Казанский государственный университет

Факультет вычислительной математики и кибернетики

Кафедра теоретической кибернетики

В.С. Кугураков, Р.К. Самитов, Р.Б. Ахтямов, В.Р. Байрашева

Практикум работы на ЭВМ.

Задание 3. Представление данных и методы разработки алгоритмов.

Казань 2007.

УДК (075.8) 004.3

Кугураков Владимир Сергеевич
Самитов Ринат Касимович
Ахтямов Рауф Баграмович
Байрашева Венера Рустамовна

Практикум работы на ЭВМ. Задание 3. Представление данных и методы разработки алгоритмов. Казань: КГУ. 2007 - с.

Пособие предназначено для студентов, обучающихся по специальности «Прикладная математика и информатика» и направлению «Информационные технологии», а также для преподавателей, ведущих практические занятия по информатике, алгоритмическим языкам и программированию.

Компьютерная верстка и дизайн обложки: Ахтямова Светлана Станиславовна.

1 Типовые задачи

1.1 *Моделирование автом*ата

Автоматом $\Re$ называется некоторое устройство, которое математически описывается тремя множествами X, Y, S и диаграммой D работы:

$X = \{x_1, x_2, \ldots, x_n\}$ – *множество входных символов*;

$Y = \{y_1, y_2, \ldots, y_m\}$ – *множество выходных символов*;

$S = \{s_1, s_2, \ldots, s_k\}$ – *множество состояний*.

Диаграмма строится так. Рисуется K кружков и внутри каждого из них помещается по символу из множества S. Из каждого кружка выводится n стрелок, которые доводятся до кружков (стрелка может выйти из кружка и зайти в него же, две стрелки могут соединять одну и ту же пару кружков). Около каждой стрелки пишется пара символов a/b, где $a \in X$ и $b \in Y$, с единственным условием: разные стрелки, выходящие из одного (любого) кружка, помечаются парами с разными символами из X. Таким образом, $\Re = \{X, Y, S, D\}$.

Работа автомата складывается из *тактов* (номер такта обозначается буквой t, $t = 1, 2, 3, \ldots$). На каждом такте на вход автомата подается один из входных символов: на первом такте символ $a_1 \in X$, на втором – $a_2 \in X$ и т.д. Последовательность входных символов $a_1, a_2, \ldots, a_r$ называется *входным словом* и обозначается через A. В ответ на входное слово автомат вырабатывает (на своем выходе) последовательность $b_1, b_2, \ldots, b_r$ символов из Y, т.е. образуется *выходное слово*, обозначаемое через B.

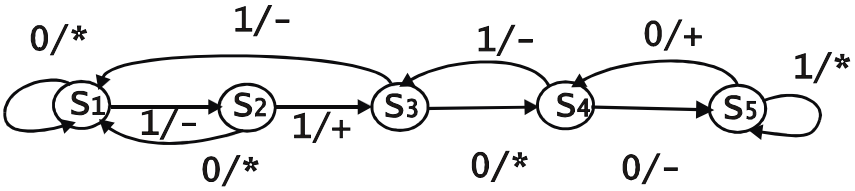
Выходное слово образуется так. Перед первым тактом автомат устанавливается в состоянии $c_0 \in S$. В общем случае – перед тактом t автомат оказался в состоянии $c_{t-1} \in S$. Тогда в диаграмме находится кружок с символом c_{t-1} и выходящая из него стрелка с парой, содержащей символ a_t. Второй символ из пары – это и есть выходной символ b_t, а к следующему $(t + 1)$ – му такту автомат оказывается в том состоянии ($c_t \in S$), к которому приводит эта стрелка.

Задание. Построить программу, моделирующую работу конкретного автомата $\Re$ и решающую для этого автомата задачу М.

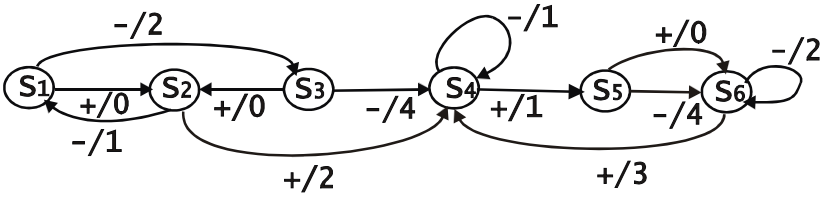
Исходные данные

I. *Автомат* $\Re$:

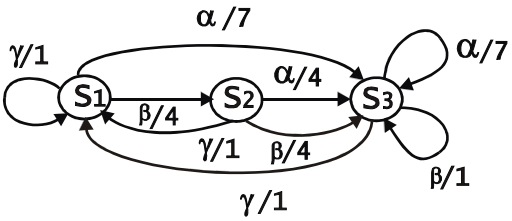
а) $X = \{0, 1\}$, $Y = \{*, +, -\}$, $S = \{s_1, s_2, s_3, s_4, s_5\}$



