

- **Множество** - это математический объект, сам являющийся набором, совокупностью, собранием каких-либо объектов, которые называются элементами этого множества и обладают общим для всех их характеристическим свойством. Множество, состоящее из конечного числа элементов, называется **конечным**. Множество, не являющееся конечным, называется **бесконечным**. Бесконечное множество называется **счетным**, если оно равномощно множеству \mathbb{N} всех натуральных чисел. Говорят, что все элементы счетного множества можно пронумеровать. В противном случае бесконечное множество называется **несчетным**. Множество, не содержащее элементов, называется пустым и обозначается \emptyset или $\{ \}$. Два множества A и B называются равными, если они состоят из одних и тех же элементов ($A=B$).

Пример множества заданного перечислением: $D = \{1, 2, 3, 5, 9\}$

Пример множества заданного характеристическим свойством: $A = \{x \mid x \in \mathbb{N} \text{ и } x < 10\}$

Пример множества заданного с помощью порождающей функцией: $A = \{x^2 \mid x \in \mathbb{N}\}$

• **Операции над множествами.**

1. Включение A в B ($A \subseteq B$ или $B \supseteq A$) истинно, если каждый элемент множества A принадлежит множеству B . В этом случае A называется <i>подмножеством</i> B , а B — <i>надмножеством</i> A .
2. Строгое включение A в B ($A \subset B$ или $B \supset A$) истинно, если $A \subseteq B$ и $A \neq B$. Если $A \neq \emptyset$ и $A \subset B$, то A есть собственное подмножество B .
3. Объединение A и B ($A \cup B$) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат A или B , т.е. $A \cup B = \{x \mid x \in A \text{ или } x \in B\}$.
4. Пересечение A и B ($A \cap B$) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат каждому из множеств A и B , т.е. $A \cap B = \{x \mid x \in A \text{ и } x \in B\}$.
5. Разность A и B ($A - B$) есть множество, состоящее из всех тех и только тех элементов множества A , которые не принадлежат множеству B , т.е. $A - B = \{x \mid x \in A \text{ и } x \notin B\}$.
6. Симметрическая разность A и B ($A \Delta B$) есть множество, состоящее из всех тех и только тех элементов множества A , которые не принадлежат множеству B и только тех элементов множества B , которые не принадлежат множеству A , т.е. $A \Delta B = \{x \mid x \in A \text{ и } x \notin B \text{ или } x \in B \text{ и } x \notin A\}.$ $A \Delta B = (A \cup B) - (A \cap B) = (A - B) \cup (B - A).$
7. Дополнение A до универсума U (A) есть множество, состоящее из всех тех и только тех элементов универсума U , которые не принадлежат множеству A , т.е. $A = \{x \mid x \notin A\}.$ $A = U - A.$
8. Декартовым (прямым) произведением множеств A и B называется множество $A \times B$ всех упорядоченных пар (a, b) , в которых элемент $a \in A$, а элемент $b \in B$

Примеры:

$$A = \{i, j, k\}, B = \{k, m\}. A \cap B = \{k\}$$

$$A = \{1, 3, 5\}, B = \{-1, 0, 1\}. A \cup B = \{-1, 0, 1, 3, 5\}$$

$$A = \{a, b, c, d\}, B = \{1, a, d, 5\}. A \setminus B = \{b, c\}$$

$$A = \{d, 5, f\}, B = \{-1, d\}. A \times B = \{(d, -1), (d, d), (5, -1), (5, d), (f, -1), (f, d)\}$$

- **Способы хранения множеств в памяти ЭВМ**

Для представления конечного множества в памяти ЭВМ можно использовать различные способы. Будем считать, что универсум U представляет собой KU натуральных чисел: $U = \{x \mid x \in N \text{ и } x \leq KU\}$.

1. Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A неупорядочены.
2. Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A упорядочены по возрастанию.
3. Элементы множества A хранятся в переменной A типа массив, элементы которого типа *boolean*. Если $i \in A$, то $Ai = true$, иначе $Ai = false$. Количество элементов в массиве A равно мощности универсума.
4. Множество A представляется двоичным кодом C , в котором: $Ci = 1$, если $i \in A$ и $Ci = 0$, если $i \notin A$, где Ci — i -й разряд кода C . В зависимости от мощности универсума код C может храниться в простой переменной или в массиве.
5. Для хранения множества можно использовать множественный тип.

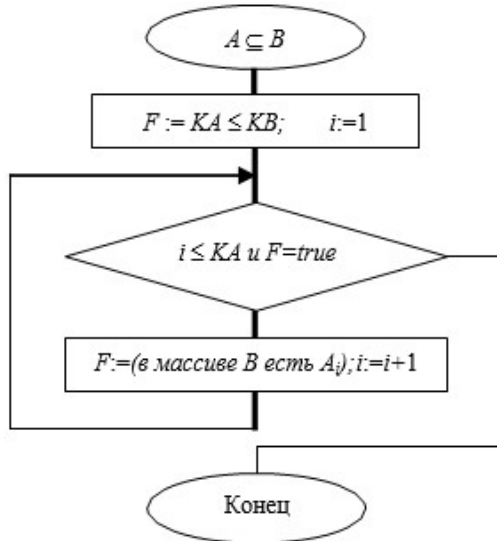
- Программная реализация операций над множествами

Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A не упорядочены.

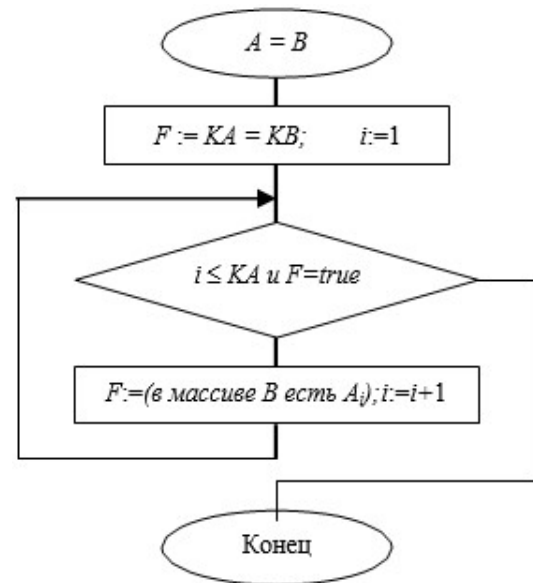
Алгоритм вычисления включения A в B ($A \subseteq B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.

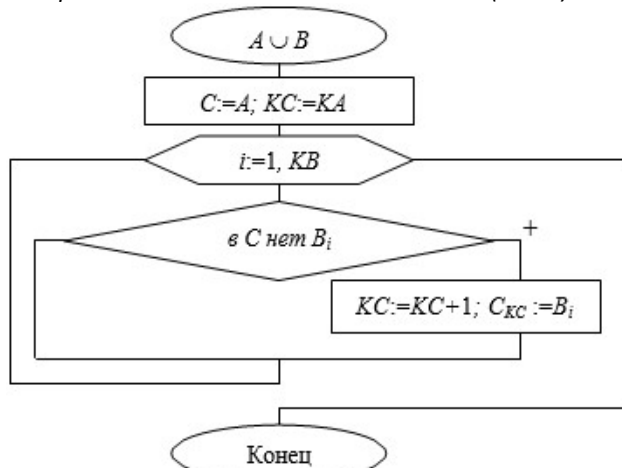
Выход: $F = \text{true}$, если $A \subseteq B$, иначе $F = \text{false}$.



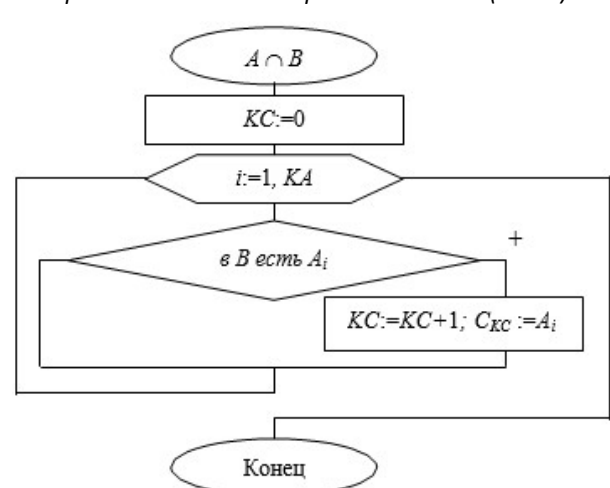
Алгоритм вычисления равенства A и B ($A = B$)



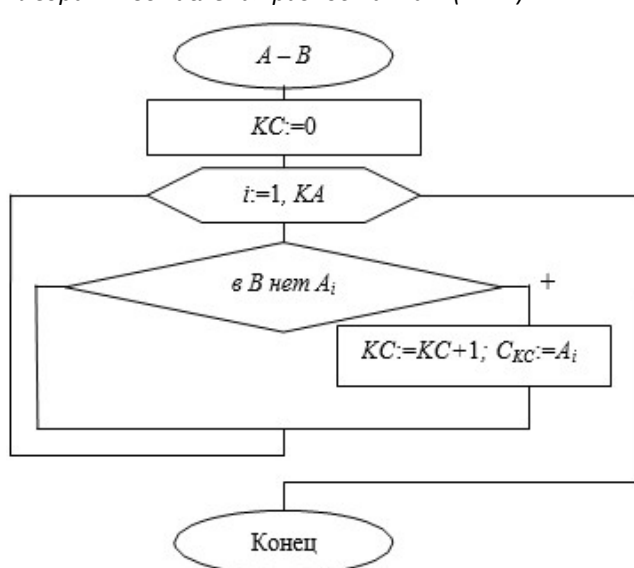
Алгоритм вычисления объединения A и B ($A \cup B$).



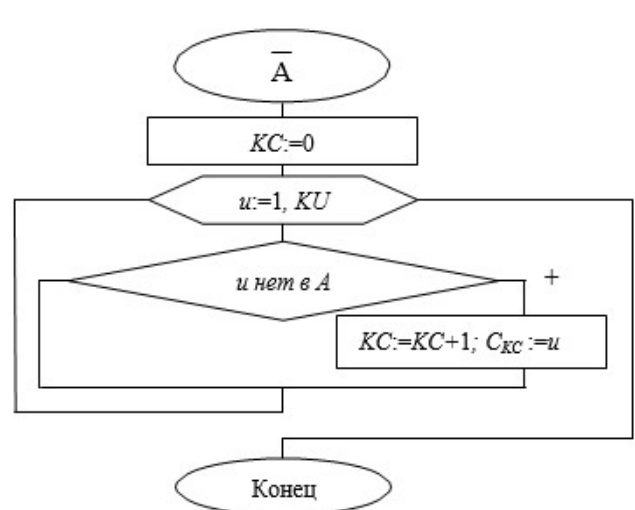
Алгоритм вычисления пересечения A и B ($A \cap B$).



Алгоритм вычисления разности A и B ($A - B$).



Алгоритм вычисления дополнения A



• Свойства операций над множествами

1. Идемпотентность: $A \cup A = A$; $A \cap A = A$
2. Коммутативность: $A \cup B = B \cup A$; $A \cap B = B \cap A$; $A \Delta B = B \Delta A$
3. Ассоциативность:
 $A \cup (B \cup C) = (A \cup B) \cup C$; $A \cap (B \cap C) = (A \cap B) \cap C$
 $A \Delta (B \Delta C) = (A \Delta B) \Delta C$
4. Дистрибутивность для пересечения и объединения:
 $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
 $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
5. Дистрибутивность для пересечения и разности:
 $A \cap (B - C) = (A \cap B) - (A \cap C)$
 $(A \cap B) - C = (A - C) \cap (B - C)$
6. Дистрибутивность для пересечения и симметрической разности:
 $A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$
7. Дистрибутивность для разности:
 $(A - B) - C = (A - C) - (B - C)$
 $(A - (B - C)) \text{ не всегда равно } (A - B) - (A - C)$
8. Законы поглощения: $(A \cap B) \cup A = A$; $(A - B) \cup A = A$; $(A \cup B) \cap A = A$
9. Законы склеивания: $A \cap \overline{B} \cup A \cap B = A$; $(A \cup \overline{B}) \cap (A \cup B) = A$
10. Свойства нуля:
 $A \cup \emptyset = A$; $A \cap \emptyset = \emptyset$
 $A - \emptyset = A$; $\emptyset - A = \emptyset$; $A \Delta \emptyset = A$
11. Свойства единицы:
 $A \cup U = U$; $A \cap U = A$
 $A - U = \emptyset$; $U - A = \overline{A}$; $A \Delta U = \overline{A}$
12. Инволютивность: $\overline{\overline{A}} = A$
13. Законы де Моргана: $\overline{A \cap B} = \overline{A} \cup \overline{B}$, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.
14. Законы де Моргана для разности, пересечения и объединения:
 $A - (B \cup C) = (A - B) \cap (A - C)$
 $A - (B \cap C) = (A - B) \cup (A - C)$
15. Свойства дополнения:
 $A \cup \overline{A} = U$; $A \cap \overline{A} = \emptyset$
 $A \Delta \overline{A} = U$; $A - \overline{A} = A$
16. Определения разности: $A - B = A \cap \overline{B} = (A \cap B) \Delta A$
17. Определения симметрической разности:
 $A \Delta B = (A \cup B) - (A \cap B) = (A - B) \cup (B - A) = (A \cap \overline{B}) \cup (B \cap \overline{A})$
18. Определения объединения: $A \cup B = (A \Delta B) \Delta (A \cap B) = (A - B) \Delta B = (B - A) \Delta A$
19. Определения пересечения: $A \cap B = (A \cup B) - (A \Delta B) = A - (A - B) = B - (B - A)$
20. Разложение Шеннона:
 $f(X, A_1, \dots, A_n) = \overline{X} \cap f(\emptyset, A_1, \dots, A_n) \cup X \cap f(U, A_1, \dots, A_n)$, где $f(X, A_1, \dots, A_n)$ — теоретико-множественное выражение, содержащее множества X, A_1, \dots, A_n ; $f(\emptyset, A_1, \dots, A_n)$ — выражение, получаемое из $f(X, A_1, \dots, A_n)$ путем подстановки вместо множества X пустого множества \emptyset ; $f(U, A_1, \dots, A_n)$ — выражение, получаемое из $f(X, A_1, \dots, A_n)$ путем подстановки вместо множества X универсума U .

• Нормальные формы Кантора

Выражение, задающее множество M в виде объединения различных элементарных пересечений, называется **нормальной формой Кантора** (НФК) множества M . Любое множество, которое может быть получено из порождающих множеств с помощью операций над множествами, может быть представлено в НФК.

Преобразуем выражение $\overline{A \Delta B - C}$ в НФК.

$$\begin{aligned}\overline{A \Delta B - C} &= \overline{(A \cap B \cup A \cap \bar{B}) - C} = \overline{(A \cap B \cup A \cap \bar{B}) \cap C} = \\ &= \overline{(A \cap B \cup A \cap \bar{B}) \cup C} = \overline{A \cap B \cap A \cap \bar{B} \cup C} = \\ &= \overline{(A \cup B) \cap (A \cup \bar{B}) \cup C} = \overline{\bar{A} \cap \bar{A} \cup \bar{A} \cap B \cup B \cap A \cup B \cap \bar{B} \cup C} = \\ &= \overline{\bar{A} \cap \bar{B} \cup B \cap A \cup C}.\end{aligned}$$

Любое множество, которое может быть получено из порождающих множеств с помощью операций над множествами, может быть представлено объединением некоторого количества различных конституент. Такое представление множества называется **совершенной НФК**.

Приведём пример совершенной НФК $A \cap \bar{B} \cap \bar{C} \cup \bar{A} \cap B \cap C \cup \bar{A} \cap \bar{B} \cap \bar{C} \cup A \cap B \cap \bar{C} \cup A \cap B \cap C$ которая примет вид $100 \cup 011 \cup 000 \cup 110 \cup 111$.

Конституенты, представленные двоичными векторами, разобьём на группы.

Номер группы			
0	1	2	3
000	100	011 110	111

Проверяем на возможность склеивания только конституенты из соседних групп. Конституенты склеиваются, если соответствующие им двоичные вектора различаются только одним i -м разрядом. Конституенты, участвовавшие в склеивании, отмечаем крестиком справа, а результат склеивания дописываем в таблицу в соответствующую группу:

Номер группы			
0	1	2	3
000+	100+	011+ 100+	111+
-00	1-0	-11 11-	

Объединение всех простых импликант образует **сокращенную НФК**: $\bar{B} \cap \bar{C} \cup A \cap \bar{C} \cup B \cap C \cup A \cap B$

Минимальной НФК множества M называется НФК, имеющая минимальную сложность (Количество вхождений первичных термов в выражение, задающее множество, определяет сложность представления множества) по отношению ко всем другим НФК этого же множества.

Если из сокращенной НФК исключить все лишние простые импликанты, то получим **тупиковую НФК**. Тупиковых НФК может быть несколько. Для получения тупиковой НФК используют импликантную матрицу Квайна, в которой строки соответствуют простым импликантам, а столбцы — конституентам. Тупиковая НФК, имеющая минимальную сложность, является **минимальной НФК**.

Импликантная матрица Квайна

Простые импликанты	Конституенты				
	100	011	000	110	111
-00	+		+		
1-0	+			+	
-11		+			+
11-				+	+

Для нахождения всех тупиковых НФК обозначим строки матрицы буквами a, d, c и d . Каждому столбцу поставим в соответствие объединение строк, которые его покрывают, а всей таблице — пересечение этих объединений: $(a \cup b) \cap c \cap a \cap (b \cup d) \cap (c \cup d)$. Используя свойства дистрибутивности, идемпотентности и поглощения преобразуем это выражение в объединение элементарных пересечений: $a \cap c \cap b \cup a \cap c \cap d$. Каждое элементарное пересечение определяет минимальное покрытие импликантной матрицы, т. е. тупиковую НФК. Итак, получили две тупиковые НФК: $\bar{B} \cap \bar{C} \cup B \cap C \cup A \cap \bar{C}$ и $\bar{B} \cap \bar{C} \cup B \cap C \cup A \cap B$.

• Методы доказательства теоретико-множественных тождеств

Равенство, левая и правая части которого представляют собой теоретико-множественное выражение, верное для любых входящих в них множеств, называют теоретико-множественным тождеством. Рассмотрим различные методы доказательства теоретико-множественных тождеств:

Метод двух включений

Чтобы доказать равенство множеств X и Y , достаточно доказать два включения $X \subseteq Y$ и $Y \subseteq X$, т.е. доказать, что из предположения $x \in X$ (для произвольного x) следует, что $x \in Y$, и, наоборот, из предположения $x \in Y$ следует, что $x \in X$.

Докажем этим методом тождество: $A \Delta B = (A \cup B) - (A \cap B)$.

Пусть $x \in A \Delta B$. Тогда, согласно определению симметрической разности, $x \in (A - B) \cup (B - A)$. Это означает, что $x \in (A - B)$ или $x \in (B - A)$. Если $x \in (A - B)$, то $x \in A$ и $x \notin B$, т.е. $x \in A \cup B$ и при этом $x \notin A \cap B$. Если же $x \in (B - A)$, то $x \in B$ и $x \notin A$, откуда $x \in A \cup B$ и $x \notin A \cap B$. Итак, в любом случае из $x \in (A - B) \cup (B - A)$ следует $x \in A \cup B$ и $x \notin A \cap B$, т.е. и $x \in (A \cup B) - (A \cap B)$. Таким образом, доказано, что $A \Delta B \subseteq (A \cup B) - (A \cap B)$.

Покажем обратное включение $(A \cup B) - (A \cap B) \subseteq A \Delta B$.

Пусть $x \in (A \cup B) - (A \cap B)$. Тогда $x \in A \cup B$ и $x \notin A \cap B$. Из $x \in A \cup B$ следует, что $x \in A$ или $x \in B$. Если $x \in A$, то с учетом $x \notin A \cap B$ имеем $x \notin B$, и поэтому $x \in A - B$. Если же $x \in B$, то опять-таки в силу $x \notin A \cap B$ получаем, что $x \notin A$ и $x \in B - A$. Итак, $x \in A - B$ или $x \in B - A$, т.е. $x \in (A - B) \cup (B - A)$. Следовательно, $(A \cup B) - (A \cap B) \subseteq A \Delta B$.

Оба включения имеют место и тождество доказано.

