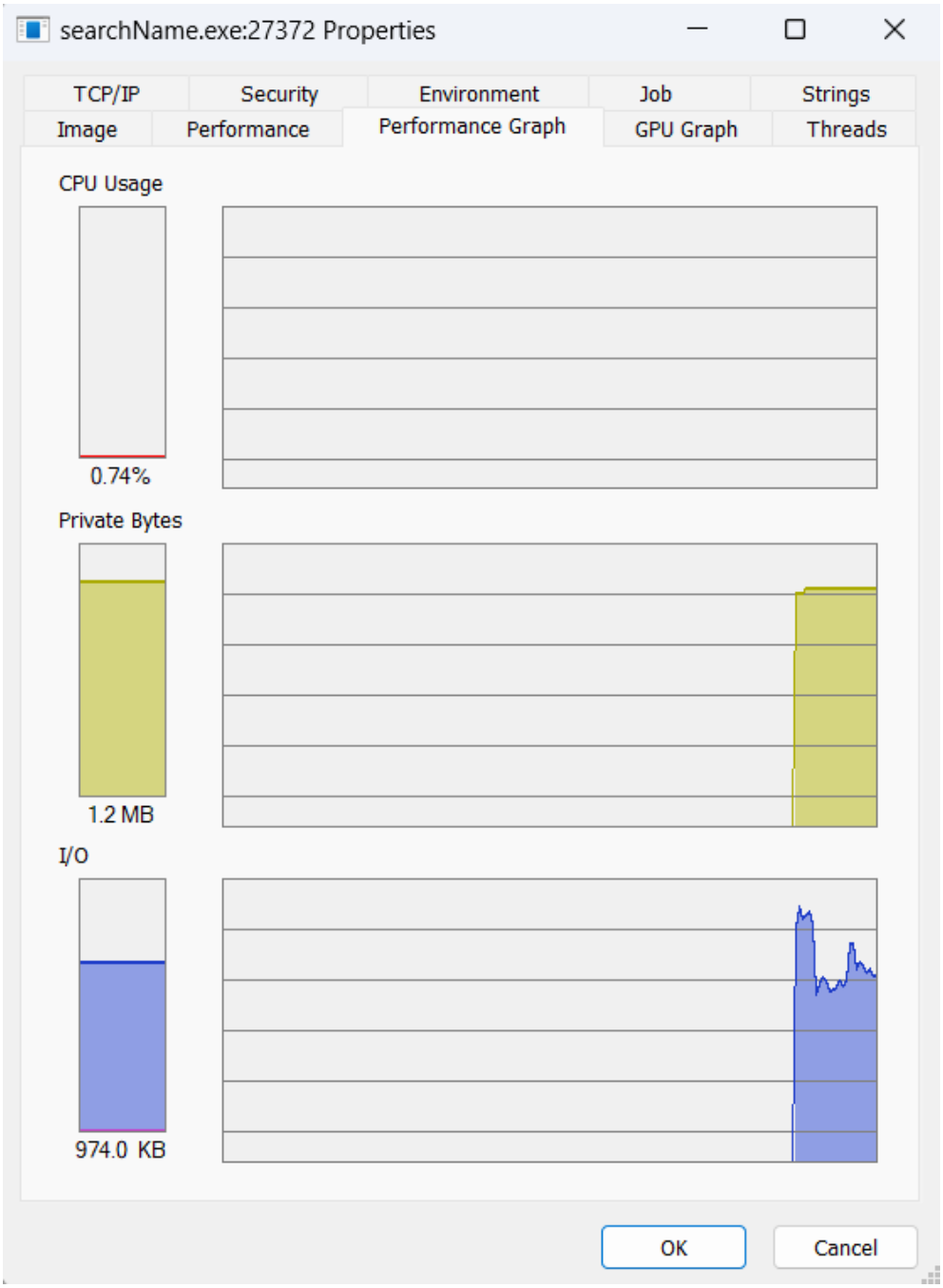
```
.>dedup.exe random_numbers256.txt dedup_numbers.txt 10 8
Общее время выполнения программы: 134982 мс
Среднее время выполнения итерации: 13362 мс
Process was active for 135023 мс
.>
```