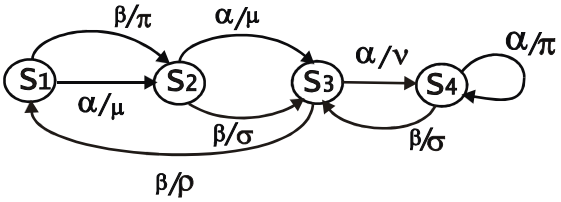
б) $X = \{+, -\}$, $Y = \{0, 1, 2, 3, 4\}$, $S = \{s_1, s_2, s_3, s_4, s_5, s_6\}$



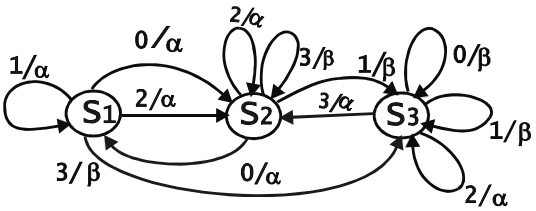
в) $X = \{\alpha, \beta, \gamma\}$, $Y = \{1, 4, 7\}$, $S = \{s_1, s_2, s_3\}$



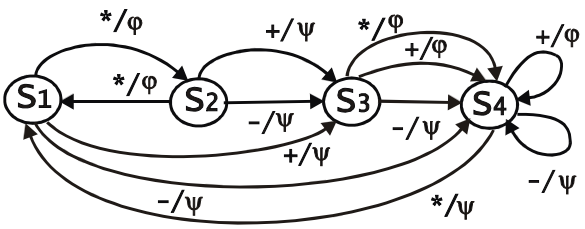
г) $X = \{\alpha, \beta\}$, $Y = \{\mu, \nu, \pi, \rho, \sigma\}$, $S = \{s_1, s_2, s_3, s_4\}$



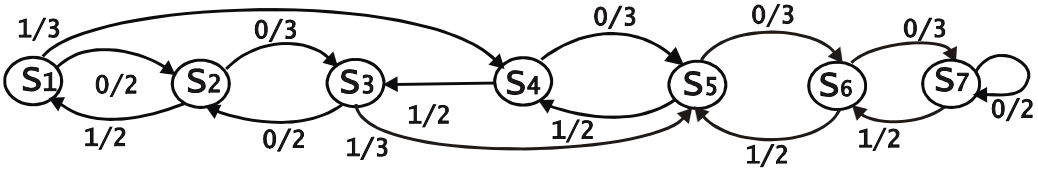
д) $X = \{0, 1, 2, 3\}$, $Y = \{\alpha, \beta\}$, $S = \{s_1, s_2, s_3\}$



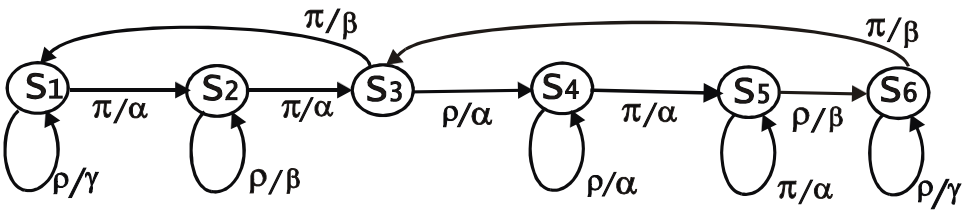
е) $X = \{*, +, -\}$, $Y = \{\varphi, \psi\}$, $S = \{s_1, s_2, s_3, s_4\}$



ж) $X = \{0, 1\}$, Y = {2, 3}, S = $\{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$



з) X = {π, ρ}, Y = {α, β , γ}, S = $\{s_1, s_2, s_3, s_4, s_5, s_6\}$



II. *Задача М*. При заданном начальном состоянии $c_0 \in S$ и заданном входном слове A длины r определить:

а) количество символа y_1 в выходном слове B;

б) количество «подслов» вида $y_1 y_2 y_2$ в выходном слове B;