Метод эквивалентных преобразований

Теоретико-множественные тождества можно доказывать, используя свойства операций над множествами. Для этого нужно преобразовать левую часть в правую, или правую — в левую.

Докажем этим методом тождество: $A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$.

Преобразуем левую часть к правой.

1. По определению $B \Delta C = (B - C) \cup (C - B)$, поэтому $A \cap (B \Delta C) = A \cap ((B - C) \cup (C - B))$.
2. Используя свойство дистрибутивности для \cap и \cup получим $A \cap (B - C) \cup A \cap (C - B)$.
3. Используя свойство дистрибутивности для \cap и $-$ получим $((A \cap B) - (A \cap C)) \cup ((A \cap C) - (A \cap B))$.

Это есть определение симметрической разности для $(A \cap B)$ и $(A \cap C)$, т.е. $A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$.

Тождество доказано.

Метод характеристических функций (Арифметические характеристические функции)

Метод характеристических функций доказательства справедливости теоретико-множественного тождества заключается в выражении характеристических функций обеих его частей через характеристические функции входящих в него множеств. Тождества верны тогда и только тогда, когда характеристические функции левой и правой частей совпадают.

Докажем этим методом тождество: $(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C)$.

С одной стороны, $\chi_{(A \Delta B) \cap C}(x) = \chi_{A \Delta B}(x) \cdot \chi_C(x) = (\chi_A(x) + \chi_B(x) - 2\chi_A(x) \cdot \chi_B(x)) \cdot \chi_C(x) = \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x) \cdot \chi_C(x)$.

С другой стороны, $\chi_{(A \cap C) \Delta (B \cap C)}(x) = \chi_{A \cap C}(x) + \chi_{B \cap C}(x) - 2 \cdot \chi_{A \cap C}(x) \cdot \chi_{B \cap C}(x) = \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_C(x) \cdot \chi_B(x) \cdot \chi_C(x) = \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x) \cdot \chi_C(x)$.

АХФ левой и правой частей тождества совпадают. Следовательно, тождество верно.

Метод характеристических функций (Логические характеристические функции)

Доказательство методом ЛХФ проводится так же, как и методом АХФ. Для проверки равенства ЛХФ можно использовать их таблицы истинности.

Докажем этим методом тождество: $(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C)$.

Для левой части: $\lambda_{(A \Delta B) \cap C}(x) = \lambda_{A \Delta B}(x) \wedge \lambda_C(x) = (\lambda_A(x) \oplus \lambda_B(x)) \wedge \lambda_C(x)$.

Для правой части: $\lambda_{(A \cap C) \Delta (B \cap C)}(x) = \lambda_{A \cap C}(x) \oplus \lambda_{B \cap C}(x) = \lambda_A(x) \wedge \lambda_C(x) \oplus \lambda_B(x) \wedge \lambda_C(x)$.

• Решение теоретико-множественных уравнений

Алгоритм решения следующий:

1. Преобразовать исходное уравнения в уравнение с пустой правой частью.
 2. Выполнить разложение Шеннона левой части уравнения по неизвестному множеству X .
 3. Упростить φ^\emptyset и φ^U , используя свойства нуля, единицы и дополнения.
 4. Вычислить φ^\emptyset и $\overline{\varphi^U}$ при заданных исходных множествах.
 5. Если множество φ^\emptyset является подмножеством множества $\overline{\varphi^U}$, следовательно, уравнение имеет решения.
 6. Определить какое множество является минимальным по мощности частным решением, а какое максимальным.
 7. Для получения общего решения необходимо каждое подмножество $\overline{\varphi^U} - \varphi^\emptyset$ объединить с φ^\emptyset .
- И желательно выполнить проверку решения.

Рассмотрим пример решения теоретико-множественного уравнения

$$B \cap (\overline{X} \cup \overline{C}) \Delta A = (\overline{A} \cup X) \Delta B \cap X$$

при $A = \{3, 4, 5, 6, 8, 10\}$, $B = \{1, 2, 7, 8, 9, 10\}$, $C = \{1, 2, 3, 4, 5, 10\}$ и $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

1. Преобразуем исходное уравнение в уравнение с пустой правой частью:

$$B \cap (\overline{X} \cup \overline{C}) \Delta A \Delta (\overline{A} \cup X) \Delta B \cap X = \emptyset.$$

2. Выполним разложение Шеннона левой части уравнения по неизвестному множеству X :

$$\overline{X} \cap \varphi^\emptyset \cup X \cap \varphi^U = \emptyset,$$

где $\varphi^\emptyset = B \cap (\overline{\emptyset} \cup \overline{C}) \Delta A \Delta (\overline{A} \cup \emptyset) \Delta B \cap \emptyset$,

$$\varphi^U = B \cap (\overline{U} \cup \overline{C}) \Delta A \Delta (\overline{A} \cup U) \Delta B \cap U.$$

3. Упростим φ^\emptyset и φ^U , используя свойства нуля, единицы и дополнения: $\varphi^\emptyset = B \Delta A \Delta \overline{A} = B \Delta U = B$,

$$\varphi^U = B \cap \overline{C} \Delta A \Delta U \Delta B = B \cap \overline{C} \Delta A \Delta B.$$

4. Вычислим φ^\emptyset и $\overline{\varphi^U}$ при заданных исходных множествах:

$$\varphi^\emptyset = \{1, 2, 7, 8, 9, 10\} = \{3, 4, 5, 6\},$$

$$\begin{aligned} \overline{\varphi^U} &= \overline{\{1, 2, 7, 8, 9, 10\} \cap \{1, 2, 3, 4, 5, 10\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{1, 2, 7, 8, 9, 10\}} = \\ &= \overline{\{1, 2, 7, 8, 9, 10\} \cap \{6, 7, 8, 9\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{3, 4, 5, 6\}} = \\ &= \overline{\{7, 8, 9\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{3, 4, 5, 6\}} = \overline{\{3, 4, 5, 6, 7, 9, 10\} \Delta \{3, 4, 5, 6\}} = \\ &= \overline{\{7, 9, 10\}} = \{1, 2, 3, 4, 5, 6, 8\}. \end{aligned}$$

5. Множество $\varphi^\emptyset = \{3, 4, 5, 6\}$ является подмножеством множества $\overline{\varphi^U} = \{1, 2, 3, 4, 5, 6, 8\}$, следовательно, уравнение имеет решения.

6. Минимальным по мощности частным решением является множество $\varphi^\emptyset = \{3, 4, 5, 6\}$, а максимальным — $\overline{\varphi^U} = \{1, 2, 3, 4, 5, 6, 8\}$.

7. Для получения общего решения необходимо каждое подмножество множества $\overline{\varphi^U} - \varphi^\emptyset = \{1, 2, 8\}$ объединить с $\varphi^\emptyset = \{3, 4, 5, 6\}$. Общее решение: $\{\{3, 4, 5, 6\}, \{3, 4, 5, 6, 1\}, \{3, 4, 5, 6, 2\}, \{3, 4, 5, 6, 1, 2\}, \{3, 4, 5, 6, 8\}, \{3, 4, 5, 6, 1, 8\}, \{3, 4, 5, 6, 2, 8\}, \{3, 4, 5, 6, 1, 2, 8\}\}$.

Самостоятельно выполните проверку решения.

- Комбинаторные объекты: подмножества, перестановки (без повторений и с повторениями), размещения (без повторений и с повторениями), сочетания (без повторений и с повторениями). Теоремы о количестве комбинаторных объектов

Комбинаторный объект - набор, составленный из элементов множества и обладающий свойством, определяющим заданный комбинаторный объект.

Множество A называется **подмножеством** множества B , если истинно включение A в B ($A \subseteq B$ = истина).

<p>Перестановка – это комбинация элементов из N разных элементов, взятых в определенном порядке. В перестановке важен порядок следования элементов, и в перестановке должны быть задействованы все N элементов.</p>	<p>1: 123 2: 132 3: 213 4: 231 5: 312 6: 321</p>
<p>Перестановку можно представить последовательностью из n мест. Для того чтобы получить одну перестановку, нужно выполнить одно за другим n действий: заполнить 1-е место в последовательности, 2-е место и так до n-го места. Для выполнения 1-го действия (заполнения 1-го места) можно взять любой элемент из множества M и поставить его на 1-е место, т.е. его можно выполнить n-способами, и после этого в множестве M останется $n - 1$ элемент. Для выполнения 2-го действия (заполнения 2-го места) можно взять любой элемент из оставшихся в множестве M и поставить его на 2-е место, т.е. его можно выполнить $n - 1$-способами, и после этого в множестве M останется $n - 2$ элемента и т.д. По правилу произведения все n действий могут быть выполнены $n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1 = n!$ способами, следовательно, количество всех различных перестановок n-элементного множества равно $n!$.</p>	
<p>Размещением n-элементного множества по k местам называется расположенные в определенном порядке элементы k-элементного подмножества n-элементного множества. Количество мест не превышает количества элементов в множестве.</p>	<p>Все возможные размещения из множества элементов $\{1,2,3\}$ по 2.</p> <p>1: 1 2 2: 1 3 3: 2 1 4: 2 3 5: 3 1 6: 3 2</p>
<p>Размещение можно представить последовательностью из k мест. Для того чтобы получить одно размещение, нужно выполнить одно за другим k действий: заполнить 1-е место в последовательности, 2-е место и так до k-го места. Для выполнения 1-го действия (заполнения 1-го места) можно взять любой элемент из множества M и поставить его на 1-е место, т.е. его можно выполнить n-способами, и после этого в множестве M останется $n - 1$ элемент. Для выполнения 2-го действия (заполнения 2-го места) можно взять любой элемент из оставшихся в множестве M и поставить его на 2-е место, т.е. его можно выполнить $n - 1$-способами, и после этого в множестве M останется $n - 2$ элемента и так далее до k-го места. По правилу произведения все k действий могут быть выполнены $n(n - 1)(n - 2) * \dots * (n - k + 1) = \frac{n!}{(n - k)!}$ способами, следовательно, количество всех различных размещений n-элементного множества M по k местам равно</p> $\frac{n!}{(n - k)!}$	

<p>Размещением с повторениями n-элементного множества по k местам называется k-элементная последовательность, состоящая из элементов n-элементного множества M, причем элементы в последовательности могут повторяться. Количество мест может превышать количество элементов в множестве.</p>	
<p>Для того чтобы получить одно размещение с повторениями, нужно выполнить одно за другим k действий: заполнить 1-е место в последовательности, 2-е место и так до k-го места. Для выполнения каждого действия можно взять любой элемент из множества M и поставить его на соответствующее место, т.е. каждое из k действий можно выполнить n способами. По правилу произведения все k действий могут быть выполнены $n \cdot n \cdot \dots \cdot n = n^k$ способами, следовательно, количество всех различных размещений с повторениями n-элементного множества по k местам равно n^k.</p>	
<p>Сочетанием из n элементов по k называется k-элементное подмножество n-элементного множества.</p>	<p>123 135 234 256 124 136 235 345 125 145 236 346 126 146 245 356 134 156 246 456</p>
<p>Сочетания в лексикографическом порядке можно порождать, используя метод поиска с возвратом. Начиная с 1-го места будем последовательно формировать элементы сочетания C. На первое место в сочетании можно ставить элементы из множества $\{1..n - k + 1\}$. Формирование i-го элемента сочетания опишем рекурсивным алгоритмом. В цикле перебираются элементы множества $\{b..n - k + i\}$, которые можно поставить на i-е место. Если на i-е место поставлен элемент x, то минимальный элемент, который можно поставить на $i + 1$-е место, равен $x + 1$. Если заполнено k-е место, то сочетание сформировано и выводим его, иначе заполняем следующее $i + 1$-е место по алгоритму.</p>	
<p>Перестановкой с повторениями называется расположенные в определенном порядке элементы мультимножества (Множество, которое может содержать одинаковые элементы)</p>	
<p>Рассмотрим одну перестановку мультимножества и заменим в ней все одинаковые элементы разными. Тогда, число различных перестановок, которые можно составить из рассматриваемой, равно $n_1! \cdot n_2! \cdot \dots \cdot n_k!$. Проделав это для каждой перестановки, получим $n!$ перестановок. Следовательно, $P_n(n_1, n_2, \dots, n_k) \cdot n_1! \cdot n_2! \cdot \dots \cdot n_k! = n!$. Отсюда</p> $P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! \cdot n_2! \cdot \dots \cdot n_k!}$ <p>Алгоритм порождения всех перестановок с повторениями практически не отличается от алгоритма порождения всех перестановок множества, за исключением того, что в алгоритме обрабатывается мультимножество.</p>	

Сочетанием с повторениями из n элементов по k называется мультимножество мощности k , составленное из элементов n -элементного множества.

Например, из элементов множества $M = \{1,2,3\}$ можно составить такие сочетания по два с повторениями: $\{1,1\}$, $\{1,2\}$, $\{1,3\}$, $\{2,2\}$, $\{2,3\}$, $\{3,3\}$.

Каждое сочетание полностью определяется, если указать, сколько раз входит каждый элемент множества в сочетание. Поставим в соответствие каждому сочетанию последовательность нулей и единиц, составленную по следующему правилу: ставим подряд столько единиц, сколько раз входит первый элемент множества в сочетание, далее ставим нуль, и после него пишем столько единиц, сколько раз входит второй элемент множества в это сочетание и т.д. Например, написанным выше сочетаниям из трех элементов по два будут соответствовать такие последовательности: 1100, 1010, 1001, 0110, 0101, 0011. Таким образом, каждому сочетанию с повторениями из n по k соответствует последовательность из k единиц и $n - 1$ нулей. Количество таких последовательностей равно числу способов, которыми можно выбрать $n - 1$ мест для нулей из $n + k - 1$ общего числа мест $\binom{n+k-1}{n-1}$.

Рассмотрим алгоритм порождения сочетаний с повторениями в неубывающем лексикографическом порядке. Сочетания с повторениями из трех элементов по три, в неубывающем лексикографическом порядке запишутся следующим образом:

111 133
112 222
113 223
122 233
123 333

Сочетания с повторениями в лексикографическом порядке можно порождать, используя метод поиска с возвратом. Начиная с 1-го места будем последовательно формировать элементы сочетания F . На первое место в сочетании можно ставить элементы из множества $\{1..n\}$. Формирование i -го элемента сочетания опишем рекурсивным алгоритмом. В цикле перебираются элементы множества $\{b..n\}$, которые можно поставить на i -е место. Если на i -е место поставлен элемент x , то минимальный элемент, который можно поставить на $i + 1$ -е место, то же равен x . Если заполнено k -е место, то сочетание сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму.

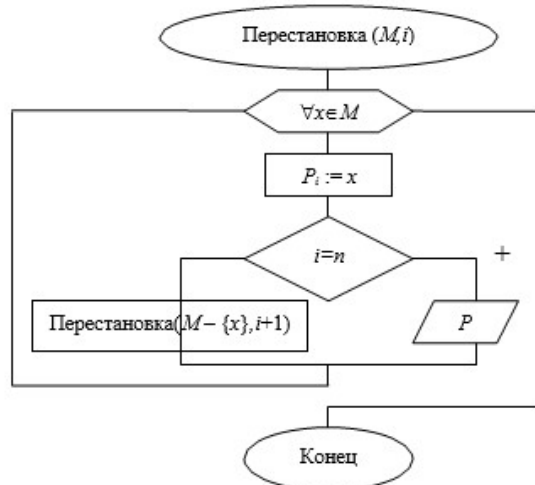
- Порождение комбинаторных объектов методом поиска с возвращением

Алгоритм порождения перестановок n -элементного множества.

Вход: M — множество, элементы которого можно поставить на i -е место; i — заполняемое место в последовательности P .

Выход: последовательность всех перестановок n -элементного множества.

Глобальные параметры: P — формируемая перестановка; n — мощность множества.



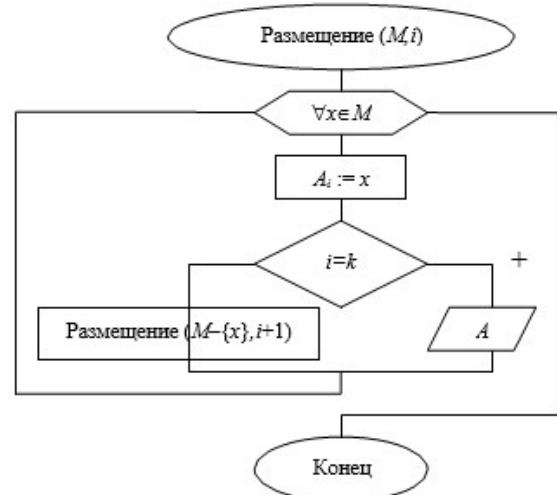
В цикле перебираются элементы множества M , которые можно поставить на i -е место. Если заполнено последнее место, то перестановка сформирована и выводим ее, иначе заполняем следующее $i + 1$ -е место по алгоритму, элементами множества M , за исключением элемента, поставленного на i -е место.

Алгоритм порождения размещений n -элементного множества по k местам.

Вход: M — множество, элементы которого можно поставить на i -е место; i — заполняемое место в последовательности P .

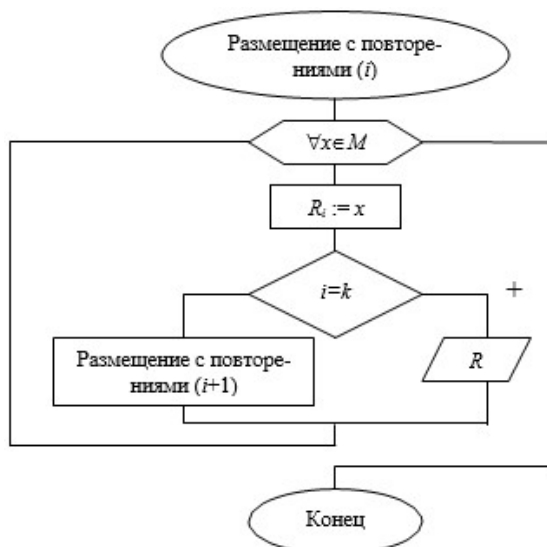
Выход: последовательность всех размещений n -элементного множества по k местам.

Глобальные параметры: A — формируемое размещение; k — количество мест.

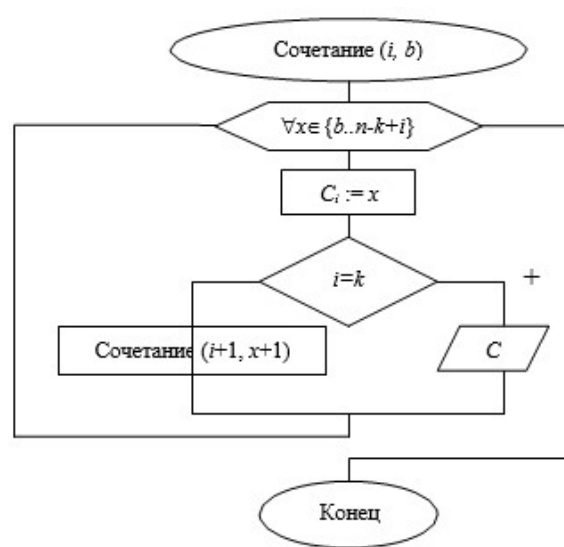


В цикле перебираются элементы множества M , которые можно поставить на i -е место. Если заполнено k -е место, то размещение сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму, элементами множества M , за исключением элемента, поставленного на i -е место.

