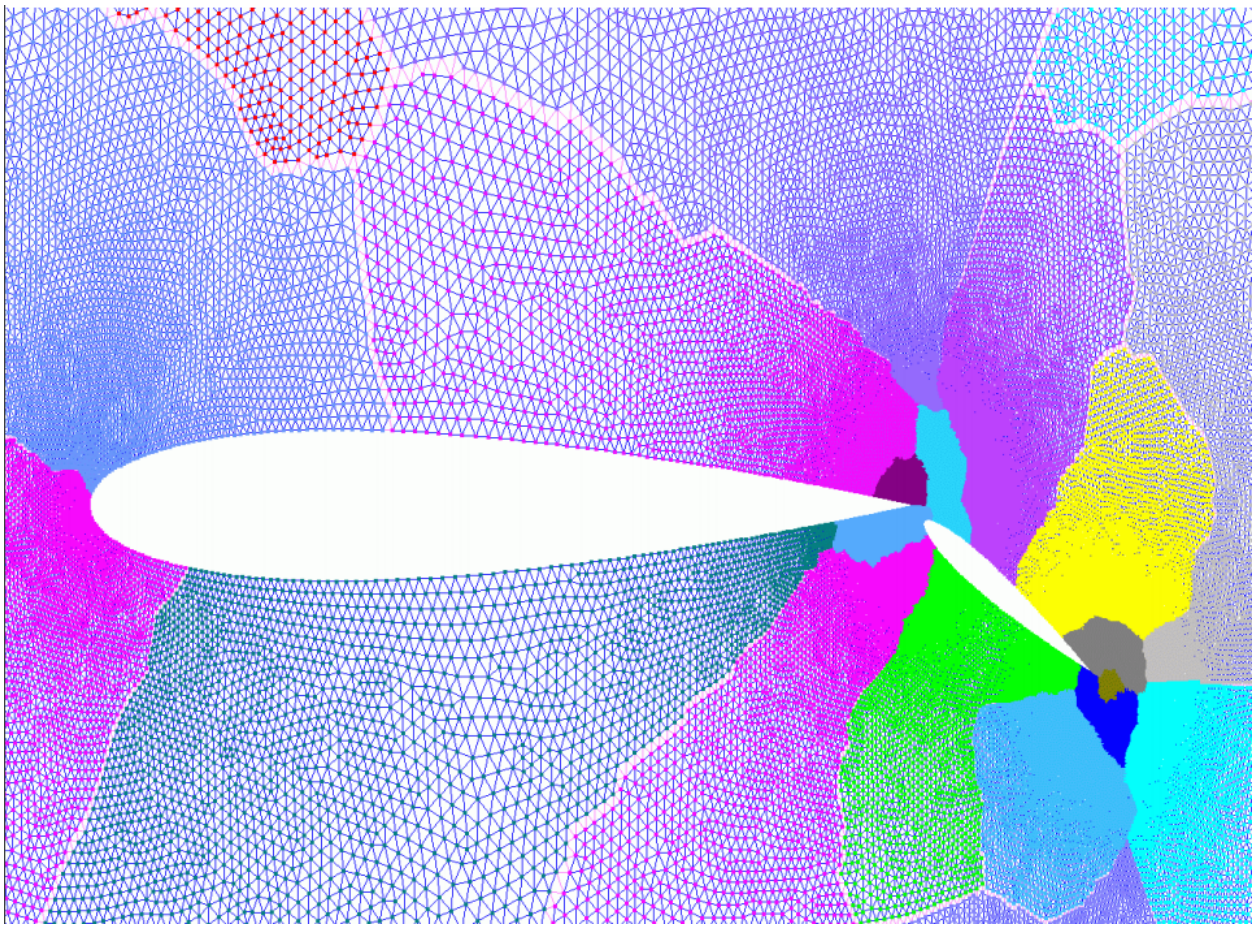
ОТЧЕТ

Задача распределения узлов между процессорами.



Кузьменко Илья

Последняя дата редактирования: 23.04.2023.

