Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

Кубанский государственный технологический университет

(ФГБОУ ВПО «КубГТУ»)

Кафедра информационных систем и программирования

Факультет компьютерных технологий и автоматизированных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

по дисциплине Дискретная математика

(наименование дисциплины)

на тему Задачи и алгоритмы дискретной математики

(тема курсового проекта (работы))

Выполнила студентка группы 12-КБ-ПИ1

        Дроздова Татьяна Геннадьевна

(ф.и.о.)

Допущена к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы Е.А.Симоненко

Защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

(подпись, дата, расшифровка подписи)

Краснодар

2013

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

Кубанский государственный технологический университет

(ФГБОУ ВПО «КубГТУ»)

Кафедра  ИСП

(наименование кафедры)

УТВЕРЖДАЮ

Зав. кафедрой ИСП

профессор \_\_\_\_\_\_\_\_Л.А.Видовский

« » \_\_\_\_\_\_\_\_\_\_ 2013 г.

З А Д А Н И Е

на курсовое проектирование

Студентке: Дроздовой Татьяне Геннадьевне группы 12-КБ-ПИ1, 2 курса

(Ф.И.О.) (№ группы и курса)

факультета компьютерных технологий и автоматизированных систем

специальности 230700 (Прикладная Информатика)

(шифр и наименование)

Тема работы: Задачи и алгоритмы дискретной математики

Содержание задания:изучить темы «Укладка множества» и **«Эйлеров цикл», провести исследование алгоритмов и  реализовать (написать программы)** на одном из языков программирования.

Объём курсовой работы:

а) пояснительная записка;

б) приложение.

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 работы: с "\_\_\_" \_\_\_\_\_\_по"\_\_\_"\_\_\_\_20\_\_г.

Срок защиты:                      "\_\_\_"\_\_\_\_20\_\_г.

Дата выдачи задания:                 "\_\_\_"\_\_\_\_20\_\_г.

Дата сдачи работы на кафедру:           "\_\_\_"\_\_\_\_20\_\_г.

Руководитель работыЕ.А.Симоненко

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

(подпись, дата)

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего профессионального образования

Кубанский государственный технологический университет

агентство по образованию

(ФГБОУ ВПО КубГТУ)

**Реферат**

Пояснительная записка курсовой работы 17 с., 4 источника.

МНОЖЕСТВА, АЛГОРИТМ, ПОИСК ПУТИ, ПОИСК ЭЙЛЕРОВА ЦИКЛА, ГРАФ

В данной курсовой работе рассматриваются две темы из курса «Дискретная математика»- «Укладка множества» и «Эйлеров цикл». Цель курсовой работы – реализация алгоритмов решения данных задач на одном из языков программирования.

# Содержание

[Введение 5](#_Toc375858958)

[1 Нормативные ссылки 6](#_Toc375858959)

[Основная часть 7](#_Toc375858960)

[Глава 1. Укладка множества 7](#_Toc375858961)

[1.1 Формулировка задачи 7](#_Toc375858962)

[1.2 Алгоритм укладки множества вещей в ранец 7](#_Toc375858963)

[Глава 2. Эйлеров цикл 10](#_Toc375858964)

[2.1 Формулировка задачи 10](#_Toc375858965)

[2.2 Алгоритм поиска эйлерова цикла (Флёри) 10](#_Toc375858966)

[Список используемой литературы 12](#_Toc375858967)

[Приложение А. Листинг программы 13](#_Toc375858968)

[Приложение Б. Листинг программы 17](#_Toc375858969)

# Введение

В данной курсовой работе рассматриваются два вопроса из курса «Дискретная математика» - «Укладка множества» и «Эйлеров цикл».

Множество есть совокупность различных элементов, мыслимая как единое целое. Задачи укладки множества возникают в приложениях, имеющих строгие ограничивающие условия на разрешенное разбиение. Главной особенностью задач укладки множества является условие, согласно которому ни один элемент не может быть покрыт больше, чем одним выбранным подмножеством.

Эйлеров цикл – это такой цикл, который проходит ровно один раз по каждому ребру, возвращаясь в вершину, из которой начали путь. Граф содержит эйлеров цикл, если он связный и все его вершины имеют четную степень, либо две его вершины имеют нечетную степень.

# 1 Нормативные ссылки

В пояснительной записке использованы ссылки на следующие государственные стандарты:

* ГОСТ Р 1.5-2004. Стандарты национальные РФ. Правила построения,

изложения, оформления и обозначения;

* ГОСТ 2.301-68 ЕСКД. Форматы;
* ГОСТ Р 7.0.5-2008 СИБИД. Библиографическая ссылка. Общие

требования и правила составления;

* ГОСТ 7.12-93 СИБИД. Библиографическая запись. Сокращения слов

на русском языке. Общие требования и правила;

* ГОСТ 7.9-95 СИБИД. Реферат и аннотация. Общие требования;
* ГОСТ 7.82-2001 СИБИД. Библиографическая запись.

Библиографическое описание электронных ресурсов. Общие

требования и правила составления.

