МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

Институт прикладной математики и механики

Высшая школа прикладной математики и вычислительной физики

Отчёт по дисциплине:
“Дискретная математика”
Лабораторная работа № 2
Реализация булевых функций

Выполнил: _____ студент группы 3630201/90001 Волынец А.В.
Принял: _____ Востров А.В.

Санкт-Петербург — 2020

Содержание

Введение

В данной лабораторной работе требуется по таблице истинности булевой функции построить сокращенное дерево решений и наиболее компактную бинарную диаграмму решений, а также синтаксическое дерево и логическую схему на основе дизъюнктора, конъюктора и инвертора. Реализовать программно вычисление значений булевой функции через бинарную диаграмму решений. Программно с помощью таблицы истинности построить СДНФ и СКНФ. Вычислить программно по СДНФ значения булевой функции. Вычисление значений функции по СДНФ и БДР (бинарной диаграмме решений) должны совпадать с таблицей истинности. Также для данной функции необходимо программно построить полином Жегалкина.

Исходная функция, заданная с помощью вектора значений по возрастанию аргументов:

$$f(x, y, z, k) = (0000010110011111)$$

1 Математическое описание задачи

1.1 Вычисление СДНФ и СКНФ по таблице истинности

Definition. *Совершенной дизъюнктивной нормальной формой (СДНФ) булевой функции* $f(x_1, \ldots, x_n)$ *называется формула* F *в базисе Буля вида:*

$$F : f(x_1, \ldots, x_n) = \bigvee_{\{(\sigma_1,\ldots,\sigma_n)|f(\sigma_1,\ldots,\sigma_n)=1\}} x_1^{\sigma_1} \ldots x_n^{\sigma_n}$$

Definition. *Совершенной конъюктивной нормальной формой (СКНФ) булевой функции* $f(x_1, \ldots, x_n)$ *называется формула* G *в базисе Буля вида:*

$$G : f(x_1, \ldots, x_n) = \bigwedge_{\{(\sigma_1,\ldots,\sigma_n)|f(\sigma_1,\ldots,\sigma_n)=0\}} x_1^{\overline{\sigma_1}} \vee \ldots \vee x_n^{\overline{\sigma_n}}$$

С помощью таблицы истинности всегда можно найти СДНФ и СКНФ по определению. Для функции f:

$$f(x, y, z, k) = (0000010110011111)$$

СДНФ$(f) = \overline{x}y\overline{z}k \vee \overline{x}yzk \vee x\overline{y}\overline{z}\overline{k} \vee x\overline{y}zk \vee xy\overline{z}\overline{k} \vee xy\overline{z}k \vee xyz\overline{k} \vee xyzk$

СКНФ$(f) = (x \vee y \vee z \vee k)\left(x \vee y \vee x \vee \overline{k}\right)(x \vee y \vee \overline{z} \vee k)\left(x \vee y \vee \overline{z} \vee \overline{k}\right)(x \vee \overline{y} \vee z \vee k)(x \vee \overline{y} \vee \overline{z} \vee k)$ $\left(\overline{x} \vee y \vee z \vee \overline{k}\right)(\overline{x} \vee y \vee \overline{z} \vee k)$

1.2 Построение полинома Жегалкина

Definition. *Полиномом Жегалкина булевой функции* $f(x_1, \ldots, x_n)$ *называется формула* F *в базисе Жегалкина* $(\wedge, +)$*:*

$$F : f(x_1, \ldots, x_n) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_{123\ldots n} x_1 \ldots x_n$$

Для функции f полином Жегалкина имеет вид:

$$f(x, y, z, k) = \alpha_0 + \alpha_1 k + \alpha_2 z + \alpha_3 zk + \alpha_4 y + \alpha_5 yk + \ldots + \alpha_{15} xyzk$$

Построение методом треугольника:

$$\begin{aligned}
\alpha_0 &= 0000010110011111 \\
\alpha_1 &= 000011101010000 \\
\alpha_2 &= 00010011111000 \\
\alpha_3 &= 0011010000100 \\
\alpha_4 &= 010111000110 \\
\alpha_5 &= 11100100101 \\
\alpha_6 &= 0010110111 \\
\alpha_7 &= 011101100 \\
\alpha_8 &= 10011010 \\
\alpha_9 &= 1010111 \\
\alpha_{10} &= 111100 \\
\alpha_{11} &= 00010 \\
\alpha_{12} &= 0011 \\
\alpha_{13} &= 010 \\
\alpha_{14} &= 11 \\
\alpha_{15} &= 0
\end{aligned}$$

