Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

Кубанский государственный технологический университет

(ФГБОУ ВПО «КубГТУ»)

Факультет компьютерных технологий и автоматизированных систем

Кафедра информационных систем и программирования

**КУРСОВАЯ РАБОТА**

по дисциплине: Дискретная математика

на тему: Задачи и алгоритмы дискретной математики

Выполнила студентка группы 12-КБ-ПИ1 Сысоев Антон Аркадьевич

(Ф.И.О.)

Допущен к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы Е.А.Симоненко

(подпись, дата, расшифровка подписи)

Защищён \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.А.Симоненко

А.Г.Волик

Краснодар

2013

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

Кубанский государственный технологический университет

(ФГБОУ ВПО «КубГТУ»)

Факультет компьютерных технологий и автоматизированных систем

Кафедра информационных систем и программирования

УТВЕРЖДАЮ

Зав. кафедрой ИСиП

профессор Л.А. Видовский

«\_\_\_» 2013 г.

**ЗАДАНИЕ**

на курсовую работу

Студенту группы 12-КБ-ПИ-1 Сысоеву Антону Аркадьевичу

факультета Компьютерных технологий и автоматизированных систем

направления 230700.62 «Прикладная информатика» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема работы: «Задачи и алгоритмы дискретной математики»

Содержание задания: Изучить темы «**Потоки в сетях: алгоритм Форда-Фалкерсона**» и «**Независимое множество вершин», провести исследование алгоритмов и**  **реализовать (написать программы)** на одном из языков программирования.

Объём работы:

а) пояснительная записка \_\_\_\_\_12\_\_\_\_\_ стр.

б) программы 2.

Рекомендуемая литература: Скиена С. «Алгоритмы. Руководство по разработке », Кристофидес Н. «Теория графов. Алгоритмический подход»,

Окулов С.М. «Дискретная математика. Теория и практика решения задач по информатике».

Срок выполнения: с «02» сентября 2013 г. по «21» декабря 2013 г.

Срок защиты: «28» декабря 2013 г.

Дата выдачи задания: «02» сентября 2013 г.

Дата сдачи работы на кафедру: с «02» декабря 2013 г. по «28» декабря 2013 г.

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Задание принял студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**РЕФЕРАТ**

Пояснительная записка к курсовой работе 11 страниц.

**ПОТОКИ В СЕТЯХ, АЛГОРИТМ ФОРДА-ФАЛКЕРСОНА, МНОЖЕСТВО, НЕЗАВИСИМОЕ МНОЖЕСТВО ВЕРШИН.**

В данной курсовой работе рассматриваются две темы из курса «Дискретная математика» - «**Потоки в сетях: алгоритм Форда-Фалкерсона»** и **«Независимое множество вершин».** Цель курсовой работы – реализация алгоритмов решения данных задач на одном из языков программирования.

**Содержание**

[Введение 5](#_Toc375728761)

[Основная часть 7](#_Toc375728762)

[Глава 1. Потоки в сетях: алгоритм Форда-Фалкерсона 7](#_Toc375728763)

[1.1 Формулировка задачи 7](#_Toc375728764)

[1.2Результат выполнения программы: 7](#_Toc375728765)

[Глава 2. Независимое множество вершин 8](#_Toc375728766)

[2.1Формулировка задачи 8](#_Toc375728767)

[2.2Результат выполнения программы: 8](#_Toc375728768)

[Список используемой литературы 9](#_Toc375728769)

[Приложение A – Листинг программы 10](#_Toc375728770)

[Приложение В – Листинг программы 12](#_Toc375728771)

# Введение

В данной курсовой работе рассматриваются два вопроса из курса «Дискретная математика» - «**Потоки в сетях: алгоритм Форда-Фалкерсона**» и «**Независимое множество вершин**». Цель данной курсовой работы состоит в изучении алгоритмов решения задач данного типа и их реализация на одном из языков программирования.

# Основные определения:

***Независимое множество вершин*** (известное также как ***внутренне устойчивое множество***) есть множество вершин графа **G**, такое, что любые две вершины в нем не смежны (никакая пара вершин не соединена ребром).

Независимое множество называется ***максимальным***, когда нет другого независимого множества, в которое оно бы входило.

