Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

ФГОУ ВПО Кубанский государственный технологический университет

(КубГТУ)

Кафедра Информационных систем и программирования

Факультет Компьютерных технологий и автоматизированных систем

**КУРСОВАЯ РАБОТА**

По дисциплине Дискретная математика

На тему Задачи и алгоритмы дискретной математики.

Выполнил студент группы 12-КБ-ПИ1 Батлук Александр Павлович

Допущен(а) к защите :

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.А.Симоненко

(подпись, дата, расшифровка подписи)

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

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Е.А.Симоненко

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Г. Волик

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата, расшифровка подписи)

Краснодар

2013

ФГОУ ВПО Кубанский государственный технологический университет

(КубГТУ)

Кафедра Информационных систем и программирования

Факультет Компьютерных технологий и автоматизированных систем

УТВЕРЖДАЮ

Зав. кафедрой Видовский Л.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, подпись, расшифровка подписи)

**ЗАДАНИЕ**

на курсовую работу

Студенту Батлуку Александру Павловичу группы 12-КБ-ПИ1

факультета Компьютерных технологий и автоматизированных систем

направления 230700 – Прикладная информатика

Тема работы Задачи и алгоритмы дискретной математики.

Содержание задания: Изучить темы: Перестановки. Генерация перестановок в лексикографическом порядке. Получение перестановки по её номеру. Получение номера перестановки. Постановка задачи и алгоритм нахождения разрывающего множества вершин и рёбер.

Объём курсовой работы:

а) пояснительная записка стр. 10;

Рекомендуемая литература Уайс «Алгоритмы», Скиена «Алгоритмы».

Срок выполнения курсовой: с 2 сентября 2013г. по 21 декабря 2013г.

Срок защиты: с 2 декабря 2013г. по 28 декабря 2013г.

Дата выдачи задания: 2 сентября 2013г.

Дата сдачи работы на кафедру: с 2 декабря 2013г. по 28 декабря 2013г.

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

(подпись)

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

(подпись)

РЕФЕРАТ

АЛГОРИТМ, КОМБИНАТОРИКА, ГРАФЫ, ПОИСК В ГЛУБИНУ, ПЕРЕСТАНОВКИ, ПРОЦЕДУРЫ, ФУНКЦИИ, C#, PASCAL, ДИНАМИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Стр. 10, рис. 2, библ. 1.

В данной курсовой работе мы рассматриваем задачи «Перестановки, Генерация перестановок в лексикографическом порядке» и «Разрывающее множество графа»

Основным моментом проведенного исследования были практическое применение алгоритмов.

Данная работа сделала возможным оценить каждый алгоритм с практической точки зрения и спроектировать задачи на различных языказ программирования

Содержание

Введение..................................................................................................................5

Перестановки................................................................................................6

Разрывающее множество.............................................................................7

Список использованных источников....................................................................8

Приложение А.........................................................................................................9

Приложение B........................................................................................................10

Введение

В данной курсовой работе рассмотрены два вопроса, касающихся программирования. Первый из них – Перестановки . Второй – Решение задачи разрывающего множества, методом поиска в глубину.

Данные задачи относятся к числу основных задач курса дискретной математики. Генерация перестановок — очень полезный алгоритм, используемый во множестве других задач. Следовательно приведенная задача имеет смысл при использовании динамического программирования при решении других задач.

Задача на граф является задачей повышенной сложности, при ее решении были применены знания, использованные при решении прочих задач, данная тема мало изучен и ее решение полезно и важно

1. Перестановки

Задача 1.

Постановка задачи:

Разработайте программу, которая генерирует все перестановки из n элементов в лексикографическом порядке.

Алгоритм:

1. Просматриваем а1, ..., аn с конца до тех пор, пока не попадется ai<ai+1. Если таковых нет, то генерация закончена.  
2. Рассматриваем ai+1, ai+2, ..., an. Найдем первый с конца am больший ai и поменяем их местами.  
3. ai+1, ai+2, ..., an переставим в порядке возрастания (для этого достаточно её переписать с конца).  
4. Печатаем найденную перестановку.  
5. Возвращаемся к пункту 1.

Первой при этом будет перестановка <1,2,..,n>, последней - <n,..,2,1>.

Входные данные:

Реализация алгоритма в Приложении А.  
В с++ есть специальная возможность — Next.Permulation, позволяющая генерировать следующую перестановку в лексикографическом порядке. В данной задаче рассмотрен данный алгоритм на языке программирования c++.

