МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Ижевский государственный технический университет
имени М.Т. Калашникова»

Кафедра «Программное обеспечение»

О.Л. Макарова

Дискретная математика

методические указания для выполнения лабораторной работы №1

«Моделирование основных операций над множествами»

направление 231000 «Программная инженерия»

Ижевск, 2013 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ по курсу
«Дискретная математика»

Общие указания

Целью лабораторных работ является:

- закрепление теоретических знаний по дисциплине «Дискретная математика»;
- приобретение практических навыков по моделированию объектов, изучаемых в курсе, а также операций над ними.

Предлагаемые лабораторные работы проводятся со студентами всех форм обучения кафедры «Программное обеспечение» в ходе изучения курса «Дискретная математика». На первом лабораторном занятии студенты получают номер варианта на весь семестр, в соответствии с которым выполняют индивидуальные задания.

В ходе лабораторной работы студент должен ознакомиться с теоретическим материалом (см. список литературы [1-8]) и написать программу в соответствии с заданием *своего номера варианта*. За каждую выполненную работу студенту начисляется определенное количество баллов. Помимо лабораторных работ студент может выполнить дополнительные задания, за что получает соответственно *дополнительные баллы*.

Программа может быть написана на языке *Pascal* (по желанию студент может использовать языки *C, C++,* возможно использование среды программирования *Delphi*). Текст программы должен в *обязательном порядке содержать комментарии*.

Предварительно представить отчет по лабораторной работе можно в электронном виде по почте ol@istu.ru (в формате .doc, .docx или .pdf; дополнительно к сообщению необходимо прикрепить тексты исходных и exe-файлов). Окончательный вариант представляется студентом преподавателю в *печатном виде.*

Содержание отчета

Отчет должен соответствовать требованиям оформления документов такого типа и содержать следующие пункты:

1. Титульный лист с указанием темы лабораторной работы и номера варианта
2. Постановку задачи
3. Алгоритм решения задачи (математическая постановка)
4. Описание программы:
   - структура входных данных;
   - структура выходных данных;
   - алгоритм программы.
5. Листинг текста программы
6. Тестовые примеры
7. Выводы

Примечание: если в задании не указан метод решения задачи или структура входных данных, то выбор метода и структуры данных должен быть сделан самостоятельно студентом, обоснование выбора должно быть отражено в отчете.

ЛАБОРАТОРНАЯ РАБОТА № 1
Моделирование основных операций над множествами

1.1. Цель работы

Изучить методы представлений множеств в программах, операции над множествами и их свойства, алгоритмы с использованием множеств; представление множеств характеристическими векторами и их практическую реализацию.

1.2. Теоретические сведения

Представить в программе какой-либо объект - это значит описать в терминах системы программирования структуру данных, используемую для хранения информации о данном объекте, а также алгоритмы, реализующие присущие данному объекту операции. Таким образом, говоря о множествах «представление» подразумевает описание способа хранения информации о принадлежности элементов множеству и описания алгоритмов для вычисления объединения, пересечения, разности и других введенных операций. Заметим, что любой объект может быть описан по-разному, причем нельзя указать какой из известных способов описания является на данный момент для рассматриваемой задачи наилучшим. В одних случаях выгоднее использовать одно представление, а в других - другое. Выбор зависит от особенностей представляемого объекта, состава и частоты использования операций в конкретной задаче и т.д. Умение выбрать наилучшее представление и является основой искусства программирования. *Хороший программист отличается тем, что он знает много разных способов представления объектов и умело выбирает наиболее подходящий.*

1.2.1. Основные определения[1,2,3]

Множество - есть любое собрание определенных и различимых между собой объектов нашей интуиции или интеллекта, мыслимое как единое целое. Эти объекты называются *элементами* множества. Множества обозначаются прописными латинскими буквами $(A, B, C, \ldots)$, а их элементы – строчными $(a, b, c, \ldots)$. Множество как совокупность дискретных объектов обозначается $A = \{a, b, c\}$.

Множества могут быть конечными (группа студентов) или бесконечными. Множества, элементами которых являются тоже множества, называют *классом (семейством, системой) множеств*.

