Курсовая работа

По предмету

"Программирование на языке высокого уровня"

Тема: "Разработка программы нахождения всех полных подграфов (клик) данного графа"

Содержание

Введение

1. Описание алгоритма нахождения клик

2. Разработка структуры программы

2.1 Постановка задачи

2.2 Структура программы

2.3 Описание классов

2.3.1 Класс vertexmatrix

2.3.2 Класс graph

2.3.3 Класс from1

2.3.4 Класс toolwindow

2.3.5 Класс MatrixWindow

2.3.6 Класс cliqueswindow

3. Реализация на C#

3.1 Реализация алгоритма Брона-Кербоша

3.2 Использование нестандартных компонентов

3.3 Реализация алгоритма удаления ребра графа мышью

3.4 Тестирование реализации алгоритма Брон-Кербоша

3.5 Системные требования

Заключение

Список использованной литературы и источников

Приложение

Введение

Клика – полный подграф неориентированного графа. Другими словами, клика графа есть подмножество его вершин, такое, что между каждой парой вершин этого подмножества существует ребро и, кроме того, это подмножество не принадлежит никакому большому подмножеству с тем же свойством.

Подграф графа — граф, содержащий некое подмножество вершин данного графа и некое подмножество инцидентных им рёбер.

Граф – совокупность непустого множества вершин и множества пар вершин.

Неориентированный граф – упорядоченная пара G: = (V,E), для которой выполнены следующие условия:

V это непустое множество вершин или узлов

E это множество пар (в случае неориентированного графа неупорядоченных) вершин, называемых ребрами.

К примеру, для графа на рисунке 1 кликами будут являться следующие множества вершин: {1,2,3},{4,2,5},{2,3,5},{3,5,6}. Порядок следования вершин значения не имеет.

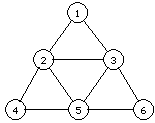


Рисунок 1 – неориентированный граф из шести вершин.

### 1. Описание алгоритма нахождения клик

В качестве алгоритма поиска клик был выбран алгоритм Брона-Кербоша ("Algorithm 457: Finding All Cliques of an Undirected Graph".), заявленный как один из самых быстрых алгоритмов поиска клик (Cazals, F.; Karande, C. (2008), "A note on the problem of reporting maximal cliques"). Алгоритм разработан голландскими математиками Броном и Кербошем (Bron and Kerbosh) в 1973 году.

Алгоритм использует тот факт, что всякая клика в графе является его максимальным по включению полным подграфом. Начиная с одиночной вершины (образующей полный подграф), алгоритм на каждом шаге пытается увеличить уже построенный полный подграф, добавляя в него вершины из множества кандидатов. Высокая скорость обеспечивается отсечением при переборе вариантов, которые заведомо не приведут к построению клики, для чего используется дополнительное множество, в которое помещаются вершины, которые уже были использованы для увеличения полного подграфа.

На листинге 1.1 приведена реализация алгоритма псевдокодом. M – текущее независимое множество, K – множество кандидатов (вершин, способных образовать клику. На начальном этапе это множество содержит все вершины графа), P – множество отсеянных вершин, которые не могут более добавляться в M, v – просматриваемая вершина, G(i) – множество вершин, смежных с i.

Листинг 1.1 – реализация алгоритма Брона-Кербоша псевдокодом

while K != 0 or M != 0:

if K != 0:

v = K.first

push M, K, P, v

M = M + {v}

K = K – G(v) – {v}

P = P – G(v)

else:

if P == 0:

print M

pop v, P, K, M

K = K – {v}

P = P + {v}

2. Разработка структуры программы

# 2.1 Постановка задачи

Задачей программы ставится нахождение всех полных подграфов (клик) данного неориентированного графа. Помимо этого программа должна обладать следующими возможностями:

* Создание и редактирование неориентированного графа.
* Загрузка и сохранение графа в файл, как в виде матрицы смежности, так и в собственном формате файла программы.
* Обеспечивать интерактивность графа (граф можно создавать/изменять при помощи мыши, изменения в графе сразу же отображаются на экране).
* Обладать средствами экспорта изображения графа в графические файлы формата png и bmp (формат jpg мало подходит для этих целей, так как на однородном фоне хорошо заметны артефакты конверсии изображения).
* Предоставлять средства просмотра найденных в графе клик, а также экспорта их в графические файлы или сохранения в матричном или графовом формате.
* Обладать средствами генерации отчета о найденных кликах.
* Предоставлять возможность распечатать созданный граф.

# 2.2 Структура программы

Программа состоит из главного окна приложения, в котором осуществляется создание и редактирование графа, окна свойств графа, отображающего его матрицу смежности и параметры и окна поиска и отображения клик.

Для реализации операций над графом и его матрицей смежности было создано два класса: Graph и VertexMatrix, реализующие все операции применимые к графу в данном приложении.

# 

# 2.3 Описание классов

### 2.3.1 Класс VertexMatrix

Конструкторы класса

VertexMatrix(int dimension) - создает заполненную нулями матрицу смежности порядка dimension.

Также класс имеет статический метод-конструктор для создания матрицы из текстового файла:

static VertexMatrix FromTextFile(string filename) - создает матрицу вершин графа из текстового файла с именем, указанным в filename.