в) сколько раз автомат, работая, окажется в состоянии S_1 и выдаст на выходе при этом символ y_1;

г) выдаст ли автомат на выходе символ y_1 в такты t = 10, 20, 30 (т.е. будет ли $b_{10} = y_1 \vee b_{20} = y_1 \vee b_{30} = y_1$);

д) максимальную длину подслов B, состоящих только из символа y_1 (т.е. максимальную длину повторения символа y_1);

е) тот такт, на котором автомат впервые окажется в состоянии S_2, и выдаст при этом на выходе символ y_2;

ж) окажется ли автомат в своей работе хотя бы по одному разу в каждом из своих состояний и в выходном слове будут ли представлены все символы из y;

з) сколько раз каждый из символов множества y окажется в выходном слове B.

1.2. *Моделирование машины Тьюринга*.

Всякая машина Тьюринга М состоит из:

- ячейки с бесконечным в обе стороны числом разрядов …, x_{-2} , x_{-1} , x_0 , x_1 , x_2 , … В каждом разряде может быть записан один из n символов множества $A = a_0, a_1, \ldots, a_{n-1}$, причем символ a_0 называется пустым ($n \geq 2$);

- устройство, которое в каждый момент времени обозревает один из разрядов ячейки, а само может находиться в состояниях $\{q_0, q_1, \ldots, q_{m-1}\} = Q$; состояние q_0 называется «стоп-состояние»;

- программы работы – это таблица из n строк и $m-1$ столбцов. Строки соответствуют символам из множества A, столбцы – символам из Q; в каждой клетке помещен один из символов A, один из символов Q и одна из букв Л, П, Н.

Машина предназначена для переработки информации в ячейке. Переработка совершается тактами, номер такта обозначается буквой t ($t = 1,2,3, \ldots$). Перед первым тактом устройство устанавливается в состояние q_1 на обозревание разряда x_0, а начальная информация записывается в ячейку так, что в разрядах $x_0, x_1, \ldots, x_k$ – не пустые символы, в остальных разрядах – пустой символ a_0.

В общем случае – если перед тактом t устройство оказалось в состоянии $q \in Q$ и обозревает разряд x_l, то работа одного такта сводится к следующему:

- если $q = q_0$, то машина останавливается и информация в ячейке считается результатом работы машины;

- если $q \neq q_0$, то берется символ a из разряда x_l и находится клетка в программе, соответствующая символу a и состоянию q, - в этой клетке три символа (a', q', R);

- в разряд x_l помещается символ a' (вместо a); устройство переходит в состояние q' и начинает обозревать разряд x_{l+1}, если R = П, или x_{l-1}, если R = Л, или x_l, если R = Н.

По завершении такта t машина автоматически переходит к выполнению действий такта $t+1$.

Задание. Построить программу, моделирующую работу конкретной машины Тьюринга М и решающую для этой машины задачу Г. В каждой задаче рассматривать работу машины в течение не более заданного числа Т тактов.

Исходные данные

I. Машина Тьюринга М (всюду $a_0 = \Lambda$ и $s_0 = !$):

а) $A = \{\Lambda, 0, 1, *\}$ б) $A = \{\Lambda, +, -, \times\}$

$Q = \{!, q_1, q_2, q_3\}$

	q_1	q_2	q_3
Λ	$\Lambda q_1 П$	$\Lambda q_1 П$	$1 q_2 Н$
0	$0 q_1 П$	$* q_3 Л$	$\Lambda q_2 Н$
1	$\Lambda q_3 П$	$1 q_2 Л$	$1 q_3 П$
*	Λ!Н	$* q_2 Л$	$* q_3 П$

$Q = \{!, q_1, q_2, q_3, q_4\}$

	q_1	q_2	q_3	q_4
Λ	$\Lambda q_1 П$	$\Lambda q_1 П$	$+ q_2 Н$	Λ!П
+	$- q_3 П$	$+ q_2 Л$	$+ q_3 П$	$+ q_4 Н$
–	$- q_1 П$	$- q_1 П$	$+ q_2 Н$	$+ q_4 Л$
×	$\times q_3 Н$	$\times q_2 Л$	$\times q_3 П$	$\Lambda q_4 Н$