Откуда следует, что полином Жегалкина:

$$f(x, y, z, k) = yk + x + xk + xz + xyz$$

1.3 Дерево решений

Для данной функции построено сокращенное дерево решений, которое получается из бинарного дерева решений заменой узлов дающих одно значение этим значением:

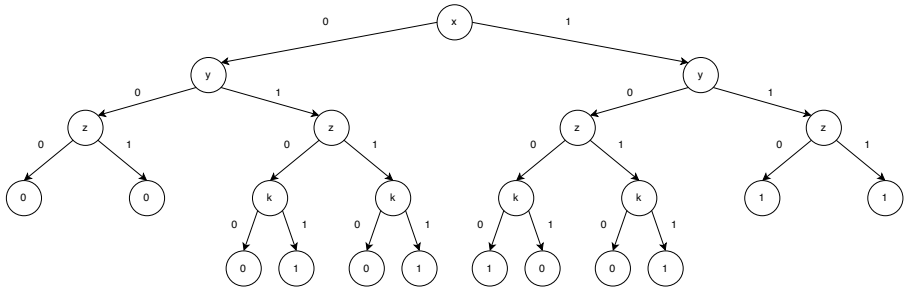

Рис. 1: Сокращенное дерево решений

1.4 Бинарная диаграмма решений

Бинарная диаграмма решений получается из дерева решений допущением вхождения нескольких дуг в один узел. Для функции f построена бинарная диаграмма решений:

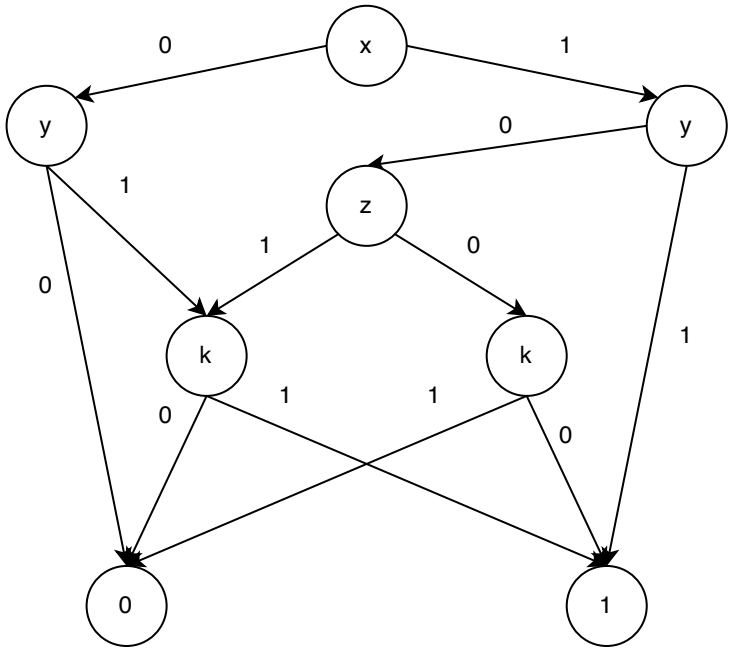

Рис. 2: Бинарная диаграмма решений

1.5 Синтаксическое дерево

Для минимальной формы функции $f = xy \vee yk \vee xyk \vee x\overline{z}\overline{k}$ построено синтаксическое дерево:

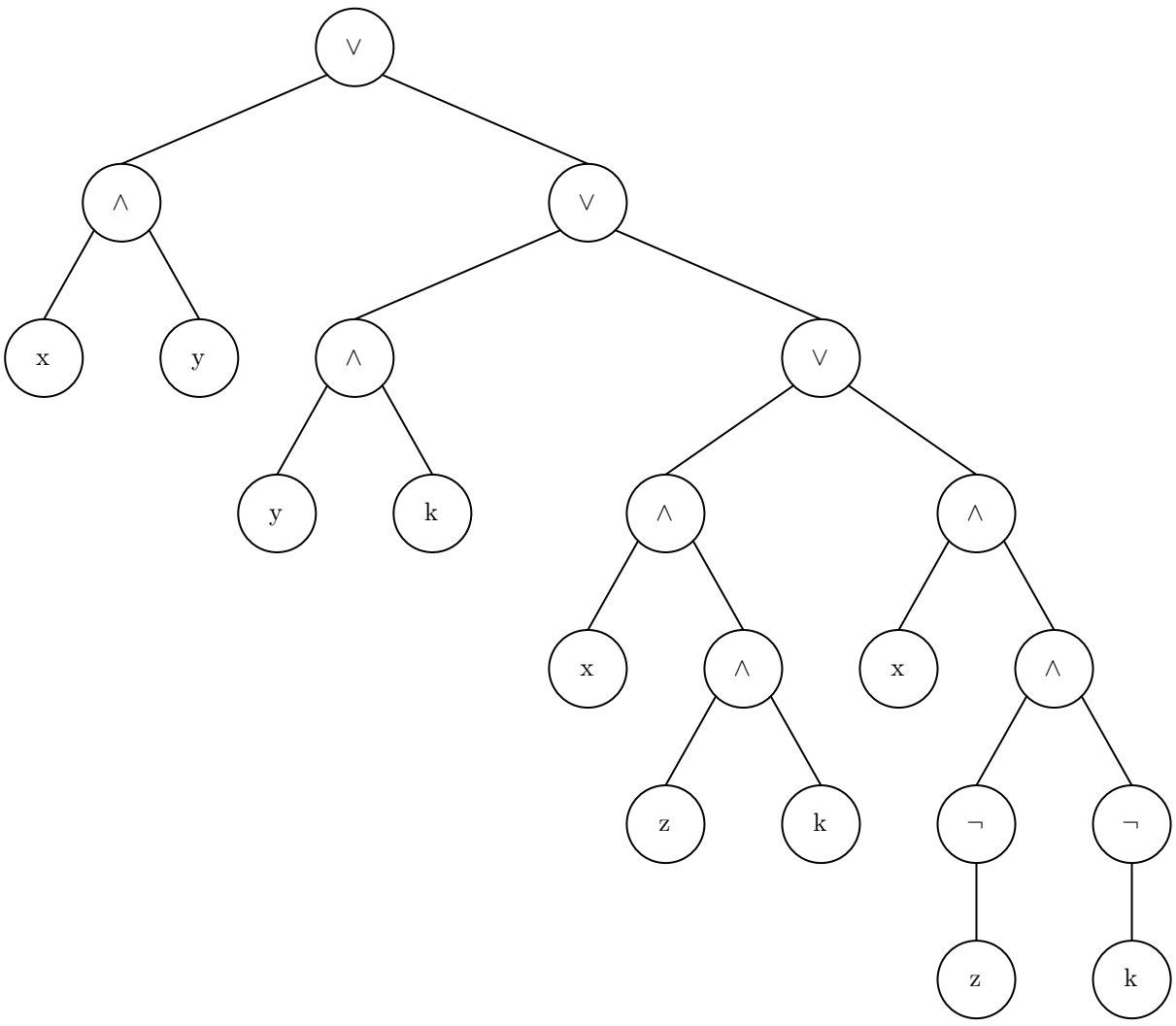

Рис. 3: Синтаксическое дерево для минимальной формы

1.6 Логическая схема

Для минимальной формы функции $f = xy \vee yk \vee xyk \vee x\overline{z}\overline{k}$ построена логическая схема на основе дизъюнктора, конъюктора и инвертора (рис. 4):