Формат файла: квадратная матрица, состоящая из нулей и единиц, например:

011

101

110

Примечание: в большинстве кодировок символы цифр сохраняют свои коды, поэтому проблемы с загрузкой матрицы из текстового файла маловероятны.

Public методы

byte Get(int column, int row) - возвращает значение ячейки матрицы в столбце column строки row. В случае если значения column или row превышают порядок матрицы, генерируется исключение IndexOutOfRangeException.

void Set(int column, int row, byte value) - Установить значение ячейки матрицы в столбце column строки row равным value. В случае если значения column или row превышают порядок матрицы, генерируется исключение IndexOutOfRangeException.

void AddVertex() - добавляет к матрице новую строку и столбец, тем самым, расширяя порядок матрицы. Новая строка и столбец заполняются нулями.

void DeleteVertex(int index) - удаляет из матрицы строку и столбец с индексом index, тем самым, понижая ее порядок. Строки и столбцы с индексами index+1, если таковы имеются, занимают место удаленных.

void SaveToTextFile(string filename) - создает текстовый файл с матрицей. Формат файла был описан выше.

Private методы

void AddRow() - добавляет строку к матрице.

void AddColumn() - добавляет столбец к матрице.

Свойства

int Dimension - возвращает порядок матрицы. Это свойство только для чтения.

Private свойства

List<List<byte>> mat - сама матрица.

List<byte> row - используется для добавления строк к матрице.

int rlength - длина строки матрицы.

int clength - длина столбца матрицы.

int mat\_dimension - порядок матрицы.

# 

# 2.3.2 Класс Graph

Конструкторы класса

Graph(VertexMatrix matrix) - cоздает граф из матрицы смежности matrix.

Graph(VertexMatrix mat, int radius) – cоздает граф размером radius из матрицы смежности mat.

По умолчанию вершины графа располагаются по окружности радиуса Radius. Первая вершина графа располагается в направлении девяти часов.

Public методы

int AddVertex(PointF coords) - добавляет к графу вершину с координатами coords, при этом порядок матрицы графа увеличивается на единицу. Возвращает индекс добавленной вершины.

void DeleteVertex(int index) - удаляет из графа вершину с индексом index. При этом из матрицы графа также удаляется соответствующие вершине строка и столбец.

int GetVertexIndexFromPoint(PointF p) - возвращает индекс вершины графа, которой принадлежит точка с координатами p. В случае если такой вершины не найдено, возвращает -1.

int[] GetVerticesFromNodePoint(PointF node) – возвращает массив размерностью 2, в которых находятся индексы вершин графа, ребру которых (если такое существует) принадлежит точка с координатами node. Если таких вершин не найдено или они не соединены ребром, функция возвращает null.

void SetVertexCoordinats(int index, PointF coord) - устанавливает координаты вершины с индексом index равными coord.

void ArrangeByCircle()

void ArrangeByCircle(int radius) - располагает вершины графа по окружности радиусом Radius.

Image DrawVerticesToImage(int[] indexes) - рисует вершины с индексами indexes графа в объект класса Image. Размеры области рисования вычисляются из нахождения вершин с координатами максимально и минимально удаленными от осей X и Y.

Возвращает объект класса Image, в котором было произведено рисование.

void Draw(Graphics g) - рисует граф в области g.

void SaveToFile(string filename) - сохраняет граф в бинарный файл. Описание формата представлено в таблице 2.3.1.

Таблица 2.1 - Формат файла .g

|  |  |  |
| --- | --- | --- |
| Смещение (байт) DEC | Размер (байт) | Содержимое |
| 0 | 2 | Сигнатура файла .g: 0x0A0D |
| 2 | 2 | Версия файла ( 0 ) |
| 4 | 2 | Число вершин в графе (порядок матрицы смежности). |
| 6 | Число вершин графа в квадрате | Матрица смежности графа. Хранится построчно |
|  | sizeof(float)\*2 \* число вершин в графе | Координаты вершин графа. Хранятся построчно: x1y1x2y2…xnyn |

static Graph FromFile(string filename) - создает граф из файла графа.

List<List<int>> FindAllCliques() - возвращает список списков вершин графа, образующих клики.

Private методы

PointPlace pointClassify(PointF point, PointF origin, PointF dest) - возвращает перечисление PointPlace, указывающую в каком положении относительно отрезка, начинающегося в точке origin и оканчивающемуся в точке dest находится точка.

Перечисление PointPlace:

enum PointPlace : int

{

LEFT = 0,

RIGHT = 1,

BEYOND = 3,

BEHIND = 4,

BETWEEN = 5,

ORIGIN = 6,

DESTINATION = 7,

}

bool pointInTriangle(PointF p, PointF a, PointF b, PointF c) - возвращает true, если точка p принадлежит треугольнику с координатами вершин a, b, c. В противном случае возвращает false.

void SubtractSet(List<int> set, int vert) - удаляет вершину c индексом vert из списка set.

void SubtractSet(List<int> set1, List<int> set2) - удаляет из списка set1 элементы, содержащиеся в set2 (если таковые присутствуют).

List<int> G(int vert) - возвращает список вершин, не смежных с вершиной с индексом vert.

Свойства

int Radius - возвращает размер графа.

int VertexRadius - возвращает радиус вершины.

