Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Санкт-Петербургский государственный морской технический университет»

Статья на тему

«Применении теории графов для решения некоторых информатических задач»

Выполнил студент гр. 2250. Синцова У.А.

Руководитель преподаватель УрГЭУ Синцова С.Г

г. Санкт-Петербург,2018 г

На сегодняшний момент тяжело представить современный мир без тех технологий, которыми мы пользуемся. Каждая компания и организация использует различные технические устройства, множество систем работает по написанным программам. В связи с этим очень популярна стала такая специальность, как программирование и соответствующих специалистах. Данная специальность требует определенных знаний, а именно знаний языков программирования, алгоритмов информатики, но все ли помнят, что информатика основывается в первую очередь на математике.

Дискре́тная матема́тика — часть математики, изучающая дискретные математические структуры, в спектр рассматриваемых тем входит: логика, теория множеств, алгоритмы, теории графов, деревья, комбинаторика, алгебра, включающая группы, полугруппы, решетки и т.п.

Это означает, что для изучения основополагающих алгоритмов, используемых компьютерными специальностями, студентам понадобится солидный опыт в решении таких задач. Действительно в большинстве университетов курс дискретной математики на уровне бакалавриата является неотъемлемой частью обучения в области компьютерной науки.

Здесь мы будем опираться на раздел дискретной математики теория графов, в которой используются ориентированные графы, циклы, пути Эйлера, пути Гамильтона, плоские и взвешенные графы.

Рассмотрим некоторые случаи приближенные к программированию, основанные на теории графов.

В современном мире так же важную роль играет кодирование информации. Чтобы передать код по каналу передачи информации без потерь используются, так называемые, статистические коды, которые позволяют повысить производительность источника дискретных сообщений. Одним из методов кодирования является метод Хаффмана.

Код по методу Хаффмана строится следующим образом:

1. Распределяем сообщения из ансамбля по убыванию его вероятностей.
2. Находим два сообщения с наименьшими вероятностями соединяем их и считаем их общую вероятность.
3. Пункт 2 повторяется до тех пор, пока общая вероятность не будет равна 1.
4. Затем рассматриваются два смежных ребра графа и ребру с большей вероятностью ставится 1, а ребру с меньшей 0.
5. Итоговый код записывается, как путь от конечной вершины до нужной вероятности, при этом записываются веса ребер, а не наименования вершин.

Рассмотрим некоторую конкретную задачу из курса теории информации.

Дано условие: Дискретный источник сообщений выдает сообщения из ансамбля $\{X_j\}$,где j = 1,2,…,N с вероятностями (см. таблицу)

Xj	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}
$P(x_j)$	0,011	0,146	0,102	0,015	0,192	0,015	0,041	0,079	0,033	0,034	0,028

Продолжение

X_{12}	X_{13}	X_{14}	X_{15}	X_{16}	X_{17}	X_{18}
0,008	0,108	0,048	0,050	0,027	0,012	0,051

Закодировать данное сообщение кодом Хаффмана. Определить среднюю длину кодовой комбинации, среднее количество информации, содержащееся в одном элементе кода и избыточность кода. (При определении минимальной средней длины кодовой комбинации следует воспользоваться приближенной формулой).

Решение

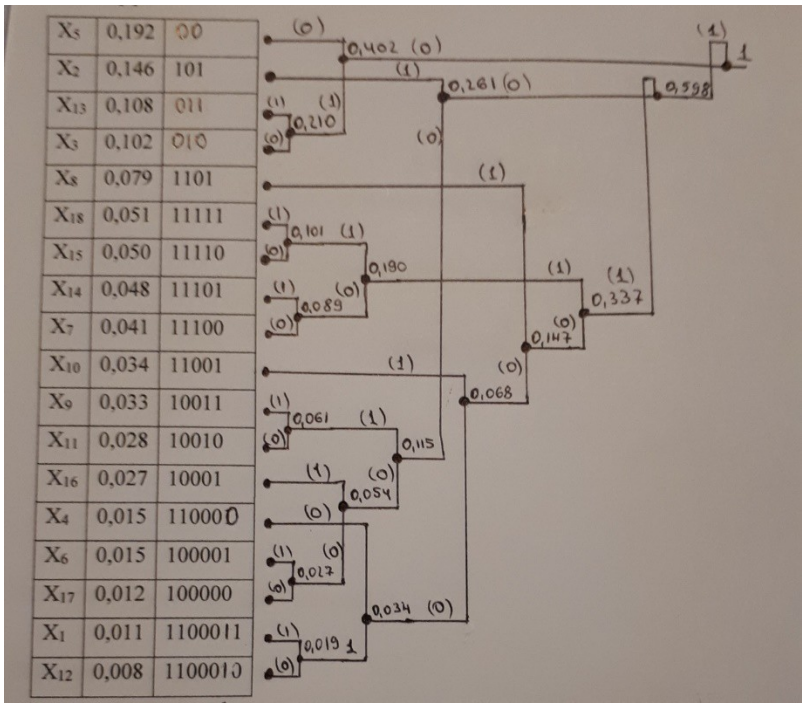

Рисунок 1 – построение графа Хаффмэна.