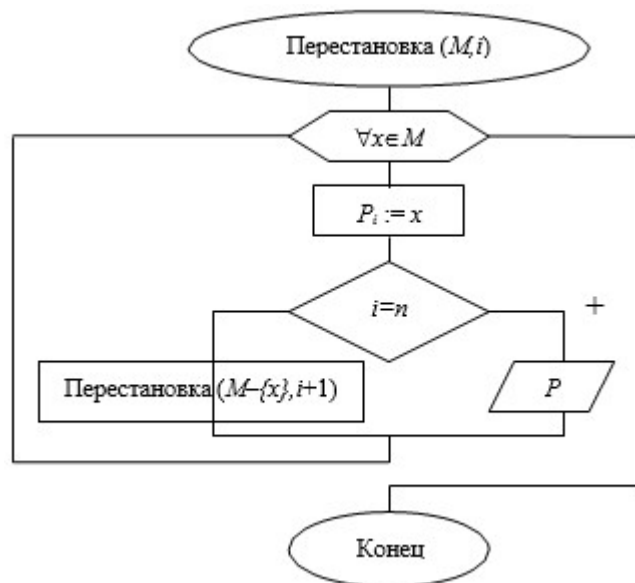
Размещение с повторениями n -элементного множества по k местам



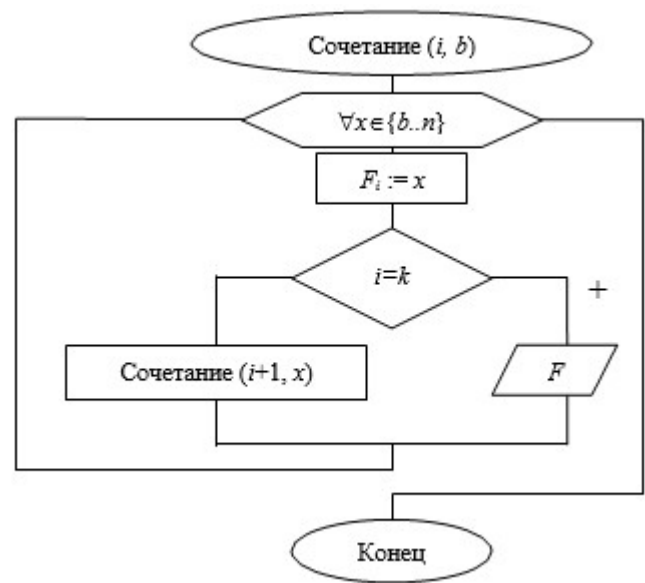
Алгоритм порождения сочетаний из n элементов по k .



Алгоритм порождения перестановок с повторениями



Алгоритм порождения сочетаний с повторениями из n элементов по k .



• Комбинаторные объекты и задачи выбора

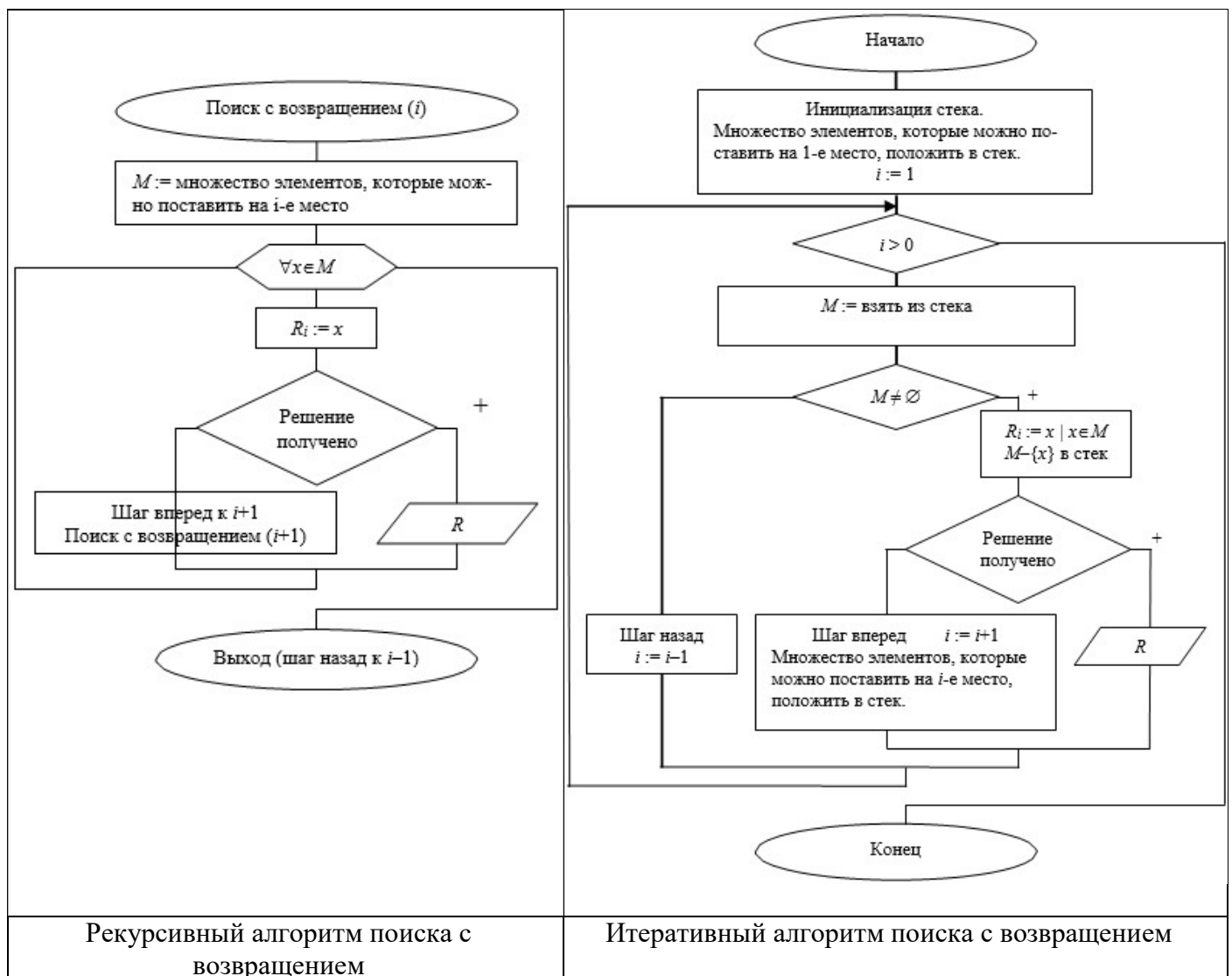
Под *комбинаторным объектом* принято понимать набор, составленный из элементов множества и обладающий свойством, определяющим заданный комбинаторный объект. Основными простейшими комбинаторными объектами являются подмножество, перестановка, размещение, сочетание, разбиение.

Задачи, для которых существует конечное множество M объектов, содержащее решение задачи, относятся к классу задач выбора. Элементы множества M называются *траекториями* задачи. Каждой траектории можно поставить в соответствие некоторую, как правило, числовую характеристику, называемую *функционалом*, позволяющую распознать траекторию, являющуюся решением задачи.

В общем случае, в постановке задачи выбора может потребоваться:

1. определить, существует ли решение;
2. найти все решения задачи;
3. определить значение функционала “лучшего” решения;
4. найти одно или все “лучшие” решения.

Алгоритм решения задачи выбора представляет собой порождение комбинаторных объектов, содержащих решение задачи, вычисление функционала для каждого объекта и определение принадлежности объекта множеству решений. Для того, чтобы преобразовать алгоритм порождения комбинаторных объектов в алгоритм решения задачи выбора, достаточно заменить блок вывода сформированного объекта на блок, в котором вычисляется значение функционала для полученного объекта и определяется принадлежность объекта множеству решений.



• Соответствия, виды соответствий

Определение 1. **Соответствием** между множествами A и B называют произвольное подмножество декартова произведения $F \subset A \times B$.

$$A \times B = \{(x, y) \mid x \in A \text{ и } y \in B\}$$

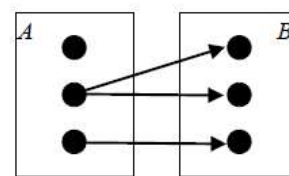
Определение 2. **Отображением** из множества A в множество B называется однозначное соответствие между A и B , т.е. такое соответствие, что для любого $a \in A$ найдётся ровно одно $b \in B$, соответствующее a .

Определение 3. **Инъективным** соответствием называется соответствие, при котором для любых $a_1 \neq a_2$ множества $F(a_1)$ и $F(a_2)$ не пересекаются. Инъективное отображение называется *инъекцией*.

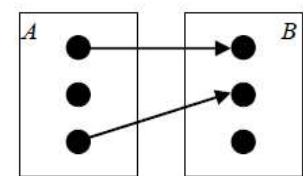
Определение 4. **Сюръективным** соответствием называется соответствие, при котором любой элемент B соответствует хотя бы одному элементу A , т.е. любой $b \in B$ лежит в $F(a)$ для некоторого $a \in A$. Сюръективное отображение называется *сюръекцией*.

Определение 5. **Биекцией** называется отображение, являющееся одновременно инъекцией и сюръекцией.

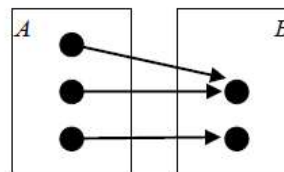
Утверждение 6. Соответствие является биекцией тогда и только тогда, когда каждому элементу A соответствует ровно один элемент B , а также каждый элемент B соответствует ровно одному элементу A .



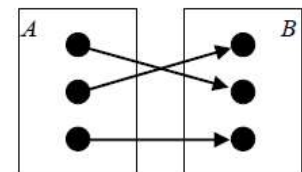
Соответствие, но не функция



Инъекция, но не сюръекция



Сюръекция, но не инъекция



Биекция

Рис.3.2. Различные виды соответствий

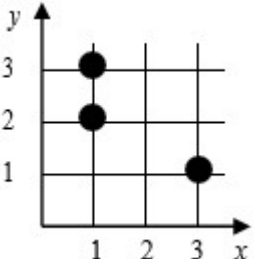
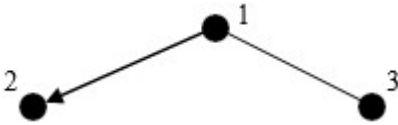
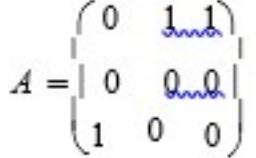
- **Отношения:**

Обобщенным понятием отношения является n -арное отношение R — подмножество прямого произведения n множеств: $R \subseteq A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) \mid a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}$.

К слову “отношение” часто добавляют прилагательное “бинарное”, тем самым подчёркивая, что в отношение входят два элемента. Если задано отношение R , то часто вместо $(x, y) \in R$ пишут xRy , например $x = y$, $x < y$ и так далее.

Бинарным отношением R называется подмножество $R \subseteq A^2$ и является отношением на множестве A . Бинарное отношение является частным случаем бинарного соответствия, поэтому оно может быть не функцией, функцией, отображением, сюръекцией, инъекцией, биекцией.

Способы задания отношений (Не надо переписывать!):

<p>Перечислением всех упорядоченных пар, принадлежащих отношению. $A = \{(1,2), (1,3), (3,1)\}$, $B = \{(3,1), (2,1), (1,3)\}$, $C = \{(1,3), (1,2), (3,1)\}$.</p>	
<p>Заданием характеристического свойства, выделяющего упорядоченные пары данного отношения среди упорядоченных пар других отношений. $A = \{(x,y) \mid x \in N \text{ и } y \in N \text{ и } x \leq 10 \text{ и } y < 10 \text{ и } x \text{ — чётно и } y \text{ — нечётно}\}$.</p>	
<p>Описанием порождающей процедуры с указанием множества (или множеств), которое пробегает параметр (или параметры) этой процедуры. $A = \{(x,y^2) \mid x \in N, y \in N\}$.</p>	
<p>Пару (x,y) можно изобразить точкой на координатной плоскости с абсциссой x и ординатой y, а бинарное отношение A — множеством точек, называемым <i>графиком</i> отношения A График отношения $A = \{(1,2), (1,3), (3,1)\}$ можно представить так:</p>	
<p>Бинарное отношение A можно представить в графической форме в виде <i>графа</i>, в котором <i>вершины</i> (кружочки или точки) соответствуют элементам множества и если $(x,y) \in A$, то вершины, соответствующие x и y, соединяются <i>дугой</i> (стрелочкой) от x к y. Если же $(x,y) \in A$ и $(y,x) \in A$, то вершины, соответствующие x и y соединяются <i>ребром</i> (линией, не имеющей направления). Граф отношения $A = \{(1,2), (1,3), (3,1)\}$ представлен на рисунке справа.</p>	
<p>Бинарное отношение A можно задать <i>характеристической функцией</i> $A(x,y)$, которая принимает истинное значение, если $(x,y) \in A$ и ложное — в противном случае. Характеристическую функцию отношения на конечном множестве мощностью N можно представить <i>матрицей</i> размером $N \times N$. Матрица характеристической функции отношения $A = \{(1,2), (1,3), (3,1)\}$ представлена на рисунке справа</p>	

Операции над отношениями

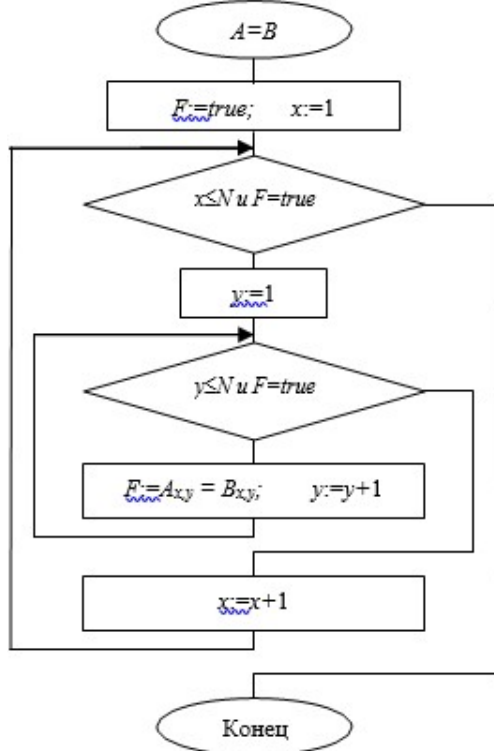
1. Включение A в B ($A \subseteq B$ или $B \supseteq A$) истинно, если каждая упорядоченная пара (x,y) отношения A принадлежит отношению B .
2. Равенство A и B ($A = B$) истинно, если отношения A и B состоят из одних и тех же упорядоченных пар, или если $A \subseteq B$ и $B \subseteq A$.
3. Строгое включение A в B ($A \subset B$ или $B \supset A$) истинно, если $A \subseteq B$ и $A \neq B$.
4. Объединение A и B ($A \cup B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар, которые принадлежат A или B , т.е. $A \cup B = \{(x,y) \mid (x,y) \in A \text{ или } (x,y) \in B\}$.
5. Пересечение A и B ($A \cap B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар, которые принадлежат каждому из отношений A и B , т.е. $A \cap B = \{(x,y) \mid (x,y) \in A \text{ и } (x,y) \in B\}$.
6. Разность A и B ($A - B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар отношения A , которые не принадлежат отношению B , т.е. $A - B = \{(x,y) \mid (x,y) \in A \text{ и } (x,y) \notin B\}$.
7. Симметрическая разность A и B ($A \Delta B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар отношения A , которые не принадлежат отношению B и только тех упорядоченных пар отношения B , которые не принадлежат отношению A , т.е. $A \Delta B = \{(x,y) \mid (x,y) \notin A \text{ и } (x,y) \in B \text{ или } (x,y) \in A \text{ и } (x,y) \notin B\}$.
8. Дополнение A до универсального отношения U (\bar{A}) есть отношение, состоящее из всех тех и только тех упорядоченных пар универсального отношения U , которые не принадлежат отношению A , т.е. $\bar{A} = \{(x,y) \mid (x,y) \notin A\}$.

Операции 1 — 8 над отношениями аналогичны операциям над множествами.

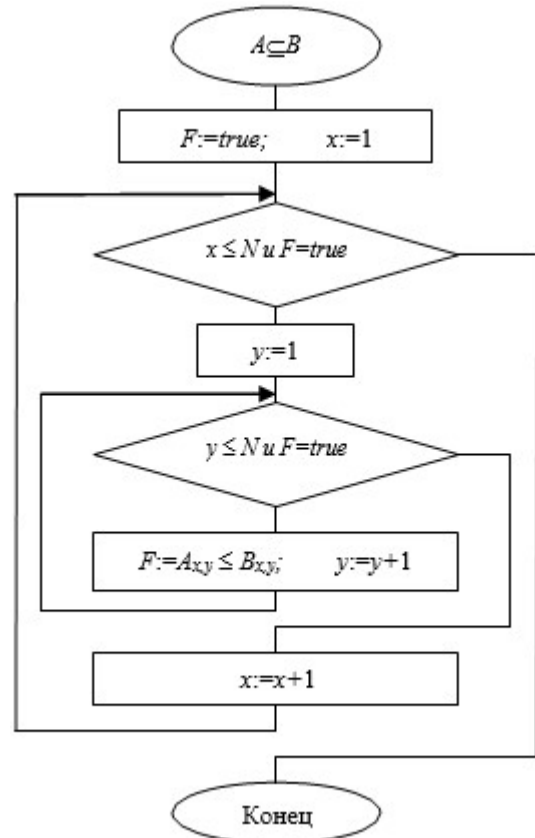
9. Обращением A (A^{-1}) называется отношение, которое состоит из всех тех и только тех упорядоченных пар (x,y) , для которых (y,x) принадлежит отношению A , т.е. $A^{-1} = \{(x,y) \mid (y,x) \in A\}$.
10. Композицией (суперпозицией или умножением) A и B ($A \circ B$) называется отношение, состоящее из всех тех и только тех упорядоченных пар (x,y) , для которых найдётся хотя бы один элемент z такой, что $(x,z) \in A$ и $(z,y) \in B$, т.е. $A \circ B = \{(x,y) \mid \text{существует } z \text{ такой, что } (x,z) \in A \text{ и } (z,y) \in B\}$.
11. Степенью отношения A называется его композиция с самим собой: $A^0 = I$, $A^1 = A$, $A^2 = A \circ A$, $A^n = A \circ A^{n-1}$. Пара (x,y) принадлежит отношению A^n , если в графе отношения A есть цепочка из n дуг (ребер), соединяющая вершину x с вершиной y .

- Программная реализация операций над отношениями

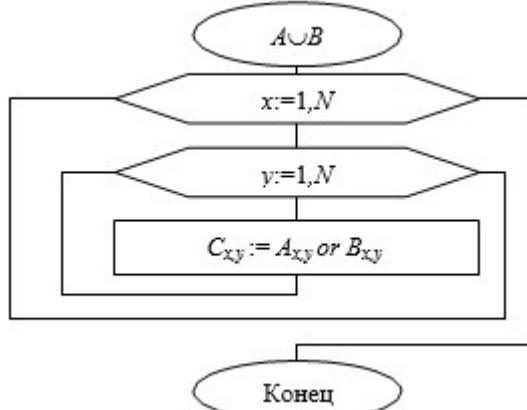
Алгоритм вычисления равенства отношений $A \cup B$ ($A = B$).
 Вход: A — двумерный массив, хранящий матрицу отношения A ; B — двумерный массив, хранящий матрицу отношения B ; N — мощность множества.
 Выход: $F = \text{true}$, если $A = B$, иначе $F = \text{false}$.



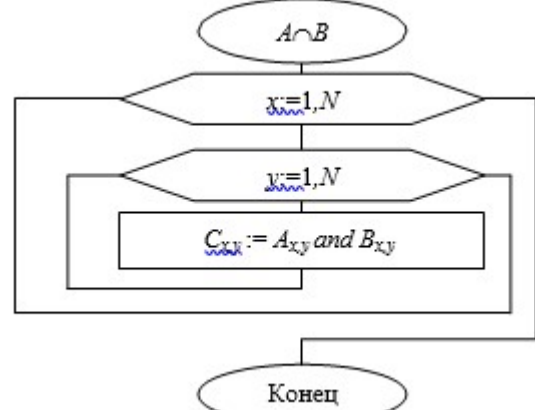
Алгоритм вычисления включения отношения A в B .



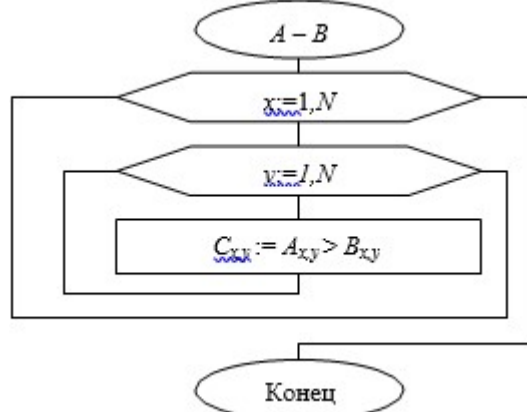
Алгоритм вычисления объединения отношений A и B



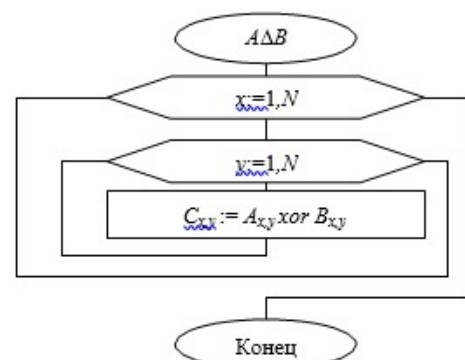
Алгоритм вычисления пересечения отношений A и B .