в) $A = \{\Lambda, 0, 1, 2, 3, *\}$

$Q = \{1, q_1, q_2\}$

	q_1	q_2
Λ	1!Н	$* q_1 П$
*	$* q_1 Л$	$\Lambda q_1 Л$
0	$1 q_2 Н$	$1 q_1 Л$
1	$2\ q_1 Н$	$0 q_2 П$
2	3!Н	$1 q_2 П$
3	$0 q_1 Л$	$0 q_1 Л$

г) $A = \{\Lambda, \alpha, \beta, \gamma, \delta\}$

Q = $\{!, q_1, q_2, q_3\}$

	q_1	q_2	q_3
Λ	$\alpha q_2 Л$	δ!Н	$\alpha q_2 Л$
α	$\alpha q_1 П$	$\alpha q_2 Л$	$\beta q_3 Л$
β	$\beta q_1 П$	$\beta q_2 Л$	$\gamma q_{1Н}$
γ	$\delta q_1 П$	$\alpha\ q_3 Н$	$\delta q_1 П$
δ	$\gamma q_1 П$	$\beta q_3 Л$	Λ!П

д) $A = \{\Lambda, \varphi, \psi\}$

$Q = \{!, q_1, q_2, q_3, q_4, q_5\}$

	q_1	q_2	q_3	q_4	q_5
Λ	$\varphi q_5 П$	$\varphi q_4 Н$	$\psi q_5 Н$	$\psi q_1 П$	$\Lambda q_1 Л$
φ	$\psi q_2 Н$	$\varphi q_2 П$	$\varphi q_3 П$	$\psi q_4 Л$	φ!Н
ψ	$\psi q_3 Н$	$\psi q_2 П$	$\psi q_3 П$	$\varphi q_4 Л$	$\varphi q_2 Н$

е) $A = \{\Lambda, +, –, \times\}$

$Q = \{!, q_1, q_2, q_3\}$

	q_1	q_2	q_3
Λ	$+ q_1 П$	$\times q_2 Л$	$- q_3 П$
–	$\times q_2 П$	$\Lambda q_1 Н$	$- q_3 Л$
×	$+ q_3 Н$	$+ q_2 Л$	$- q_2 Л$
+	$- q_2 Л$	$\times q_2 П$	Λ!Н

ж) $A = \{\Lambda, \mu, \nu, \rho, \sigma\}$

Q = $\{!, q_1, q_2, q_3, q_4\}$

	q_1	q_2	q_3	q_4
Λ	$\mu q_1 Л$	$\Lambda q_2 П$	$\Lambda q_3 Л$	$\nu q_4 Л$
μ	$\nu q_2 Л$	$\rho q_1 Л$	$\nu q_2 П$	$\nu q_1 П$
ν	$\rho q_3 Л$	$\rho q_4 Л$	$\mu q_4 П$	ρ!П
ρ	$\sigma q_4 Л$	$\rho q_3 П$	$\sigma q_1 Н$	$\sigma q_2 Л$
σ	$\mu q_4 П$	σ!Н	$\mu q_1 Н$	$\Lambda q_3 Н$

з) A = $\{\Lambda, 0, 1, 2, +, –\}$

Q = $\{!, q_1, q_2, q_3\}$

	q_1	q_2	q_3
Λ	$0 q_2 Л$	$1 q_3 П$	$- q_1 П$
0	$1 q_2 П$	$2 q_2 П$	$- q_2 Л$
1	$2 q_3 П$	$+ q_2 П$	$+ q_2 Л$
2	$+ q_3 П$	$2 q_1 П$	$+ q_1 П$
+	$- q_3 Л$	$0 q_1 Л$	$\Lambda q_3 Н$
–	Λ!Н	$0 q_1 Л$	Λ!Н

II. Задача Г. При заданной начальной информации в ячейке машины определить, моделируя ее работу в течение не более Т тактов:

а) сколько раз устройство оказывалось в состоянии q_1;

б) сколько раз устройство находилось в каждом из своих состояний;

в) сколько разрядов ячейки было использовано машиной при своей работе (сколько ячеек обозревалось устройством);

г) количество разрядов ячейки, содержащих не пустой символ, – по окончании работы машины;

д) окажется ли устройство во время работы машины в каждом из своих состояний;

е) какие символы окажутся в разряде x_0 после 10-го, 20-го, 30-го такта (если машина ранее не остановится);

ж) сколько раз выбиралась клетка таблицы, соответствующая символу a_1 и состоянию q_2;

з) сколько поворотов совершило устройство (поворот – это изменение направления движения устройства вдоль ячейки).