Группа 304, МГУ ВМК ВМ

ilyexakuzmenko@gmail.com

~~~~~~~~~~~~~~~~~~~~~~~~~СОДЕРЖАНИЕ~~~~~~~~~~~~~~~~~~~~~~~~~

~~~~~~~~~~~~~~~~~~~~~~~~ПОСТАНОВКА ЗАДАЧИ~~~~~~~~~~~~~~~~~~~~~~~~

-> Задача:

Реализовать программу, которая разбивает квадратную матрицу вычислительных узлов размерности mxm между N процессорами так, чтобы трудоемкости на каждом процессоре приблизительно были одинаковыми.

-> Дополнительное условие:

Изначально трудоемкость каждого узла равна 1. Из левого нижнего узла матрицы начинает двигаться круг, внутри которого трудоемкость умножается на 10.
Программа должна эффективно разделить узлы между процессорами.

-> Входные данные:

```
N = 5; // Число процессоров
m = 10; // Количество строк матрицы
n = m; // Количество столбцов матрицы
nnodes = n * m; // Общее количество узлов
ncon = 1; // Количество весов (трудоемкость)

Radius = 4; // Радиус круга, задаваемый количеством узлов (шт).
Speed = 1; //1,4 Количество узлов, которое пройдет центр круга по диагонали за
            секунду (шт/сек).
Time = 2; // Время, которое двигался круг по диагонали (сек).
```

~~~~~~~~~~~~~~~~~~~~~~ПРОГРАММНАЯ РЕАЛИЗАЦИЯ~~~~~~~~~~~~~~~~~~~~~~

1. Введение.

Для реализации решения поставленной задачей была использована дополнительная внешняя библиотека *METIS*.

– *METIS (Matrix Elementary Transformation on Irregular Structures)* - это пакет программ для разбиения графов на части и решения различных задач, связанных с ними, таких как распределение задач на процессоры в распределенных вычислениях или уменьшение размерности больших графов для более эффективного анализа. METIS основан на алгоритмах разбиения графов на части, которые оптимизируют множество параметров, таких как размер каждой части, количество связей между частями и распределение веса на процессорах.

METIS написан на языке C и доступен как открытое программное обеспечение под лицензией Apache.

2. Наименования файлов.

Полноценная программа написана на 2 языках программирования: *C++* и *Python*, и состоит из 3 файлов:

a) *MAIN.cpp* - главный файл, который запускает все остальные файлы;
b) *redistributor.cpp* - файл, который содержит работу с матрицей.
c) *graph.py* - файл, который строит графики с распределением вычислительных узлов.

В ходе выполнения программа создает 2 дополнительных файла в текущей директории:

a) *matrix.csv* - файл, содержащий матрицу с разбиением узлов между процессорами, где элемент матрицы - номер процессора, который его обрабатывает.
b) *matrixCirlce.csv* - файл, содержащий матрицу с маской круга.

3. Компиляция.

Для того, чтобы протестировать работу программы, необходимо:

1) Установить METIS.
2) Поместить все файлы в одну директорию.
3) Один раз скомпилировать файл *redistributor.cpp*:

```
$  g++ redistributor.cpp -lmetis
```

4) Установить в файле *graph.py* пути к создавшимся в ходе компиляции файлам *matrix.csv* и *matrixCirlce.csv.*

5) Скомпилировать файл *MAIN.cpp* и запустить получившийся файл:

```
$  g++ MAIN.cpp
$  ./a.out
```

4. Вспомогательные функции.

Файл *redistributor.cpp* содержит несколько вспомогательных функций:

a) Функция:

```
void getMAINvalues(idx_t N1, idx_t m1, idx_t n1, idx_t nnodes1, idx_t ncon1, idx_t Radius1, idx_t Speed1, idx_t Time1){ ... }
```

– функция, связывающая файл *redistributor.cpp* и *MAIN.cpp*.

b) Функция:

```
idx_t* movingCircle (idx_t R, idx_t V, idx_t T){ ... }
```

– функция возвращает матрицу трудоемкостей, где внутри круга трудоемкость 10, а вне круга 1. Круг, центр которого располагался в начальный момент времени в НИЖНЕМ ЛЕВОМ узле, двигается по диагонали, стартуя из НИЖНЕГО ЛЕВОГО узла.

R - Радиус круга, задаваемый количеством узлов,

V - количество узлов, проходимых центром круга по диагонали за секунду,

T - количество секунд - время, которое двигался круг по диагонали.

c) Функция:

```
void printAdjncy(idx_t* adjncy, idx_t* xadj){ ... }
```

– Функция выводит массив, содержащий список всех соседних вершин для каждой вершины графа. Используется для отладки программы.

adjncy - массив, содержащий список всех соседних вершин для каждой вершины графа.

xadj - массив, содержащий индексы начала каждой строки в массиве *adjncy*.

5. Ход программы. Файл *redistribution.cpp.*

Для того, чтобы программа находила необходимое разбиение, было пройдено несколько этапов:

1) Выделение памяти.

2) Заполнение *xadj* и *adjncy*.

3) Вычисление маски трудоемкостей *circle*.

4) Заполнение *vwgt*.

vwgt - массив, содержащий веса для каждого узла.

5) Вызов функции *METIS_PartGraphRecursive()* для разбиения графа на части:

```
METIS_PartGraphRecursive(&nnodes, &ncon, xadj, adjncy, vwgt, adjwgt,
NULL, &N, NULL, NULL, options, &objval, part);
```

6) Изменение значения элемента в матрице *T*.

7) Вывод маски *circle* в файл *matrixCircle.csv.*

8) Вывод матрицы *T* в файл *matrix.csv*.

9) Очистка памяти.

~~~~~~~~~~~~~~~~~~~~~~РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ~~~~~~~~~~~~~~~~~~~~~~

Для наглядности разбиения был добавлен файл graph.py. Результаты его работы будут прикреплены ниже.

Для демонстрации работы программы будем рассматривать входные значения и результат работы программы.

Введем обозначения:

N - число процессоров;

m - размерность квадратной матрицы узлов;

R - радиус круга, задаваемый количеством узлов (считая центр, то есть при R=1 в маске круга должна быть только 1 ячейка - центр) (*шт*) ;

V - скорость - количество узлов, которые прошел центр круга по диагонали за 1 секунду (*шт/сек*)

T - время, которое двигался круг по диагонали (*сек*).

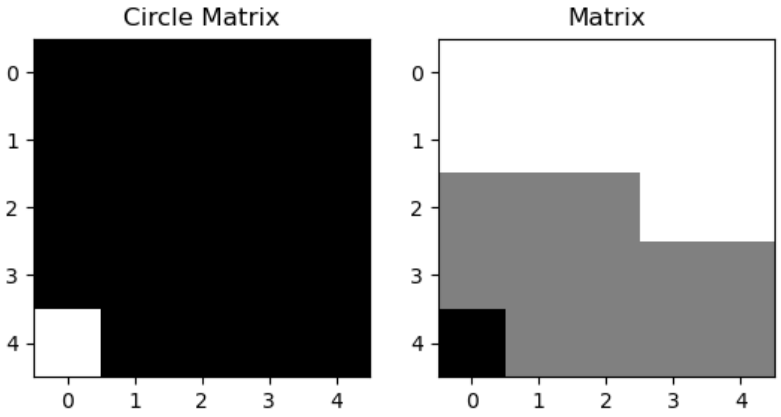
Примеры:

1. *Ввод: N = 3, m = 5, R = 1, V = 0, T = 1:*

Вывод:

Маска круга:

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
2 0 0 0 0



Матрица распределения:

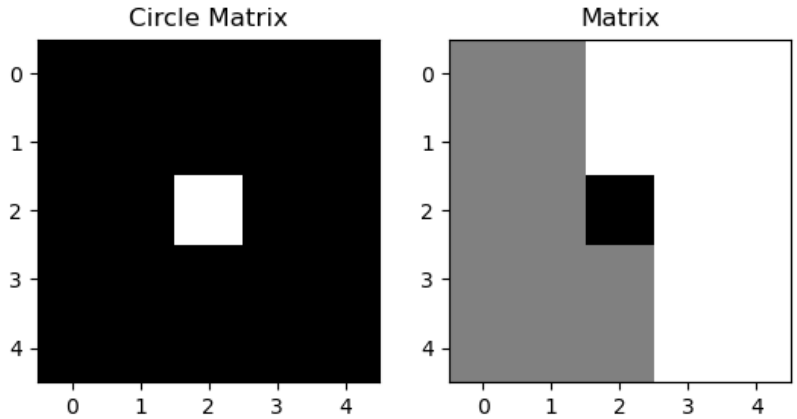
[[3, 3, 3, 3, 3],
[3, 3, 3, 3, 3],
[2, 2, 2, 3, 3],
[2, 2, 2, 2, 2],
[1, 2, 2, 2, 2]]

2. *Ввод: N = 3, m = 5, R = 1, **V = 2**, T = 1:*

Вывод:

Маска круга:

0 0 0 0 0
0 0 0 0 0
0 0 2 0 0
0 0 0 0 0
0 0 0 0 0



Матрица распределения:

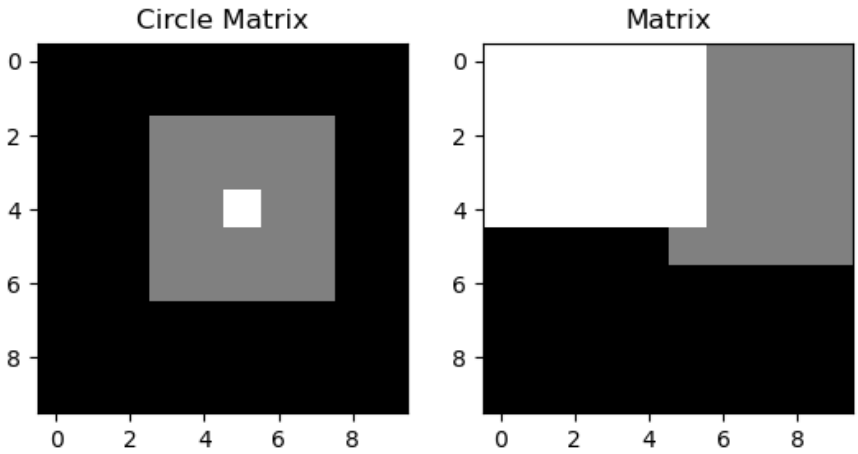
[[2, 2, 3, 3, 3],
[2, 2, 3, 3, 3],
[2, 2, 1, 3, 3],
[2, 2, 2, 3, 3],
[2, 2, 2, 3, 3]]

3. *Ввод: N = 3,* ***m = 10****,* ***R = 3****,* ***V = 5****, T = 1:*

Вывод:

Маска круга:

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 2 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0



Матрица распределения:

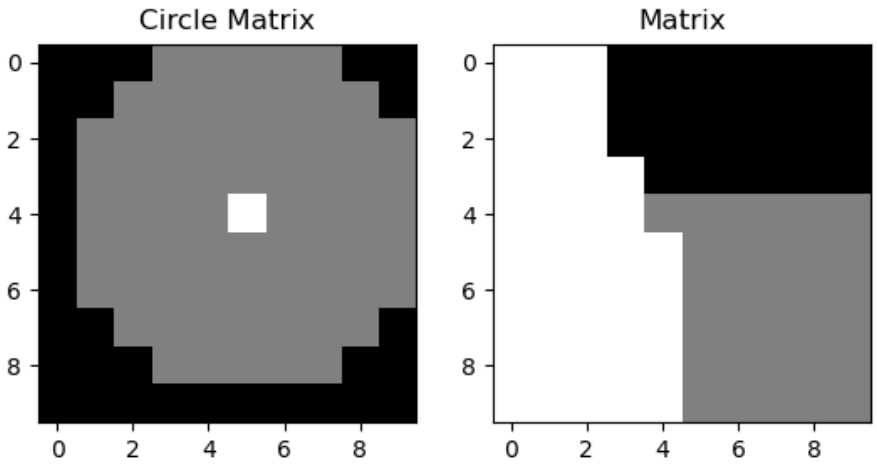
[[3, 3, 3, 3, 3, 3, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 3, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 3, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 3, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 3, 2, 2, 2, 2],
[1, 1, 1, 1, 1, 2, 2, 2, 2, 2],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]