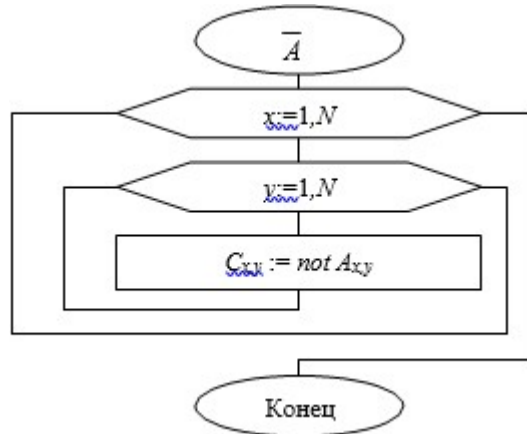
Алгоритм вычисления разности отношений A и B



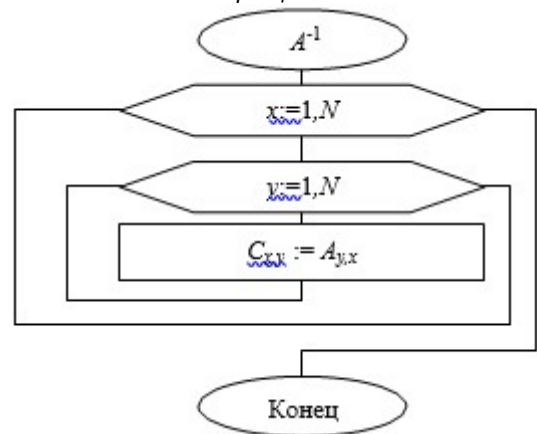
Алгоритм вычисления симметрической разности отношений A и B .



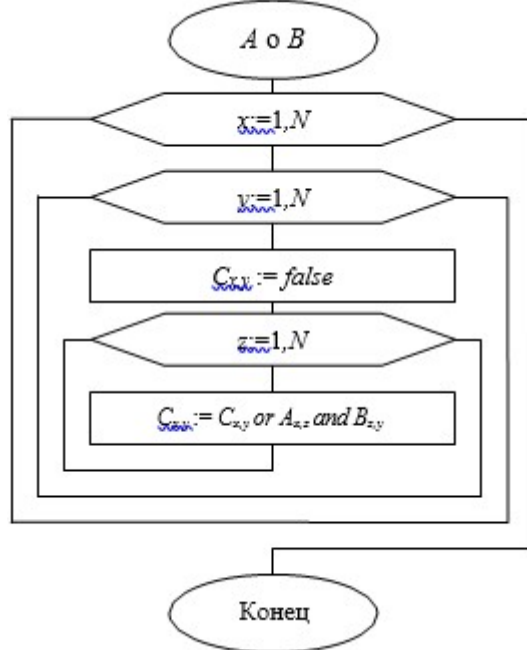
Алгоритм вычисления дополнения отношения A



Алгоритм вычисления обращения отношения A



Алгоритм вычисления композиции отношений A и B



● Свойства отношений

Пусть R — бинарное отношение на множестве A , I — тождественное отношение, U — универсальное отношение. Вместо $(x,y) \in R$ будем иногда использовать обозначение xRy . Отношения могут обладать следующими *основными* свойствами:

1. R *рефлексивно*, если xRx для всех $x \in A$ (граф такого отношения имеет петли при каждой вершине). Рефлексивность отношения R определяется истинностью значения выражения $I \subseteq R$.

2. R *антирефлексивно*, если $(x,x) \notin R$ для всех $x \in A$ (граф такого отношения не имеет петель). Антирефлексивность отношения R определяется истинностью значения выражения $R \cap I = \emptyset$. Нерефлексивное отношение в общем случае не является антирефлексивным. Отношение, граф которого содержит вершины с петлями и без петель — нерефлексивное и не антирефлексивное.

3. R *симметрично*, если из xRy следует yRx для всех $x,y \in A$ (такое отношение изображается неориентированным графом, возможно с петлями). Симметричность отношения R определяется истинностью значения выражения $R = R^{-1}$.

4. R *антисимметрично*, если из xRy следует $(y,x) \notin R$ для всех $x,y \in A$ (такое отношение изображается ориентированным графом, возможно с петлями). Антисимметричность отношения R определяется истинностью значения выражения $R \cap R^{-1} \subseteq I$. Отношение, граф которого содержит ориентированные дуги и неориентированные рёбра, не является ни симметричным, ни антисимметричным.

5. R *транзитивно*, если из xRz и zRy следует xRy для всех $x,y,z \in A$. Транзитивность отношения R определяется истинностью значения выражения $R \circ R \subseteq R$.

6. R *антитранзитивно*, если из xRz и zRy следует $(x,y) \notin R$ для всех $x,y,z \in A$. Антитранзитивность отношения R определяется истинностью значения выражения $(R \circ R) \cap R = \emptyset$.

7. R *полно*, если из $x \neq y$ следует, что xRy или yRx для всех $x,y \in A$ (такое отношение изображается полным графом). Полнота отношения R определяется истинностью значения выражения $R \cup I \cup R^{-1} = U$.

В пример можно привести 0-ую матрицу.	В пример можно привести 1-ую матрицу.
Она является: Транзитивной, антитранзитивной Симметрична, антисимметрична Антирефлексивна	Она является: Толерантной, эквивалентной, строгого и нестрогого порядка.

- **Отношения эквивалентности и порядка**

Следующие *производные* свойства отношений определяются совокупностью основных свойств.

Отношение R является отношением:

- *толерантности*, если оно рефлексивно, симметрично;
- *эквивалентности*, если оно рефлексивно, симметрично и транзитивно (обозначается \sim);
- *порядка*, если оно антисимметрично и транзитивно;
- *нестромого порядка*, если оно отношение порядка (антисимметрично и транзитивно) и рефлексивно (обозначается \leq);
- *строгого порядка*, если оно отношение порядка (антисимметрично и транзитивно) и антирефлексивно (обозначается $<$);
- *линейного порядка*, если оно отношение порядка (антисимметрично и транзитивно) и полно;
- *нестромого линейного порядка*, если оно отношение нестромого порядка (антисимметрично, транзитивно и рефлексивно) и полно;
- *строгого линейного порядка*, если оно отношение строгого порядка (антисимметрично, транзитивно и антирефлексивно) и полно.

В пример можно привести 0-ую матрицу.	В пример можно привести 1-ую матрицу.
Она является: Транзитивной, антитранзитивной Симметрична, антисимметрична Антирефлексивна => является отношением порядка, строгого порядка.	Она является: Толерантной, эквивалентной, строгого и нестромого порядка.

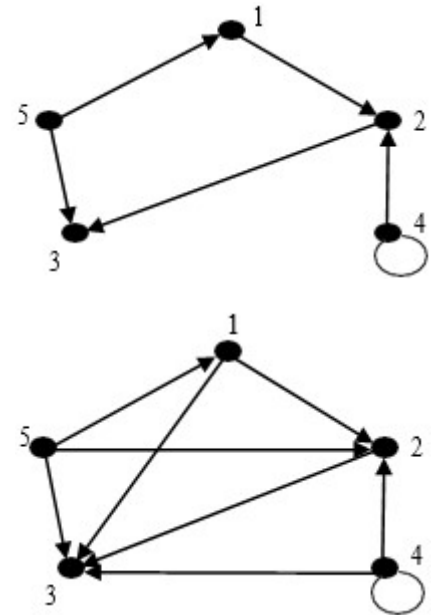
• Замыкание отношений

Замыканием отношения A относительно свойства S , называется отношение C_S , наименьшее по мощности из всех отношений, обладающих свойством S и содержащих A в качестве подмножества. Для того, чтобы получить замыкание C_S отношения A относительно свойства S , в него нужно добавить минимально возможное количество пар, после чего полученное отношение будет обладать свойством S .

Алгоритмы вычисления рефлексивного и симметричного замыкания довольно просты. Чтобы получить рефлексивное замыкание C_{ref} отношения A , достаточно в отношение A добавить все пары вида (x, x) , т.е. $C_{ref} = A \cup I$. Чтобы получить симметричное замыкание C_{sim} отношения A , нужно просмотреть все пары отношения A и если пара $(x, y) \in A$, то пару (y, x) добавить в отношение, т.е. $C_{sim} = A \cup A^{-1}$.

Рассмотрим пример. Отношение A задано графом:

Анализируя вершину 1 видим дуги из 5 в 1 и из 1 в 2, образующие путь длины два и проходящий через вершину 1, поэтому добавляем дугу из 5 в 2. В вершину 2 теперь входят три дуги из 1, 4 и 5, а выходит одна дуга в вершину 3, поэтому добавляем дуги из 1 в 3, из 4 в 3, а дуга из 5 в 3 уже есть. Из третьей вершины дуги не выходят, поэтому ничего не добавляем. Из четвертой вершины ребро идет в нее же и выходит во вторую, в этом случае новые дуги не добавляются. В пятую вершину дуги не входят, поэтому новые дуги не вводим. Таким образом исследованы все вершины графа, добавлены все необходимые дуги и граф (рис.3.18, б) транзитивного замыкания получен.

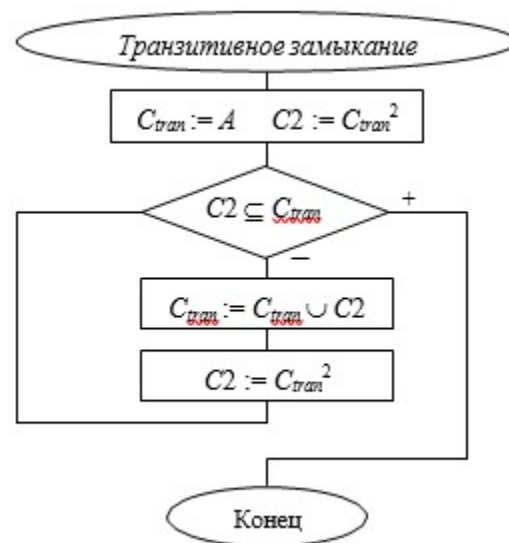


• Нахождение транзитивного замыкания.

Транзитивное замыкание C_{tran} отношения A вычисляется сложнее.

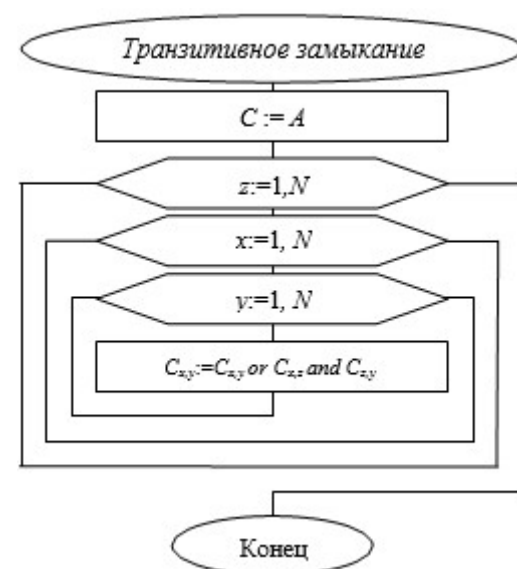
1 Способ

Предположим, что отношение A обладает свойством транзитивности, тогда $C_{tran} = A$. Для определения транзитивности C_{tran} вычислим $C2 = C_{tran}^2$ и проверим истинность $C2 \subseteq C_{tran}$. Если $C2 \subseteq C_{tran}$ истинно, то C_{tran} транзитивно и представляет собой транзитивное замыкание отношения A . Если же $C2 \subseteq C_{tran}$ ложно, то C_{tran} не транзитивно и к нему нужно добавить пары, принадлежащие $C2$: $C_{tran} := C_{tran} \cup C2$. В результате добавления пар, принадлежащих $C2$, отношение C_{tran} может стать транзитивным или не транзитивным. Для проверки опять вычислим $C2 = C_{tran}^2$ и повторим процесс.



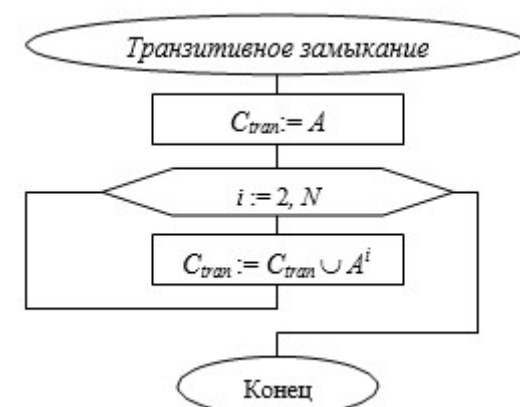
2 Способ (алгоритм Уоршалла)

Рассмотрим графический способ получения транзитивного замыкания, основанный на преобразовании графа исходного отношения. Для того, чтобы построить граф отношения C_{tran} , являющимся транзитивным замыканием отношения A , изготовим копию графа отношения A . Затем будем отыскивать такие тройки вершин x , y и z , для которых в графе есть дуги из x в z и из z в y . Для каждой такой тройки добавим в граф дугу из x в y , если такой дуги еще нет. Когда таким способом уже нельзя добавить новых дуг, каждые две вершины, соединенные цепочкой дуг, оказываются соединенными одной дугой. Следовательно, построение закончено и граф транзитивного замыкания отношения A получен.



Для того чтобы систематизировать процесс добавления дуг, будем последовательно рассматривать все вершины графа с первой до последней. Для каждого пути длины два, проходящего через рассматриваемую вершину, проведем дугу из начальной вершины этого пути в его конечную вершину. Граф транзитивного замыкания отношения A будет построен после однократного исследования всех вершин графа. Повторное исследование вершин не может добавить новых дуг (доказано Уоршаллом в 1962 году).

3 способ (объединения степеней):



● Разбиение множества на классы эквивалентности

Пусть задано отношение эквивалентности R на множестве M и $x \in M$. Подмножество элементов множества M , эквивалентных x , называется *классом эквивалентности* для x : $[x] = \{y \mid y \in M \text{ и } yRx\}$.

Алгоритм построения разбиения множества M на классы эквивалентности.

Вход: множество M и отношение эквивалентности R .

Выход: S — разбиение множества M на классы эквивалентности.

1. $A := M; S := \emptyset$;
2. Пока $A \neq \emptyset$ выполнять:
 Для $A, x \in A$ и R построить $[x]$ по алгоритму 3.1,
 добавить его в S и вычесть из A ;
3. Конец.

Для любого элемента x множества M класс эквивалентности не пуст, так как содержит в себе по крайней мере элемент x ($x \sim x$).

Любые два эквивалентных элемента x и y образуют равные классы эквивалентности, т.е. если $x \sim y$, то $[x] = [y]$.

Неэквивалентные элементы x и y образуют непересекающиеся классы эквивалентности.

Всякое отношение эквивалентности R на множестве M определяет единственное разбиение множества M на классы эквивалентности. Множество всех классов эквивалентности называется *фактормножеством* множества M по эквивалентности R .

Алгоритм построения класса эквивалентности $[x]$.

Вход: M — множество; $x \in M$; R — отношение эквивалентности.

Выход: $[x]$ — класс эквивалентности для x .

1. $[x] := \{x\}$;
2. Для всех $y \in M$ выполнить:
 если yRx , то включить y в $[x]$;
3. Конец.

Подмножество элементов множества M , эквивалентных x , называется *классом эквивалентности* для x : $[x] = \{y \mid y \in M \text{ и } yRx\}$.

Применим описанный алгоритм к матрице отношения делимости:

Суммируя элементы по столбцам матрицы, получим массив $W = (0, 1, 1, 1, 2, 2, 2, 3)$. В этом массиве первый элемент равен нулю, следовательно, соответствующий элемент множества (1) принадлежит нулевому уровню. Заменим его значением -1 и вычтем из массива W строку матрицы, соответствующую элементу 1 множества. Получим $W = (-1, 0, 0, 0, 2, 2, 2, 3)$. В этом массиве второй, третий и четвертый элементы равны нулю, следовательно, соответствующие элементы множества (2, 3 и 5) принадлежат первому уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 2, 3 и 5 множества. Получим $W = (-1, -1, -1, -1, 0, 0, 0, 3)$. В этом массиве пятый, шестой и седьмой элементы равны нулю, следовательно, соответствующие элементы множества (6, 10 и 15) принадлежат второму уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 6, 10 и 15 множества. Получим $W = (-1, -1, -1, -1, -1, -1, -1, 0)$. Последний элемент массива равен нулю, следовательно, элемент множества (30) принадлежит третьему уровню. Заменим последний элемент значением -1 . Неотрицательных элементов теперь в массиве нет, конец алгоритма.

	1	2	3	5	6	10	15	30
1	0	1	1	1	0	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	0	1	0
5	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0

- **Формирование отношения эквивалентности по разбиению**

Всякое разбиение S множества M определяет единственное отношение эквивалентности R на множестве M . Ниже приведены два алгоритма, определяющие отношение эквивалентности R на множестве M по заданному разбиению S .

Алгоритм построения отношения эквивалентности R по разбиению S множества M .

Вход: M — множество; S — разбиение множества M .

Выход: R — отношение эквивалентности на множестве M , определяемое разбиением S .

1. $R := \emptyset$;
2. Для всех пар $(x, y) \in M^2$ выполнить:
если x и y принадлежат одному и тому же подмножеству разбиения S , то (x, y) включить в R ;
3. Конец.

Алгоритм построения отношения эквивалентности R по разбиению S множества M .

Вход: M — множество; S — разбиение множества M .

Выход: R — отношение эквивалентности на множестве M , определяемое разбиением S .

1. $R := \emptyset$;
2. Для всех подмножеств $S_i \in S$ выполнить:
каждую пару (x, y) , такую, что $x \in S_i$ и $y \in S_i$, включить в R ;
3. Конец.

• Упорядоченные множества.

Множество вместе с заданным на нем отношением порядка называют **упорядоченным множеством**. Множество M с заданным на нем отношением порядка \leq записывается как пара (M, \leq) .

Элементы x и y упорядоченного множества (M, \leq) называют *сравнимыми* по отношению порядка \leq , если $x \leq y$ или $y \leq x$ (напомним, что запись $x \leq y$ означает, что пара (x, y) принадлежит отношению \leq). В противном случае элементы x и y называют *несравнимыми*. Если отношение \leq является отношением линейного порядка, то все элементы множества M попарно сравнимы и множество M называется *линейно упорядоченным*.

Элемент, $a \in M$ называют *наибольшим элементом* множества M , если для всех $x \in M$ верно $x \leq a$.

Элемент $b \in M$ называют *максимальным элементом* множества M , если для всех $x \in M$ верно $x \leq b$ или x и b несравнимы.

Аналогично определяются *наименьший* и *минимальный* элементы множества.

Минимальный и максимальный элемент существует в любом упорядоченном множестве, причем как максимальных, так и минимальных элементов может быть больше одного. Наибольший и наименьший элемент в упорядоченном множестве может и не существовать. Если же наибольший (наименьший) элемент существует, то он единственный и является максимальным (минимальным).

Примером упорядоченного множества может служить множество $\{1, 2, 3, 5, 6, 10, 15, 30\}$ с отношением делимости, т.е. пара (x, y) принадлежит отношению делимости, если x является делителем y (y делится на x без остатка). Матрица характеристической функции отношения делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$ представлена на рисунке справа.

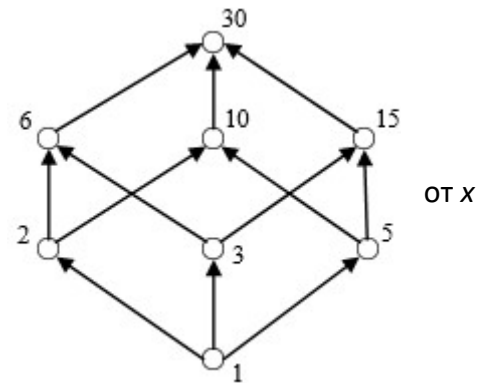
	1	2	3	5	6	10	15	30
1	0	1	1	1	0	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	0	1	0
5	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0

- **Отношение доминирования.**

Отношением \triangleleft доминирования, ассоциированного с отношением порядка \leq будем называть отношение, состоящее из всех таких пар (x, y) , что $x < y$ и не существует такого элемента z , что $x < z < y$. Чтобы получить матрицу отношения \triangleleft , нужно проанализировать каждую единицу в матрице отношения $<$, и, если единица соответствует паре (x, y) и существует такой элемент z , что $x < z < y$, то заменить ее нулем. Матрица характеристической функции отношения доминирования, ассоциированного с отношением делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$, представлена на рисунке.