Private свойства

VertexMatrix gmatrix - матрица вершин графа.

List<PointF> vertices - список координат вершин графа.

Font font - шрифт, используемый для номеров вершины графа. Используется шрифт Verdana высотой 9 пунктов.

int graph\_rad - ширина графа. По умолчанию равна 60.

int vertex\_rad - радиус вершины графа. По умолчанию равен 10.

bool ellipse - определяет, располагать ли вершины графа по окружности радиусом graph\_rad.

# 

# 2.3.3 Класс From1

Конструктор

Form1() - cоздает экземпляр класса Form1.

Public методы

Класс не имеет public методов.

Private методы

IDockContent GetContentFromPersistString(string persistString) - метод, необходимый для подготовки компонента DockPanel к работе и обеспечивает возможность размещения в ней докингого окна класса MatrixWindow.

Параметр persistString – имя класса докингого окна.

void Form1\_Load(object sender, EventArgs e) - обработчик события Load окна.

void saveDocument(bool saveAs) - отображает меню "Сохранить", предоставляющее возможность сохранить граф в файл. Если граф создан не из файла, пользователю предоставляется возможность самостоятельно выбрать имя, тип и путь к сохраняемому файлу посредством стандартного диалога сохранения файла Windows. В случае, если параметр saveAs равен true, будет вызвано стандартное окно "Сохранить как" Windows.

void openDocument() - отображает стандартное диалоговое окно открытия файла Windows. Если до вызова этой функции был создан граф или производились изменения в существующем, будет выведено диалоговое окно с предложением сохранить граф.

void newDocument() - закрывает текущий документ (если таковой имеется) и создает пустой граф для последующей с ним работе в программе. Если до вызова этой функции был создан граф или производились изменения в существующем, будет выведено диалоговое окно с предложением сохранить граф.

void UNDOAction() - отменяет последнее совершенное пользователем редактирование графа.

void REDOAction() - отменяет отмену последнего совершенного пользователем редактирования графа.

void AddToUNDO(Graph graph) - добавляет граф graph в стек отмены.

void dockPanel\_Paint(object sender, PaintEventArgs e) - событие Paint объекта класса dockPanel.

void DoToolAction(AppTool t, PointF coords) - выполняет действие, предписанное инструментом t с начальными координатами coоrds.

void dockPanel\_MouseDown(object sender, MouseEventArgs e) - обработчик события MouseDown объекта класса dockPanel.

Используется для получения координат мыши и вызова метода DoToolAction.

void dockPanel\_MouseMove(object sender, MouseEventArgs e) - обработчик события MouseMove объекта класса. Используется для получения координат мыши инструментами "Курсор" и "Добавление ребер".

void dockPanel\_MouseUp(object sender, MouseEventArgs e) - обработчик события MouseUp объекта класса DockPanel. Используется для уведомления инструмента "Добавление ребра" о том, что действие закончилось.

void toolStripButtonCursor\_CheckStateChanged(object sender, EventArgs e) - изменяет выбранный инструмент на инструмент "Курсор".

void toolStripButtonAddVertex\_CheckStateChanged(object sender, EventArgs e) - изменяет выбранный инструмент на инструмент "Добавление вершин".

void toolStripButtonDelVertex\_CheckStateChanged(object sender, EventArgs e) - изменяет выбранный инструмент на инструмент "Удаление вершины".

void toolStripButtonAddNode\_CheckStateChanged(object sender, EventArgs e) - изменяет выбранный инструмент на инструмент "Добавление ребер".

void toolStripButtonDelNode\_CheckStateChanged(object sender, EventArgs e) - изменяет выбранный инструмент на инструмент "Удаление ребер".

void toolStripButtonUndo\_Click(object sender, EventArgs e) - обработчик клика по кнопке "Отменить" панели инструментов. Нажатие этой кнопке приведет к откату состояния графа на предыдущее.

void toolStripButtonRedo\_Click(object sender, EventArgs e) - обработчик клика по кнопке "Повторить" панели инструментов. Нажатие этой кнопки приведет к отмену отмены изменений в графе.

void PropertiesWindowToolStripMenuItem\_CheckStateChanged(object sender, EventArgs e) - обработчик клика по пункту меню Вид -> Окно свойств. Скрывает или показывает окно "Свойства графа".

void ViewToolStripMenuItem\_DropDownOpening(object sender, EventArgs e) - обработчик события DropDownOpening панели меню Вид - > Окно свойств. В случае если окно свойств видимо обработчик отмечает элемент меню.

void расположитьВершиныПоОкружностиToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по элементу меню Граф - > Расположить вершины по окружности. Вызов этого меню приведет к расположению графа по радиусу окружности равному свойству Radius графа.

void SaveToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Сохранить. Этот же обработчик имеет кнопка "Сохранить" на панели инструментов.

void OpenToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Открыть. Этот же обработчик имеет кнопка "Открыть" панели инструментов.

void NewToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Новый. Этот же обработчик имеет кнопка "Новый" на панели инструментов.

void printToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Печать. Этот же обработчик имеет кнопка "Печать" на панели инструментов.

void pageToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Предварительный просмотр. Этот же обработчик имеет кнопка "Предварительный просмотр" на панели инструментов. Открывает окно предварительного просмотра документа перед печатью.