4. *Ввод: N = 3, m = 10, **R = 5**, V = 5, T = 1:*

Вывод:

Маска круга:

0 0 0 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 2 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0



Матрица распределения:

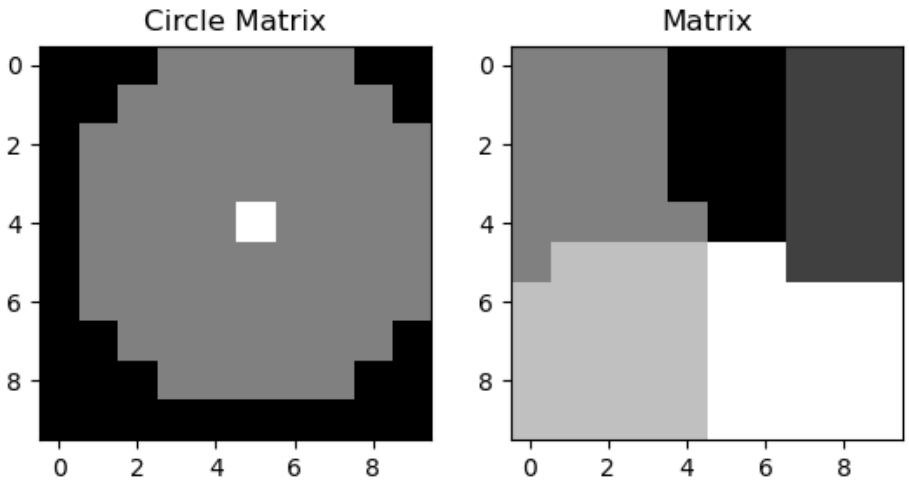
[[3, 3, 3, 1, 1, 1, 1, 1, 1, 1],
[3, 3, 3, 1, 1, 1, 1, 1, 1, 1],
[3, 3, 3, 1, 1, 1, 1, 1, 1, 1],
[3, 3, 3, 3, 1, 1, 1, 1, 1, 1],
[3, 3, 3, 3, 2, 2, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 2, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 2, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 2, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 2, 2, 2, 2, 2],
[3, 3, 3, 3, 3, 2, 2, 2, 2, 2]]

5. *Ввод:* ***N = 5***, *m = 10, R = 5, V = 5, T = 1:*

Вывод:

Маска круга:

0 0 0 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 2 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 0
0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0



Матрица распределения:

[[3, 3, 3, 3, 1, 1, 1, 2, 2, 2],
[3, 3, 3, 3, 1, 1, 1, 2, 2, 2],
[3, 3, 3, 3, 1, 1, 1, 2, 2, 2],
[3, 3, 3, 3, 1, 1, 1, 2, 2, 2],
[3, 3, 3, 3, 3, 1, 1, 2, 2, 2],
[3, 4, 4, 4, 4, 5, 5, 2, 2, 2],
[4, 4, 4, 4, 4, 5, 5, 5, 5, 5],
[4, 4, 4, 4, 4, 5, 5, 5, 5, 5],
[4, 4, 4, 4, 4, 5, 5, 5, 5, 5],
[4, 4, 4, 4, 4, 5, 5, 5, 5, 5]]