	1	2	3	5	6	10	15	30
1	0	1	1	1	0	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	0	1	0
5	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0

Отношение доминирования удобно представлять графически в виде *диаграммы Хассе*. На этой диаграмме элементы множества изображаются кружочками. При этом если $x \triangleleft y$, то кружочек, изображающий элемент y , располагается выше кружочка, изображающего элемент x , и соединяются стрелочкой, ведущей к y . На рисунке представлена диаграмма Хассе отношения доминирования, ассоциированного с отношением делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$.



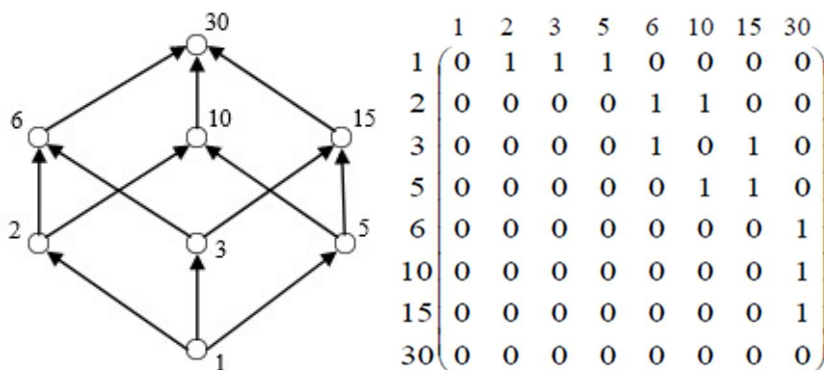
- **Топологическая сортировка.**

На диаграмме Хассе элементы множества располагаются по уровням. Элемент u располагается на нулевом уровне, если не существует такого элемента x , что $x \triangleleft u$. Элементы нулевого уровня соответствуют нулевым столбцам матрицы отношения доминирования. Если из матрицы удалить строки и столбцы, соответствующие элементам нулевого уровня, то элементы, соответствующие нулевым столбцам полученной матрицы, представляют собой элементы второго уровня. Продолжая процесс, получим элементы каждого уровня. Распределение элементов множества по уровням называют **топологической сортировкой**. При программной реализации топологической сортировки не стоит удалять строки и столбцы матрицы и каждый раз анализировать всю матрицу с целью поиска нулевых столбцов. Можно поступить следующим образом:

- 1) найти суммы элементов по всем столбцам матрицы и сохранить их в массиве W ;
- 2) элементы множества, соответствующие нулевым элементам массива W , считать элементами очередного уровня;
- 3) нулевые элементы массива W заменить отрицательным числом;
- 4) если в массиве W есть неотрицательные элементы, то вычесть из него поэлементно каждую строку матрицы, соответствующую элементу полученного в п. 2 уровня и перейти к п. 2, иначе конец алгоритма.

Рассмотрим следующий пример:

Суммируя элементы по столбцам матрицы, получим массив $W = (0, 1, 1, 1, 2, 2, 2, 3)$. В этом массиве первый элемент равен нулю, следовательно, соответствующий элемент множества (1) принадлежит нулевому уровню. Заменим его значением -1 и вычтем из массива W строку матрицы, соответствующую элементу 1 множества. Получим $W = (-1, 0, 0, 0, 2, 2, 2, 3)$. В этом массиве второй, третий и четвертый элементы равны нулю, следовательно, соответствующие элементы множества (2, 3 и 5) принадлежат первому уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 2, 3 и 5 множества. Получим $W = (-1, -1, -1, -1, 0, 0, 0, 3)$. В этом массиве пятый, шестой и седьмой элементы равны нулю, следовательно, соответствующие элементы множества (6, 10 и 15) принадлежат второму уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 6, 10 и 15 множества. Получим $W = (-1, -1, -1, -1, -1, -1, -1, 0)$. Последний элемент массива равен нулю, следовательно, элемент множества (30) принадлежит третьему уровню. Заменим последний элемент значением -1 . Неотрицательных элементов теперь в массиве нет, конец алгоритма.



• Графы и родственные им объекты

Пусть V — непустое множество, $V^{\{2\}}$ — множество всех его двух-элементных подмножеств. Пара (V, E) , где $E \subseteq V^{\{2\}}$ называется графом (неориентированным графом).

Часто рассматриваются следующие **родственные графам объекты**.

1. Если элементами множества E являются упорядоченные пары, то граф называется ориентированным (или орграфом). В этом случае элементы множества E называются дугами.
2. Если элементом множества E может быть пара одинаковых (не различных) элементов множества V , то такой элемент множества E называется петлей, а граф называется псевдографом.
3. Если E является мультимножеством, содержащим несколько одинаковых элементов, то эти элементы называются кратными ребрами, а граф называется мультиграфом.
4. Если элементом множества E являются не обязательно двухэлементные, а любые подмножества множества V , то такие элементы множества E называются гиперребрами, а граф называется гиперграфом. На диаграмме гиперграфа гиперребро представляет собой замкнутую линию, охватывающую инцидентные гиперребру вершины.
5. Если задана функция $F: V \rightarrow M$ и/или $H: E \rightarrow M$, то множество M называется множеством *весов*, а граф называется *взвешенным*. На диаграмме весами отмечаются вершины и/или ребра. В качестве весов обычно используются числа или символы.

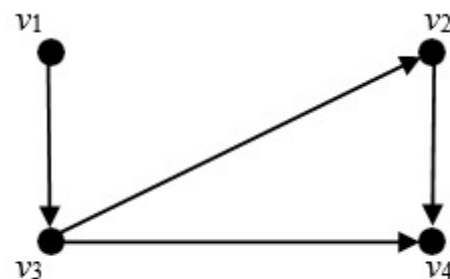


Рис.4.3. Диаграмма орграфа

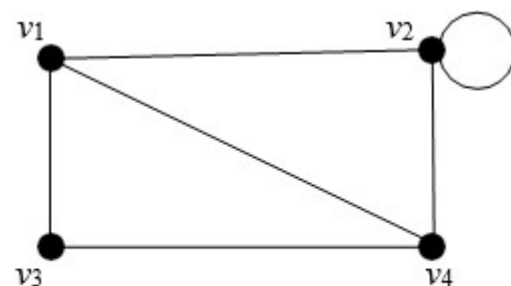


Рис.4.4. Диаграмма псевдографа

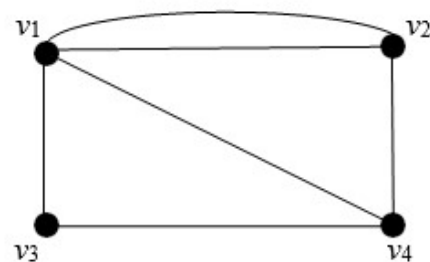


Рис.4.5. Диаграмма мультиграфа

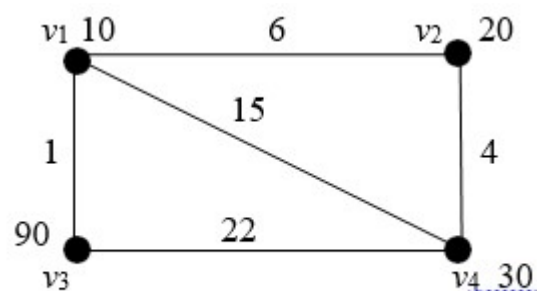


Рис.4.7. Диаграмма взвешенного графа

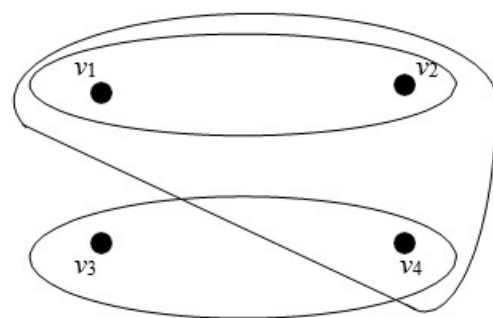
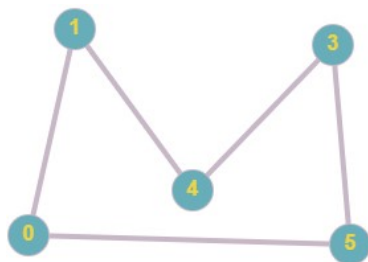


Рис.4.6. Диаграмма гиперграфа

- Способы задания

Граф можно представить графически в виде диаграммы: каждая вершина представляется точкой или окружностью, а каждое ребро представляется линией, соединяющей его концевые вершины.



Матричные способы представления графов:

1. Матрицей смежности графа $G = (V, E)$ называется матрица A порядка $n \times n$, где $n = |V|$. Элемент матрицы смежности $a_{ij} = 1$, если $(v_i, v_j) \in E$, $v_i, v_j \in V$ и $a_{ij} = 0$, если $(v_i, v_j) \notin E$. В матрице смежности строки и столбцы соответствуют вершинам графа.

2. Матрицей инцидентности графа $G = (V, E)$ называется матрица B порядка $n \times m$, где $n = |V|$, $m = |E|$. Элементы матрицы инцидентности b_{ij} определяются следующим образом: $b_{ij} = 1$, если i -я вершина является началом j -й дуги, $b_{ij} = -1$, если i -я вершина является концом j -й дуги, $b_{ij} = 0$, если i -я вершина и j -я дуга не инцидентны.

3. Матрицей дуг (списком дуг) графа $G = (V, E)$ называется матрица C порядка $2 \times m$, где $m = |E|$. Элементы матрицы C определяются следующим образом: $c_{1k} = v_i$ и $c_{2k} = v_j$, если $e_k = (v_i, v_j)$, т.е. если i -я вершина является началом k -й дуги, а j -я вершина является концом k -й дуги.

1	2
0, 1, 0, 0, 1,	1, 0, 0, 0, 1
1, 0, 0, 1, 0,	1, 1, 0, 0, 0
0, 0, 0, 1, 1,	0, 0, 1, 1, 0
0, 1, 1, 0, 0,	0, 1, 1, 0, 0
1, 0, 1, 0, 0,	0, 0, 0, 1, 1

- **Изоморфизм**

Два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются *изоморфными*, если существует взаимнооднозначное соответствие $p : V_1 \rightarrow V_2$ между вершинами графов, сохраняющее смежность, т.е. если вершины v_i и v_j смежны в графе G_1 , то вершины $p(v_i)$ и $p(v_j)$ смежны в графе G_2 , если же вершины v_i и v_j не смежны в графе G_1 , то вершины $p(v_i)$ и $p(v_j)$ не смежны в графе G_2 .

Изоморфные графы можно изобразить диаграммами, например, как ниже, которые различаются только обозначениями вершин.



Рис.4.10. Диаграммы изоморфных графов

- Поиск маршрутов, цепей, циклов методом поиска с возвращением

Получить все маршруты W длины l в графе G , начинающиеся вершиной v можно, используя метод поиска с возвращением. Сначала на первое место маршрута W поставим вершину v . Затем, начиная со второго элемента, последовательно формируем элементы маршрута. Формирование i -го элемента маршрута опишем рекурсивным алгоритмом, блок-схема которого представлена справа. В цикле перебираются вершины, смежные вершине, стоящей на $i - 1$ -м месте в маршруте W , т.е. элементы множества $\Gamma(W_{i-1})$, которые можно поставить на i -е место. Если заполнено последнее $l + 1$ -е место, то маршрут получен и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму. Слегка модернизировав этот алгоритм, можно получить не только маршруты, но также цепи и циклы:

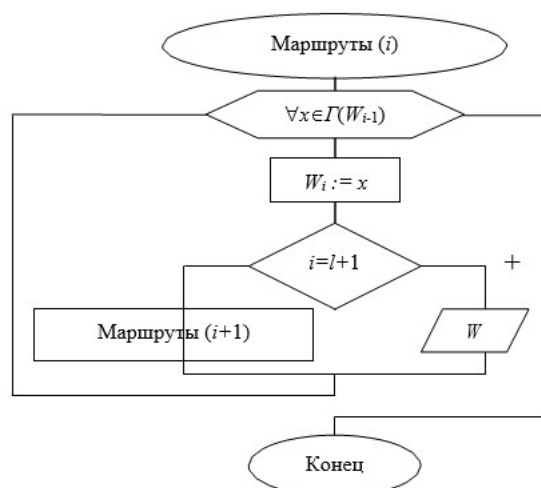


Рис.4.11. Рекурсивный алгоритм получения всех маршрутов W длины l в графе G , начинающихся вершиной v

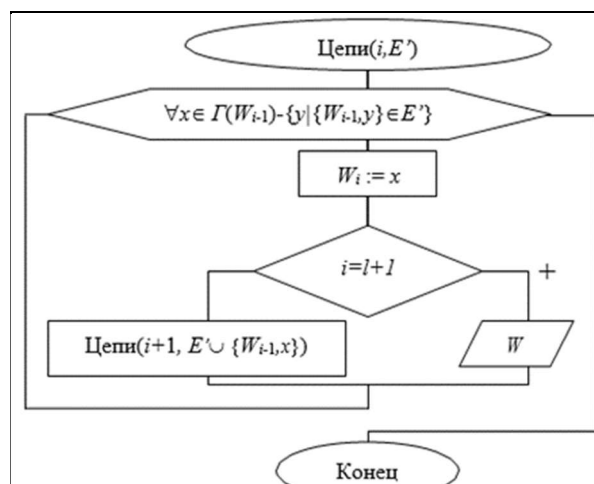


Рис.4.12. Рекурсивный алгоритм получения всех цепей W длины l в графе G , начинающихся вершиной v

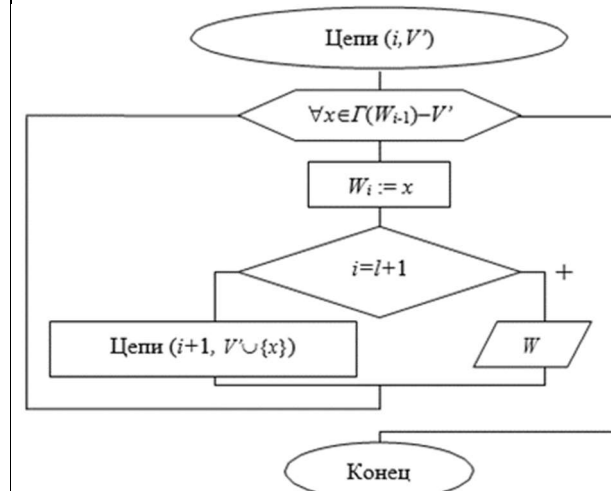
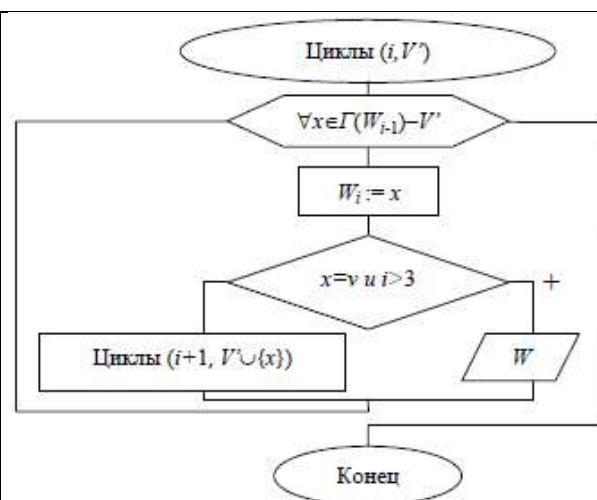
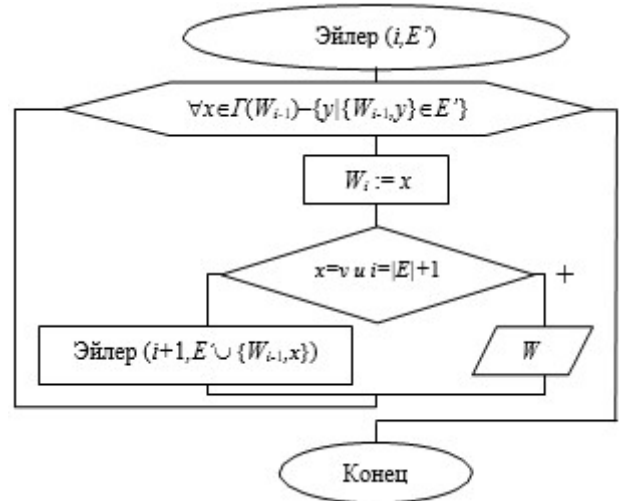


Рис.4.13. Рекурсивный алгоритм получения всех простых цепей W длины l в графе G , начинающихся вершиной v

- **Эйлеровы и гамильтоновы циклы.**

- Если граф имеет цикл, содержащий все ребра графа по одному разу, то такой цикл называется **эйлеровым циклом**, а граф — **эйлеровым графом**. Эйлеров цикл содержит не только все ребра, но и все вершины графа (возможно, по несколько раз). Принадлежность графа классу эйлеровых легко устанавливается теоремой Эйлера: связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны.

Эйлеров цикл представляет собой замкнутую цепь, содержащую все ребра графа, поэтому можно получить все эйлеровы циклы используя алгоритм получения всех цепей заданной длины, изменив в нем условие получения решения. Алгоритм получения всех эйлеровых циклов представлен блок-схемой на рисунке справа.



Алгоритм получения всех эйлеровых циклов W в графе G .

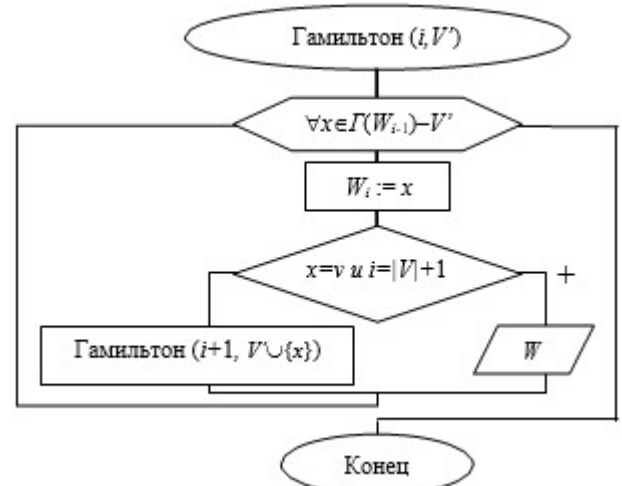
Вход: i — заполняемое место в цикле W ; E' — множество ребер, включенных в цикл.

Выход: последовательность всех эйлеровых циклов W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемый цикл; v — исходная вершина.

- Если граф имеет простой цикл, содержащий все вершины графа (по одному разу), то такой цикл называется **гамильтоновым циклом**, а граф — **гамильтоновым графом**. Вопрос о принадлежности графов к классу гамильтоновых решается, как правило, очень трудно. Известны лишь *достаточные* условия гамильтоновости графов, например, теорема Оре: *если для любой пары x и y несмежных вершин графа G порядка $n \geq 3$ выполняется условие $d(x) + d(y) \geq n$, то G — гамильтонов граф*; или теорема Дирака: *если для любой вершины x графа G порядка $n \geq 3$ выполняется условие $d(x) \geq n/2$, то G — гамильтонов граф*.

Убедиться в том, что граф гамильтонов можно, например, отыскав в нем один из гамильтоновых циклов, используя метод поиска с возвращением. На рисунке представлена блок-схема рекурсивного алгоритма нахождения всех гамильтоновых циклов графа.



Алгоритм получения всех гамильтоновых циклов W в графе G .

Вход: i — заполняемое место в цикле W ; V' — множество вершин, включенных в цикл.

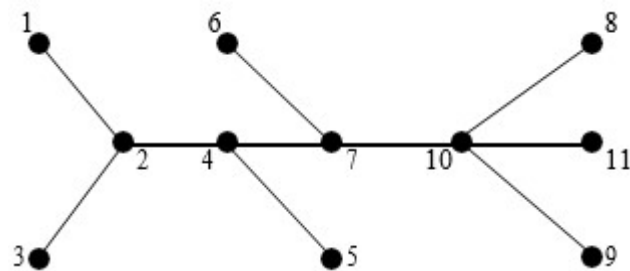
Выход: последовательность всех гамильтоновых циклов W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемый цикл.

