Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное 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 «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе № 4 по дисциплине «Анализ алгоритмов»

Тема Параллельные вычисления

Студент Маслюков П. В.

Группа ИУ7-52Б

Оценка (баллы) _____

Преподаватель Волкова Л. Л.

Москва — 2024 г.

Содержание

Введение **3**

1 Аналитическая часть **4**

1.1 Кластеризация алгоритмом k-средних 4

1.2 Параллельный алгоритм кластеризации методом k-средних 5

1.3 Вывод 5

2 Конструкторская часть **6**

2.1 Разработка алгоритмов 6

2.2 Распараллеливание программы 8

2.3 Вычислительная сложность 8

2.4 Вывод 8

3 Технологическая часть **9**

3.1 Средства реализации 9

3.2 Сведения о модулях программы 9

3.3 Реализация алгоритмов 9

3.4 Функциональные тесты 11

3.5 Вывод 12

4 Исследовательская часть **13**

4.1 Технические характеристики 13

4.2 Демонстрация работы программы 14

4.3 Время выполнения алгоритмов 14

4.4 Вывод 15

Заключение **16**

Список использованных источников **17**

Введение

По мере развития вычислительных систем программисты столкнулись с необходимостью производить параллельную обработку данных для улучшения отзывчивости системы, ускорения производимых вычислений и рационального использования вычислительных мощностей. Благодаря развитию процессоров стало возможным использовать один процессор для выполнения нескольких параллельных операций, что дало начало термину «Многопоточность».

Целью данной лабораторной работы является изучение принципов и получение навыков организации параллельного выполнения операций.

1 Аналитическая часть

В данном разделе будут описаны алгоритмы кластеризации: кластеризация методом k-средних и параллельный алгоритм кластеризации методом k-средних.

1.1 Кластеризация алгоритмом k-средних

Первые применения алгоритма k-средних были описаны в работе Джеймса МакКуина в 1967 году. При заранее известном числе кластеров k алгоритм k-средних начинает с некоторого начального разбиения документов и уточняет его, оптимизируя целевую функцию – среднеквадратичную ошибку кластеризации [1].

Действие алгоритма начинается с выбора k начальных центров кластеров. Обычно исходные центры кластеров выбираются случайным образом. Затем каждый документ присваивается тому кластеру, чей центр является наиболее близким документу, и выполняется повторное вычисление центра каждого кластера как центроида, или среднего своих членов. Такое перемещение документов и повторное вычисление центроидов кластеров продолжается до тех пор, пока не будет достигнуто условие остановки. Условием остановки может служить следующее:

- достигнуто пороговое число итераций;
- центроиды кластеров больше не изменяются;
- достигнуто пороговое значение ошибки кластеризации.

На практике используют комбинацию критериев остановки, чтобы одновременно ограничить время работы алгоритма и получить приемлемое качество.

1.2 Параллельный алгоритм кластеризации методом k-средних

Чтобы уменьшить врмея выполнения алгоритма, следует распараллелить ту часть алгоритма, которая содержит высчитывает расстояние от элемента массива до центра кластеризации. Вычисление результата не зависит от результата подсчета для других элементов. Поэтому можно распараллелить часть кода, где проходят эти действия. Каждый поток будет выполнять вычисления расстояния для элемента массива.

1.3 Вывод

Были рассмотрены алгоритмы кластеризации методом k-средних и возможность его оптимизации с помощью распараллеливания потоков. Была расмотрена технология параллельного программирования и организация взаимодействия параллельных потоков.

2 Конструкторская часть

В данном разделе будет представлена схема алгоритма кластеризации методом k-средних и также будет рассмотрен способ распараллеливания алгоритма. Кроме того будет проанализирована вычислительная сложность алгоритма.

2.1 Разработка алгоритмов

На рисунке 2.1 приведена схема алгоритма кластеризации методом k-средних.