DialogResult userWantsToSaveChanges() - выводит окно с предложением сохранить изменения в документе. Варианты ответа "Да", "Нет", "Отмена". Возвращает структуру DialogResult, содержащую вариант выбранного ответа.

bool closeApp() - функция вызывается при закрытии приложения. Вызывает вышеописанную функцию и, в случае утвердительного ответа, возвращает true. В остальных случаях возвращает false.

void ExitToolStripMenuItem\_Click(object sender, EventArgs e) - обработчик клика по пункту меню Файл - > Выход. Вызывает закрытие приложения.

Private свойства

delegate IDockContent DeserializeDockContent ddc - необходим для подготовке контрола dockPanel к работе.

MatrixWindow matrixWindow - окно "Граф".

AppTool currTool - перечисление, определяющее текущий выбранный инструмент.

bool mouseDown - определяет, нажата ли левая кнопка мыши. Используется для работы с инструментами "Курсор" и "Добавить ребро".

int selVertexIndex- индекс выбранной вершины, с которой работает инструмент.

int selVertexIndex2 - индекс второй выбранной вершины, с которой работает инструмент "Добавить ребро".

Point nodePointStart - координата первой выбранной вершины, с которой работает инструмент "Добавить ребро".

Point nodePointEnd - координата второй вершины, с которой работает инструмент "Добавить ребро".

bool documentModified - определяет, были ли произведены изменения в графе. Следует отметить, что если изменения имели быть, но были отменены, документ считается не модифицированным.

string openedDocumentPath - путь к файлу графа или матрице (если граф был создан из файла, else – "").

Stack<Graph> stackUNDO - стек отмены. В него помещается экземпляр графа, перед тем как произвести в нем изменения.

Stack<Graph> stackREDO - стек повтора. В него перемещаются элементы из предыдущего стека при произведении пользователем отмены совершенного им действия.

Следует отметить, что вместо двух стеков может использоваться список элементов типа Graph, что сэкономит ресурсы.

Константы класса

const string GRAPH\_OPENSAVE\_DIALOG\_FILTER = "Граф (\*.g)|\*.g|Матрица (\*.txt)|\*.txt" - константа, определяющая свойство Filter у стандартных диалогов открытия/сохранения файла.

const string PROGRAM\_NAME = "Cliques" - заголовок окна.

# 

# 2.3.4 Класс ToolWindow

Данный класс не содержит никаких пользовательских свойств или методов

2.3.5 Класс MatrixWindow

Конструкторы класса

MatrixWindow() - cоздает экземпляр объекта класса MatrixWindow.

Public методы

VertexMatrix GetMatrixFromDataGrid() - возвращает содержащуюся в окне матрицу.

void FillDataGrid(VertexMatrix mat) - заполняет окно матрицей mat.

void SetDefValues() - устанавливает в окне значения "Размер графа", "Размер вершин" и "Число вершин" равными по умолчанию, это 60, 10 и 0 соответственно. Граф в главном окне при этом не меняется.

void BlockGraphProp(bool block) - устанавливает возможность изменять свойства графа в окне. При block = true изменять свойства нельзя, else можно.

Private методы

void dataGridView1\_ColumnAdded(object sender, DataGridViewColumnEventArgs e) - обработчик события ColumnAdded контрола, содержащего матрицу графа. Добавляет к новосозданной колонке ее порядковый номер.

void ClearDataGrid() - очищает контрол, отображающий матрицу графа. Граф при этом не изменяется.

void dataGridView1\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)- обработчик события RowsAdded контрола, отображающего матрицу графа. Нумерует новосозданные ячейки.

void dataGridView1\_CellMouseDown(object sender, DataGridViewCellMouseEventArgs e) - обработчик события CellMouseDown контрола, отображающего матрицу графа. Меняет значение в кликнутой ячейке на противоположное. Значения на главной диагонали матрицы изменению не подвергаются.

Public свойства

Класс не имеет public свойств.

Private свойства

Класс не имеет private свойств.

# 2.3.6 Класс CliquesWindow

Конструкторы класса

CliquesWindow() - cоздает экземпляр класса CliquesWindow.

Public методы

Класс не имеет public методов.

Private методы

void CliquesWindow\_Load(object sender, EventArgs e) - обработчик события Load формы. Если public свойство graph не равно null вызывает процедуру поиска клик. В противном случае выводит на экран уведомление об отсутствии в графе клик.

void FindAllCliques(Graph g) - находит все клики в графе g.

Image ScaleImage(Image source, int width, int height) - изменяет масштаб изображения source на width и height. Возвращает полученное изображение.

ShowCliqueMatrix(int[] c) - отображает матрицу вершин c клики.

void SaveClique(int index) - отображает диалог сохранения клики с индексом index в файл.

void CreateReport(string filename) - создает файлы отчета обо всех найденных в графе кликах.

Public свойства

Graph graph - граф, в котором будет производиться поиск клик.

Private свойства

List<List<int>> cliques - список вершин найденных клик.

ImageList imgList - сюда заносятся изображения найденных клик.

Константы класса

int VIEW\_IMAGE\_WIDTH = 150 - ширина изображения клики.

int VIEW\_IMAGE\_HEIGHT = 150 - высота изображения клики.