Search name

10000 повторений и 1 поток

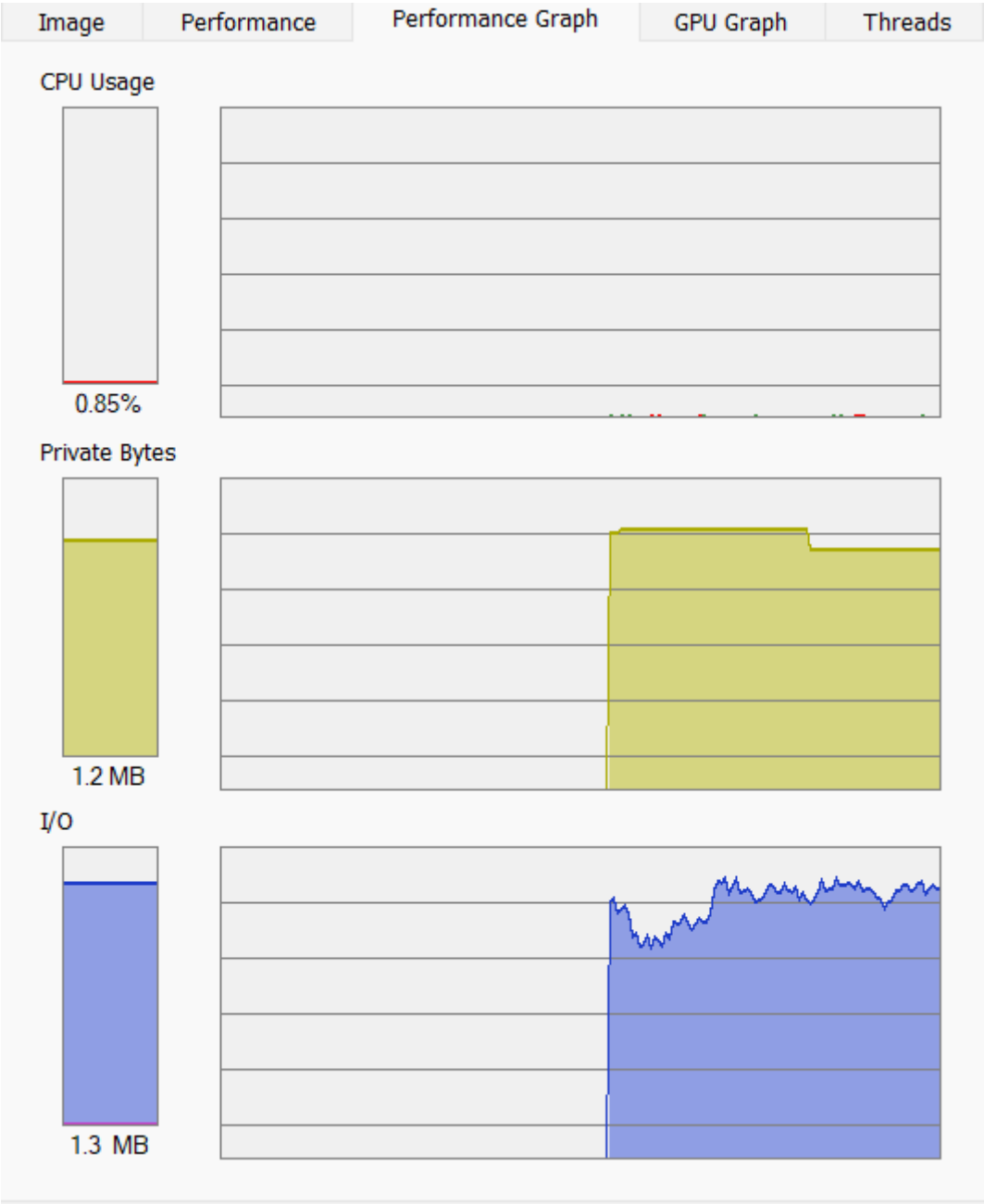
```
Общее время выполнения программы: 29687 мс
Среднее время выполнения итерации: 2 мс
Process was active for 29748 мс
.>
```



