**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Информационная безопасность (ИУ-8) .

**Алгоритмы и структуры данных**

**Домашняя работа.**

**зАДАНИЕ № 19:**

Алгоритм укладки полимино в прямоугольник.

**Преподаватель:**

Чесноков В. О.

**Студент**:

Петропавлов Р. И.

ИУ8-53

Москва 2017

***Условие Задачи:***

Пусть у вас есть несколько плоских фигур различной формы. Каждая фигура вырезана из листочка в клеточку строго по линиям и не содержит отверстий (т. е. является полимино). Напишите алгоритм укладки этих фигур в прямоугольник. Фигуры можно вращать на угол, кратный 90º и отражать по горизонтали или вертикали.

***Теоретическая часть:***

**Полимино**, или *полиомино* (*polyomino*) — плоские геометрические фигуры, образованные путём соединения нескольких одноклеточных квадратов по их сторонам. Это полиформы, сегменты которых являются квадратами.

**Замощение плоскости** (tiling) - представление плоскости в виде множества непересекающихся полимино.

Задача об укладки полимино в прямоугольник похожа на одну из неразрешимых задач - задача о замощении. (<https://neerc.ifmo.ru/wiki/index.php?title=Примеры_неразрешимых_задач:_задача_о_замощении>) - основной информационный источник.

***Теоремы:***

1.Задача о замощении ***четверти*** плоскости полимино неразрешима.

2.Задача о замощении ***половины*** плоскости полимино неразрешима.

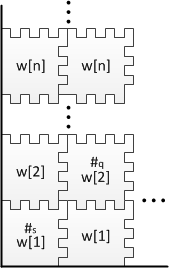
3.Задача о замощении ***целой*** плоскости полимино неразрешима.

***Доказательство (теоремы № 1):***

[Сведём](http://neerc.ifmo.ru/wiki/index.php?title=M-%D1%81%D0%B2%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C) задачу останова к данной задаче. Пусть дана [машина Тьюринга](http://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%A2%D1%8C%D1%8E%D1%80%D0%B8%D0%BD%D0%B3%D0%B0)  =\langle \Sigma, Q, \Pi, B  \in \Pi, s,\delta: Q \times \Pi \rightarrow Q \t и слово  \in \Sigma^*. Требуется определить, остановится ли данная МТ на входе http://neerc.ifmo.ru/wiki/images/math/1/d/6/1d69a7408de15f155cd6b3d17c3bed5e.png.

Для того, чтобы доказать неразрешимость задачи о замощении, для заданной машины Тьюринга M и слова w построим набор полимино, которым можно замостить четверть плоскости, если МТ не остановится на заданном слове. Если же МТ останавливается, то четверть плоскости полученным набором замостить невозможно.

Будем эмулировать процесс выполнения МТ на входе w \in \Sigma^*путем построения вертикальных рядов, каждый из которых эквивалентен конфигурации МТ на определенном этапе выполнения. Первый ряд эквивалентен начальной конфигурации МТ, а каждый следующий ряд соответствует следующей конфигурации

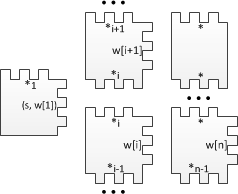
[](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_init.png)

Теперь на основе заданной МТ будем строить набор полимино, которые будут иметь следующий вид:

[](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_gen.png)

На каждой стороне такого полимино находится определенное число выступов/впадин. Каждому символу из алфавита, состоянию и паре из состояния и символа сопоставим некоторое уникальное число (можно ограничить  \le |\Pi| + |Q| + |\Pi \times Q| + 1) – это и будет количество выступов/впадин находящихся на одной стороне полимино.

Сначала построим набор полимино, который задаёт начальную конфигурацию:

[](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_start.png)

где http://neerc.ifmo.ru/wiki/images/math/6/c/8/6c8b7754aa40f5503dea2645a547fe28.png – уникальные числа для каждых соседних двух полимино из начальной конфигурации. Первое полимино характеризует начальное состояние, последующие за ним кодируют входное слово, и завершающее полимино требуется для корректного замощения оставшейся части ряда.