6. *Ввод:* ***N = 7***, ***m = 100***, ***R = 20***, ***V =40***, *T = 1:*

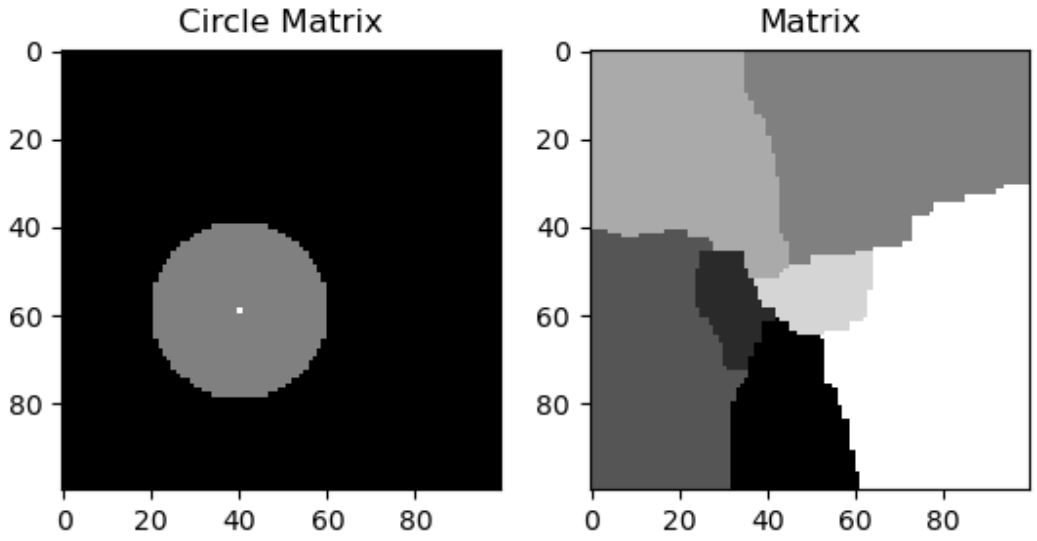
Вывод:

Маска круга:

Слишком большая.

Матрица распределения:

Слишком большая.



7. *Ввод:* ***N = 10***, *m = 100,* ***R = 40***, *V =40, T = 1:*

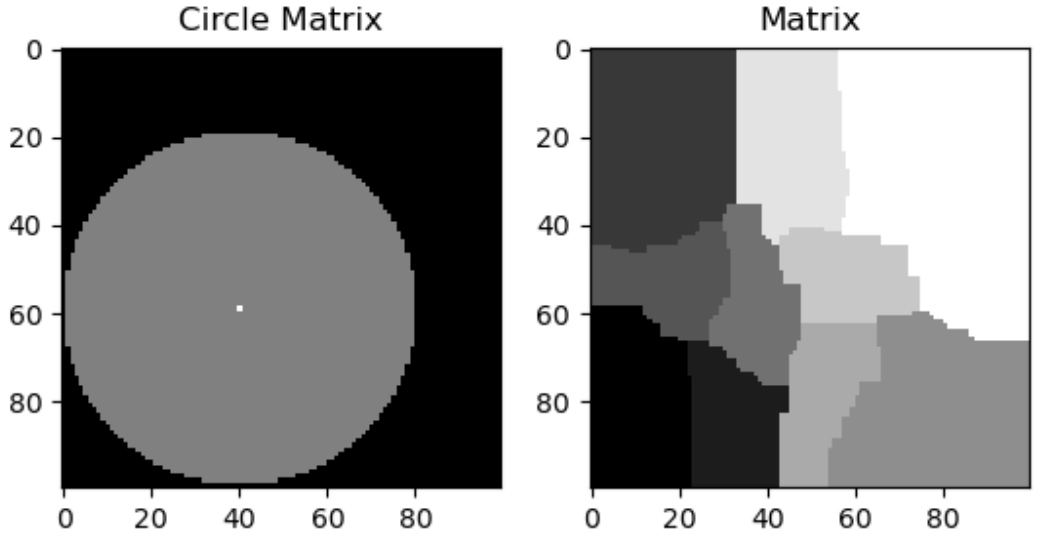
Вывод:

Маска круга:

Слишком большая.

Матрица распределения:

Слишком большая.