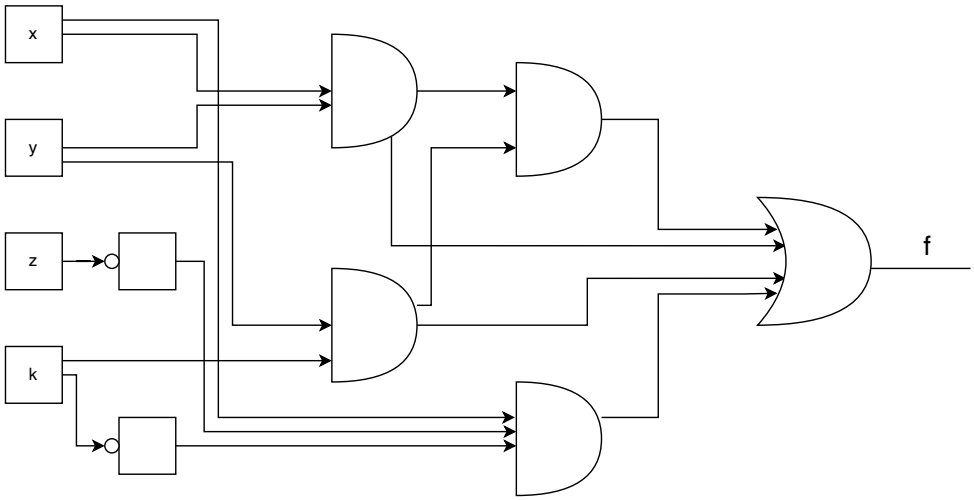

Рис. 4: Логическая схема

2 Особенности реализации

Для хранения данных используются глобальные переменные, поэтому многие функции не имеют параметров. Булева функция f задаётся с помощью `vector<char>`:

```
vector<char> func{0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1};
```

2.1 Программное построение СДНФ и СКНФ

Для построения СДНФ по таблице истинности используется следующий алгоритм 3.3 ([1], стр. 135):

Алгоритм 3.3 Построение СДНФ

Вход: вектор x : **array** $[1..n]$ **of string** идентификаторов переменных,
вектор F_1 : **array** $[0..2^n - 1]$ **of** 0..1 значений функции при установленном порядке кортежей.
Выход: последовательность символов, образующих запись формулы СДНФ для заданной функции.

```
f := false { признак присутствия левого операнда дизъюнкции }
for i from 0 to 2^n − 1 do
  if F1[i] = 1 then
    if f then
      yield ' ∨ ' { добавление в формулу знака дизъюнкции }
    else
      f := true { это первое слагаемое в дизъюнкции }
    end if

    g := false { признак присутствия левого операнда конъюнкции }
    for j from 1 to n do
      if g then
        yield '∧' { добавление в формулу знака конъюнкции }
      else
        g := true { это первый сомножитель в конъюнкции }
      end if
      v := (i div 2^(j−1)) mod 2  { значение j-го разряда кортежа с номером i }
      if v = 0 then
        yield '¬' { добавление в формулу знака отрицания }
      end if
      yield x[j] { добавление в формулу идентификатора переменной }
    end for
  end if
end for
```

В программе массиву индентификаторов соответствует:

```
const char* ids = "xyzk";
```

Результат алгоритма сохраняется в переменной sdnf типа `string`. Построение СКНФ происходит по аналогичному алгоритму.

2.2 Вычисление значения функции по СДНФ

Вход: строковое представление СДНФ, аргумент булевой функции типа int

Выход: bool - значение функции при данном аргументе.

При первом вызове функции инициализируется статический двумерный массив `vector<vector<char> table`, в котором хранятся кортежи, при которых функция принимает значение 1.

```
bool calcOnSDNF(int i){
    static size_t count = 0;
    static vector<vector<char>> table;
    if (count == 0){
        int term = -1;
        for (int k = 0; k < sdnf.size(); k++) {
            if (sdnf[k] == 'x') {
                table.emplace_back();
                if (k > 0 and sdnf[k - 1] == '!')
                    table[++term].push_back(0);
                else
                    table[++term].push_back(1);
            }
            else if (sdnf[k] == 'y' or sdnf[k] == 'z' or sdnf[k] == 'k'){
                if (k > 0 and sdnf[k - 1] == '!')
                    table[term].push_back(0);
                else
                    table[term].push_back(1);
            }
        }
        count++;
    }
    ...
```

Далее значение аргумента `i` переводится в двоичную систему счисления в массив `vector<char> value`. После чего данный кортеж сравнивается с кортежами в массиве table:

```
...
    vector<char> value(toBin(i));
    bool res = true;
    for (int k = 0; k < table.size(); k++){
        res = true;
        for (int j = 0; j < len; j++){
            if (table[k][j] != value[j]){
                res = false;
                break;
            }
        }
        if (res) return true;
    }
    return res;
}
```