### 3. Реализация на C#

# 3.1 Реализация алгоритма Брона-Кербоша

Реализация алгоритма Брона-Кербоша на С# представлена в Листинге 3.1. Представленные на нем функции являются методами класса Graph.

Листинг 3.1 – Реализация алгоритма Брона-Кербоша на С#.

public List<List<int>> FindAllCliques()

{

List<List<int>> output = new List<List<int>>();

//сюда помещаются вершины, образующие клику

List<int> M = new List<int>();

//список вершин графа

List<int> K = new List<int>();

//список "отработанных" вершин

List<int> P = new List<int>();

//вершина

int v;

Stack<int> stackV = new Stack<int>();

Stack<List<int>> stackM = new Stack<List<int>>();

Stack<List<int>> stackK = new Stack<List<int>>();

Stack<List<int>> stackP = new Stack<List<int>>();

//список несмежных с вершиной вершин

List<int> GS = new List<int>();

//заполняем список вершинами графа

for (int i = 0; i < gmatrix.Dimension; i++)

K.Add(i);

while (K.Count != 0 || M.Count != 0)

{

if (K.Count != 0)

{

v = K[0];

stackM.Push(M.GetRange(0, M.Count));

stackK.Push(K.GetRange(0, K.Count));

stackP.Push(P.GetRange(0, P.Count));

stackV.Push(v);

M.Add(v);

GS = G(v);

SubtractSet(K, GS);

SubtractSet(K, v);

SubtractSet(P, GS);

}

else

{

if (P.Count == 0) //клика найдена

output.Add(M.GetRange(0,M.Count));

M = stackM.Pop();

K = stackK.Pop();

P = stackP.Pop();

v = stackV.Pop();

SubtractSet(K, v);

P.Add(v);

}

}

return output;

}

/\* вычитает вершину из множества \*/

void SubtractSet(List<int> set, int vert)

{

for (int i = 0; i < set.Count; i++)

{

if (set[i] == vert)

set.RemoveAt(i);

}

}

/\* вычитает второе множество из первого \*/

void SubtractSet(List<int> set1, List<int> set2)

{

for (int i = 0; i < set1.Count; i++)

for (int j = 0; j < set2.Count; j++)

if (set1.Count != 0 && i < set1.Count)

if (set1[i] == set2[j])

set1.RemoveAt(i);

}

/\* возвращает список вершин, не смежных с vert \*/

List<int> G(int vert)

{

List<int> ret = new List<int>();

for (int i = 0; i < gmatrix.Dimension; i++)

if (gmatrix.Get(i, vert) == 0)

ret.Add(i);

return ret;

}

Примечание: gmatrix – матрица смежности вершин, реализованная объектом класса VertexMatrix. Свойство Dimension определяет порядок матрицы (число вершин графа).

# 3.2 Использование нестандартных компонентов

В программе был использован элемент, не входящий в поставку .NET Framework 2.0 – DockPanel. Компонент является opensource и написан китайским разработчиком Weifen Luo (http://sourceforge.net/users/weifenluo) под лицензией MIT. Подробное описание компонента, а также исходные коды находятся на прилагаемом к курсовой компакт-диске в папке DockingWindowsComponent.

Назначение компонента

Компонент предназначен для создания "плавающих" окон, окон, способных "прилипать" к краям невидимой панели, а также быть независимыми от нее. Подобный элемент интерфейса, например, использует пакет MS Visual Studio (окна "Properties", "Solution Explorer", etc) и другие программы. Достоинства такой организации интерфейса состоит в том, что не требуемые в данный момент окна могут быть скрытыми и не занимать экранное место, а, при необходимости отображения окна, компактно располагаются по краям родительского окна, что экономит общее экранное место, или же находятся в отсоединенном ("плавающем") состоянии. Панель невидима на экране и занимает всю доступную площадь окна.

Внесенные в компонент изменения

В компонент были внесены изменения, адаптирующие его под решаемую задачу. Поскольку, как уже было сказано выше, компонент занимает всю площадь экрана, было принято решение выполнять отрисовку графа именно на нем. Первоначальный вариант панели не имел средств для рисования на ней, поэтому были внесены изменения в метод OnPaint класса DockPanel, которые находятся в файле DockPanel.cs. На листинге 3.2 показан этот метод до изменения, на листинге 3.3 – после. Также было запрещено обработка события OnPaintBackground, что позволило избежать мерцания рабочей области.

Листинг 3.2 - Первоначальное состояние метода OnPaint класса DockPanel.

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

if (DockBackColor == BackColor)

return;

Graphics g = e.Graphics;

SolidBrush bgBrush = new SolidBrush(DockBackColor);

g.FillRectangle(bgBrush, ClientRectangle);

}

Листинг 3.3 - Состояние метода OnPaint класса DockPanel после редактирования.

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

}

Как видно из листинга 3.2.2 изменение свелось к удалению запрете закраски панели собственным цветом фона. Это приведет к невозможности установки свойства BackColor компонента, но оно не используется в данном приложении.

Использование компонента в данном приложении

В данном приложении элемент DockPanel использовался для создания окна, отображающего свойства графа (класс MatrixWindow). Для этого сначала было создано окно-пустышка класса ToolWindow, наследуемое от класса DockContent. Это позволило окну ToolWindow стать "плавающим". Далее от класса ToolWindow был наследован MatrixWindow.