Алгоритм отрабатывает примерно за 2мс, что соответствует ожидаемому результату

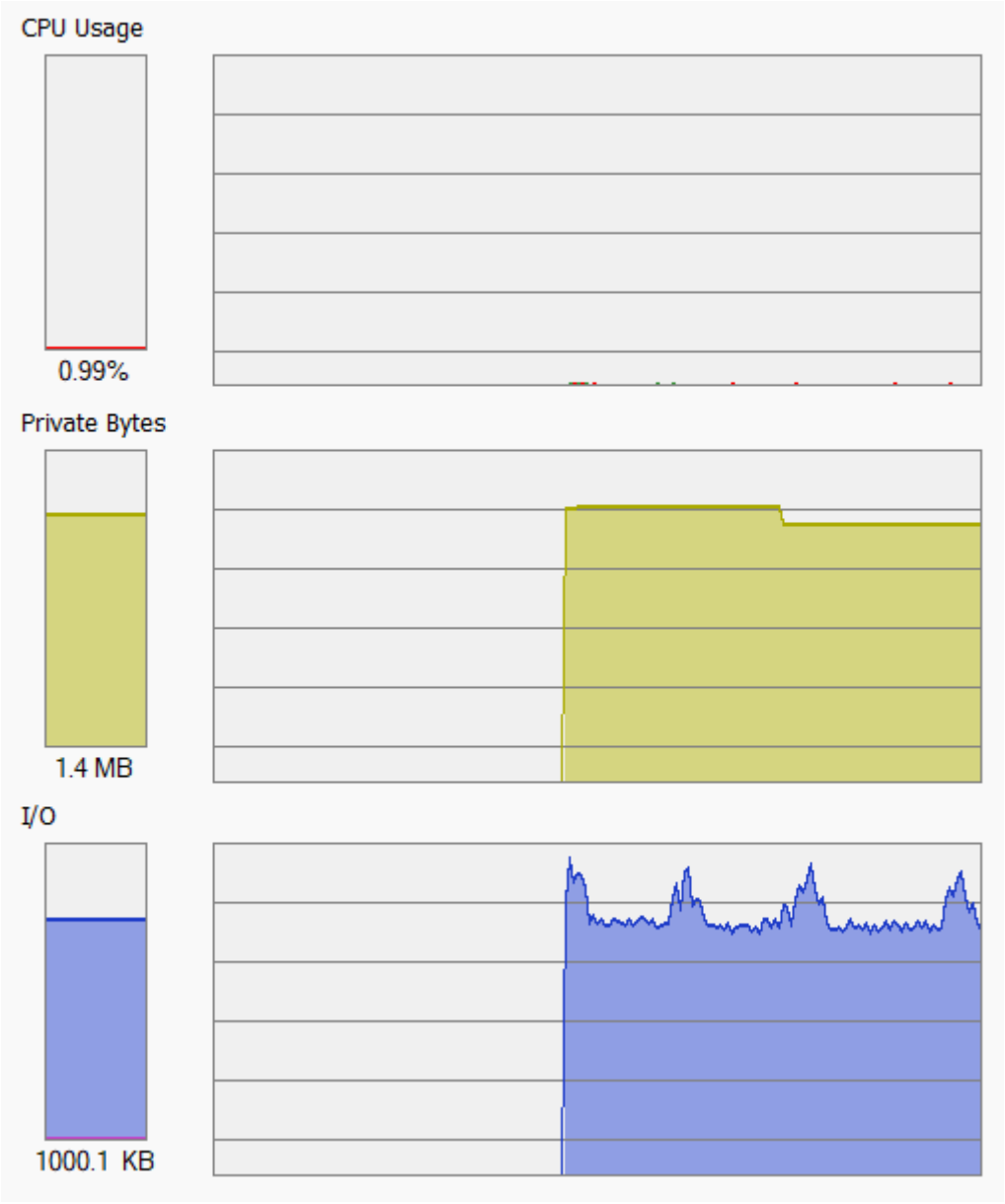
10000 повторений и 4 потока

```
Общее время выполнения программы: 97756 мс
Среднее время выполнения итерации: 9 мс
Process was active for 97812 мс
.>
```