Примечание. В любой задаче Г ответ должен сопровождаться тем, закончила ли работу машина в состоянии «стоп» или ее работа оборвана на такте Т.

1.3 *Моделирование программ в операторных машинах*.

Вычислительная машина имеет одну ячейку памяти x, в которой может храниться любой элемент информационного множества $A=\{a_1, a_2, a_3, \dots, a_n\}$. Над содержимым ячейки можно производить преобразования из множества $F = \{f_1, \dots, f_n\}$ и проверять логические условия из множества $P = \{p_1, \dots, p_m\}$, так что F и P можно назвать системой команд машины.

Задается множество программ R в такой машине с описанием их выполнения.

Задание. Составить программу для ЭВМ, которая способна ввести запись любой программы $П \in R$ в конкретной операторной машине, ввести исходную информацию $a \in A$ и выполнить все операции согласно программе A. Составленная программа должна печатать либо результат вычисления $b = П(a)$, если программа заканчивает свою работу, либо сообщение о «Зацикливании» программы П.

Исходные данные

I. Определение машины:

а) $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$; $F = \{f_1, f_2, f_3\}$; $P = \{p_1, p_2\}$.

Операторы и условия задаются таблично

a	0	1	2	3	4	5	6	7	8	9
$f_1(a)$	1	0	3	2	6	9	7	8	5	4
$f_2(a)$	5	2	6	1	7	3	9	4	8	0
$f_3(a)$	2	3	4	7	8	9	1	5	6	9
$p_1(a)$	*и*	*л*	*и*	*и*	*л*	*л*	*л*	*и*	*и*	*Л*
$p_2(a)$	*л*	*и*	*и*	*л*	*л*	*л*	*л*	*л*	*и*	*и*

б) $A = \{0, 1, \wedge, \vee, \equiv, \neg, (,)\}$; $F = \{f_1, f_2\}$; $P = \{p_1, p_2, p_3\}$.

Операторы и логические условия задаются таблично.

a	0	1	$\wedge$	$\vee$	$\equiv$	$\neg$	(	)
$f_1(a)$	1	$\vee$	$\equiv$	$\neg$	(	0	)	$\wedge$
$f_2(a)$	(	)	0	1	$\equiv$	$\wedge$	$\vee$	$\neg$
$p_1(a)$	*и*	*и*	*л*	*л*	*л*	*л*	*л*	*л*
$p_2(a)$	*л*	*л*	*л*	*л*	*л*	*л*	*и*	*и*
$p_3(a)$	*л*	*л*	*и*	*и*	*и*	*и*	*л*	*л*

в) $A = \{+, –, *, =, /, [,]\}$; $F = \{f_1, f_2, f_3\}$; $P = \{p_1, p_2\}$.

Операторы и логические условия задаются таблично

a	+	–	*	=	/	[	]
$f_1(a)$	–	*	/	+	[	]	=
$f_2(a)$	*	*	[	[	[	–	–
$f_3(a)$	/	/	/	/	–	–	–
$p_1(a)$	*и*	*и*	*и*	*и*	*и*	*л*	*л*
$p_2(a)$	*и*	*и*	*л*	*л*	*л*	*и*	*и*

г) $A = \{-99, -98, \ldots, -1, 0, +1, \ldots, 98, 99\}$; $F = \{f_1, f_2, f_3\}$; $P = \{p_1, p_2, p_3\}$,

где $f_1(a) = \begin{cases} a+1, & \text{если } a \le 98, \\ -99, & \text{если } a = 99; \end{cases}$ $p_1(a) = \begin{cases} и, & \text{если } a \text{ четно}, \\ л, & \text{если } a \text{ нечетно}; \end{cases}$

$$f_2(a)=\begin{cases}a-1, & \text{если } a\ge -98,\\ 99, & \text{если } a=-99;\end{cases} \qquad p_2(a)=\begin{cases}и, & \text{если } a>0,\\ л, & \text{если } a\le 0;\end{cases}$$

$$f_3(a)=\begin{cases}0, & \text{если } a=0,\\ 1, & \text{если } a\ne 0;\end{cases} \qquad p_3(a)=\begin{cases}и, & \text{если } 10\le a\le 40,\\ л, & \text{в остальных случаях.}\end{cases}$$

д) $A = \{0, 1, 2, 3, \dots, 999, 1000\}$; $F = \{f_1, f_2, f_3, f_4\}$; $P = \{p_1, p_2\}$.

$$f_1(a)=\begin{cases}a, & \text{если } a>500,\\ 2a, & \text{если } a\le 500;\end{cases} \qquad p_1(a)=\begin{cases}и, & \text{если } a=1000,\\ л, & \text{если } a\ne 1000;\end{cases}$$

$$f_2(a)=\begin{cases}a+2, & \text{если } a\le 998,\\ 0, & \text{если } a=999 \text{ или } 1000;\end{cases} \qquad p_2(a)=\begin{cases}и, & \text{если } a \text{ четно и}\\ & \text{больше } 250,\\ л, & \text{в остальных случаях;}\end{cases}$$

$f_3(a) \equiv 1$, $f_4(a) \equiv 500$.