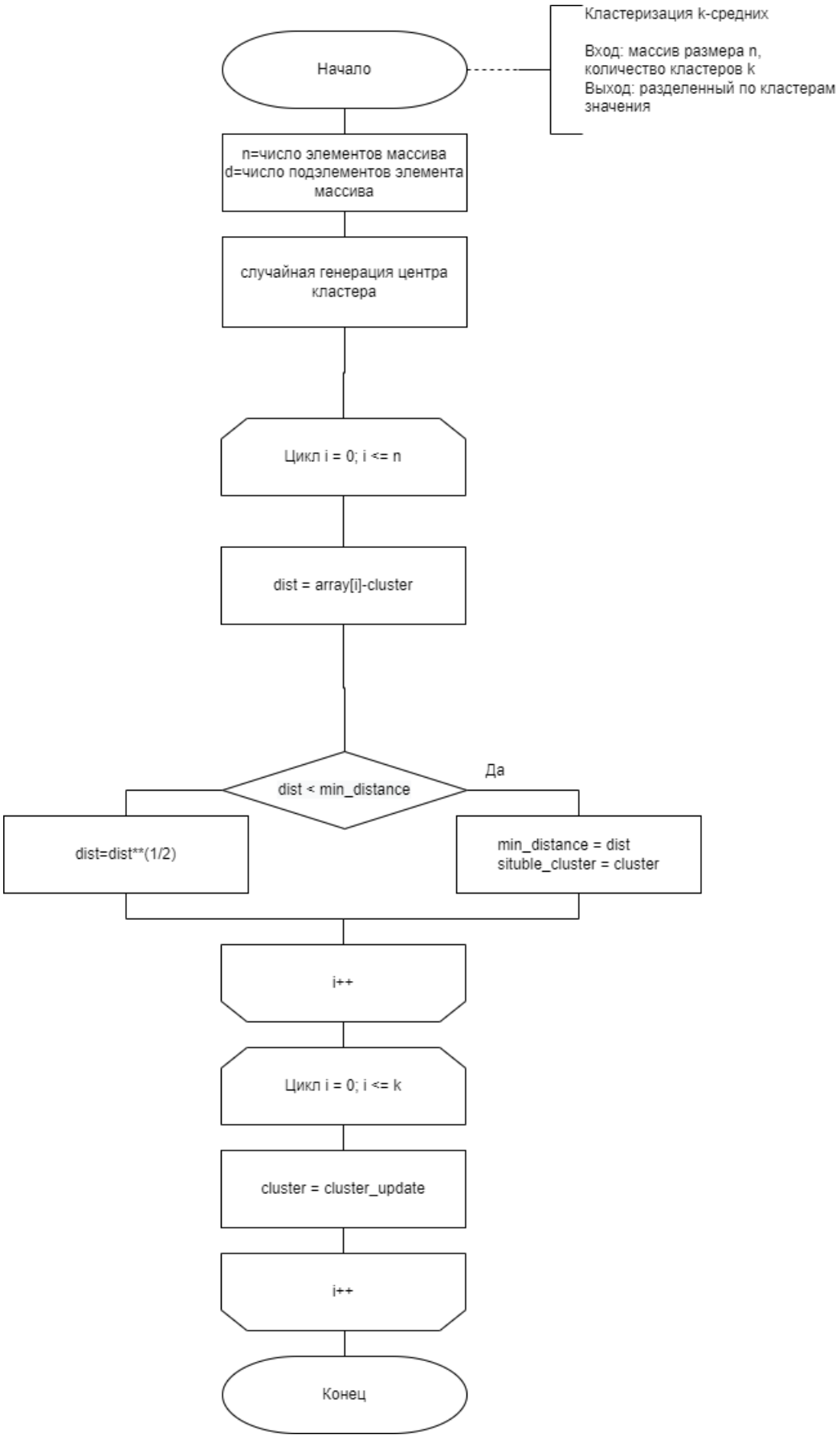

Рисунок 2.1 – Алгоритм кластеризации методом k-средних

2.2 Распараллеливание программы

Распараллеливание программы должно ускорять время работы. Это достигается за счет реализации параллельных потоков в узких участках программы. В алгоритме кластеризации методом k-средних данным участком будет являться участок, в котором происходит подсчет расстояния от элемента массива до центра кластера.

2.3 Вычислительная сложность

Алгоритм k-средних линейно зависит от всех своих факторов: от количества документов, количества кластеров, количества терминов и количества итераций. Для сохранения линейной сложности ($O(|D|)$) при комбинировании с иерархическим с целью эффективного задания начальных центроидов кластеров, предлагается квадратичный иерархический алгоритм применить к выборке документов размером $\sqrt{|D|}$

2.4 Вывод

Была представлена схема алгоритма кластеризации. Был рассмотрен способ рспараллеливания алгоритма. Также была проанализированна вычислительная сложность.

3 Технологическая часть

В данном разделе будут указаны средства реализации, будут представлены реализации алгоритмов, а также функциональные тесты.