2.3 Вычисление значения функции по БДР

Бинарная диаграмма решений реализована в виде класса `BinaryDiagram` и вспомогательного класса вершины `Node`:

```
struct Node{
    Node* l_child; //указатель на узел по нулевому ребру
    Node* r_child; //указатель на узел по единичному ребру
    bool val;
    Node(): l_child(nullptr), r_child(nullptr), val(false){}
    Node(Node* l, Node* r, bool v = false): l_child(l), r_child(r), val(v){}
};

struct BinaryDiagram{
    const int N = 9;// число вершин графа
    std::vector<Node*> nodes;
    ...
}
```

В конструкторе класса устанавливаются потомки узлов БДР через вспомогательный метод SetChildren. БДР, использующееся в программе отличается от привиденного на рис. 2. Для реализации алгоритма вычисления значения по БДР нужно добавить фиктивные узлы между внутренними узлами, если путь (количество ребер) меньше чем количество переменных. Так было добавлен узел с номером 8 между узлами 1 и 4, как показано на рис. 4. Номерам узлов соответствуют индексы массива nodes.

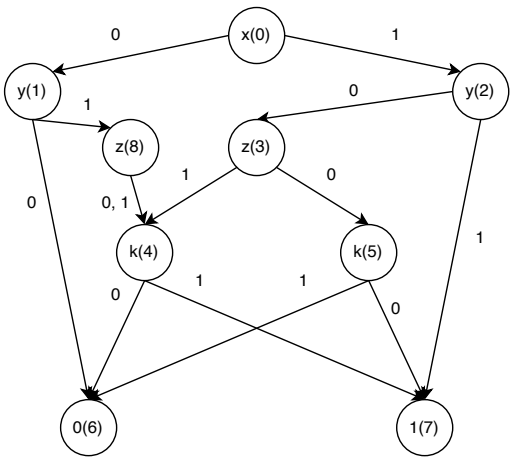

Рис. 5: Внутренее представление БДР

Ниже приведен конструктор класса Binary Diagram:
Вход: нет параметров
Выход: нет возвращаемого значения, иниацилизируется массив nodes, задающий БДР

```
    BinaryDiagram(): root(nullptr), nodes(N, nullptr){
        for (int i = 0; i < N; i++){
            nodes[i] = new Node;
        }
        setChildren(0, 1, 2);
        setChildren(1, 6, 8);
        setChildren(8, 4, 4);
        setChildren(2, 3, 7);
```

```
        setChildren(3, 5, 4);
        setChildren(4, 6, 7);
        setChildren(5, 7, 6);
        nodes[6]->val = false;
        nodes[7]->val = true;
    }
```

Ниже приведен вспомогательный для конструктора метод setChildren:

Вход: i - номер родителя, l - номер левого ребенка, r - номер правого ребёнка.

Выход: нет возвращаемого значения, метод устанавливает поля l_child и r_child структуры node с номером i в массиве nodes.

```
    void setChildren(int i, int l, int r){
        nodes[i]->l_child = nodes[l];
        nodes[i]->r_child = nodes[r];
    }
```

Вычисление значения функции реализовано с помощью следующей процедуры:

Вход: корень БДР Node*, аргумент типа int, для которого нужно вычислить значение функции.

Выход: значение функции при данном аргументе - bool

```
bool BinaryDiagram::calc(int i){
    vector<char> value(toBin(i));
    Node* p = nodes[0];
    for (int k = 0; k < value.size(); k++){
        if (p->r_child == nullptr and p->l_child == nullptr)
            return p->val;
        else if (value[k] == 0)
            p = p->l_child;
        else
            p = p->r_child;
    }
    return p->val;
}
```

Метод calc проходит по дереву решений, пока не дойдет до вершины самого нижнего яруса, в которой хранится значение функции при данном аргументе.