# 

# 3.3 Реализация алгоритма удаления ребра графа мышью

Задача: удалить ребро графа, лежащее под курсором мыши (Рис. 3.2).

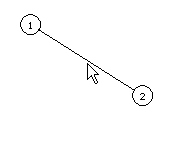


Рисунок 3.2 - Мышь находится рядом с ребром графа, который требуется удалить.

Решение:

Поскольку определять пересечение курсора мыши с линией не удобно, так как при тонкой линии требуется точное позиционирование мыши, по двум точкам строятся два треугольника, которые вместе образуют прямоугольник, который делит пополам ребро графа (рисунок 3.3).

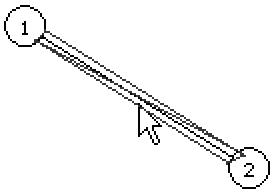


Рисунок 3.3 - Прямоугольник, образованный двумя треугольниками, делит пополам ребро графа. Координаты мыши принадлежат одному из треугольников.

Так задача сводится к одной из классических задач аналитической геометрии – определение принадлежности точки треугольнику.

Рассмотрим алгоритм определения принадлежности точки к одному треугольнику. Для начала необходимо узнать, к какой области принадлежит точка (Рис. 3.4).



Рисунок 3.4. Области, в которых может лежать точка относительно линии.

Этим в классе Graph занимается частный метод pointClassify(Point source, Point dest), который возвращает один из элементов перечисления PointPlace, которое определяет область нахождения точки.

Перечисление PointPlace:

enum PointPlace : int

{

LEFT = 0,

RIGHT = 1,

BEYOND = 3,

BEHIND = 4,

BETWEEN = 5,

ORIGIN = 6,

DESTINATION = 7,

}

Листинг 3.4 – Метод pointClassify класса Graph.

PointPlace pointClassify(PointF point, PointF origin, PointF dest)

{

PointF a = dest;

a.X -= origin.X;

a.Y -= origin.Y;

PointF b = point;

b.X -= origin.X;

b.Y -= origin.Y;

double sa = a.X \* b.Y - b.X \* a.Y;

if (sa > 0.0)

return PointPlace.LEFT;

if (sa < 0.0)

return PointPlace.RIGHT;

if ((a.X \* b.X < 0.0) || (a.Y \* b.Y < 0.0))

return PointPlace.BEHIND;

if (Math.Sqrt(a.X \* a.X + a.Y \* a.Y) < Math.Sqrt(b.X \* b.X + b.Y \* b.Y))

return PointPlace.BEYOND;

if (point == origin)

return PointPlace.ORIGIN;

if (point == dest)

return PointPlace.DESTINATION;

return PointPlace.BETWEEN;

}

Далее достаточно определить лежит ли точка в области, образованной ребрами треугольника. Эту задачу выполняет метод pointInTriangle, который принимает координаты треугольников и точку, принадлежность треугольникам которой следует проверить. Метод возвращает true, если точка принадлежит треугольнику и false в противном случае.

Листинг 3.5 – Метод pointInTriangle класса Graph.

bool pointInTriangle(PointF p, PointF a, PointF b, PointF c)

{

return ((pointClassify(p, a, b) != PointPlace.LEFT) &&

(pointClassify(p, b, c) != PointPlace.LEFT) &&

(pointClassify(p, c, a) != PointPlace.LEFT));

}

Более подробное описание алгоритмов можно найти по следующим ссылкам:

1. http://algolist.manual.ru/maths/geom/belong/poly2d.php
2. http://algolist.manual.ru/maths/geom/datastruct.php

### 3.4 Тестирование реализации алгоритма Брон-Кербоша

Тестирование алгоритма производилось:

* на пустом графе
* на графе с единственной вершиной
* не графе с тремя не соединенными ребрами вершинами
* на графе из двух вершин, соединенных ребром
* на сложном графе

Стратегия тестирования

Сперва, с помощью определения понятия "клика", были найдены клики данного графа, после чего результаты сравнивались с результатом работы программы.

1. Тестирование на пустом графе.

Теоретические расчеты: поскольку граф пуст (множество его вершин есть пустое множество) клик в нем нет.

Практический результат: клик в графе не найдено, получено соответствующее уведомление (рисунок 3.5).

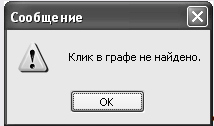


Рисунок 3.5. Сообщение об отсутствии клик в графе.

Результат: теоретические и практические расчеты сходятся – на данном наборе алгоритм работает верно.

2. Тестирование на графе с единственной вершиной.

Теоретические расчеты: граф не содержит клик - подмножество его вершин, такое, что между каждой парой вершин этого подмножества существует ребро и, кроме того, это подмножество не принадлежит никакому большому подмножеству с тем же свойством.

Практический результат: клик в графе не найдено, получено соответствующее уведомление (рисунок 3.5).

Результат: теоретические и практические расчеты сходятся – на данном наборе алгоритм работает верно.

3.Тестирование на графе с тремя не соединенными ребрами вершинами.

Теоретические расчеты: аналогичны расчетом в пункте 2.

Практический результат: клик в графе не найдено, получено соответствующее уведомление (рисунок 3.5).