5000 повторений и 8 потоков

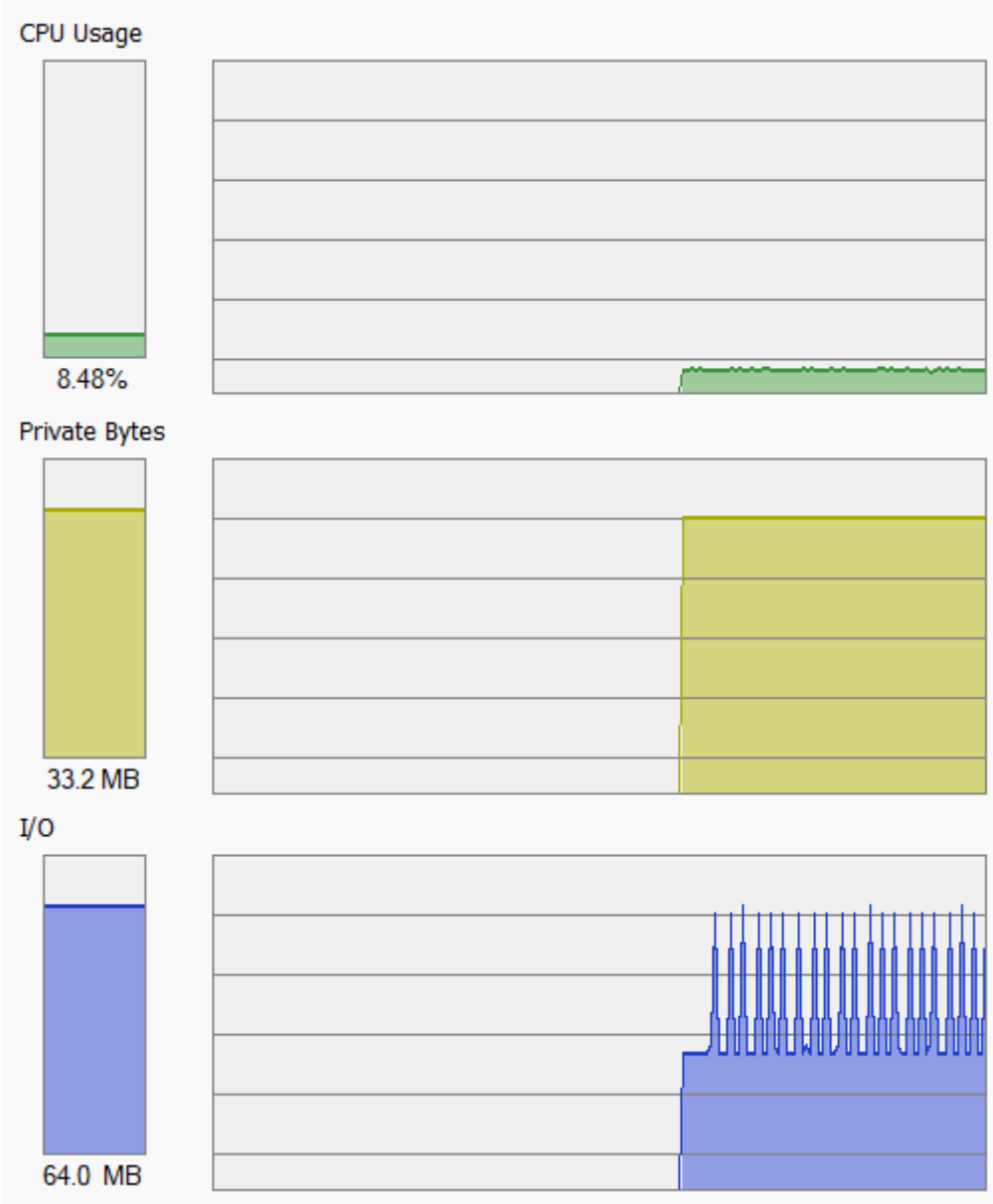
```
Общее время выполнения программы: 114027 мс
Среднее время выполнения итерации: 22 мс
Process was active for 114081 мс
.>
```