Средняя длина кодовой комбинации определяется по формуле:

$$n_{cp} = \sum_{i=1}^{m} n_i * p(x_i)$$

$n_{ср}$=0,3840+0,4380+0,3240+0,3060+0,3160+0,2550+0,2500+0,1700+0,1650+0,1400+0,1350+

+0,0900+0,0900+0,0720+0,0770+0,0560=3,713 элемента

Количество информации, приходящееся на 1 элемент кода, считается по формуле:

$$I_{cp}(x)=\frac{H(x)}{n_{cp}}=\frac{-\sum_{i=1}^{m} P(x_i)\cdot \log P(x_i)}{\sum_{i=1}^{m} n_i * p(x_i)}$$

$I_{ср=}$3,667/3,713=0,9878 бит;

Минимальное среднее количество элементов считается по формуле:

$$n_{min}{}^{2}=\frac{H(x)}{logK}$$

$n_{min}{}^{2}$=3,667712/1=3,667712 элемента;

Рассчитаем избыточность по формуле:

$$R=1-\frac{n_{min}{}^{2}}{n_{cp}}$$

R=1-3,667/3,713=0,0122.

Рассмотрим еще одно задание, тесно связанное с графами. В объектно-ориентированном программировании есть три основных принципа это наследование, инкапсуляция и полиморфизм. Здесь мы рассмотрим наследование. Наследование – это процесс, посредством которого один объект может приобретать свойства другого объекта. И сразу рассмотрим пример:

Допустим нам нужно написать программу, которая создает архив документов при этом документы делятся на приказы и письма и у каждого из них есть свои свойства, при этом они наследуют родительские свойства объектов. Чтобы лучше понять иерархию классов построим граф наследования.

ДОКУМЕНТ

ПРИКАЗЫ ПИСЬМА

Представим код программы решающее эту задачу:

```
public class Document //Создание материнского класса Документ
   {
      public DateTime docDate;
      public String docNumber;
      public String docScannedFile;


      public Document(DateTime date, String number)
      {
         docDate = date;
         docNumber = number;
         docScannedFile = null;
      }
      public void attachScanedFile(String newFileName)
      {
         docScannedFile = newFileName;
      }
      public virtual String toString()
      {
         return this.GetType().Name + "Дата: " + docDate.ToString() + ", номер: " + docNumber;
} }
```

public class Order : Document // создание дочернего класса приказ, который наследует свойства материнского класса документ и при этом имеет свои персональные свойства//

```
   {//В программе всем свойствам объектов присваиваются рандомные значения//
      public String issuer;
      public String otvetstvenny;
      public Order(DateTime date, String number, String issuedBy, String otv)
```

```
        : base(date, number)
    {
        issuer = issuedBy;
        otvetstvenny = otv;
    }
    public override string toString()
    {
        String result = base.ToString();
        return result + "Отправитель: " + issuer + "Получатель: " + otvetstvenny;
    }}
public class Letter : Document // создание дочернего класса письмо по аналогии //
{
    public String from;
    public Letter(DateTime date, String number, String sender)
        : base(date, number)
    { from = sender; }
    public override string toString()
    { String result = base.ToString();
        return result + "Откуда: " + from; }}
class Program //По итогу программа выводит все три класса со всеми ихсвойствами//
{
    static void Main(string[] args)
    {
        Document doc = new Document(DateTime.Now, "Д-101");
        Letter Letter1 = new Letter(DateTime.Now, "01.06", "ООО \"Собачка\"");
```

```
Order Order1 = new Order(DateTime.Now, "Од-14", "Специалист \"Н\"", "Глав.бух. \"К\"");

Console.WriteLine(doc.toString());

Console.WriteLine(Letter1.toString());

Console.WriteLine(Order1.toString());

Console.ReadLine();}}}
```

Разобрав две задачи, мы можем с легкостью сказать, что знание, как минимум, теории графов нужны абсолютно любому IT-специалисту. Но, как мы знаем, без знаний основополагающих дискретной математики теорию графов не выучить. Даже на примере первой разобранной задачи мы можем сказать, что нам нужны такие знания, как законы алгебры логики, элементы комбинаторики и т.д.

Литература:

1. James A. Anderson Discrete Mathematics with Combinatorics, Издательский дом «Вильямс», 2018 г, Москва, Санкт-Петербург, Киев, с.957
2. Володичева М. И., Григорьев-Голубев В. В. Дискретная математика с пакетами MATHEMATICA и MATHCAD: учеб. Пособие. – СПб.: Изд-во СПбГМТУ, 2015. – 329 с.