Далее строим полимино для всех элементов алфавита  \in \Pi:

[olyomino alph.png](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_alph.png)

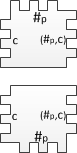
В нем количество впадин слева равно количеству выступов справа. Такой тип полимино передает содержимое ленты МТ следующему ряду.

Теперь построим полимино для функции перехода delta (a, c) = \langle p, d, D \rangle, где  \in Q, p \in Q, c \in \Pi, d \in \Pi, D\in \{\leftarrow, \downarrow, \rightarrow \}:

[](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_delta.png)

На рисунке изображены (сверху вниз) полимино соответствующие значениям  = \{\leftarrow, \downarrow, \rightarrow \}. Вместе со следующим типом они эмулируют перемещение головки МТ.

Далее построим следующий тип полимино:

[](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_delta2.png)

Эти полимино получают на вход символ алфавита http://neerc.ifmo.ru/wiki/images/math/c/e/f/cef160c9876cc23aa59313fa78238223.png от предыдущего ряда и состояние http://neerc.ifmo.ru/wiki/images/math/5/4/1/541365d1bab99f1a2a885ee6bfa42451.png от соседнего полимино, а затем передает следующему ряду пару из состояния и символа.

Построим последний тип полимино, характеризующие состояния #_Y и #_N:

[olyomino halt.png](http://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B0%D0%B9%D0%BB:Polyomino_halt.png)

Такое полимино имеет уникальное число выступов справа. Ни одно другое полимино из полученного набора не сможет к нему присоединиться, и процесс дальнейшего замощения будет невозможен.

Полученный алгоритм сведения получает на вход МТ и слово, а на выход выдает соответствующий им набор полимино.

Таким образом, четверть плоскости можно замостить тогда и только тогда, когда закодированная МТ не останавливается на данном входе. Иными словами есть бесконечное количество конфигураций, не переходящих в конечное состояние. Это значит, что мы сможем замощать плоскость ряд за рядом бесконечное количество раз, что в результате замостит плоскость.

Если же МТ остановится, то и замостить четверть плоскости мы не сможем из-за того, что конечное полимино не имеет продолжения. Значит задача о замощении полимино не разрешима.

***Математическая интерпретация:***

Обозначим S – прямоугольник размера ,

Изобразим на S решетку состоящую из единичных квадратов.

Пусть ХУ ориентированная ломанная на S, с началом в точке Х и концом в точке У, образованная линиями решетки.

Возьмем группу G{A,B}, движению ломанной вверх сопоставим А, вправо – В, соответственно – вниз,– влево. Операцией определим соответствующее движение в заданном направлении. Таким образом можно описать наше полимино, с обязательным условием совпадения начала и конца. Обозначим полимино через Р.

Наша задача, составить функцию:

, где G – функция, присваивающая каждому из полимино координату начала так, что S полностью замощена элементами Р.

***План решения задания:***

При решении задачи, будет использоваться лишь **Теорема №1** потому, что: очевидно, что если данным набором полимино можно замостить квадрант (то есть четверть плоскости), то им можно замостить также полуплоскость и всю плоскость; однако, из замощаемости, скажем, полуплоскости, замощаемость квадранта не вытекает.

Для того, чтобы определить, можно ли соединить полимино будет проводиться анализ описание полимино через элементы множества выбранной группы.

Для фигур имеющих общие элементы с несколькими другими фигурами будет применен метод ветвей и границ. Основное достоинство метода в том, что конфигурации фигур после склейки, не оптимальные для данной задачи, будут отброшены.

Для начала будет проведена проверка входных данных, для того чтобы отсеять условия с неправильными полимино, с прямоугольниками не позволяющими замостить их без наложения и прямоугольниками не позволяющими вместить самый широкий и самый высокий элементы.