Для конечного множества $A = \{a_1, a_2, \ldots, a_n\}$ количество элементов n называется *мощностью* множества и обозначается $|A|$ $(|A| = n)$.

Множество, состоящее из одного элемента, обозначается $\{a\}$. Множество, не содержащее элементов, называется *пустым* и обозначается $\emptyset$ (например, $A = \emptyset$).

Множество, содержащее все элементы, находящиеся в рассмотрении, называется *универсальным* или *универсумом*, обозначается U. *Булеаном* называется множество всех подмножеств множества A и обозначается $P(A)$ или 2^A.

Говорят, что множество A *вложено* (*включено*, является *подмножеством*): множества B , если все элементы множества A являются элементами множества B (обозначается $A \subseteq B$).

Над множествами определены следующие операции:

Название	Определение
Объединение множеств A и B	$A \cup B = \{a \mid a \in A$ или $a \in B\}$
Пересечение множеств A и B	$A \cap B = \{a \mid a \in A$ и $a \in B\}$
Разность множеств B и A	$B \backslash A = \{a \mid a \in B$ и $a \notin A\}$
Дополнение множества A	$\bar{A} = U \backslash A = \{a \mid a \notin A\}$
Симметрическая разность множеств A и B	$A \triangle B = A \div B = (A \cup B) \backslash (A \cap B)$

Более наглядно представить соотношения между множествами можно с помощью диаграмм Эйлера-Венна (см. рис. 1.1).

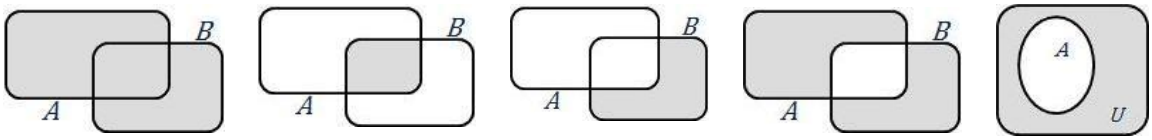

Рис. 1.1

1.2.2. Представление множеств [1,5,8]

Существуют два основных подхода к представлению множеств в памяти.

1. При первом подходе хранят описание каждого элемента, действительно присутствующего в множестве, как это делается, когда выписываются все элементы множества и заключаются в фигурные скобки.

2. При втором подходе изначально определяются все потенциально возможные элементы множества (U), а затем для любого подмножества универсума для каждого возможного члена указывается, принадлежит ли он на самом деле данному подмножеству или нет.

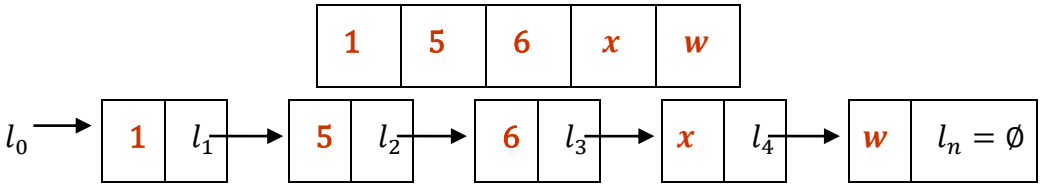

Рис. 1.2 Смежное и связанное представление множества в памяти

При первом подходе представления множеств используют как смежное, так и связанное размещение его элементов в памяти (рис. 1.2).

1.2.3. Смежное представление[8]

С вычислительной точки зрения простейшим представлением конечного множества $s_1, s_2, \ldots, s_n$ является точный список его элементов, расположенных по порядку в смежных ячейках памяти. В языках высокого уровня — это одномерные, двухмерные и т. д. массивы данных. Наряду с очевидными преимуществами последовательное представление имеет и некоторые значительные недостатки. Смежное представление становится неудобным, если требуется изменить последовательность путем включения новых и исключения имеющихся там элементов, которое требует перемещения многих элементов. С точки зрения

времени обработки такое перемещение элементов может оказаться дорогостоящим из-за сложности операций включения и удаления $O(n)$.

1.2.4. Связанное представление[8]

Неудобство включения и исключения элементов при смежном представлении происходит из-за того, что порядок следования элементов задается неявно требованием, чтобы смежные элементы последовательности находились в смежных ячейках памяти. В результате многие элементы последовательности во время включения или исключения должны передвигаться.