е) $A = [0, 1]$, т.е. все реальные числа из сегмента $[0, 1]$; $F = \{f_1, f_2, f_3, f_4\}$; $P = \{p_1, p_2\}$.

$f_1(a) = |\sin(1 + a^2)|$;

$$f_2(a)=\frac{1+a^2}{1+a^2+a^4}; \qquad p_1(a)=\begin{cases}и, & \text{если } a\le 0{,}5,\\ л, & \text{если } a>0{,}5;\end{cases}$$

$f_3(a)=\sqrt{a}$;

$$f_4(a)=e^{-a}; \qquad p_2(a)=\begin{cases}и, & \text{если } a=0{,}75,\\ л, & \text{если } a\ne 0{,}75.\end{cases}$$

ж) $A = [-1, 1]$, т.е. все реальные числа a такие, что $-1 \le a \le 1$; $F = \{f_1, f_2, f_3\}$; $P = \{p_1, p_2, p_3\}$.

$$f_1(a) = \cos(a-1); \qquad p_1(a)=\begin{cases}и, & \text{если } a<0,\\ л, & \text{если } a\ge 0;\end{cases}$$

$$f_2(a) = 0{,}5\ln(a+2); \qquad p_2(a)=\begin{cases}и, & \text{если } |a|=1,\\ л, & \text{если } |a|\ne 1;\end{cases}$$

$$f_3(a) = |a|; \qquad p_3(a)=\begin{cases}и, & \text{если } a\ne 0,\\ л, & \text{если } a=0.\end{cases}$$

II. *Определение множества R программ.*

а) Всякая программа Π множества R задается совокупностью из L команд ($L \geq 1$) с номерами i, $1 \leq i \leq L$. Команда задается либо тройкой $<i, f, l>$, где $f \in F$ и $0 \leq l \leq L$, либо четверкой $<i, f, l, k>$, где $p \in P$, $0 \leq l \leq L$, $0 \leq k \leq L$. Выполнение программы определяется так. Перед началом выполнения в ячейку машины записывается элемент $a \in A$ и первой выполняется команда с номером $i = 1$. В общем случае, в ячейке элемент $b \in A$, и выполнению подлежит команда с номером j, тогда:

- если $j = 0$, то программа останавливается и b считается результатом ее работы;

- если $j \neq 0$ и команда с этим номером $<j, f, l>$, то в ячейке записывается новый элемент $b' = f(b)$ и следующей выполняется команда с номером l;

- если $j \neq 0$ и команда с этим номером $<j, p, l, k>$, то в ячейке сохраняется тот же элемент b; проверяется условие p: если $p(b) = u$, то следующей выполняется команда с номером l, если $p(b) = л$ – команда с номером k.

Очевидно, в машинах а) – е) программа зацикливается тогда и только тогда, когда она не остановилась после выполнения $L \cdot k$ команд. В машинах ж) – з) полагать, что программа зацикливается, если она не остановилась после выполнения $L \cdot \sigma$ команд (σ задается).

б) Всякая программа Π множества R задается совокупностью из L команд ($L \geq 1$) с номерами i, $1 \leq i \leq L$. Команда – это либо пара $< i, f >$, где $f \in F$, либо тройка $< i, p, l >$, где $p \in P$ и $0 \leq l \leq L$. Выполнение программы определяется так. Перед началом выполнения в ячейку записывается элемент $a \in A$ и первой выполняется команда с номером $i = 1$. В общем случае, пусть в ячейке оказался элемент $b \in A$ и выполняется команда с номером j, тогда:

– если $j = 0$ или $j = L + 1$, то программа прекращает выполнение и b считается результатом ее работы;

– если $j \neq 0$ и команда $< j, f >$, то в ячейку засылается новый элемент $b' = f(b)$ и следующей выполняется команда с номером $j + 1$;

– если $j \neq 0$ и команда $< i, p, l >$, то в ячейке элемент сохраняется и проверяется условие p: при $p(b) = u$ следующей выполняется команда с номером l; при $p(b) = л$ – с номером $j + 1$.