Результат: теоретические и практические расчеты сходятся – на данном наборе алгоритм работает верно.

4.Тестирование на графе из двух вершин, соединенных ребром.

Теоретические расчеты: граф удовлетворяет понятию "клика".

Практические результаты: найдена одна клика, представляющая собой данный граф. Результат: теоретические и практические расчеты сходятся – на данном наборе алгоритм работает верно.

Тестирование на сложном графе.

В программе был создан граф, представленный на рисунке 3.6.

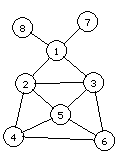


Рисунок 3.6. Сложный граф, используемый в тесте.

Матрица смежности графа:

100011

10111000

11001100

01001100

01110100

00111000

10000000

10000000

Теоретические расчеты: алгоритмом должны быть найдены следующие клики: {1,2,3},{1,7},{1,8},{2,3,5},{2,4,5},{3,5,6},{4,5,6}.Практические результаты: программой был сгенерирован отчет, представленный на листинге 3.6.

Листинг 3.6. Отчет, сгенерированный программой.

Graph untitled.g

Vertices count: 8

Matrix:

100011

111000

001100

001100

110100

111000

000000

000000

Cliques count: 7

Clique 1

Vertices: 1 2 3

Matrix:

1

1

0

Clique 2

Vertices: 1 7

Matrix:

Clique 3

Vertices: 1 8

Matrix:

Clique 4

Vertices: 2 3 5

Matrix:

1

1

0

Clique 5

Vertices: 2 4 5

Matrix:

1

1

0

Clique 6

Vertices: 3 5 6

Matrix:

1

1

0

Clique 7

Vertices: 4 5 6

Matrix:

1

1

0

Результат: алгоритм работает верно.

# 

# 3.5 Системные требования

Требования к аппаратному обеспечению:

Процессор с тактовой частотой 1000 МГц.

Не менее 256 Мб оперативной памяти.

Монитор с разрешением 1024 x 768.

Клавиатура, мышь.

Требования к программному обеспечению:

OS Windows Xp/Vista/7

NET Framework 2.0

### 

### Заключение

На языке программирования C# была выполнена реализация алгоритма Брона-Кербоша по поиску клик в неориентированном графе. Также были реализованы средства создания и редактирования неориентированного графа, а также поиска и отображения его клик и создания отчета о найденных кликах. Также были получены следующие навыки:

* Умение применять и модифицировать opensource-компоненты.
* Навык работы с динамическими структурами данных.
* Навык организации печати документов средствами .NET Framework.

Список использованной литературы и источников

1. Bron C., Kerbosh J. (1973), Algorithm 457 — Finding all cliques of an undirected graph, Comm. of ACM, 16, p. 575—577
2. Cazals, F.; Karande, C. (2008), "A note on the problem of reporting maximal cliques", Theoretical Computer Science 407 (1): 564–568,doi:10.1016/j.tcs.2008.05.010
3. Graph Drawing: Algorithms for the Visualization of Graphs.
4. Drawing Graphs : Methods and Models (Lecture Notes in Computer Science).
5. http://sourceforge.net/projects/dockpanelsuite/
6. http://nzeemin.livejournal.com/184415.html
7. http://algolist.manual.ru/maths/geom/datastruct.php
8. http://algolist.manual.ru/maths/geom/belong/poly2d.php

Приложение

Руководство пользователя

Описание интерфейса

На рисунке 1 представлено главное окно приложения. Цифрами отмечены:

1. Рабочая область приложения.
2. Созданный на рабочей области граф.
3. Главное меню приложения.
4. Панель инструментов приложения.
5. Окно, отображающее матрицу смежности и параметры графа.
6. Матрица смежности графа.
7. Параметры графа.

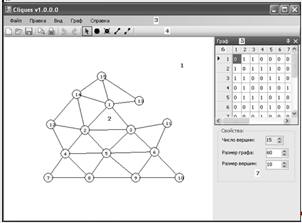


Рисунок 1. Главное окно приложения.

Панель инструментов программы (Рис. 2):

1. Кнопка создание нового документа.
2. Открытие нового документа.
3. Сохранение изменений в документе.
4. Предварительный просмотр документа перед печатью.
5. Кнопка печати документа.
6. Отменить сделанное изменение (если таковое имело быть).
7. Отменить отмену изменение (если таковое имело быть).
8. Инструмент "Курсор".
9. Инструмент "Добавить вершину".
10. Инструмент "Удалить вершину".
11. Инструмент "Добавить ребро".
12. Инструмент "Удалить ребро".



Рисунок 2. Панель инструментов приложения.

Инструменты:

Инструмент "Курсор" необходим для изменения координат вершин графа. Для изменения координат вершины графа достаточно навести курсор на вершину, и зажать левую кнопку мыши. Теперь, пока кнопка не была опущена, координаты вершины будут меняться в соответствии с изменениями координат мыши. Для окончания перемещения графа необходимо отпустить кнопку мыши.

Инструмент "Добавить вершину" создает новую вершину в кликнутой рабочей области. Строка и столбец этой вершины в матрице заполняются нулями.

Инструмент "Удалить вершину" удаляет из графа вершину, по которой кликнули мышью.