Связанным представлением множеств можно добиться большей гибкости. При связанном размещении множеств каждому элементу s_i ставится в соответствие указатель l_i на следующую подобную пару (s_i, l_i). Вводятся начальный указатель l_0, который указывает на первый элемент и последний l_n – признак конца. Связанное представление множеств удобно при операциях включения и удаления элементов, так как достаточно работать только с указателями элементов. Но при этом мы теряем возможность работать с элементами множеств как с массивами, когда по номеру i можно непосредственно обратиться к элементу s_i. В связанном размещении такой возможности не существует, и доступ к элементам последовательности не является прямым и эффективным. Например, при поиске среднего элемента последовательности, даже при известной ее длине, требуется просмотреть по связанному списку половину множества. Однако, следует отметить, что при представлении множеств в виде списков трудоемкость операции $\in$ составит $O(n)$, а операций $\cap$, $\cup$, $\subset$ составит $O(nm)$, где n и m - мощности участвующих в операциях множеств. Если элементы множеств упорядочить, то трудоемкость всех операций составит $O(n)$.

1.2.5. Множественный тип

В языке *Pascal* множественный тип данных похож на перечислимый тип данных. В математике множество - любая совокупность элементов произвольной природы, в программировании - набор элементов, не организованных в порядке следования. В качестве базовых типов могут использоваться: перечислимые типы данных, символьный и байтовый типы или диапазонные типы на их основе. Такие ограничения связаны с формой представления множества в языке и объясняются тем, что функция `Ord` для элементов используемого базового типа должна быть в пределах от 0 до 255.

Множество имеет зарезервированное слово `set of` и вводится следующим описанием

```
Type
  < имя типа > = set of < имя базового типа >;
Var
  < идентификатор,... >:< имя типа >;
```

Например:

```
Type
  SetByte = set of byte;
  SetChisla = set of 10 .. 20;
  Symbol = set of char;
  Season = (winter, spring, summer,autumn);
Var
  Index : SetChisla=[12, 15, 17, 20..30];
  FIO : StringSet := [‘Иванов’, ‘Петров’, ‘Сидоров’]
```

Значения в списке могут отсутствовать, тогда множество является пустым:

```
bs:=[];
```

Пустое множество совместимо по присваиванию с множеством любого типа. Для множеств имеет место структурная эквивалентность типов. Множества целых и множества на базе типа и его диапазонного подтипа или на базе двух диапазонных типов одного базового типа неявно преобразуются друг к другу. Если при присваивании `s:=s1` во множестве `s1` содержатся элементы, которые не входят в диапазон значений базового типа для множества `s`, то они отсекаются.

Например:

```
var st: set of 3..9;

  st := [1..5,8,10,12]; // в st попадут значения [3..5,8]
```

Операция `in` проверяет принадлежность элемента множеству:

```
if Wed in bestdays then ...
```

Для множеств определены операции:

+	объединение	=	равенство	<	строгое вложение
−	разность	<>	неравенство	>=	нестрого содержит
*	пересечение	<=	нестрогое вложение	>	строго содержит

Процедура `write` при выводе множества выводит все его элементы, при этом данные, если это возможно, будут отсортированы по возрастанию.

В операциях над множествами могут участвовать и отдельные элементы множества.

Например, допустима следующая запись, где два элемента и множество объединяются в новое множество:

```
WarmSeason := May+Summer+September;
B: = B-['c'];
```

Существуют еще две операции для работы с множествами, которые могут быть описаны либо конструкциями, либо функциями языка:

	функция	конструкция
добавления элемента x к множеству s	`Include(s,x)`	`s+=[x]`
Удаление элемента x из множества s	`Exclude(s,x)`	`s-=[x]`

1.2.6. Алгоритмы слияния[1,5,8]

Эффективной реализацией операций над множествами, представленных в виде списков, основана на весьма общем алгоритме, известном как алгоритм *слияния*. Суть этого алгоритма состоит в том, что оба множества просматриваются параллельно, причем на каждом шаге продвижение происходит в том множестве, в котором текущий элемент больше (или меньше).