# Основная часть

# Глава 1. Укладка множества

## 1.1 Формулировка задачи

Имеется nгрузов. Для каждого i-го груза определён вес wi> 0 и ценность ci> 0, i = 1, 2, …, n. Нужно упаковать в рюкзак ограниченной грузоподъёмностиW те грузы xi, при которых суммарная ценность упакованного рюкзака была бы максимальной.

## 1.2 Алгоритм укладки множества вещей в ранец

Если все числа wi, ci, W натуральные, то мы можем решить эту задачу при помощи динамического программирования следующим образом:

Рассмотрим для пары чисел (K,S) такой, что 0 ≤ K ≤ N, 0 ≤ S ≤ W, задачу P(K, S): найти набор вещей максимальной суммарной стоимости при условии, что мы можем выбирать из первых K вещей, и суммарный вес выбранных вещей не должен превосходить S. Ясно, что наша исходная задача — это в точности P(N, W).

Пусть A(K, S) — оптимальная суммарная стоимость для задачи P(K,S). Задача P(0,S) тривиальна: мы не можем взять ни одной вещи, следовательноA(0, S) = 0. Решим теперь задачу P(K+1, S) при условии, что известны решения задач P(K,T) для всех T. Заметим, что все наборы из вещей 1, …, K+1 суммарным весом не более S можно разделить на две группы:

1. Наборы, в которые не входит вещь K+1. Такие наборы есть в точности наборы из вещей 1, …, K суммарным весом не более S. Наибольшая стоимость такого набора равна A(K, S).
2. Наборы, в которые входит вещь K+1. Любой такой набор получается добавлением этой вещи к набору из вещей 1,…, K суммарным весом не более S-wK+1. Наибольшая стоимость такого набора равна A(K,S-wK+1) + cK+1. (Заметим, чтоэтотслучайвозможентолькокогда S ≥ wK+1.)

Следовательно, искомая величина A(K+1, S) есть максимум из наибольших стоимостей для двух случаев, т. е. A(K+1, S) = max(A(K,S), A(K,S-wK+1) + cK+1). Мы получаем следующий алгоритм:

Brifcase(N, W, C, w)

**для**S**от** 0 **до**W

A[0][S] ← 0

**для**K**от** 1 **до**N

**для**S**от** 0 **до**W

A[K][S] ← A[K-1][S]

**если** S ≥ w[K], **то**

A[K][S] ← max(A[K][S], A[K-1][S-w[K+1]] + c[K+1])

**вернуть**A[N][W]

Найдем теперь какой-либо оптимальный набор вещей. Для этого заметим, что оптимальный набор для задачи P(K,S) включает в себя вещь K тогда и только тогда, когда A(K,S)>A(K-1,S). Значит, достаточно дописать к нашему алгоритму такой код:

S←W

**для**K**от**N**до** 1

**если** A[K][S] > A[K-1][S], **то**

**вывести** K

S ← S – w[K]

Пример входных данных

Кол-во товара:3

Вес и цена первого товара: 6, 20

Вес и цена второго товара: 4, 35

Вес и цена третьего товара: 5, 40

Вместимость рюкзака: 12

Пример выходных данных

Максимальная ценность: 75

Взяты предметы: 2, 3

Реализация алгоритма в Приложении А

# Глава 2. Эйлеров цикл

## 2.1 Формулировка задачи

Требуется найти такой цикл, который проходит ровно один раз по каждому ребру, возвращаясь в вершину, из которой начали путь.

## 2.2 Алгоритм поиска эйлерова цикла (Флёри)

Начиная с некоторой вершины, произвольным образом «идем» по смежным ребрам графа, удаляя пройденное ребро и вершину, ставшую после удаления ребра изолированной. Не проходим по ребру, если после его удаления граф становится несвязным.

Процесс завершается после обхода всех ребер, а значит и вершин графа, и обязательно в исходной вершине, степень которой к этому моменту становится нулевой. Порядок, в котором были пройдены ребра, определяет конфигурацию найденного цикла.

Еcли граф задан списком смежности, то формализованная запись алгоритма имеет вид:

begin

selectv∈V; // Выбрать произвольную вершину

v→S;// Занести ее в стек

whileS≠ 0 do// Пока стек не пуст, выполнять действия:

{

ifG(s1) ≠ 0 // если окружение вершины s1, находящейся в

//верхушке стека не пусто, то

{

then

{

selectu∈G(s1); //Выбрать одну из ее «соседок»

u→S; //занести ее в стек

G(s1) := G(s1) – {u}; // а затем удалить пройденное ребро

G(u):= G(u) – {s1};

}

else// иначе:

{

S→v; // получить из стека

output(v);// и вывести очередную вершину цикла

}

}

}

end.

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

Реализация алгоритма в Приложения В

# Список используемой литературы

1.Кристофидес Н. Теория графов. Алгоритмический подход. – Пер. с англ. – М.: Мир, 1978. – 432 с.;-глава9.5.

2. Седжвик Р. Алгоритмы на C++. – Пер. с англ. – М.: Вильямс, 2011. – 1056 с.;-глава 17.7.