Инструмент "Добавить ребро" создает новое ребро графа. Для этого нужно выделить одну из вершин, которой будет принадлежать ребро, мышью и, не отпуская левую кнопку, перетащить курсор на вторую вершину.

Инструмент "Удалить ребро" удаляет ребро графа, по которому кликнули мышью.

На рисунке 3 представлено окно "Граф", задачей которого ставится отображение матрицы и параметров графа. Оно позволяет добавлять и удалять вершины, а также устанавливать или удалять ребра графа. Окно является "плавающим" (Docking), что означает, что оно способно "прилипать" к краям главного окна, а также быть полностью независимым от него. Отобразить или скрыть это окно возможно при помощи функции меню Вид - > Окно свойств.

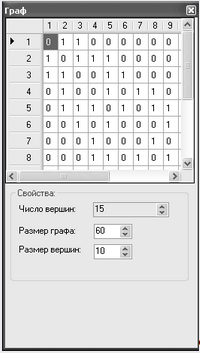


Рисунок 3. Докинговое окно "Граф".

На рисунке 4 представлено окно "Клики графа", задачей которого ставится поиск и отображения всех клик графа, а также сохранение отдельных клик в графический, текстовый или графовый (\*.g) файл, а также создание отчета обо всех найденных кликах. Доступ к окну можно получить из меня Граф -> Операции -> Поиск клик. Следует отметить, что если граф не создан или в нем отсутствуют клики, пользователь не получит доступ к окну, получив соответствующее уведомление.

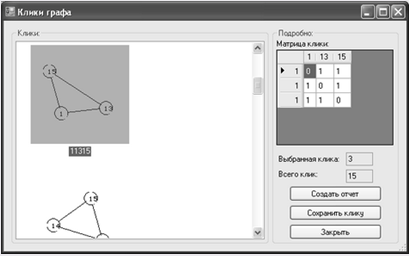


Рисунок 4. Окно "Клики графа".

Назначение кнопок:

Кнопка "Создать отчет" создает отчет обо всех найденных кликах в текстовом формате.

Кнопка "Сохранить клику" сохраняет выбранную клику в одном из выбранных форматах (граф, матрица, графический файл).

Кнопка "Закрыть" закрывает окно и дает доступ к главному окну приложения.

Формат отчета о найденных кликах

Graph: имя\_файла\_графа

Vertices: число\_вершин\_в\_графе

Matrix:Матрица графа

Cliques count: число\_клик\_в\_графе

Clique номер\_клики

Vertices: вершины,\_образующие\_клику

Matrix:Матрица клики.

Кликовый блок содержит число записей, равное числу клик в графе.

Работа с программой

Запуск программы

Запуск программы осуществляется стандартным способом запуска приложений Windows.

Создание графа

Граф создается либо путем манипуляции инструментами, либо из файла матрицы или графа. Для того чтоб создать граф из файла матрицы или графа, следует кликнуть по кнопке "Открыть" панели инструментов, либо щелкнуть по пункту меню Файл - > Открыть. В появившимся диалоговом окне открытия файла выбрать нужный и нажать кнопку "Открыть"

Редактирование графа

Редактирование графа может осуществляться как инструментами, так и при помощи окна "Граф". Данное окно позволят задавать связи между вершинами, а также изменять размер графа.

Отмена сделанных изменений

Для отмены произведенных в графе изменений следует нажать кнопку "Отменить" на панели инструментов, либо воспользоваться горячей клавишей Ctrl+Z. Также эта функция доступна из меню "Правка", пункт "Отменить".

Отмена отмены сделанных изменений

Для отмены отмены произведенных в графе изменений следует нажать кнопку "Повторить" на панели инструментов, либо воспользоваться горячей клавишей Ctrl+Y. Также эта функция доступна из меню "Правка", пункт "Повторить"

Поиск клик в графе

Для вызова окна поиска клик следует обратиться к пункту меню "Граф" - > "Операции с графом" - > "Поиск клик".

Если в графе были найдены клики, появится окно "Клики графа", содержащее все изображения клик, а также их матрицы смежности. В противном случае будет показано уведомление об отсутствии в графе клик.

Создание отчета о найденных кликах

Для создания отчета о найденных кликах следует вызвать окно "Клики графа" (Меню "Граф"->"Операции с графом" ->"Поиск клик"). Если в графе были найдены клики, нажать кнопку "Создать отчет", тем самым, вызвав стандартное диалоговое окно сохранения файла. В нем следует ввести имя создаваемого отчета и нажать на кнопку "Сохранить". Программой будет сгенерирован отчет в текстовом формате, содержащий информацию об исходном графе и всех содержащихся в нем кликах. Также будет создан графический файл в формате png, содержащий все изображения найденных клик.

Сохранение созданного графа

Для сохранения созданного графа следует нажать на кнопку "Сохранить" панели инструментов или воспользоваться горячей клавишей Ctrl+S. Эти действия приведут к вызову стандартного диалога сохранения файла Windows.

Выход из программы

Выход из программы осуществляется по горячей клавише Alt+F4, либо при помощи стандартных методов закрытия оконных приложений Windows. Также завершить работу с программой можно при помощи меню "Файл" -> "Выход". Следует отметить, что если, во время работы с программой, создавался новый или модифицировался уже существующий граф, будет вызвано диалоговое окно с предложением сохранить выполненные изменения.