8. *Ввод:* ***N = 20***, ***m = 1000***, ***R = 300***, ***V =600***, *T = 1:*

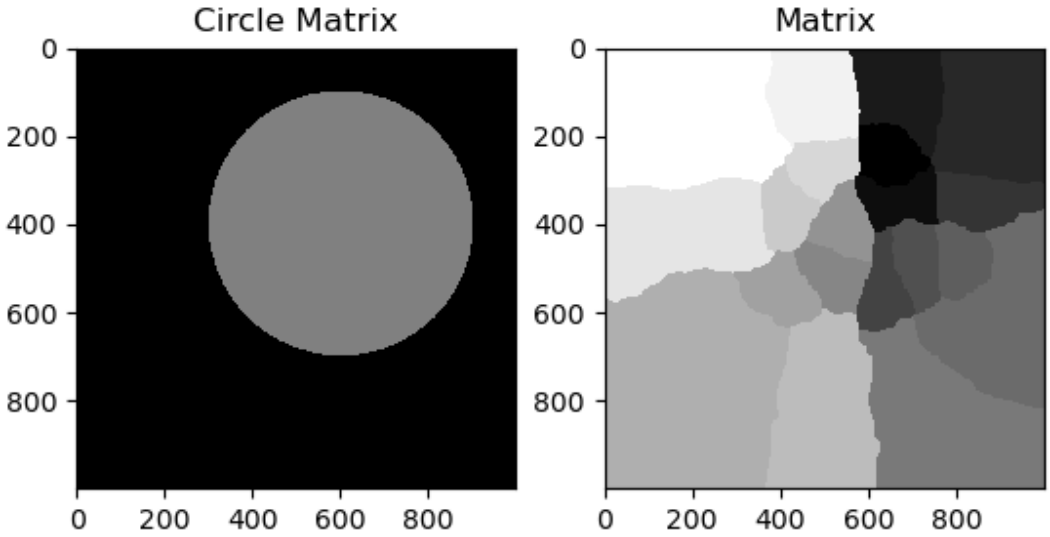
Вывод:

Маска круга:

Слишком большая.

Матрица распределения:

Слишком большая.



9. *Ввод: N = 20, m = 1000, R = 300,* ***V =0****, T = 1:*

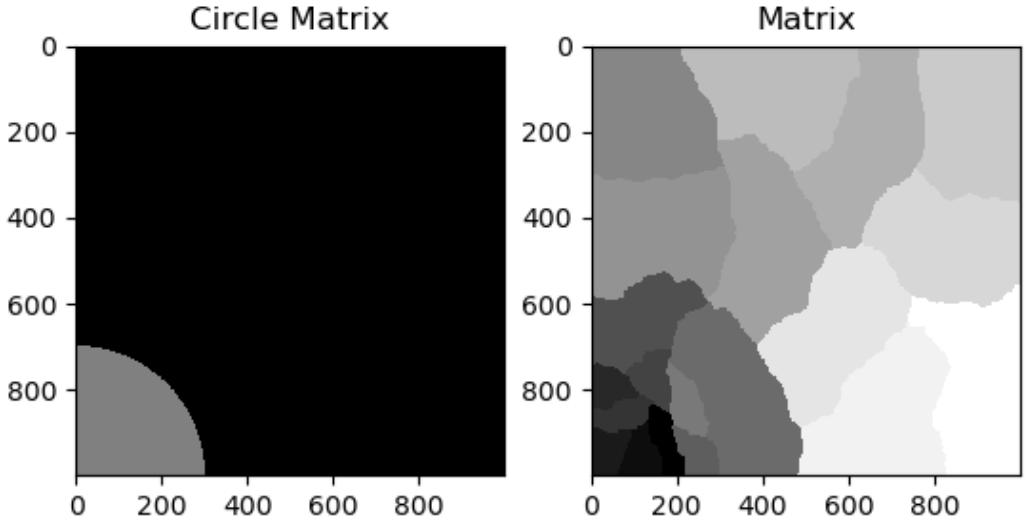
Вывод:

Маска круга:

Слишком большая.

Матрица распределения:

Слишком большая.



СПИСОК ЛИТЕРАТУРЫ

1. *Якобовский М.В.* Параллельные вычисления: рациональная декомпозиция сеточных графов

2. *Корнилина М.А., Якобовский М.В.* Оценка накладных расходов при выполнении расчётов на локально измельчаемых сетках // Препринты ИПМ им. М.В.Келдыша. 2022. № 102. 36 с.

3. *George Karypis* METIS: A Software Package for Partitioning UnstructuredGraphs, Partitioning Meshes, and ComputingFill-Reducing Orderings of Sparse Matrices URL: *https://usermanual.wiki/Pdf/manual.588322308/html#pf1a*