3.1 Средства реализации

Реализация данной лабораторной работы выполнялась при помощи языка программирования Python. В текущем лабораторном задании необходимо определить процессорное время, затрачиваемое на выполнение программы, а также ввести многопоточное программирование. Все необходимые инструменты для этого имеются в выбранном языке программирования[2].

Замеры времени проводились при помощи функции process_time из библиотеки time [3]. Многопоточное программирование было реализованно с помощью модуля ThreadPoolExecutor.

3.2 Сведения о модулях программы

Программа состоит из следующих модулей:

- main.py - главный файл программы, предоставляющий пользователю меню для выполнения основных функций;
- func.py - файл, содержащий функции кластеризации и сравнения времени.

3.3 Реализация алгоритмов

Алгоритм кластеризации методом k-средних и его параллельная реализация приведены в листингах 3.1, 3.2.

Листинг 3.1 – Алгоритм кластеризации методом k-средних

```
def clusterization(array, k):
    n = len(array)
    dim = len(array[0])
    cluster = [[0 for i in range(dim)] for q in range(k)]
    cluster_content = [[] for i in range(k)]
    for i in range(dim):
        for q in range(k):
            cluster[q][i] = randint(0, MAX_CLUSTER_VAL)
    cluster_content = data_distribution(array, cluster, n, k, dim)
    privious_cluster = copy.deepcopy(cluster)

    while 1:
        cluster = cluster_update(cluster, cluster_content, dim)
        cluster_content = data_distribution(array,
           cluster,n,k,dim)
        if cluster == privious_cluster:
            break
        privious_cluster = copy.deepcopy(cluster)
    visualisation_3d(cluster_content)
```

Листинг 3.2 – Параллельная реализация алгоритма кластеризации

```
def asinc_clusterization(array, cluster, n, k, dim):
    for i in range(n):
    min_distance = float('inf')
    situable_cluster = -1
    for j in range(k):
        distance = 0
        for q in range(0,dim, 3):

        with ThreadPoolExecutor() as executor:
            res = executor.map(summing(array[i][q], cluster[j][q]
                ))
            for num in res:
                distance += num
        distance = distance ** (1 / 2)
        if distance < min_distance:
            min_distance = distance
            situable_cluster = j

    cluster_content[situable_cluster].append(array[i])
```

3.4 Функциональные тесты

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы кластеризации методом k-средних. Все тесты пройдены успешно.

Таблица 3.1 – Функциональные тесты

Массив входных чисел	Количествово кластеров	Распределение
$\begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix}$	$\begin{pmatrix}2\end{pmatrix}$	$\begin{pmatrix}1 & 2 & \\ 3 & 4 & 5\end{pmatrix}$
$\begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix}$	$\begin{pmatrix}5\end{pmatrix}$	$\begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix}$
$\begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix}$	$\begin{pmatrix}1\end{pmatrix}$	$\begin{pmatrix}1 & 2 & 3 & 4 & 5\end{pmatrix}$

3.5 Вывод

Были реализованы функции алгоритмов кластеризции. Было проведено функциональное тестирование указанных функций.

4 Исследовательская часть

В данном разделе будут приведены примеры работы программы, и будет проведен сравнительный анализ реализованных алгоритмов по затраченному процессорному времени.

4.1 Технические характеристики

Тестирование проводилось на устройстве со следующими техническими характеристиками:

- операционная система Windows 10 pro;
- память 32 Гб;
- процессор Intel(R) Core(TM) i5-12400 12th Gen 2.50 ГГц.

Тестирование проводилось на компьютере, включенном в сеть электропитания. Во время тестирования компьютер был нагружен только встроенными приложениями окружения, а также непосредственно системой тестирования.

4.2 Демонстрация работы программы

На рисунке 4.1 приведен пример работы программы.