- Деревья и их свойства.

Связный ациклический граф называется *деревом*. Несвязный ациклический граф называется *лесом*. Лес представляет собой множество деревьев. Дерево и лес обладают следующими свойствами:

- 1) количество ребер любого дерева на единицу меньше количества вершин;
- 2) количество ребер любого леса меньше количества вершин на количество деревьев в нем;
- 3) любые две вершины, принадлежащие различным деревьям — не смежны;
- 4) любую пару вершин дерева соединяет единственная простая цепь;
- 5) если в дереве любую пару несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл;
- 6) если две вершины леса, принадлежащих различным деревьям, соединить ребром, то количество деревьев уменьшится на единицу.



- Количество деревьев с n вершинами

Количество различных деревьев, которые можно построить на n вершинах равно количеству таких последовательностей (размещений с повторениями из n элементов по $n - 2$ местам), т.е. n^{n-2} .

- **Связность, компоненты связности, алгоритм Краскала.**

Граф называется связным, если в нем для каждой пары вершин найдется соединяющая их цепь, иначе – несвязным. Две вершины графа называются связанными, если существует соединяющая их цепь. В связном графе все пары вершин связаны.

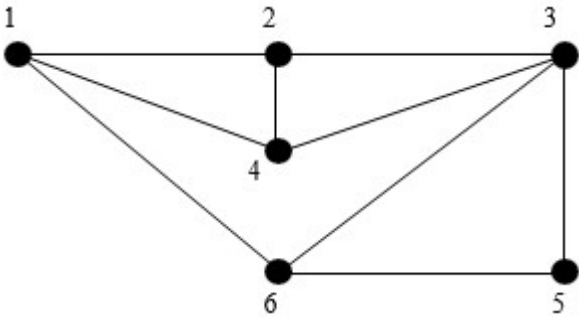
Отношение связности вершин является отношением эквивалентности: оно рефлексивно (каждая вершина соединена с собой цепью нулевой длины), симметрично (всякая цепь, записанная в обратном порядке, также является цепью) и транзитивно (если имеется цепь, соединяющая вершины x и y и цепь, соединяющая вершины y и z , то соединение этих цепей в вершине y даст маршрут из вершины x в вершину z , из которого можно выделить цепь, соединяющую вершины x и z), поэтому оно разбивает множество вершин на классы эквивалентных вершин. Подграф, построенный на множестве вершин одного класса эквивалентности, представляет собой **связную компоненту графа**. Связный граф имеет одну компоненту связности, несвязный — более одной.

Отношение связности вершин графа $G=(V,E)$ можно задать матрицей связности C порядка $n \times n$, где $n=|V|$. Элемент матрицы связности $c_{ij}=1$, если вершины i и j связаны и $c_{ij}=0$, если вершины i и j не связаны. Рассмотрим два способа получения матрицы связности.

Алгоритм Краскала:

1. Начальная установка. Лес не содержит ни одного ребра и ни один из букетов не сформирован.
2. Выбрать любое ребро, включить его в лес и сформировать букет, включив в него концевые вершины выбранного ребра.
3. Пока в графе есть необработанные ребра, выполнять п.4.
4. Выбрать необработанное ребро. После этого выбора возможны четыре различных случая:
 - А. Обе концевые вершины выбранного ребра принадлежат одному и тому же букету. В этом случае ребро не включается в лес.
 - Б. Одна из концевых вершин выбранного ребра принадлежит некоторому букету, а другая концевая вершина не принадлежит ни одному из уже сформированных букетов. В этом случае выбранное ребро включается в лес и его концевая вершина, не принадлежащая ранее ни одному букету, включается в букет, которому принадлежит другая концевая вершина рассматриваемого ребра.
 - В. Ни одна из концевых вершин не принадлежит ни одному из сформированных букетов. В этом случае выбранное ребро включается в лес и формируется новый букет из его концевых вершин.
 - Г. Концевые вершины выбранного ребра принадлежат различным букетам. В этом случае выбранное ребро включается в лес, а оба букета, которым принадлежат его концевые вершины, объединяются в один новый букет и количество букетов при этом уменьшается.

Рассмотрим пример работы алгоритма Краскала:



Шаг	Ребро	Решение	Массив В						Примечание
			1	2	3	4	5	6	
1			1	2	3	4	5	6	Начальная установка
2	{1,2}	Включать в лес	1	1	3	4	5	6	На шаге 1 $B_1 \neq B_2$
3	{1,4}	Включать в лес	1	1	3	1	5	6	На шаге 2 $B_1 \neq B_4$
4	{2,3}	Включать в лес	1	1	1	1	5	6	На шаге 3 $B_2 \neq B_3$
5	{2,4}	Не включать	1	1	1	1	5	6	На шаге 4 $B_2 = B_4$
6	{3,4}	Не включать	1	1	1	1	5	6	На шаге 5 $B_3 = B_4$
7	{3,5}	Включать в лес	1	1	1	1	1	6	На шаге 6 $B_3 \neq B_6$
8	{3,6}	Включать в лес	1	1	1	1	1	1	Дерево построено
9	{5,6}	Не включать	1	1	1	1	1	1	На шаге 8 $B_5 = B_6$
10	{6,1}	Не включать	1	1	1	1	1	1	На шаге 9 $B_6 = B_1$

● Покрывающее дерево минимальной стоимости, алгоритмы построения

Пусть задан граф со взвешенными ребрами и его покрывающее дерево (дерево, являющееся подграфом графа G и содержащее все его вершины). Сумма весов ребер, принадлежащих покрывающему дереву, называется *стоимостью дерева*. Покрывающее дерево графа, имеющее наименьшую стоимость по сравнению со всеми другими покрывающими деревьями этого же графа, называется *покрывающим деревом минимальной стоимости*.

Алгоритм Краскала строит покрывающее дерево минимальной стоимости, если ребра графа просматривать в порядке неубывания их весов. Поэтому для нахождения покрывающего дерева минимальной стоимости можно предварительно использовать алгоритм сортировки ребер, а затем применить алгоритм Краскала. Тогда на каждом шаге будет выбираться ребро с минимальным весом, которое можно включить в покрывающее дерево. Такой выбор можно осуществить и в процессе выполнения алгоритма Краскала без предварительной сортировки ребер. В этом случае при выборе ребра используем алгоритм нахождения ребра минимального веса среди ребер, не включенных в дерево, при этом все ребра, образующие цикл с ребрами, включенными в дерево, исключаются из дальнейшего рассмотрения.

Алгоритм Краскала.

1. Начальная установка. Лес не содержит ни одного ребра и ни один из букетов не сформирован.
2. Выбрать любое ребро, включить его в лес и сформировать букет, включив в него концевые вершины выбранного ребра.
3. Пока в графе есть необработанные ребра, выполнять п.4.
4. Выбрать необработанное ребро. После этого выбора возможны четыре различных случая:
 - А. Обе концевые вершины выбранного ребра принадлежат одному и тому же букету. В этом случае ребро не включается в лес.
 - Б. Одна из концевых вершин выбранного ребра принадлежит некоторому букету, а другая концевая вершина не принадлежит ни одному из уже сформированных букетов. В этом случае выбранное ребро включается в лес и его концевая вершина, не принадлежащая ранее ни одному букету, включается в букет, которому принадлежит другая концевая вершина рассматриваемого ребра.
 - В. Ни одна из концевых вершин не принадлежит ни одному из сформированных букетов. В этом случае выбранное ребро включается в лес и формируется новый букет из его концевых вершин.
 - Г. Концевые вершины выбранного ребра принадлежат различным букетам. В этом случае выбранное ребро включается в лес, а оба букета, которым принадлежат его концевые вершины, объединяются в один новый букет и количество букетов при этом уменьшается.

Рассмотрим еще один алгоритм построения покрывающего дерева минимальной стоимости, не требующий предварительной сортировки ребер. Это **алгоритм Прима** (алгоритм ближайшего соседа). Он отличается от алгоритма Краскала тем, что в нем формируется только один букет. На очередном шаге выбирается ребро минимального веса среди ребер, у которых одна концевая вершина принадлежит букету, а другая — не принадлежит. Такое ребро включается в дерево, а концевая вершина, не принадлежащая букету, включается в него.

Алгоритм Прима.

Вход: $G = (V, E)$ — взвешенный граф, где V — множество вершин, E — множество ребер.

Выход: T — покрывающее дерево минимальной стоимости.

1. Начальная установка.
Каждой вершине v_i приписать $t(v_i) = 0$;
 $B := \{v\}$, v — произвольная вершина графа включается в букет B ; каждой вершине v_i , не принадлежащей букету, приписать $d(v_i) = \infty$, $y := v$, y — последняя вершина, включенная в букет;
 $T := \emptyset$, дерево пусто.
2. Пересчет чисел $d(v_i)$ и меток $t(v_i)$.
 $\forall v_i \in \Gamma(y) - B$ пересчитать $d(v_i)$:
 $d(v_i) := \min(d(v_i), \text{вес}\{y, v_i\})$.
Если число $d(v_i)$ изменилось (уменьшилось), то $t(v_i) := y$.
3. Из всех вершин $v_i \notin B$ выбрать вершину v_j с наименьшим числом $d(v_j)$;
 $B := B \cup \{v_j\}$, включить выбранную вершину в букет;
 $y := v_j$, y — последняя вершина, включенная в букет;
 $T := T \cup \{t(v_i), v_j\}$, добавить ребро в дерево.
4. Если все вершины графа включены в букет ($B = V$), то конец алгоритма, иначе перейти к п.2

• Поиск в орграфе в глубину и в ширину

Одним из методов поиска в орграфе является *поиск в глубину*. Порядок — “посещения” вершин при поиске в глубину определяется следующими правилами:

1. Посещаем начальную вершину v_i и считаем ее текущей.
2. Если v_i — текущая вершина, v_j — вершина, смежная вершине v_i и еще не посещалась, то посещаем ее и считаем текущей.
3. Если исходная вершина v_i — текущая вершина и все смежные с ней уже посещались, то поиск заканчивается.
4. Если v_i — текущая вершина и все смежные с ней уже посещались, то текущей считаем вершину, из которой пришли в вершину v_i .

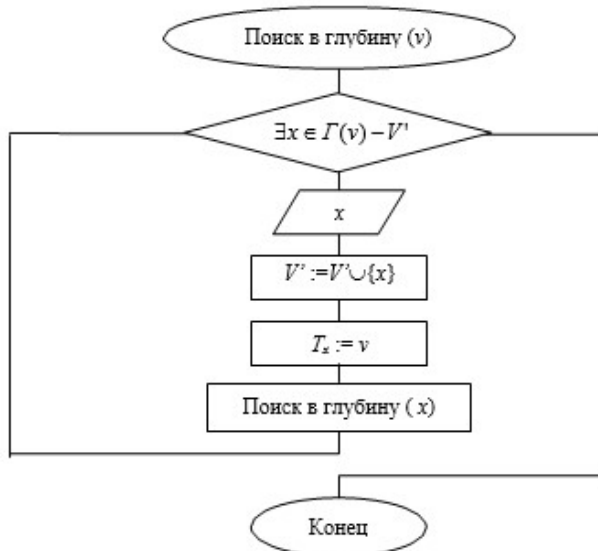
Для программной реализации метода поиска в глубину можно использовать рекурсивный или итеративный алгоритм. В алгоритмах используется множество V' вершин, достижимых из начальной. В исходном состоянии оно содержит только начальную вершину. В итеративном алгоритме для запоминания вершин, предшествующих текущей, используется стек. Дерево поиска можно сохранить в массиве T , количество элементов которого равно количеству вершин орграфа. Элемент T_i представляет собой вершину, предшествующую вершине i в дереве. Если вершина i — корень дерева, то $T_i = 0$. Если вершина i недостижима из начальной, $T_i = -1$. В исходном состоянии элемент массива T , соответствующий начальной вершине, равен 0, а все остальные равны -1 .

Рекурсивный алгоритм поиска в глубину.

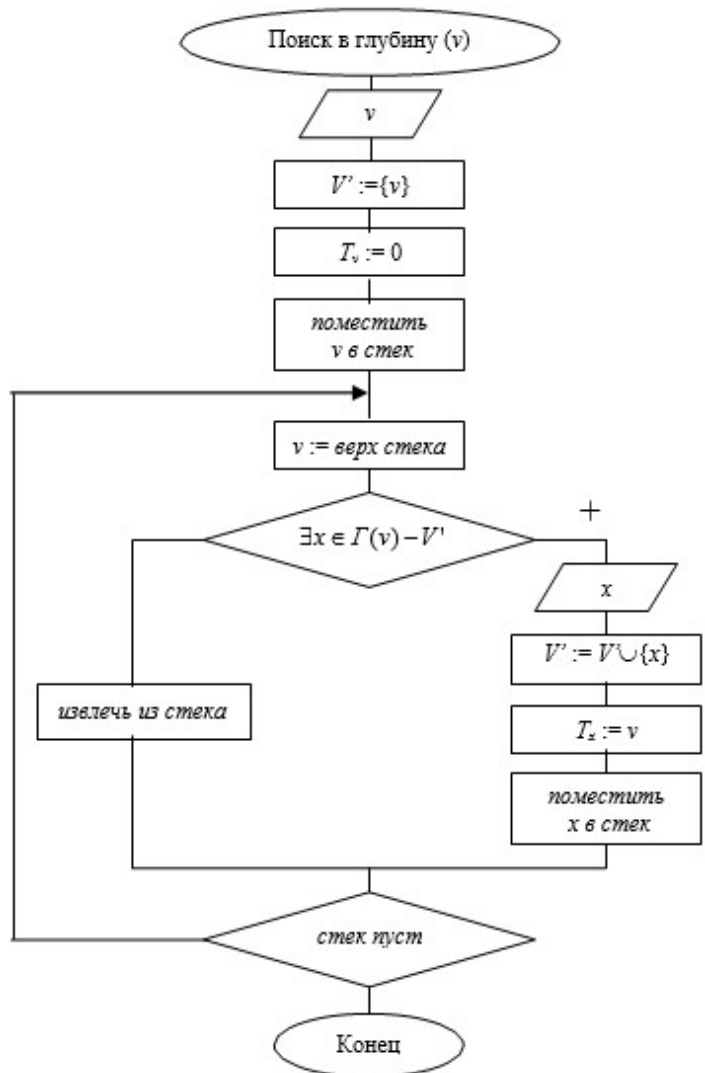
Вход: v — текущая вершина.

Выход: последовательность вершин орграфа в порядке их посещения в процессе поиска в глубину (начальная вершина выводится перед первым обращением);
 T — дерево поиска в глубину.

Глобальные параметры: V' — множество вершин, достижимых из начальной.



Итеративный алгоритм поиска в глубину



Порядок посещения вершин при поиске в ширину определяется следующими правилами:

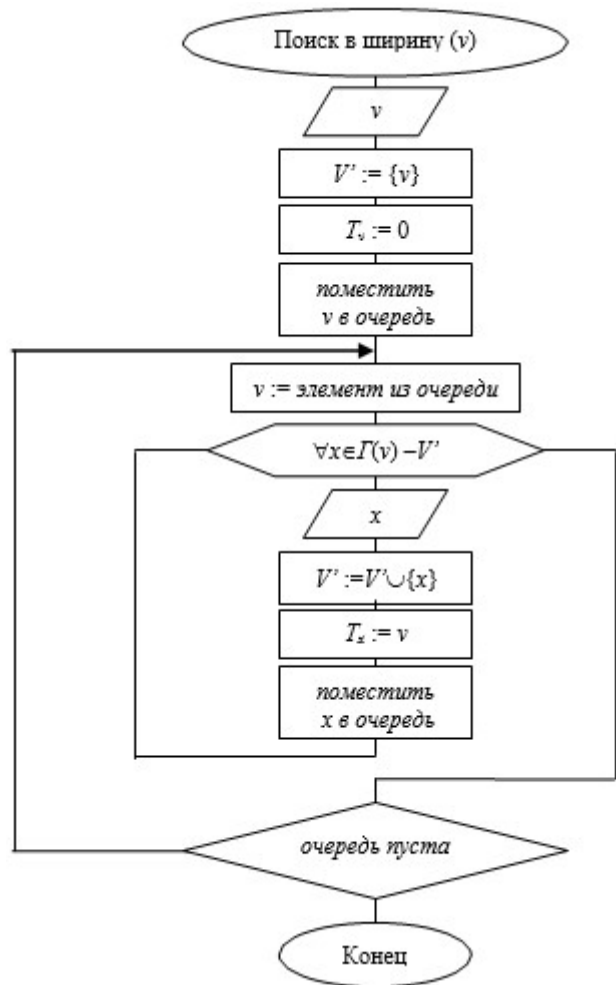
1. Посещаем начальную вершину v_i и считаем ее текущей.
2. Если v_i — текущая вершина, $\Gamma(v_i)$ — множество вершин, смежных с v_i , то вершины из множества $\Gamma(v_i)$, которые еще не посещались, последовательно посещаем, а вершину, которую посетили вслед за v_i , считаем текущей.
3. Если v_i — последняя вершина, которая посещалась при поиске, является текущей и все вершины, смежные с ней, уже посещались, то поиск заканчивается.

При программной реализации метода поиска в ширину для запоминания порядка посещения вершин используется очередь, а множество V' — для хранения вершин, достижимых из начальной. Дерево поиска в ширину можно сохранить в массиве T , в котором элемент T_i представляет собой вершину, предшествующую вершине i в дереве.

Алгоритм поиска в ширину.

Вход: v — начальная вершина.

Выход: последовательность вершин орграфа в порядке их посещения в процессе поиска в глубину; T — дерево поиска в ширину.



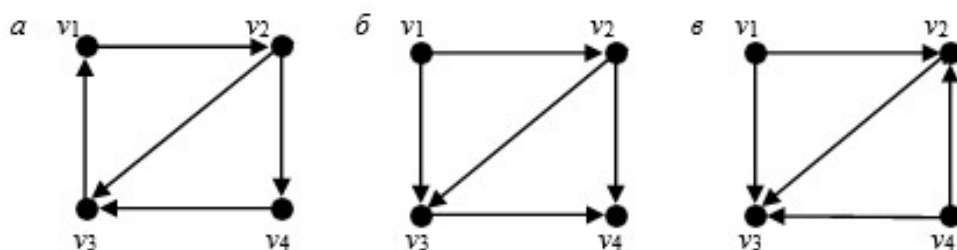
- Связность в орграфе: сильная, односторонняя, слабая.

Две вершины v_i и v_j орграфа $G = (V, E)$ называются **сильно связанными**, если v_i достижима из v_j и v_j достижима из v_i , т.е. вершины v_i и v_j **взаимодостижимы**.

Две вершины v_i и v_j орграфа $G = (V, E)$ называются **односторонне связанными**, если v_i достижима из v_j или v_j достижима из v_i .

Две вершины v_i и v_j орграфа $G = (V, E)$ называются **слабо связанными**, если они связаны в графе G' , полученном из орграфа G путем отмены ориентации дуг.

Если все вершины в орграфе сильно (односторонне, слабо) связаны, то орграф называется **сильно (односторонне, слабо) связанным**. На рисунке ниже показаны диаграммы сильно, односторонне и слабо связанных орграфов.



- **Конденсация орграфа**

Конденсацией орграфа G называют такой орграф G^* , вершинами которого служат компоненты сильной связности G , а дуга в G^* присутствует только если существует хотя бы одно ребро между вершинами, входящими в соответствующие компоненты связности.