Зацикливание в машинах а) – е) происходит тогда, когда программа не остановилась после выполнения $L{\cdot}k$ команд. В машинах ж) – з) полагать, что зацикливание происходит в случае отсутствия остановки до выполнения $L{\cdot}\sigma$ команд (σ задается).

в) Всякая программа Π множества R задается совокупностью из L команд ($L \geq 1$) с номерами i, $1 \leq i \leq L$. Команда – это $< i, p, f, l, g, k >$, где $p \in P$, $f, g \in F$, $0 \leq l$, $k \leq L$. Выполнение программы определяется так. Перед началом выполнения в ячейку записывается элемент $a \in A$ и первой выполняется команда с номером $i = 1$. В общем случае, пусть в ячейке оказался элемент $b \in A$ и выполняется команда с номером i, тогда:

– если $i = 0$, то программа останавливается и b считается результатом ее работы;

– если $i \neq 0$, то проверяется условие p: если $p(b) = u$, то в ячейку записывается элемент $b' = f(b)$ и следующей выполняется команда с номером l; если $p(b) = л$, то в ячейку записывается элемент $b'' = g(b)$ и следующей выполняется команда с номером k.

Если после выполнения $L{\cdot}k$ команд программа не остановилась, это значит, что она зациклилась (машины а) – е)); в машинах ж) – з) считать зацикливание программы в случае отсутствия остановки до выполнения $L{\cdot}\sigma$ команд (σ задается).

г) Всякая программа Π множества R и стрелок $\downarrow^i, \uparrow_i$, причем стрелка вверх ставится только после символа $p \in P$. Например,

$$\downarrow^1 p_1 \uparrow_2 f_1 f_2 f_1 p_2 \uparrow_1 f_2 \downarrow^2$$

Выполнение программы определяется так. Сначала в ячейку засылается элемент $a \in A$ и к нему применяется первый символ программы. В общем случае – пусть в ячейке находится элемент $b \in A$, к которому следует применить символ h_j, тогда:

– если h_j – это стрелка, то делается переход к следующему символу с сохранением в ячейке элемента b;

– если h_j – это $f \in F$, ячейку записывается новый элемент $b' = f(b)$ и делается переход к следующему символу;

– если h_j – это $p \in P$, то в ячейке сохраняется элемент b и проверяется условие $p(b)$: если $p(b) = u$, то совершается переход к стрелке $\downarrow^k$ (при

стрелке $\uparrow_k$, стоящей после $h_j = P$), если же $p(b) = л$, то делается переход к символу, стоящему вслед за стрелкой $\uparrow_k$.

Программа завершается тогда, когда происходит переход за ее последний символ. Считается, что программа зациклилась, если она не остановилась после выполнения σ операторов (σ задается).

д) Всякая программа $П \in R$ задается граф-схемой. Граф-схема – это совокупность вершин и соединяющих их стрелок, в которой:

– одна вершина не имеет входящих в нее стрелок (начальная вершина) и имеет только одну выходящую;

– одна вершина не имеет выходящих из нее стрелок (конечная вершина);

– из любой другой вершины выходит либо одна стрелка (такой вершине приписывается оператор $f \in F$), либо две стрелки - одна со знаком "+", другая – со знаком "–" (такой вершине приписывается предикат $p \in P$).

Выполнение программы. С исходным элементом $a \in A$ начинается движение по стрелке из начальной вершины. В общем случае, пусть идет движение по стрелке с элементом $b \in A$. Если стрелка заходит в вершину с приписанным оператором $f \in F$, то с элементом $b' = f(b)$ происходит движение по стрелке, выходящей из этой вершины. Если же заходит в вершину с приписанным предикатом $p \in P$, то проверяется условие $p(b)$: если $p(b) = и$, то дальнейшее движение происходит с тем же элементом b по стрелке с "+", выходящей из этой вершины; если $p(b) = л$, то движение с b продолжается по стрелке с "–" . Если с некоторым элементом $c \in A$ происходит движение по стрелке, заходящей в концевую вершину, то на этом выполнение программы заканчивается и считается результатом выполнения программы. Считать программу зацикленной, если при ее выполнении пройдено σ вершин и концевая вершина не достигнута.

1.4 *Построение экстремальной части графа*.