Входные данные: количество и сами элементы перестановки  
 Выходные данные: перестановки в лексикографическом порядке, расположенные в столбец![](data:None;base64,)

Рисунок 1: Пример генерации

**2. Разрывающее множество**

Задача 2.

**Постановка задачи:**

Найти наименьший набор ребер Е' или вершин V', удаление которого сделает граф ациклическим

**Алгоритм решения:**

Для поиска разрывающего множества ребер в неориентированном графе, нужно найти максимальный ациклический подграф, и те ребра, которые не вошли в него и будут разрывающим множеством

1) Совершаем обход (поиск) в глубину

Поиск в глубину ([англ.](http://ru.wikipedia.org/wiki/Английский_язык) Depth-first search, DFS) — один из методов обхода [графа](http://ru.wikipedia.org/wiki/Граф_(математика)). Алгоритм поиска описывается следующим образом: для каждой непройденной вершины необходимо найти все непройденные смежные вершины и повторить поиск для них. Используется в качестве подпрограммы в алгоритмах поиска одно- и двусвязных компонент, топологической сортировки.

2) Сохраняем вершины в порядке обхода

3) Сравниваем полученный в результате граф и исходным

4) Те ребра, которые отсутствуют — и будут являться искомым разрывающим множеством

Для решения задачи представим граф:

![](data:None;base64,)

Рисунок 3 — Входные/выходные данные

**Список используемых источников**

1. [Скиена] Скиена С. Алгоритмы. Руководство по разработке. – 2-е изд.: пер. с англ. – СПб.: БХВ-Петербург, 2011. – 720 с. Скиена «Алгоритмы».
2. [Седжвик] Седжвик Р. Алгоритмы на C++. – Пер. с англ. – М.: Вильямс, 2011. – 1056 с.
3. [Окулов: ДМ] Окулов С.М. Дискретная математика. Теория и практика решения задач по информатике: учебное пособие. – М.: БИНОМ. Лаборатория знаний, 2008. – 422 с.

**Приложение А**

**Исходный код –**

*#*include <iostream>

using namespace std;

int M[]={1,2,3,3,4,5,5,5,6}; //к примеру

void Sort(int \*A,int n);

void swap(int &a,int &b)

{

int temp=a;

a=b;

b=temp;

}

void Obrabotka(int \*A,int n)

{

for(int i=0;i<n;i++)cout<<A[i]<<" ";

cout<<endl;

}

void Perestanovki(int \*A,int n)

{

Sort(A,n);

int pos; // позиция первого элмента, нарушающего порядок убывания (с конца)

do

{

pos=n-2;

Obrabotka(A,n);

for(int i=n-1;pos>=0&&A[pos]>=A[i];i--)pos--; //находим pos

int j=n-1; //элемент для обмена

while(pos>=0&&A[pos]>=A[j])j--;

if(pos>=0)swap(A[pos],A[j]);

//разворачиваем часть R

int l=pos+1,r=n-1;

while(l<r)

{

swap(A[l],A[r]);

l++;

r--;

}

}while(pos>=0);

}

int main()

{

Perestanovki(M,9);

return 0;

}

**Приложение B**

**Исходный код –**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Graphos

**{**

class Program  
 {  
 static int[,] baseGraph = new int[4, 4]  
 {  
 {0,0,1,1},  
 {0,0,1,1},  
 {1,1,0,1},  
 {1,1,1,0}  
 };  
  
static int[] colors = new int[4];  
  
static void DFS(int currentVertex)  
 {  
 colors[currentVertex - 1] = 1;  
 for (int i = 1; i < 5; i++)  
 {  
 if (baseGraph[currentVertex - 1, i - 1] == 1 && colors[i - 1] == 0)  
 {  
 baseGraph[currentVertex - 1, i - 1] = 0;  
 baseGraph[i - 1, currentVertex - 1] = 0;  
 DFS(i);  
 }  
 }  
 colors[currentVertex - 1] = 2;  
 }  
  
 static void Main(string[] args)  
 {  
 DFS(1);  
 for (int i = 0; i < 4; i++)  
 {  
 for (int j = i; j < 4; j++)  
 {  
 if (baseGraph[i, j] == 1)  
 {  
 Console.WriteLine((i + 1).ToString() + " - " + (j + 1).ToString());  
 }  
 }  
 }  
 }  
}

}