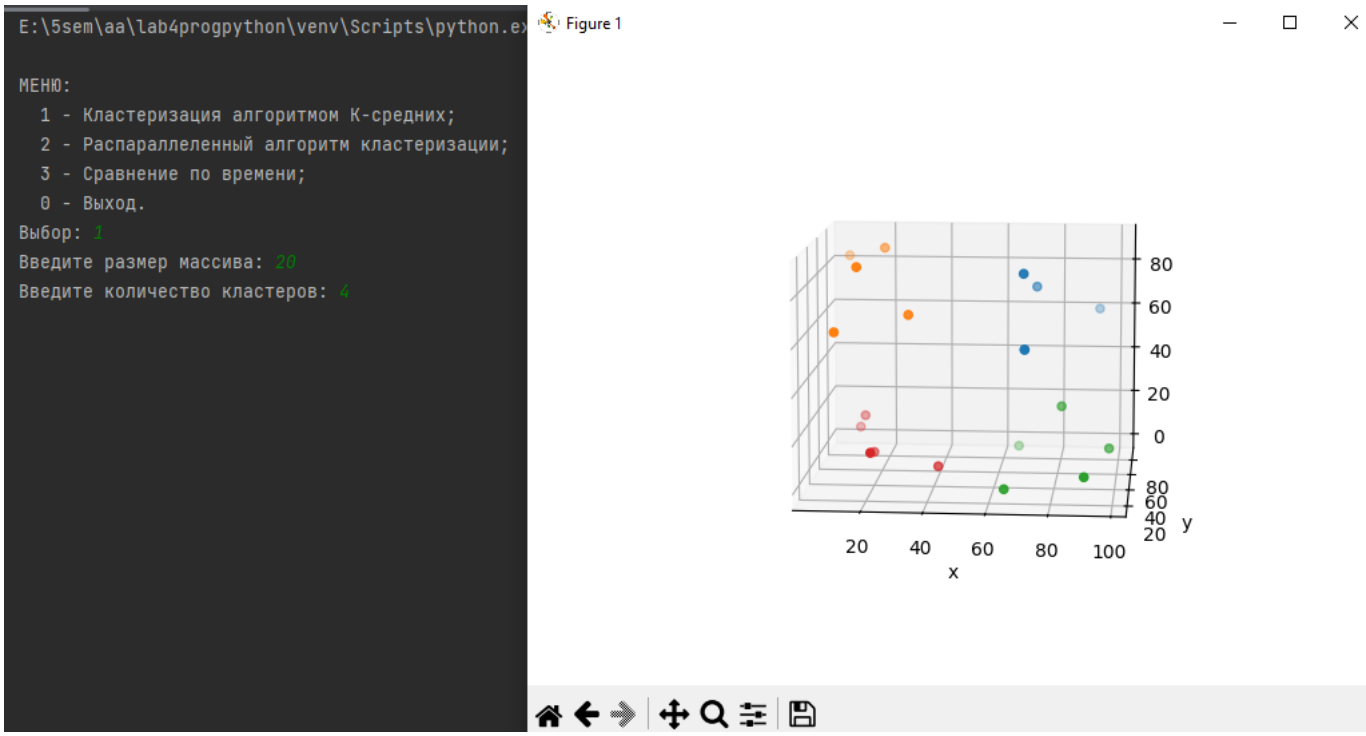

Рисунок 4.1 – Пример работы программы

4.3 Время выполнения алгоритмов

Функция process_time из библиотеки time языка программирования Python возвращает процессорное время в секундах - значение типа float.

Для замера времени:

- получить значение времени до начала выполнения алгоритма;
- получить значение времени после окончания выполнения алгоритма;
- вычесть из второго значения первое.

Замеры проводились для массивов с исходными точками размером 100, 200, 300, 400, 500. Результаты измерения времени приведены в таблице 4.1 (в мс).

Таблица 4.1 – Результаты замеров времени

Размер	Последовательная реализация	Распараллеленная реализация
100	0.0008	0.0003
200	0.0020	0.0006
300	0.0044	0.0014
400	0.0059	0.0018
500	0.0082	0.0028

4.4 Вывод

В результате проведенного эксперимента и анализа полученных значений, можно сделать вывод, что последовательная реализация алгоритма кластеризации методом k-средних работает медленнее распараллеленной версии алгоритма.

Заключение

В ходе работы были изучены алгоритмы кластеризации методом k-средних, последовательный и параллельный. Было произведено сравнение этих алгоритмов. В ходе сравнения было выявлено, что версия с реализацией параллельных потоков работает быстрее версии с последовательной реализацией.

Список использованных источников

[1] Е. И. Большакова АВТОМАТИЧЕСКАЯ ОБРАБОТКА ТЕКСТОВ НА ЕСТЕСТВЕННОМ ЯЗЫКЕ И КОМПЬЮТЕРНАЯ ЛИНГВИСТИКА. 2011.

[2] Welcome to Python [Электронный ресурс]. Режим доступа: `https://www.python.org` (дата обращения: 22.12.2023). 2023.

[3] time — Time access and conversions [Электронный ресурс]. Режим доступа: `https://docs.python.org/3/library/time.html#functions` (дата обращения: 22.12.2023). 2023.