Рассмотрим алгоритм слияния, который определяет, является ли множество A подмножеством множества B.

```
Вход: проверяемые множества A и B, которые заданы указателями a и b.
Выход: true, если A⊂B, в противном случае false.
```

```
pa:=a; pb:=b { инициализация }
while pa≠nil и pb≠nil do
  if pa.i < pb.i then
   return false {элемент множества A отсутствует в множестве B}
  else if pa.i > pb.i then
    pb:=pb.n {элемент множества A, может быть,присутствует в множестве B }
    else
     pa:=pa.n { здесь pa.i = pb.i, то есть }
     pb:=pb.n { элемент множества A ТОЧНО присутствует в множестве B}
    end
  end
return pa=nil { true, если A исчерпано, false – в противном случае}
```

Обоснование. На каждом шаге основного цикла возможна одна из трёх ситуаций: текущий элемент множества A меньше, больше или равен текущему элементу множества B. В первом случае текущий элемент множества A заведомо меньше, чем текущий и все последующие элементы множества B, а потому он не содержится в множестве B и можно завершить выполнение алгоритма. Во втором случае происходит продвижение по множеству B в надежде отыскать элемент, совпадающий с текущим элементом множества A. В третьем случае найдены совпадающие элементы и происходит продвижение сразу в обоих множествах. По завершении основного цикла возможны два варианта: либо pa $=$ nil, либо pa $\neq$ nil. Первый означает, что для всех элементов множества A удалось найти совпадающие элементы в множестве B, что означает $A \subset B$. Второй — что множество B закончилось раньше, то есть не для всех элементов множества A удалось найти совпадающие элементы в множестве B, то есть $A \not\subset B$.

Аналогично, используя алгоритм слияния можно использовать для нахождения результатов пересечения, объединения, разности множеств.

P.S.: На практике часто оказывается так, что самая изощренная реализация операции, не зависящая от представления, оказывается менее эффективной, по сравнению с самой прямолинейной реализацией, ориентированной на конкретное представление. При современном состоянии компьютерной базы вычислительной техники оказывается, что во многих задачах можно пренебречь некоторым выигрышем в эффективности, например, различием между $O(nm)$ и $O(n+m)$. То есть нынешние компьютеры настолько быстры, что можно не гнаться за самым эффективным представлением, ограничившись *«достаточно эффективным»*.

1.2.7. Характеристические векторы[1,2,4,5,8]

При втором методе множество представляется в виде вектора на смежной памяти. Пусть U — универсальное множество или универсум (т. е. все рассматриваемые множества являются его подмножествами), состоящее из n элементов. Любое подмножество $S \in U$ представляется в виде характеристического вектора из n элементов. Элемент i в этом векторе равен 1 тогда и только тогда, когда i – й элемент множества U принадлежит S, в противном случае он устанавливается равным 0.

Например, для универсума $U = \{3,5,6,7,8,9\}$ характеристический вектор множества $A = \{\text{чисел, кратных 3}\}$, имеет вид $a = (1,0,1,0,0,1)$.

Использование характеристических векторов удобно, когда формирование множества выполняется путем последовательного удаления из универсума элементов, не принадлежащих данному множеству. Кроме того, определять принадлежность i -го элемента множеству можно за время, не зависящее от его

размера. Главное неудобство заключается в их не экономичности, так как данное представление требует дополнительной памяти для хранения характеристического вектора, что для больших n (размер универсального множества U) бывает практически невыполнимо.

Основные операции над множествами, такие как объединение и пересечение, можно осуществлять как операции $\vee$ и $\wedge$ над двоичными векторами. Все остальные операции можно выразить через них.

Например, пусть $U = \{1, 2, 3, 4, 5, 6\}$, множества $A = \{1, 2, 4, 5\}$ и $B = \{3, 5\}$. Характеристическими векторами множеств A и B являются векторы $a = (1, 1, 0, 1, 1, 0)$ и $b = (0, 0, 1, 0, 1, 0)$ соответственно. Вычислим характеристический вектор множества $A \cup \bar{B}$. Он равен a или (не b) $= (1, 1, 0, 1, 1, 0)$ или $(1, 1, 0\ 1, 0, 1) = (1,1,0,1,1,1)$.