2.4 Построение полинома Жегалкина

Вход: массив значений булевой функции func `vector<char>`

Выход: строковое представление полинома Жегалкина jegal

```
void findJEGAL()
{
    vector<char> triangle(func.size(), 0);
    for (int i = 0; i < triangle.size(); i++)
        triangle[i] = func[i] - '0';
    bool f = false;
    for (int i = 0; i < (int)pow(2, len); i++)
    {
        if (i != 0)
        {
            for (int j = 0; j < (int) pow(2, len) - i; j++)
            {
                triangle[j] ^= triangle[j + 1];
```

```
            }
        }
        if (triangle[0] == 1)
        {
            if (f)
                jegal += " + ";
            else
                f = true;
            vector<char> value(toBin(i));
            if (i == 0) jegal += "1";
            else{
                for (int k = 0; k < len; k++) {
                    if (value[k]) jegal += ids[k];
                }
            }
        }
    }
}
```

Данная функция строит полином Жегалкина по методу треугольника.

3 Результаты работы программы

Ниже приведены результаты работы программы:

```
To print information about fuction (i)
To print menu (m)
To calc value on given argument (c)
To set new function (f)
To quit (q)
Source function: (0000010110011111)
Вычисленная по таблице истинности СДНФ:
(!x)y(!z)k V (!x)yzk V x(!y)(!z)(!k) V x(!y)zk V xy(!z)(!k) V xy(!z)k V xyz(!k) V xyzk
Вычисленная по таблице истинности СКНФ:
(xVyVzVk)(xVyVzV(!k))(xVyV(!z)Vk)(xVyV(!z)V(!k))(xV(!y)VzVk)(xV(!y)V(!z)Vk)((!x)VyVzV(!k))((!x)VyV(!z)Vk)
Значения функции вычисленные по СДНФ:
0000010110011111
Совпадает с таблицей истинности.
Значения функции вычисленные по БДР:
0000010110011111
Совпадает с таблицей истинности.
Полином Жегалкина:
yk + x + xk + xz + xyz
command: asdf
Unknown command, to print menu of commands type in 'm'
command: 23452
Unknown command, to print menu of commands type in 'm'
command: f
Введите количество переменных: 23456
Incorrect number: 23456. The number must be not greater than 5
Введите количество переменных: 2
Введите вектор значений: 1110
command: i
Source function: (1110)
Вычисленная по таблице истинности СДНФ:
(!x)(!y) V (!x)y V x(!y)
Вычисленная по таблице истинности СКНФ:
((!x)V(!y))
Значения функции вычисленные по СДНФ:
1110
Совпадает с таблицей истинности.
Полином Жегалкина:
1 + xy
command: c
Чтобы выйти из режима вычисления значения функции, введите 'q'.
Введите аргумент функции: 3
Значение функции в данной точке (11): 0
Введите аргумент функции: 24
Неправильный ввод, попробуйте еще раз.
Введите аргумент функции: q
command: q
```

Рис. 6: Пример работы программы

На рис. 6 приведена работа программы: сначала выводятся информация о исходной фукнции из задания, затем пользователь задаёт новую функцию двух переменных. Также приведена обработка ошибочного пользовательского ввода.

Заключение

В данной лабораторной работе для заданной функции f были построены сокращенное дерево решений, синтаксическое дерево решений, а также бинарная диаграмма решений. Были построены программно СДНФ, СКНФ, полином Жегалкина для f. Были реализованы функции для вычисления значения булевой функции через СДНФ и БДР.

Достоинства программы:

1. Масшатабируемость: возможность программно добавлять новые функции с большим количеством переменных и другими обозначениями переменных; возможность программно добавлять новые функции для работы с булевыми функциями, например минимизации функции.

2. Пользовательский интерфейс, который можно модифицировать программно при добавлении новых возможностей программы.

3. Возможность пользователя задавать функцию с меньшим или большим количеством переменных непосредственно при работе с программой

4. Реализована защита от неправильного пользовательского ввода.

Недостатки программы:

1. Бинарная диаграмма решений строится вручную и задаётся в конструкторе класса вручную, при использовании других функций пользователю необходимо программно задать БДР.

Список литературы

[1] Новиков Ф.А. Дискретная математика для программистов. Учебник для вузов. 3-е изд. - СПб.:Питер, 2001, 304.

[2] Stroustrup B. The C++ programming language, 4-th edition - Addison-Wesley, 2013, 1347.