После начальных проверок будет проведена проверка на возможность размещение исходных полимино по периметру прямоугольника. Если входные данные корректны, то возможны случаи нескольких вариантов размещения полимино по периметру. На этот случай нам поможет анализ описания полимино. Благодаря его структуре, анализ поможет нам сразу проверить все возможные ориентации фигуры в пространстве используя сериальное кодирование.

На случай вариативности расположения очередной фигуры мы используем метод ветвей и границ, который позволит нам убрать не корректные и не оптимальные случаи. Используется именно этот метод, поскольку он хорошо подходит для решения NP-сложных задач и по существу, метод является вариацией полного перебора, только с отсевом подмножеств допустимых решений, заведомо не содержащих оптимальных решений.

Возможно, при успешной реализации текстового вывода, будет также поддерживаться и вывод графический. Реализация текстового вывода будет писаться на языке C++, в случае успешного выполнения, реализация графического вывода будет писаться либо на Java, либо на Perl, поскольку они больше приспособлены к выводу в формате ASCII-графики или XHTML+SVG (какой из этих способов будет использоваться, пока неизвестно) по сравнению с тем же C++.

Мною будет реализован класс **Polyomino**, который будет содержать структуру данных, хранящую в себе:

1. двумерный массив-для представления фигуры;
2. площадь;
3. количество таких фигур;

Также, необходимо будет реализовать метод **errors**-для обработки ошибок, метод **Coordinates**-вспомогательная функция для соединения фигур, метод **input\_test-**для проверки корректно введённых данных и соответственно метод **print-**для вывода работы программы.

В случае графического представления фигур, ещё будет необходимо реализовать метод **toASCII()-**Возвращает строку, содержащую ASCII-представление фигуры,

Или же метод **toSVG()-**Возвращает строку, содержащую SVG-документ с изображением фигуры.

Какой из данных методов будет выбран пока неизвестно.

***Формат входных данных:***

1.Первая строка содержит количество полимино (n).

2.Вторая строка содержит размер массива.

3.Третья строка содержит количество полимино такого вида. Повторяется n раз

4.Далее задаётся сам массив.

Например:

4

3

1

100

100

110

3

1

111

000

000

2

1

11

01

2

1

01

01

Где 0-клетка отсутствует,1-клетка присутствует.

***Формат выходных данных:***

Каждая строка содержит положение общего полимино (положение задаётся через символы “R”=right, “U”=up, “L”=left, “D”=down)

укладку буду начинать с наименьших полимино, тут к примеру начну с конца, тогда:

1.RDDLUU

2.RRRDDLLULU

3.RRRDDDLLUULU или же RRRDDLLLLLURRU(как мы выбираем я описал в конце "План решения задания") в данном случае мы выберем второй способ.

4.мы повернем наше первое полимино на 90градусов и в итоге получим: RRRRRRDDLLLLLLUU

***Альтернативное решение:***

Одним из вариантов алгоритма для решения данной задачи может быть-**поиск с возвратом, бэктрекинг (англ. backtracking)** — общий метод нахождения решений задачи, в которой требуется полный перебор всех возможных вариантов в некотором множестве М.

Решение задачи методом поиска с возвратом сводится к последовательному расширению частичного решения. Если на очередном шаге такое расширение провести не удается, то возвращаются к более короткому частичному решению и продолжают поиск дальше. Данный алгоритм позволяет найти все решения поставленной задачи, если они существуют. Для ускорения метода стараются вычисления организовать таким образом, чтобы как можно раньше выявлять заведомо неподходящие варианты. Зачастую это позволяет значительно уменьшить время нахождения решения. Однако время нахождения решения может быть очень велико даже при небольших размерностях задачи (количестве исходных данных), причём настолько велико, что о практическом применении не может быть и речи. Поэтому, если исходные данные велики, то применяют алгоритм **Д.Кнута- Algorithm X** или как его по-другому называют **Algorithm DLX(Dancing Links X)** для поиска всех решений конкретной проблемы покрытия. Этот алгоритм работает быстрее, но требует большего объема памяти.