Следовательно, $A \cup \bar{B} = \{1, 2, 4, 5, 6\}$.

Наилучший метод представления множеств существенно зависит от операций, которые мы собираемся выполнять над ними. Типичные операции над множествами: выяснить, имеется ли конкретный элемент в данном множестве; добавить в множество новые элементы; удалить элементы из множества; выполнить обычные теоретико-множественные операции, такие как объединение или пересечение двух множеств. Следует помнить, что *при любом представлении элементов множеств необходимо сохранять информацию о способе их упорядочения.*

1.3. Порядок выполнения работы

Составить программу(ы) для решения следующих задач:

1. (1 балл) Дана строка символов St_A, состоящая из прописных букв латинского алфавита. Выдать множество-носитель данной строки.
   Входные данные: Строка A, длина строки $1 < l < 50$.
   Выходные данные: Вывести в строку множество-носитель строки A. Элементы множества разделять пробелами. В конце строки пробел не ставить.
   Пример:

Входные данные	Выходные данные
bacscode	a b c d e s
babab	a b

2. (2 балла) Используя результаты задачи 1 сгенерировать три множества A, B, C, состоящие из букв фамилии, имени и отчества студента, выполняющего работу (написать процедуру). Найти множества R, используя операции над множествами, в соответствии с номером своего варианта. В конце строки пробел не ставить.
   Пример: $R = A \backslash (B \cap C)$

Входные данные	Выходные данные
макарова ольга леонидовна	а в к м о р а г л о ь а в д е и л н о $R = \{в, к, м, р\}$

3. (2 балла) Используя результаты задачи 2 сгенерировать универсум, определив его как $U = A \cup B \cup C$. Сформировать характеристические векторы

для множеств A, B, C (написать процедуру). Получить вектор множества R из задачи 2, используя характеристические векторы множеств A, B, C. В конце строки пробел не ставить.

Пример: $R = A\backslash(B \cap C)$

Входные данные	Выходные данные
макарова ольга леонидовна	а в г д е и к л м н о р ь 1 1 0 0 0 0 1 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 0 1 0 0 1 0

4. (1 балл) Используя универсум, полученный в задаче 3, и результирующий вектор вывести множество R. В конце строки пробел не ставить.

Пример:

Входные данные	Выходные данные
а в г д е и к л м н о р ь 0 1 0 0 0 0 1 0 1 0 0 1 0	{в,к,м,р}

5. (3 балл) Сравнить результаты и сделать выводы. Оформить отчет.

Варианты множества R:

1) $(A \cup C)\backslash B$
2) $(A\backslash C)\backslash B$
3) $A \cup (C \cup B)$
4) $(A \cup B)\backslash C$
5) $A \cap (B\backslash C)$
6) $C\backslash(B \cap A)$
7) $(C\backslash A) \cup B$
8) $(A\backslash B) \cap C$
9) $A\backslash(B \cup C)$
10) $A \cap (B \cup C)$
11) $(A\backslash B) \cup C$
12) $(A\backslash C)\backslash B$
13) $(A \cup B) \cap C$
14) $(C\backslash B) \cup A$
15) $(A \cap C) \cup B$
16) $A \cup (B\backslash C)$
17) $(C\backslash B) \cup A$
18) $(A\backslash C) \cup B$

1.4. Дополнительные задачи

1. **Пересечение интервалов (1 балл)**

Даны 2 замкнутых интервала. Необходимо вывести их пересечение.

Формат входного файла: В первой строке содержится левая и правая граница первого интервала. Во второй строке – левая и правая граница второго интервала (все числа целые, по модулю не превосходящие 100; правая граница строго больше левой).

Формат выходного файла: Единственная строка выходного файла должна содержать пару чисел – левая и правая границы пересечения интервалов. Если общего интервала нет, выведите -1.