**Потоком в сети** между вершиной t (источником) и s (стоком) называется набор чисел сij, (т. е. количество условного “груза”, перевозимого из вершины с номером i в вершину с номером j),удовлетворяющих четырем условиям:

1) числа сij Ј  0, причем если сij > 0, то сji = 0 (нет встречных перевозок);

2) числа cij Ј  qij (соответствующих пропускных способностей ребер);

3) если вершина с номером i – промежуточная (не совпадает с источником и стоком), то

http://dvo.sut.ru/libr/himath/w163rabk/224.gif

4)  количество “груза”, вывозимого из источника t, должно быть равно количеству груза, ввозимого в сток s:

http://dvo.sut.ru/libr/himath/w163rabk/225.gif

# Основная часть

# Глава 1. ****Потоки в сетях: алгоритм Форда-Фалкерсона****

# 1.1 Формулировка задачи

С помощью алгоритма Форда-Фалкерсона реализовать задачу о нахождение максимального потока в сети.

# 1.2Результат выполнения программы:

|  |  |
| --- | --- |
| Пример входных данных | Пример выходных данных |
| 6  0 5 | 0 16 0 0 13 0 0 0 12 0 6 0 0 0 0 0 9 20 0 0 7 0 0 4 32 0 0 0 14 0 0 0 0 0 0 0 0 |

Реализация алгоритма в Приложении А

# Глава 2. Независимое множество вершин

# 2.1Формулировка задачи

Найти максимальное независимое множество вершин.

# 2.2Результат выполнения программы:

|  |  |
| --- | --- |
| Пример входных данных | Пример выходных данных |
| 4  0 1 1 1  1 0 0 0  1 0 0 0  1 0 0 0 | 3  2 3 4 |

Реализация алгоритма в Приложении В

# 

# 

# Список используемой литературы

1. Кристофидес Н. Теория графов. Алгоритмический подход. – Пер. с англ. – М.: Мир, 1978. – 432 с.; - глава3.

2. Седжвик Р. Алгоритмы на C++. – Пер. с англ. – М.: Вильямс, 2011. – 1056 с.; - глава 22.2.

3. Скиена С. Алгоритмы. Руководство по разработке. – 2-е изд.: пер. с англ. – СПб.: БХВ-Петербург, 2011. – 720 с.; -глава 16.2 и глава 9.3.2.

4. Окулов С.М. Дискретная математика. Теория и практика решения задач по информатике: учебное пособие. – М.: БИНОМ. Лаборатория знаний, 2008. – 422 с.;- глава 9.2.

# Приложение A – Листинг программы

На языке С++ и откомпилирована в Borland C++Builder 6:

#include <iostream.h>  
#include <memory.h>  
#include <stdio.h>  
#include <conio.h>  
char\* rus (char\* st)  
{  
unsigned char\* p = st;  
while (\*p)  
{  
if (\*p >= 192)  
if (\*p<=239)  
\*p -= 64;  
else  
\*p -= 16;  
p++;  
}  
return st;  
}  
const int MAX\_VERTICES = 40;  
int NUM\_VERTICES;   
const int INFINITY = 10000;   
int f[MAX\_VERTICES][MAX\_VERTICES];  
int c[MAX\_VERTICES][MAX\_VERTICES];  
int Flow[MAX\_VERTICES];  
int Link[MAX\_VERTICES];   
int Queue[MAX\_VERTICES]; int QP, QC;

int FindPath(int source, int target)   
{  
QP = 0; QC = 1; Queue[0] = source;  
Link[target] = -1; // особая метка для стока  
int i;  
int CurVertex;  
memset(Flow, 0, sizeof(int)\*NUM\_VERTICES);   
Flow[source] = INFINITY;   
while (Link[target] == -1 && QP < QC)  
{  
CurVertex = Queue[QP];  
for (i=0; i<NUM\_VERTICES; i++)  
if ((c[CurVertex][i] - f[CurVertex][i])>0 && Flow[i] == 0)   
{  
Queue[QC] = i; QC++;  
Link[i] = CurVertex;   
if (c[CurVertex][i]-f[CurVertex][i] < Flow[CurVertex])  
Flow[i] = c[CurVertex][i];  
else  
Flow[i] = Flow[CurVertex];  
}  
QP++;   
}  
if (Link[target] == -1) return 0;   
CurVertex = target;  
while (CurVertex != source)   
{  
f[Link[CurVertex]][CurVertex] +=Flow[target];  
CurVertex = Link[CurVertex];  
}  
return Flow[target];   
}  
int MaxFlow(int source, int target) // source - исток, target - сток  
{  
memset(f, 0, sizeof(int)\*MAX\_VERTICES\*MAX\_VERTICES);

