**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

**Алгоритмы и структуры данных**

**Домашняя работа:**

Полимино

**Преподаватель:**

Чесноков В. О.

**Студент**:

Петропавлов Р. И.

ИУ8-53

Москва 2017

Оглавление:

[Практическая часть: 3](#_Toc468903226)

[Описание классов: 3](#_Toc468903228)

1. [Data: 3](#_Toc468903234)
2. [Coordinates: 3](#_Toc468903229)
3. [Figures: 3](#_Toc468903232)
4. [Condition: 4](#_Toc468903233)
5. [Substring: 4](#_Toc468903230)
6. [FiguresVariant: 4](#_Toc468903231)

[Алгоритм: 5](#_Toc468903235)

[Пример: 7](#_Toc468903236)

[Тесты программы: 8](#_Toc468903227)

[Сложность используемых алгоритмов: 9](#_Toc468903237)

[Заключение: 9](#_Toc468903238)

# 

**Практическая часть:**

Описание классов.

1. Data
2. Coordinates
3. Figures
4. Condition
5. Substring
6. FiguresVariant

1.Data:

Основной класс программы, единственный объект которого хранит всю информацию о фигурах, возможных размерах прямоугольника, дереве состояний и площади поля.

2.Coordinates:

Класс координат в двумерном пространстве, с перегруженными операторами присваивания и сравнения.

3. Figures:

Класс, объекты которого представляют из себя фигуры полимино. Построение фигуры делается с помощью двумерного массива из нулей и единиц (который подаётся на вход и прописан в файле). В классе определены методы проверки полимино на наличие нескольких фигур в матрице, проверки на отверстия, функция генерации группового представления, и всех вращений фигуры.

4.Condition:

Класс, хранящий в себе конкретное состояние поля в текущий момент. Является элементом иерархичной структуры, описывающей все переходы поля, подчиняющихся жадной эвристике метода ветвей и границ.

5. Substring:

Класс, предоставляющий интерфейс для работы с наибольшей общей подстрокой группового представления поля и фигуры. Хранит строку, длину, индексы начала подстрок в поле и в фигуры и номер вращения фигуры.

6. FiguresVariant:

Класс, хранящий в себе все возможные лучшие варианты общих подстрок для конкретной фигуры в отдельный момент работы программы. Также содержит все подстроки максимальной длины, все возможные новые групповые представления поля, и объект класса Figures.

**Алгоритм:**

1. На вход программе подается два пути(изменить значение в исходном файле Data.cpp строка 15-16) :
   1. к файлу в котором находятся фигуры.
   2. к файлу, в который запишем ответ.
2. В файле фигуры записаны в следующем виде:
   1. Первое число в файле – количество фигур.
   2. Далее для каждой фигуры:
   * Порядковый идентификатор.
   * Размер минимального квадрата, в который можно вписать фигуру.
   * Количество фигур данного вида.
   * Двумерный массив из нулей и единиц, показывающий конфигурацию клеток в фигуре.
3. Создаем объект класса Data, конструктору на вход подаем argc и argv для проверки корректности ввода в конструкторе.
4. В конструкторе заполняются следующие поля класса Field:
   1. vector<Figures> valid\_data – заносим информацию о всех фигурах поданных на вход.
   2. vector<Coordinates> valid\_sizes – все возможные размеры исходного прямоугольника.
5. Вызываем метод bool Data::tryLocalOptimum() – для построения иерархии состояний в поле класса vector<Condition> data\_tree;
6. Далее для каждого состояние вызываем метод Data Data::countStep(int new\_state\_id), который порождает новое состояние.
   1. Внутри данного метода анализируются исходные фигуры, и та фигура, чья наибольшая общая подстрока с групповым представлением поля максимальная, занимает свое место внутри поля, групповое представление поля пересчитывается, возвращается новое состояние.(если новое состояние не проходит проверку, прежде чем перейти к фигурам с меньшими общими подстроками проверяются все вращения исходной фигуры).
7. Состояние порождаются до тех пор, пока новое не станет финальным, то есть наибольшая общая подстрока будет полностью совпадать с фигурой и групповым представлением поля.
8. Если замечено такое состояние, выполнение программы прекращается, выводится ответ.
9. Если мы перебрали все состояния, а финального так и не нашли, говорим о том, что решение не найдено.
10. Ответ выводится последовательностью букв(R – Right, D – Down, L – Left, U - Up

**Пример:**

Если на вход программе подать данную посследовательность:

|  |
| --- |
|  |
|  | 2 |
|  | 0 |
|  | 4 |
|  | 2 |
|  |  |
|  | 1 0 0 0 |
|  | 1 0 0 0 |
|  | 1 1 0 0 |
|  | 1 1 0 0 |
|  |  |
|  | 1 |
|  | 3 |
|  | 2 |
|  |  |
|  | 1 1 1 |
|  | 1 1 1 |
|  | 0 0 0 |

То на выходе мы должны получить следующий ответ:

|  |  |
| --- | --- |
|  | 0. |
|  | DDLLLLURRURR |
|  | 0. |
|  | RRRRDDDLLLLUUU |
|  | 1. |
|  | RRRRRRDDDLLLLLLUUU |
|  | 1. |
|  | RRRRRRRRDDDLLLLLLLLUUU |

**Тесты программы:**

**Сложность используемых алгоритмов**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Алгоритмы | Сложность по времени | Сложность по памяти |
| 1 | Проверка массива  а)на допустимые числа  б) на наличие нескольких фигур в массиве | а) О(n)  б) О(n^2log(n)) | а)О(n)  б) О(n) |
| 2 | Перевод в групповое представление | О(n) | О(n) |
| 3 | Найти наибольшую общую подстроку для циклических строк | О(n\*m), где m и n  – длины строк. | О(n\*m), где m и n – длины строк. |
| 4 | Проверить групповое представление | О(n^2log(n)) | О(n) |
| 5 | Посчитать площадь и все пары сторон | O(n) | O(n) |
| 6 | Основной алгоритм перебора | O(e^n) | O(e^n) |

**Заключение:**

В результате домашнего задания, мною была реализована программа на языке C++ Я на практике применил знания, полученные на курсе “Алгоритмы и структуры данных”, реализовав алгоритм, способный найти решение NP-сложной задачи. Все данные, касающиеся работы были выложены в публично доступном репозитории на Github: https://github.com/BMSTU732/Polymino