Или (как по методичке, на выбор):

Орграф $G^* = (V^*, E^*)$, в котором

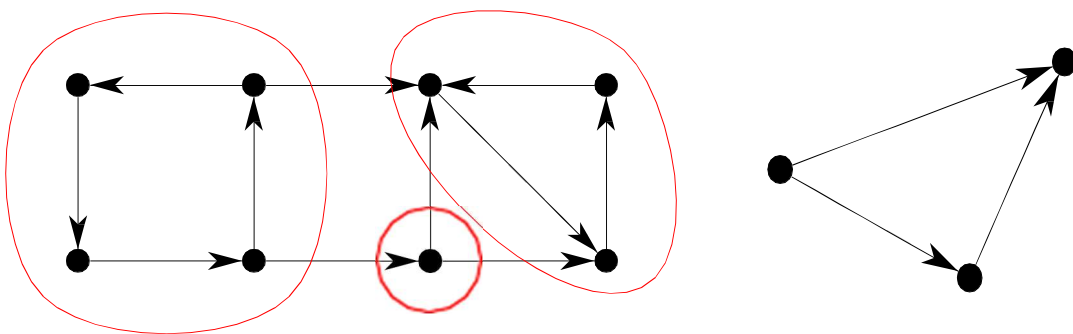
$$V^* = \{V_1, V_2, \dots, V_n\}, \text{ где } n \text{ — количество компонент сильной связности, } V_i \text{ — множество вершин } i\text{-й компоненты сильной связности,}$$
$$E^* = \{(V_i, V_j) \mid v_i \in V_i \text{ и } v_j \in V_j \text{ и } (v_i, v_j) \in E\},$$

называется конденсацией орграфа $G = (V, E)$. Конденсация G^* орграфа G получается - “стягиванием” в одну вершину каждой компоненты сильной связности.

Приведём пример конденсации:

Орграф, изображенный на рис. слева, имеет три компоненты сильной связности, обведенные красными линиями. В орграфе есть дуги, не входящие ни в одну из компонент сильной связности в этом главное отличие от компонент связности обычного графа.

“Стянув” все дуги в каждой компоненте сильной связности орграфа G , мы получим орграф без циклов, называемый конденсацией G (для орграфа на рис. справа приведена его конденсация).



- База и антибаза

Базой B орграфа G называется такое множество вершин, которое удовлетворяет следующим двум условиям:

- 1) каждая вершина орграфа G достижима хотя бы из одной вершины множества B ;
- 2) в B нет вершины, которая достижима из другой вершины множества B .

Антибазой \bar{B} орграфа G называется такое множество вершин, которое удовлетворяет следующим двум условиям:

- 1) из любой вершины орграфа G достижима хотя бы одна вершина множества \bar{B} ;
- 2) в \bar{B} нет вершины, из которой достижима другая вершина множества \bar{B} .

Примеры:

Базой орграфа G^* (рис.4.38) является множество $B^* = \{V_1, V_3\}$, а базами орграфа G (рис.4.35) являются множества $B_1 = \{1, 5\}$, $B_2 = \{2, 5\}$ и $B_3 = \{6, 5\}$.

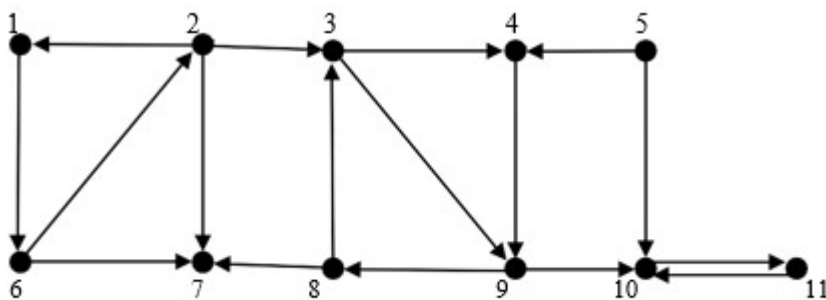


Рис.4.35. Диаграмма орграфа

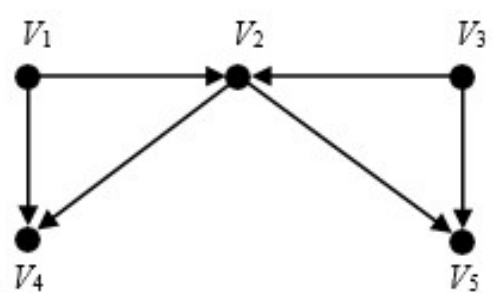


Рис.4.38. Конденсация орграфа

Антибазой орграфа G^* (рис.4.38) является множество $B^* = \{V_4, V_5\}$, а антибазами орграфа G (рис.4.35) являются множества $B_1 = \{7, 10\}$ и $B_2 = \{7, 11\}$.

• Нахождение сильносвязных компонент

Суть задачи заключается, в том что следует найти отношение достижимости R и контрдостижимости Q , затем по ним определить отношение взаимодостижимости ($H = R \cap Q$), разбиение которого на классы эквивалентности даст подмножества вершин, образующих компоненты сильной связности.

Вычислить матрицу R достижимости можно по формуле $R = I + M^+$, где

I - бинарная матрица, содержащая единицы на главной диагонали;

M - матрица смежности

M^+ - транзитивное замыкание отношения, представленного матрицей смежности орграфа, которое можно вычислить, используя алгоритм объединения степеней или алгоритм Уоршала.

Отношение Q контрдостижимости вычисляется по формуле $Q = R^{-1}$

На рисунке ниже представлена диаграмма орграфа, а ещё ниже — матрицы отношений достижимости, контрдостижимости и взаимодостижимости, а в самом конце — компоненты сильной связности.

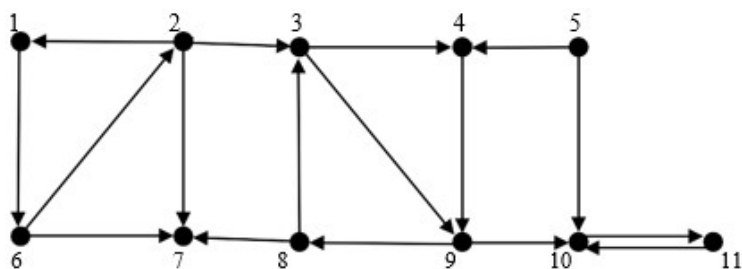


Рис.4.35. Диаграмма орграфа

$R = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$	$Q = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$	$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$
Матрица достижимости	Матрица контрдостижимости	Матрица взаимодостижимости

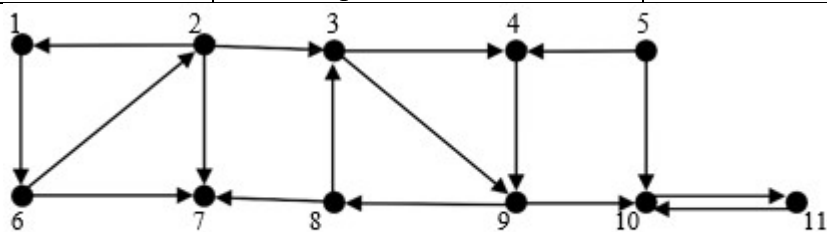


Рис.4.37. Компоненты сильной связности

● Кратчайшие пути во взвешенных орграфах

Дан взвешенный оргграф (взвешены дуги, веса дуг положительные) и две его вершины — S и F . Длиной пути от вершины S до вершины F называется сумма весов всех дуг, принадлежащих этому пути. Требуется найти в оргграфе путь от вершины S до вершины F минимальной длины, т.е. путь, длина которого не больше любого другого пути от вершины S до вершины F . Путь минимальной длины будем также называть *кратчайшим*, а длину этого пути — *кратчайшим* или *минимальным расстоянием* от начальной вершины пути до конечной.

Задачу отыскания кратчайшего пути во взвешенном оргграфе от вершины S до вершины F можно решить различными способами.

1. Кратчайший путь от вершины S до вершины F представляет собой простую цепь (любой другой путь, содержащий в себе цикл, будет иметь большую длину), поэтому для отыскания кратчайшего пути достаточно выполнить полный перебор простых цепей от вершины S до вершины F и выбрать цепь минимальной длины.

Для этого введем в алгоритм нахождения простых цепей метод поиска с возвращением дополнения, позволяющие выбрать цепь минимальной длины. Цепь W представлена последовательностью вершин с вершиной S на первом месте. В процессе формирования цепи будем вычислять ее длину L . Если в цепь из $i - 1$ вершин длины L добавить i -ю вершину x , то получим цепь из i вершин длины $L' = L + \text{длина_дуги}(W_{i-1}, x)$. Будем использовать переменную

$Lmin$ для хранения длины пути, принятого за минимальный, а переменную $Wmin$ — для хранения этого пути. Инициализируем $Lmin$ бесконечно большим числом. После нахождения очередной цепи W от вершины S до вершины F сравним ее длину L с $Lmin$ и, если окажется, что L меньше $Lmin$, запоем цепь W в переменной $Wmin$, а ее длину L — в $Lmin$.

2. Для сокращения количества анализируемых простых цепей воспользуемся идеей метода *ветвей и границ*. Этот метод позволяет сократить число анализируемых решений, когда требуется выбрать из множества всех решений наилучшее, обладающее заданным свойством. В нашем случае множество всех решений — это множество всех простых цепей от вершины S до вершины F , а наилучшее — кратчайший путь. В методе ветвей и границ решение представляет собой “ветвь”, получение решения — построение “ветви”. Если на некотором шаге построения “ветви” станет ясно, что продолжение построения “ветви” не даст наилучшего решения, то на этом шаге получения решения “ветвь” ограничивается, выполняется шаг назад и делается попытка построить “ветвь” в другом направлении. Пусть $Lmin$ — длина пути от вершины S до вершины F , принятого за минимальный. Если на некотором шаге добавление дуги (W_{i-1}, x) дает цепь, длина которой $L' \geq Lmin$, то дальнейшее построение этой цепи прекращается и выполняется шаг назад.

3. Один из эффективных алгоритмов нахождения кратчайшего пути во взвешенном оргграфе между двумя вершинами, при условии, что все дуги оргграфа имеют положительный вес, предложил в 1959 г. Едсгер Дейкстра.

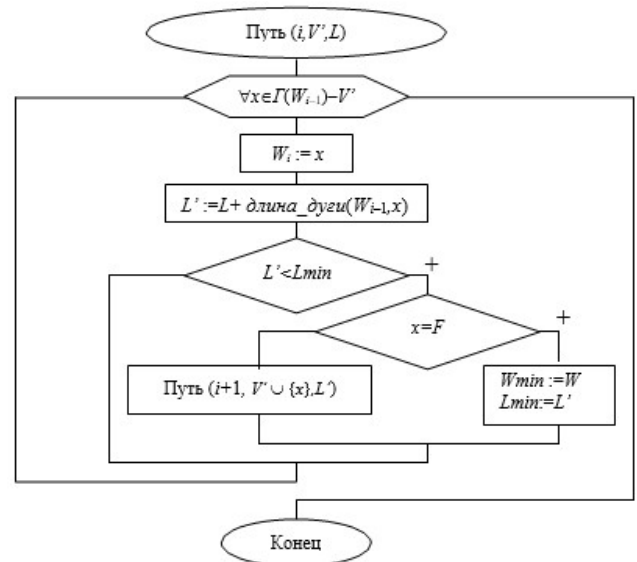
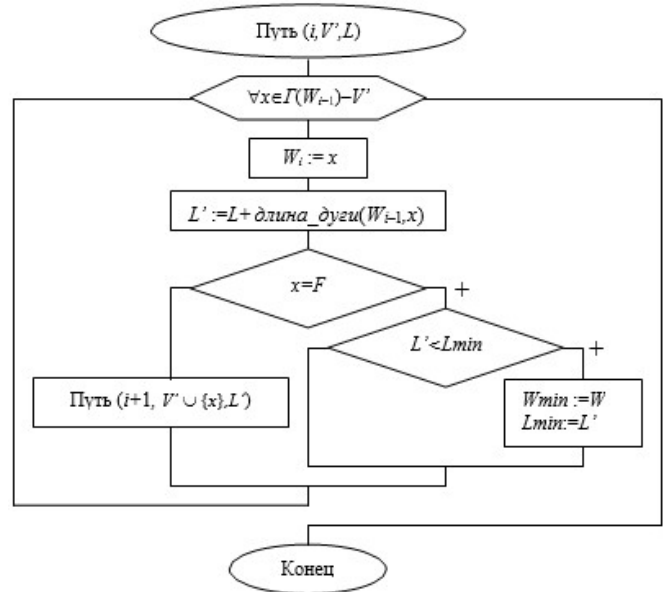
Алгоритм Дейкстры.

Вход: взвешенный оргграф;

S — начальная вершина пути;

F — конечная вершина пути;

Выход: кратчайший путь из S в F и его длина $d(F)$.



1. Начальная установка.

$V' := \{S\}, V'' := V - \{S\}, d(S) := 0, \forall x_i \in V'', d(x_i) := \infty,$

$y := S$ (y — последняя вершина, до которой найден кратчайший путь).

2. Пересчет чисел $d(x_i)$.

$\forall x_i \in \Gamma(y) \cap V''$ пересчитать $d(x_i)$:

$$d(x_i) := \min(d(x_i), d(y) + \text{длина_дуги}(y, x_i)).$$

Если величина $d(x_i)$ изменилась (уменьшилась), то это означает, что кратчайший путь в вершину x_i , вероятно, содержит дугу (y, x_i) , поэтому ее нужно запомнить как претендента на включение в кратчайший путь.

3. Поиск вершины x_i , до которой найден кратчайший путь.

Из всех вершин $x_i \in V''$ выбрать вершину x_j с наименьшим значением $d(x_j)$. Если $d(x_j) = \infty$, то пути от вершины S ко всем вершинам из множества V'' не существует, иначе $y := x_j, V' := V' \cup \{y\}, V'' := V'' - \{y\}$, отметить дугу претендента, ведущую в вершину y .

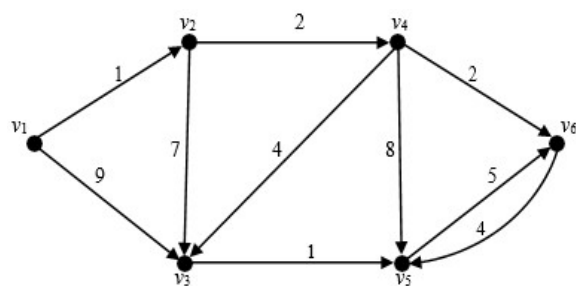
4. Проверка на завершение.

Если $y = F$, то конец алгоритма, кратчайший путь из S в F найден, его длина $d(y)$.

Если $d(x_j) = \infty$, то пути из S в F нет, конец алгоритма. Иначе перейти к п.2.

Работу алгоритма Дейкстры при поиске кратчайшего пути от вершины v_1 до вершины v_5 во взвешенном орграфе, диаграмма которого изображена на рисунке, представим в таблице ниже, отображающей изменения значений массивов V, D и T на каждом шаге выполнения алгоритма.

Номер шага	Пункт алгоритма	Состояние массивов V, D и T	Вершина y
1	1	$V=(1, 0, 0, 0, 0, 0)$ $D=(0, \infty, \infty, \infty, \infty, \infty)$ $T=(0, -1, -1, -1, -1, -1)$	1
2	2	$V=(1, 0, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, -1, -1, -1)$	1
3	3	$V=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, -1, -1, -1)$	2
4	4	$2 \neq 5$, продолжить	2
5	2	$V=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 8, 3, \infty, \infty)$ $T=(0, 1, 2, 2, -1, -1)$	2
6	3	$V=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 8, 3, \infty, \infty)$ $T=(0, 1, 2, 2, -1, -1)$	4
7	4	$4 \neq 5$, продолжить	4
8	2	$V=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 7, 3, 11, 5)$ $T=(0, 1, 4, 2, 4, 4)$	4
9	3	$V=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 7, 3, 11, 5)$ $T=(0, 1, 4, 2, 4, 4)$	6
10	4	$6 \neq 5$, продолжить	6
11	2	$V=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 7, 3, 9, 5)$ $T=(0, 1, 4, 2, 5, 4)$	6
12	3	$V=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 7, 3, 9, 5)$ $T=(0, 1, 4, 2, 5, 4)$	3
13	4	$3 \neq 5$, продолжить	3
14	2	$V=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 7, 3, 8, 5)$ $T=(0, 1, 4, 2, 3, 4)$	3
15	3	$V=(1, 1, 1, 1, 1, 1)$ $D=(0, 1, 7, 3, 8, 5)$ $T=(0, 1, 4, 2, 3, 4)$	5
16	4	$5 = 5$, конец	5



• Центры и медианы взвешенного орграфа

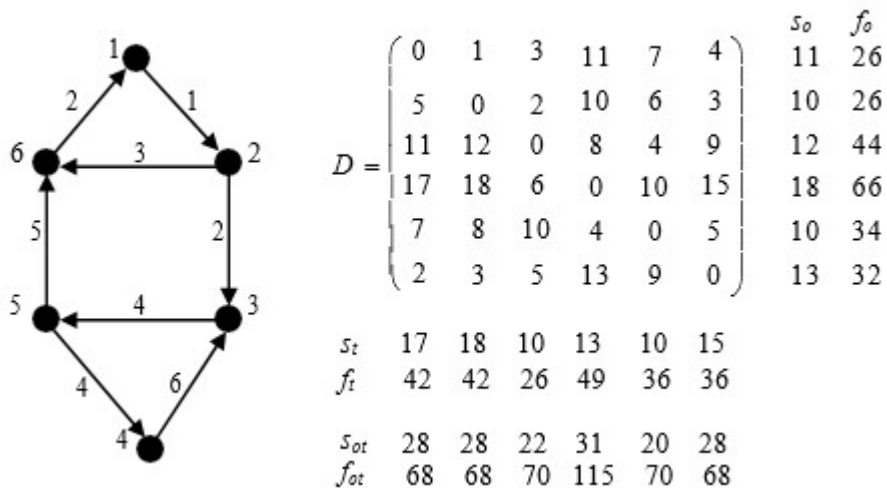
Определим для каждой вершины v_i взвешенного орграфа число $s_o(v_i)$ как максимум среди кратчайших расстояний от вершины v_i до каждой вершины орграфа, число $s_t(v_i)$ как максимум среди кратчайших расстояний до вершины v_i от каждой вершины орграфа и число $s_{ot}(v_i)$ как сумму чисел $s_o(v_i)$ и $s_t(v_i)$: $s_{ot}(v_i) = s_o(v_i) + s_t(v_i)$. Число $s_o(v_i)$ называется числом внешнего разделения вершины v_i , число $s_t(v_i)$ — числом внутреннего разделения вершины v_i и число $s_{ot}(v_i)$ — числом внешне-внутреннего разделения вершины v_i .

Вершина с минимальным числом внешнего разделения называется внешним центром, вершина с минимальным числом внутреннего разделения называется внутренним центром и вершина с минимальным числом внешне-внутреннего разделения называется внешне-внутренним центром.

Определим для каждой вершины v_i взвешенного орграфа число $f_o(v_i)$ как сумму кратчайших расстояний от вершины v_i до каждой вершины орграфа, число $f_t(v_i)$ как сумму среди кратчайших расстояний до вершины v_i от каждой вершины орграфа и число $f_{ot}(v_i)$ как сумму чисел $f_o(v_i)$ и $f_t(v_i)$: $f_{ot}(v_i) = f_o(v_i) + f_t(v_i)$. Число $f_o(v_i)$ называется внешним передаточным числом вершины v_i , число $f_t(v_i)$ — внутренним передаточным числом вершины v_i и число $f_{ot}(v_i)$ — внешне-внутренним передаточным числом вершины v_i .

Вершина с минимальным внешним передаточным числом называется внешней медианой, вершина с минимальным внутренним передаточным числом называется внутренней медианой и вершина с минимальным внешне-внутренним передаточным числом называется внешне-внутренней медианой.

На рисунке ниже представлен взвешенный орграф, матрица D кратчайших расстояний, числа внешнего s_o , внутреннего s_t и внешне-внутреннего s_{ot} разделения, а также внешние f_o , внутренние f_t и внешне-внутренние f_{ot} передаточные числа вершин орграфа. В этом орграфе вершины 2 и 5 являются внешними центрами, вершины 3 и 5 — внутренними центрами, вершина 5 — внешне-внутренним центром, вершины 1 и 2 — внешними медианами, вершина 3 — внутренней медианой и вершины 1, 2 и 6 — внешне-внутренней медианой.



● Независимые множества и клики

Рассмотрим неориентированный граф $G=(V,E)$.

Независимое множество вершин (известное также как внутренне устойчивое множество) есть множество вершин графа G , такое, что любые две вершины в нем не смежны (никакая пара вершин не соединена ребром). Независимое множество называется максимальным, когда нет другого независимого множества, в которое оно бы входило.

Множество вершин M графа $G = (V,E)$ называется кликой, если любые две вершины из этого множества смежны. Подграф графа G , построенный на множестве вершин M , являющимся кликой, представляет собой полный подграф.