3.Скиена С. Алгоритмы. Руководство по разработке. – 2-е изд.: пер. с англ. – СПб.: БХВ-Петербург, 2011. – 720 с.;-глава 18.2 и глава 15.7.

4. Домнин Л.Н. Элементы теории графов: учеб.пособие – Пенза: Изд-во Пенз. гос. ун-та, 2007 – 144с.;- глава 5.2.

# ПриложениеА.Листингпрограммы

usingSystem;

namespacebreafcase

{

class Article //товар

{

public string name; //названиетовара

public int weight; //вестовара

public int price; //ценатовара

publicbool take; //беремлитовар

public Article(string n, int w, int p)

{

name = n;

weight = w;

price = p;

}

}

class Program

{

public static Article[] articles;

public static int numsArticle; // Количествотоваров

publicstaticboolincorrEnter; //Переменная, что проверяет корректность ввода

public static string name; // Имятовара

public static int weight, price, maxWeight; // Весиценатовара, размеррюкзака

publicstatic int[,] func; //массив для хранения значений функции

// Метод выводит на экран рюкзак

staticvoidPrint()

{

Console.WriteLine("\nМаксимальнаястоимость: " + func[maxWeight, numsArticle - 1]);

Console.Write("Взяты следующие предметы: ");

foreach (Article x in articles)

if (x.take)

Console.Write(x.name + " ");

}

static void Main(string[] args)

{

int i, j; //просто переменные :)

Console.WriteLine("Данная программа поможет Вам заполнить Ваш рюкзак максимально ценными товарами\n");

//Введем количество товаров

do

{

incorrEnter = false;

Console.Write("\nКакое количество товаров вы рассматриваете для приобретения: ");

try

{

numsArticle = Convert.ToInt32(Console.ReadLine());

}

catch (FormatException)

{

incorrEnter = true;

numsArticle = 0;

}

} while (incorrEnter);

articles = new Article[numsArticle]; //Создаеммассивзаданногоразмера

//Создадимобъектытоваров

for (i = 0, j = i + 1; i <articles.Length; i++, j++)

{

Console.Write("\n\nВведите название " + j + "-го товара: ");

name = Console.ReadLine();

do

{

incorrEnter = false;

Console.Write("Введите вес товара: ");

try

{

weight = Convert.ToInt32(Console.ReadLine());

}

catch (FormatException)

{

incorrEnter = true;

weight = 0;

}

} while (incorrEnter);

do

{

incorrEnter = false;

Console.Write("Введите цену товара: ");

try

{

price = Convert.ToInt32(Console.ReadLine());

}

catch (FormatException)

{

incorrEnter = true;

price = 0;

}

} while (incorrEnter);

articles[i] = new Article(name, weight, price); //Создаемобъект

}

// Введемразмеррюкзака

do

{

incorrEnter = false;

Console.Write("\nВведите размер вашего рюкзака (контейнера): ");

try

{

maxWeight = Convert.ToInt32(Console.ReadLine());

}

catch (FormatException)

{

incorrEnter = true;

maxWeight = 0;

}

} while (incorrEnter);

func = new int[maxWeight + 1, numsArticle]; //Реализуеммассивфункции

//РеализуемалгоритмБеллмана

for (weight = 1; weight <= maxWeight; weight++) // Загружаемрюкзакеслиеговместимость = Weight

for (i = 1; i <numsArticle; i++) // беремпредметыс 1 поnumsArticle

//если вес предмета больше Weight, или предыдущий набор лучше выбираемого

if (articles[i].weight > weight)

{

func[weight, i] = func[weight, i - 1]; //тогдаберемпредыдущийнабор

articles[i].take = false;

}

else if (func[weight, i - 1] >= (func[weight - articles[i].weight, i - 1] + articles[i].price))

{

func[weight, i] = func[weight, i - 1]; //тогдаберемпредыдущийнабор

articles[i].take = false;

}

else

{

func[weight, i] = func[weight - articles[i].weight, i - 1] + articles[i].price; //иначедобавляемкпредыдущемунаборутекущийпредмет

articles[i].take = true;

}

Print();

Console.ReadKey();

}

}

## Приложение Б. Листинг программы

using System;

usingSystem.Collections.Generic;

usingSystem.Linq;

usingSystem.Text;

namespace

ConsoleApplication1

{

classProgram

{

staticvoid Main(string[] args)

{

List<int>[] a = newList<int>[4];

a[0] = newList<int> { 1, 1 };

a[1] = newList<int> { 0, 0, 2, 3 };

a[2] = newList<int> { 1, 3 };

a[3] = newList<int> { 1, 2 };

int v = 0;

Stack<int> S = newStack<int>();

S.Push(v);

while (S.Count != 0)

{

if (a[v].Count != 0)

{

S.Push(a[v][0]);

v = a[v][0];

a[a[v][0]].Remove(v);

a[v].RemoveAt(0);

}

else

{

v = S.Pop();

Console.WriteLine(v);

}

}

}

}

}