int MaxFlow = 0;   
int AddFlow;  
do  
{  
AddFlow = FindPath(source, target);  
MaxFlow += AddFlow;  
} while (AddFlow >0);   
return MaxFlow;  
}  
int main()  
{  
printf(rus("\n НАХОЖДЕНИЕ МАКСИМАЛЬНОГО ПОТОКА \n"));  
printf(rus("\n АЛГОРИТМ ФОРДА-ФАЛКЕРСОНА \n\n"));  
printf(rus("\n КУРСОВАЯ РАБОТА ПО ДИСКРЕТНОЙ МАТЕМАТИКЕ \n"));  
printf(rus("\n выполнил: Сысоев А.А. \n\n"));  
printf(rus("\n\n нажмите любую клавишу для продолжения...."));  
getch();  
clrscr();  
int source, target;  
printf(rus("\n Введите число вершин в графе\n-->"));  
scanf("%d", &NUM\_VERTICES);  
printf(rus("\n Введите значения истока и стока \n-->"));  
scanf("%d %d", &source, &target);  
printf(rus("\n Введите матрицу содержащею вместимость ребер: \n "));  
printf(rus("каждый элемент - вместимость ребра ведушего \n из вершины с номером его строки к вершине с номером его столбца\n"));  
int i, j;  
for (i=0; i<NUM\_VERTICES; i++)  
for (j=0; j<NUM\_VERTICES; j++)  
scanf("%d",&c[i][j]);  
printf(rus("\n максимальный поток равен:"));  
printf("%d", MaxFlow(source, target));  
getch();  
return 0;  
}

# Приложение В – Листинг программы

На языке Java:

import java.util.\*;

  static void findMaximumIndependentSet(List<Integer> cur, List<Integer> result, boolean[][] graph, int[] oldSet,

      int ne, int ce) {

    int nod = 0;

    int minnod = ce;

    int fixp = -1;

    int s = -1;

    for (int i = 0; i < ce && minnod != 0; i++) {

      int p = oldSet[i];

      int cnt = 0;

      int pos = -1;

      for (int j = ne; j < ce; j++)

        if (graph[p][oldSet[j]]) {

          if (++cnt == minnod)

            break;

          pos = j;

        }

      if (minnod > cnt) {

        minnod = cnt;

        fixp = p;

        if (i < ne) {

          s = pos;

        } else {

          s = i;

          nod = 1;

        }

      }

    }

    int[] newSet = new int[ce];

    for (int k = minnod + nod; k >= 1; k--) {

      int sel = oldSet[s];

      oldSet[s] = oldSet[ne];

      oldSet[ne] = sel;

      int newne = 0;

      for (int i = 0; i < ne; i++)

        if (!graph[sel][oldSet[i]])

          newSet[newne++] = oldSet[i];

      int newce = newne;

      for (int i = ne + 1; i < ce; i++)

        if (!graph[sel][oldSet[i]])

          newSet[newce++] = oldSet[i];

      cur.add(sel);

      if (newce == 0) {

        if (result.size() < cur.size()) {

          result.clear();

          result.addAll(cur);

        }

      } else if (newne < newce) {

        if (cur.size() + newce - newne > result.size())

          findMaximumIndependentSet(cur, result, graph, newSet, newne, newce);

      }

      cur.remove(cur.size() - 1);

      if (k > 1)

        for (s = ++ne; !graph[fixp][oldSet[s]]; s++)

          ;

    }

  }

  public static List<Integer> maximumIndependentSet(boolean[][] graph) {

    int n = graph.length;

    int[] all = new int[n];

    for (int i = 0; i < n; i++)

      all[i] = i;

    List<Integer> res = new ArrayList<Integer>();

    findMaximumIndependentSet(new ArrayList<Integer>(), res, graph, all, 0, n);

    return res;

  }

  // Usage example

  public static void main(String[] args) {

    int n = 4;

    boolean[][] g = new boolean[n][n];

    // create a simple cycle

    g[0][1] = g[1][0] = true;

    g[1][2] = g[2][1] = true;

    g[2][3] = g[3][2] = true;

    g[3][0] = g[0][3] = true;

    List<Integer> res = maximumIndependentSet(g);

    List<Integer> expectedResult = new ArrayList<Integer>();

    Collections.addAll(expectedResult, 0, 2);

    System.out.println(expectedResult.equals(res));

  }

}