Обычно в графе можно найти несколько клик, например, каждая пара смежных вершин является кликой. Клика называется *максимальной*, если она не является собственным подмножеством никакой другой клики. В максимальную клику нельзя добавить ни одной вершины графа так, чтобы полученное множество было кликой. Максимальных клик в графе может быть несколько. Различные максимальные клики могут не пересекаться или находиться в общем положении, могут содержать одинаковое или различное число вершин.

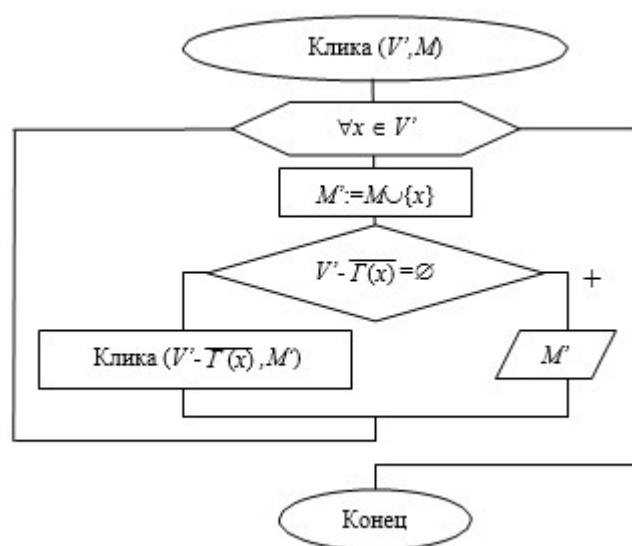
Все максимальные клики можно найти, используя метод поиска с возвращением.

Алгоритм поиска всех максимальных клик.

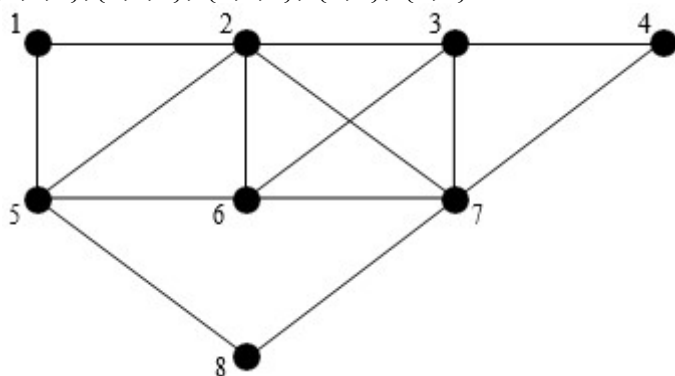
Вход: граф $G = (V,E)$;

M — клика;

V' — множество вершин, смежных с каждой вершиной множества M . Выход: последовательность максимальных клик.



В графе, диаграмма которого представлена на рисунке ниже, максимальных клик шесть: $\{1,2,5\}$, $\{2,3,6,7\}$, $\{3,4,7\}$, $\{5,6,2\}$, $\{5,8\}$, $\{7,8\}$.



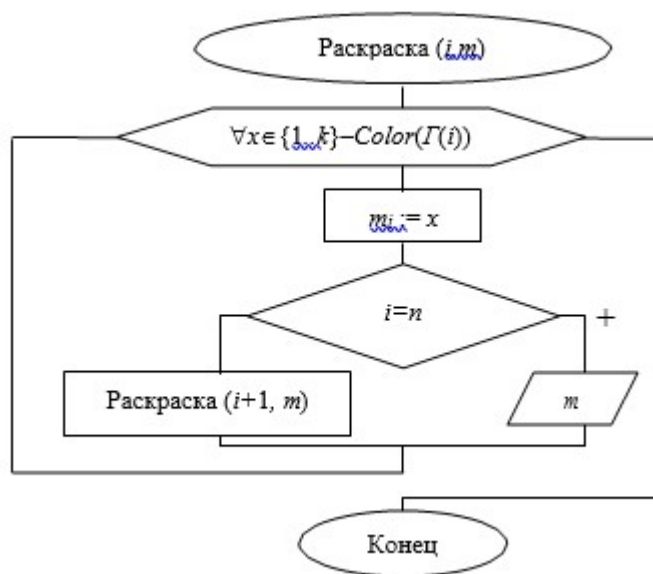
● Раскраска графа

Пусть задан граф $G = (V, E)$ и натуральное число k . Вершинной k -раскраской графа называется функция, ставящая в соответствие каждой вершине из множества V натуральное число (цвет) из множества $\{1..k\}$. Другими словами, раскраска задает цвет каждой вершине графа. Раскраска называется *правильной*, если никакие две смежные вершины не окрашены в один цвет. Граф называется *k -раскрашиваемым*, если для него существует правильная k -раскраска. *Хроматическим числом* графа называется минимальное число цветов, требующееся для правильной раскраски графа. Правильная раскраска графа в хроматическое число цветов называется *минимальной*.

Алгоритм получения всех раскрасок графа $G = (V, E)$ не более чем в k цветов.

Вход: i — окрашиваемая вершина; m — текущая раскраска.

Выход: последовательность всех раскрасок графа в k цветов, если граф G k -раскрашиваемый.



- **Булевы функции.**

Функция f , зависящая от n переменных x_1, x_2, \dots, x_n называется *булевой*, или *переключательной*, если функция f и любой из ее аргументов $x_i, i = (1, \dots, n)$ принимают значения только из множества $\{1, 0\}$. Аргументы булевой функции также называются булевыми.

- Табличные, аналитические и графовые способы задания булевых функций.

Табличные способы:

❖ Произвольная булева функция может быть задана табличным способом. При таком способе булева функция $f(x_1, \dots, x_n)$ задается **таблицей истинности**, в левой части которой представлены все возможные двоичные наборы длины n , а в правой указывается значение функции на этих наборах. Под двоичным набором $y = (y_1, y_2, \dots, y_n)$, $y_i \in \{1, 0\}$ понимается совокупность значений аргументов x_1, x_2, \dots, x_n булевой функции f . Двоичный набор имеет длину n , если он представлен n цифрами из множества $\{0, 1\}$.

x	y	f
0	0	0
0	1	0
1	0	0
1	1	1

❖ Булевы функции, зависящие от большого числа переменных, задавать таблицей истинности не удобно в силу ее громоздкости. Поэтому для задания функции многих переменных удобно использовать **модифицированную таблицу истинности**. В этом случае множество из n переменных функции разбивается на два подмножества: $\{x_1, x_2, \dots, x_j\}$ и $\{x_{j+1}, \dots, x_n\}$. Переменными x_1, x_2, \dots, x_j отмечают строки таблицы истинности, задавая в каждой строке значение соответствующего двоичного набора длины j , а переменными x_{j+1}, \dots, x_n отмечают столбцы таблицы, задавая в каждом столбце значение соответствующего двоичного набора длины $n - j$. Значение функции записывается в клетке на пересечении соответствующей строки и столбца.

Аналитические способы:

❖ СДНФ представляет собой дизъюнкцию конституент единицы (функция, принимающая значение 1 только на единственном наборе).

Для того, чтобы получить аналитическое выражение в СДНФ функции, заданной таблично, нужно составить дизъюнкцию конституент единицы для тех наборов значений аргументов, для которых значение функции равно 1, причем символ любой переменной в некоторой конституенте берется со знаком отрицания, если конкретное значение переменной в рассматриваемом наборе имеет значение 0.

❖ Если в СДНФ заменить операции дизъюнкции операциями сложения по модулю 2, то равенство сохранится. Такая форма называется **совершенной полиномиальной нормальной формой** (СПНФ).

❖ СКНФ представляет собой конъюнкцию конституент нуля. (функция, принимающая значение 0 на единственном наборе)

❖ ДНФ называется дизъюнкция элементарных конъюнкций (конъюнкция конечного числа различных между собой переменных, каждая из которых может быть с отрицанием)

❖ КНФ называется конъюнкция элементарных дизъюнкций. (дизъюнкция конечного числа различных между собой переменных, каждая из которых может быть с отрицанием)

❖ Кроме совершенных форм булевых функций существует **произвольная скобочная форма** (инфиксная запись), например: $f = \overline{x_1}(\overline{x_2}x_3 \vee \overline{x_3}(x_2 \vee \overline{x_4})) \vee x_2(x_3 \vee \overline{x_1}x_4)$

❖ Еще одно представление булевых функций получается с использованием операций конъюнкции \wedge , сложения по модулю 2 \oplus и константы 1 (**алгебра Жегалкина**). Если в произвольной формуле алгебры Жегалкина раскрыть скобки и произвести все возможные упрощения, то получится формула, имеющая вид суммы произведений, т.е. полинома по модулю 2. Такая формула называется **полиномом Жегалкина** для данной функции.

$$f = (\overline{x}y) \vee (x\overline{y}) \vee (xy) - \text{СДНФ}$$

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

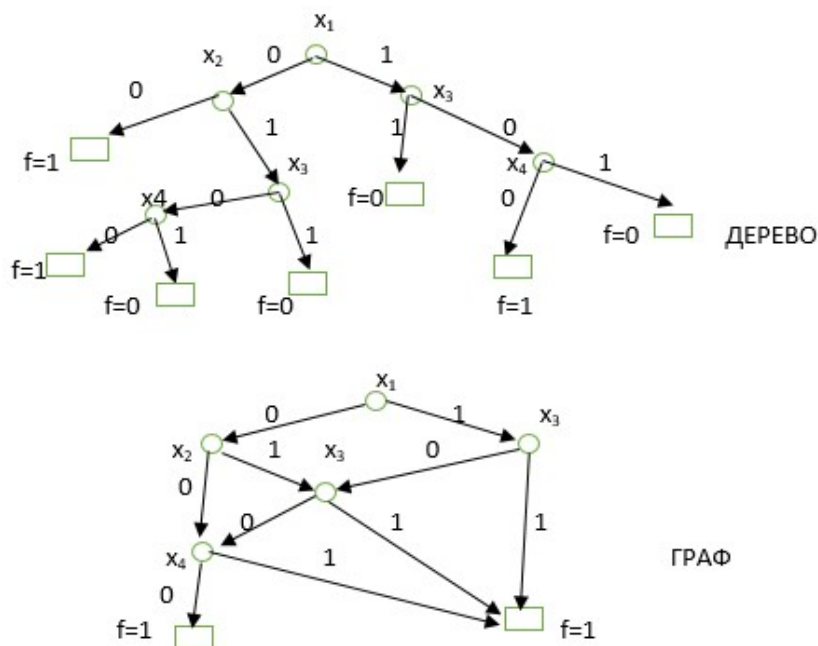
Графовые представления:

❖ *Бинарное дерево булевой функции* содержит начальную вершину (корень), в которую не входит ни одна дуга и выходят две дуги, множество условных вершин, в которые входит одна дуга и выходят две дуги, и множество заключительных вершин (листьев), в которые входит одна дуга и ни одна не выходит. Начальная и условные вершины на диаграмме изображаются кружочками, а заключительные — прямоугольниками. В начальной и в условных вершинах записываются аргументы функции, а в заключительных — значения функции. Дуги, выходящие из начальной и условных вершин, отмечаются нулем и единицей.

❖ *Бинарный граф булевой функции* отличается от бинарного дерева тем, что он содержит только две заключительные вершины и в каждую условную и заключительную вершины входят не менее одной дуги.

❖ *Бинарный граф системы булевых функций* отличается от бинарного графа одной функции тем, что он может содержать более двух заключительных вершин, в которых записаны значения всех функций системы.

❖ Булеву функцию, заданную формулой, например, в произвольной скобочной форме, можно представить *синтаксическим деревом*, в листьях которого записываются аргументы функции, а в остальных вершинах — операции.



• Построение бинарных графов булевых функций

Алгоритм построения бинарного графа булевой функции следующий

- 1) строится начальная вершина, к которой приписывается функция f ;
- 2) выбирается переменная разложения x_i и записывается в вершине, выполняется разложение Шеннона функции f по переменной x_i

$$f(x_1, x_2, \dots, x_i, \dots, x_n) = \bar{x}_i y_0 \vee x_i y_1, \text{ где } y_0 \text{ и } y_1 - \text{коэффициенты разложения:}$$
$$y_0 = f(x_1, x_2, \dots, 0, \dots, x_n);$$
$$y_1 = f(x_1, x_2, \dots, 1, \dots, x_n);$$

- 3) если в графе существует условная вершина, к которой приписана функция y_0 (или y_1), то проводится дуга в эту вершину и отмечается нулем (единицей); если в графе не существует условной вершины, к которой приписана функция y_0 (или y_1) и функция y_0 (или y_1) не является константой, то вводится новая условная вершина, к этой вершине проводится дуга, отмеченная нулем (или единицей) и к ней приписывается функция y_0 (или y_1); если функция y_0 (или y_1) является константой, то проводится дуга, отмеченная нулем (или единицей) в заключительную вершину $f=y_0$ ($f=y_1$), если такая существует, или, в противном случае, вводится такая заключительная вершина;
- 4) пп. 2 и 3 выполняются для всех концевых условных вершин.

- Вычисление значений булевых функций по бинарному графу (дереву).

➤ Если булева функция задана бинарным графом (или деревом), то можно написать программу вычисления ее значения с использованием только команд двух типов:

<условная команда> → <метка> если <аргумент> то <метка> иначе <метка>
 <операторная команда> → <метка> $F := \text{значение}$ конец

Для того, чтобы написать программу, необходимо произвольным образом пронумеровать вершины бинарного графа, начиная с начальной вершины (рис.5.7). Каждой вершине графа соответствует команда программы. Начальной и условным вершинам соответствуют условные команды, а заключительным — операторные. Если в начальной или условной вершине с номером m записан аргумент x_i , дуга, отмеченная нулем, идет в вершину с номером n , а дуга, отмеченная единицей, идет в вершину с номером k , то такой вершине соответствует команда:

m если x_i то k иначе n

Заключительной вершине с номером l , в которой записано $f = 0$, соответствует команда:

l $F := 0$ конец

Заключительной вершине с номером p , в которой записано $f = 1$, соответствует команда:

p $F := 1$ конец

➤ Рассмотрим еще один способ вычисления значения булевой функции по бинарному графу. Представим бинарный граф таблицей. Таблица имеет три столбца, нумерация которых начинается с нуля. Строки таблицы соответствуют вершинам графа. Если в начальной или условной вершине с номером m записан аргумент x_i , дуга, отмеченная нулем, идет в вершину с номером n , а дуга, отмеченная единицей, идет в вершину с номером k , то такой вершине соответствует строка таблицы с номером m , нулевой столбец которой содержит n , первый — k и второй — i .

Такую таблицу можно сохранить в двумерном массиве T и использовать для вычисления значения булевой функции по следующему алгоритму.

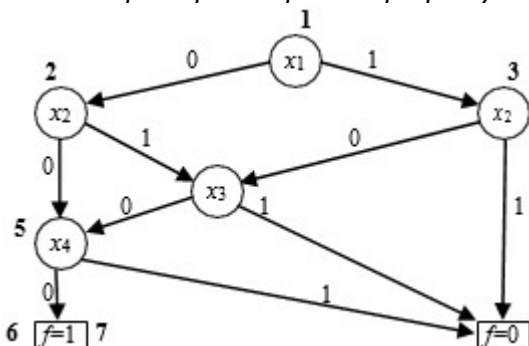
Алгоритм вычисления значения булевой функции по бинарному графу.

Вход: X — набор значений аргументов; T — таблица бинарного графа булевой функции.

Выход: F — значение булевой функции.

1. $v := 1$;
2. Пока $T[v, 0] \neq 0$ выполнять $v := T[v, X[T[v, 2]]]$;
3. $F := T[v, 2]$.

Приведём пример бинарного графа булевой функции с пронумерованными вершинами.



Программа вычисления значения булевой функции по бинарному графу будет такой:

- 1 если x_1 то 3 иначе 2
- 2 если x_2 то 4 иначе 5
- 3 если x_2 то 7 иначе 4
- 4 если x_3 то 7 иначе 5
- 5 если x_4 то 7 иначе 6
- 6 $F := 1$ конец
- 7 $F := 0$ конец

- Минимизация булевых функций в классе ДНФ. (Не полностью!)

Общая задача минимизации булевых функций может быть сформулирована следующим образом: найти аналитическое выражение заданной булевой функции в форме, содержащей минимально возможное число букв (переменных и операций). В общей постановке данная задача пока не решена, однако достаточно хорошо исследована в классе дизъюнктивно-конъюнктивных нормальных форм.

Минимальной дизъюнктивной нормальной формой булевой функции называется ДНФ, содержащая наименьшее число букв (по отношению ко всем другим ДНФ, представляющим заданную булеву функцию).

В общем случае минимизация булевой функции производится в три этапа:

1. Переход от СДНФ к сокращенной ДНФ;
2. Переход от сокращенной ДНФ к минимальной ДНФ (Исключение лишних простых импликант на сокращенных ДНФ. Устранение лишних простых импликант из сокращенной ДНФ булевой функции не является однозначным процессом, т.е. булева функция может иметь несколько тупиковых ДНФ. Тупиковые ДНФ булевой функции f , содержащие минимальное число букв, являются минимальными.);
3. Переход от минимальной ДНФ к скобочной форме.

● Скобочная минимизация булевых функций

Скобочная минимизация булевых функций (факторизация) в ряде случаев позволяет получить скобочную форму, значительно более простую, чем минимальная ДНФ. Процесс факторизации заключается в многократном выполнении операции вынесения за скобки общих членов.

Работу факторного алгоритма рассмотрим на примере.
Пусть минимальная ДНФ функции f имеет вид:

$$f = x_1 \bar{x}_2 x_6 \vee x_1 x_2 x_5 \bar{x}_6 \vee x_1 \bar{x}_2 x_4 \bar{x}_5 \vee x_1 \bar{x}_2 x_3 x_5$$

Обозначим каждую элементарную конъюнкцию буквами A, B, C, D . Тогда функцию f можно представить множеством: $F = (A, B, C, D)$.

На первом этапе работы алгоритма ищутся все попарные пересечения элементов множества F . В нашем случае имеем:

$$\begin{array}{lll} A \cap B = x_1; & A \cap C = x_1 \bar{x}_2; & A \cap D = x_1 \bar{x}_2; \\ B \cap C = x_1; & B \cap D = x_1 x_5; & C \cap D = x_1 \bar{x}_2. \end{array}$$

Выбирается та пара элементов множества F , пересечение которых дает конъюнкцию наибольшей длины. Для рассматриваемого примера выберем либо пару (A, C) , либо (A, D) , либо (C, D) , что с точки зрения результатов работы алгоритма безразлично. Выберем пару (C, D) и запишем функцию f аналитически, вынося общий элемент $x_1 \bar{x}_2$ за скобки по отношению к C и D . Получим:

$$f = x_1 \bar{x}_2 x_6 \vee x_1 x_2 x_5 \bar{x}_6 \vee x_1 \bar{x}_2 (x_4 \bar{x}_5 \vee x_3 x_5)$$

Обозначим $A_1 = x_1 \bar{x}_2 x_6$; $B_1 = x_1 x_2 x_5 \bar{x}_6$; $C_1 = x_1 \bar{x}_2 (x_4 \bar{x}_5 \vee x_3 x_5)$.

Получим $F_1 = (A_1, B_1, C_1)$.

На втором этапе алгоритма ищутся все попарные пересечения элементов множества F_1 . В нашем случае имеем:

$$A_1 \cap B_1 = x_1; \quad A_1 \cap C_1 = x_1 \bar{x}_2; \quad B_1 \cap C_1 = x_1$$

Вновь выбирается пара элементов множества F_1 , пересечение которых дает конъюнкцию наибольшей длины. Выберем пару (A_1, C_1) и запишем функцию f аналитически, вынося общий элемент $x_1 \bar{x}_2$ за скобки по отношению к A_1 и C_1 . Получим:

$$f = x_1 \bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5) \vee x_1 x_2 x_5 \bar{x}_6$$

Обозначив $A_2 = x_1 \bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5)$, $B_2 = x_1 x_2 x_5 \bar{x}_6$, получим множество $F_2 = (A_2, B_2)$. Ищем пересечение элементов множества F_2 . Получаем $A_2 \cap B_2 = x_1$. Окончательно имеем:

$$f = x_1 (\bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5) \vee x_2 x_5 \bar{x}_6)$$