Пример:

Входные данные	Выходные данные
0 5 2 7	2 5
0 5 5 10	5 5
2 5 -2 3	-1

2. **Мультимножество (1 балл)**

Даны множества A и B упорядоченных чисел с количеством элементов N и M. Объединить их в упорядоченное мультимножество C.
Формат входного файла: В первой строке – $1 < N < 32000$, $1 < M < 32000$. Во второй строке – элементы массива A, $1 < a[i] < 32000$. В третьей строке – элементы массива B, $1 < b[i] < 32000$.
Формат выходного файла: Вывести в строку массив C. В конце строки пробел не ставить.

Пример:

Входные данные	Выходные данные
3 3 1 2 3 2 4 8	1 2 2 3 4
3 3 -2 0 4 -3 0 4	-3 -2 0 0 4 4
4 3 -11 -2 0 2 -5 3 5	-11 -5 -2 0 2 3 5

3. **Слияние двух множеств 2(1 балл)**

Даны множества A и B упорядоченных чисел с количеством элементов N и M. Объединить их в упорядоченное множество C.
Формат входного файла: В первой строке – $1 < N < 32000$, $1 < M < 32000$. Во второй строке – элементы массива A, $1 < a[i] < 32000$. В третьей строке – элементы массива B, $1 < b[i] < 32000$.
Формат выходного файла: Вывести в строку массив C. В конце строки пробел не ставить.

Пример:

Входные данные	Выходные данные
3 3 1 2 3 2 4 8	1 2 3 4
3 3 -2 0 4 -3 0 4	-3 -2 0 4
4 3 -11 -2 0 2 -5 3 5	-11 -5 -2 0 2 3 5

4. **Подмножества (2 балла)**

Дано множество из N целых чисел и целое число C. Необходимо посчитать количество подмножеств исходного множества таких, что сумма элементов подмножества кратна C.
Например, если исходное множество равно {1, 2, 3, 4, 5}, то нужно выделить следующие подмножества, сумма элементов которых кратна трем: {3}, {1, 2}, {4, 5}, {1, 2, 3}, {1, 2, 3, 4, 5} и др.

Формат входного файла: В первой строке находится число N ($1 \leq N \leq 10$). В следующей строке находятся N чисел – элементы множества (натуральные числа, не превосходящие 100). В последней строке содержится число C ($1 \leq C \leq 1024$).

Формат выходного файла: Выведите одно число – искомое количество подмножеств.

Пример:

Входные данные	Выходные данные
5 1 2 3 4 5 3	11
5 1 2 2 2 1 9	0

1.5. Контрольные вопросы

1. Может ли множество содержать одинаковые элементы?
2. Равны ли множества ∅ и {∅}?
3. Что такое мощность множества?
4. Что такое универсум?
5. Перечислите плюсы и минусы смежного и связанного представления множеств в программах.
6. Что такое характеристический вектор множества?
7. Поясните работу с множествами в случае их задания характеристическими векторами.
8. Поясните алгоритм слияния множеств на примере проверки включения (объединения, пересечения, разности) множеств.
9. Какими свойствами обладают основные операции над множествами?

1.6. Литература

1. *Новиков, Ф. А.* Дискретная математика для программистов : учеб. для вузов. – 2-е изд. – СПб. : Питер, 2005. – 364 с.
2. *Хаггарти, Р.* Дискретная математика для программистов : учеб. пособие. – 2-е изд. – М. : Техносфера, 2005. – 400 с.
3. *Судоплатов, С. В.* Дискретная математика : учебник / С. В. Судоплатов, Е. В. Овчинникова. – 2-е изд. – Москва ; ИНФРА-М ; Новосибирск : НГТУ, 2005. – 256 с.
4. *Дейкстра. Э.* Дисциплина программирования. – М.: Мир, 1978. – 280 с.
5. *Кнут Д.* Искусство программирования, том 1, выпуск 2. Основные алгоритмы. : Пер. с англ. - М.: ООО "И.Д. Вильямс", 2007. – 682 с.
6. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики/ Пер. с англ. – М.: Мир, 1998. – 703 с.
7. *Кнут Д.* Искусство программирования, том 2, выпуск 2. Получисленные алгоритмы. : Пер. с англ. - М.: ООО "И.Д. Вильямс", 2007. – 788 с.
8. *Иванов Б.Н.* Дискретная математика. Алгоритмы и программы : учеб. пособие. – М. : Лаборатория Базовых Знаний, 2001 – 288 с.