CPU Usage
0.99%
Private Bytes
1.4 MB
I/O
1000.1 KB

Dedup and search

10 повторений в 2 потоках

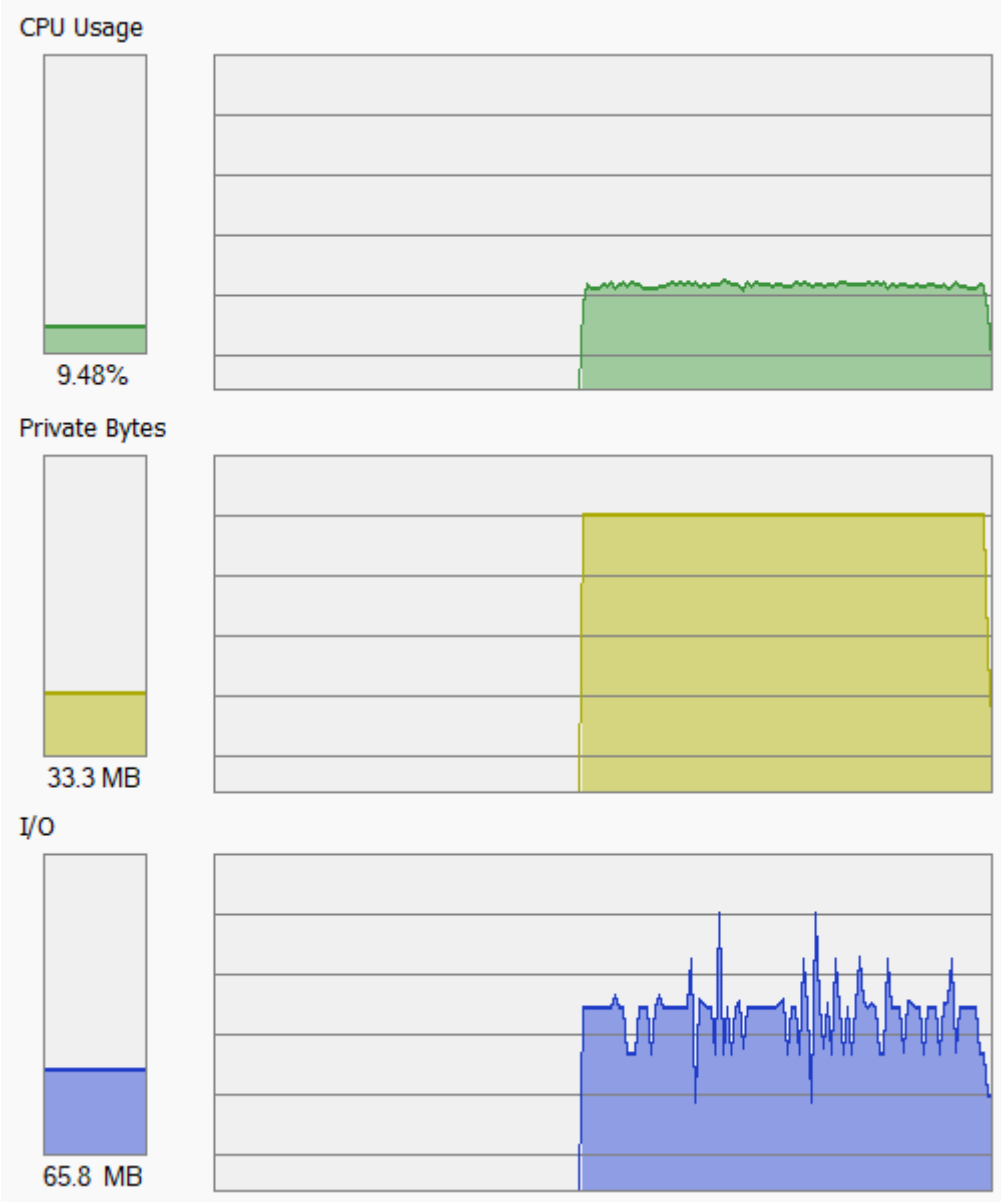
```
Общее время выполнения программы: 84462 мс
Среднее время выполнения итерации dedup: 8446 мс
Среднее время выполнения итерации search name: 1 мс
Process was active for 84949 мс
.>
```



Тут используется в два раза больше ресурсов

10 повторений 8 потоков

```
Общее время выполнения программы: 111986 мс
Среднее время выполнения итерации dedup: 11035 мс
Среднее время выполнения итерации search name: 9 мс
Process was active for 112035 мс
.>
```



Вывод

Я научилась пользоваться профилировщиками для отслеживания нагрузки параметров системы и реализовала алгоритмы, которые нагружают дисковую и файловую системы: для дисковой системы – записать из одного файла в другой все уникальные элементы. Для файловой – найти заданный файл в директориях. Также реализовала свою оболочку shell, которая не использует абстракции над системными вызовами.

Исходный код

https://github.com/E-v-e-l-i-n-e/os-labs/tree/main/lab1