Графом называется совокупность точек (вершин) $A = \{a_1, \ldots, a_n\}$ и соединяющих их линий (ребер) $V = \{v_1, v_2, \ldots, v_m\}$. Не обязательно, чтобы в графе каждая пара точек соединялась линией. Пусть D – некоторое свойство подмножеств множества A (или множества D); тогда подмножество W называется D-экстремальным (минимальным или максимальным), если W удо-

влетворяет свойству D и никакое подмножество W' ($W' \subset W$, $W' \supset W$) не удовлетворяет свойству D.

Задание. Составить программу для выделения D - экстремального подмножества в заданном графе согласно указанному алгоритму его выделения.

Исходные данные

I. Граф, задаваемый вершинами и ребрами:

а) $A = \{1, 2, 3, 4, \ldots, 13, 14, 16\}$, т.е. граф состоит из $n = 16$ вершин, обозначенных целыми числами от 1 до 16; $V = \{(1, 10), (2, 13), (3, 13), (3, 14), (4, 5), (4, 7), (4, 14), (4, 15), (5, 6), (6, 7), (7, 15), (7, 16), (8, 16), (8, 9), (10, 11), (10, 12), (11, 12), (12, 13), (14, 15), (14, 16), (15, 16)\}$, т.е. в графе $m = 21$ ребро, каждое ребро обозначается парой вершин $V_i = (a, b)$.

б) $A = \{a_1, a_2, \ldots, a_{12}\}$;

$V = \{(a_1, a_{10}), (a_1, a_3), (a_1, a_{12}), (a_2, a_3), (a_3, a_{10}), (a_4, a_{10}), (a_4, a_{12}), (a_5, a_6), (a_5, a_8), (a_5, a_9), (a_6, a_8), (a_6, a_9), (a_7, a_8), (a_8, a_9), (a_{10}, a_{11}), (a_{10}, a_{12})\}$,

т.е. граф состоит из 12 вершин и 16 ребер.

в) $A = \{\gamma_1, \gamma_2, \gamma_3, \ldots, \gamma_{14}\}$;

$V = \{(\gamma_1, \gamma_2), (\gamma_1, \gamma_4), (\gamma_2, \gamma_4), (\gamma_3, \gamma_4), (\gamma_5, \gamma_6), (\gamma_5, \gamma_9), (\gamma_5, \gamma_{11}), (\gamma_5, \gamma_{13}), (\gamma_6, \gamma_{13}), (\gamma_7, \gamma_8), (\gamma_8, \gamma_{14}), (\gamma_9, \gamma_{10}), (\gamma_9, \gamma_{14}), (\gamma_{10}, \gamma_{12}), (\gamma_{10}, \gamma_{14}), (\gamma_{11}, \gamma_{12}), (\gamma_{12}, \gamma_{14}), (\gamma_{13}, \gamma_{14})\}$,

т.е. граф состоит из 14 вершин и 18 ребер.

г) $A = \{E_1, E_2, E_3, \ldots, E_{15}\}$;

$V = \{(E_1, E_6), (E_2, E_7), (E_3, E_8), (E_4, E_9), (E_5, E_{10}), (E_6, E_7), (E_7, E_8), (E_8, E_9), (E_9, E_{10}), (E_{10}, E_6), (E_6, E_{11}), (E_7, E_{12}), (E_8, E_{13}), (E_9, E_{14}), (E_{10}, E_{15}), (E_{11}, E_{13}), (E_{12}, E_{14}), (E_{13}, E_{15}), (E_{14}, E_{11}), (E_{15}, E_{12})\}$,

т.е. граф состоит из 15 вершин и 20 ребер.

д) Произвольный граф из $n = 13$ вершин и $m = 25$ ребер, вводимый в программу.

II. Свойство D подмножеств W:

а) Свойство «опорности» подмножества вершин $W \subseteq A$: для всякой вершины $a \in A \backslash W$ найдется вершина $x \in W$ такая, что a и x соединены в графе ребром, т.е. $(a, x) \in V$.

Алгоритм построения минимального опорного подмножества состоит в выполнении n шагов:

i - ый шаг) Пусть уже построено множество W_{i-1} (при $i = 1$ множество $W_0 = A$). Проверяется, удовлетворяет ли вершина a_i условиям:

1^0) в графе нет ребер, соединяющих вершину a_i с вершинами множества W_{i-1};

2^0) в $A \backslash W_{i-1}$ имеется такая вершина y, что в графе есть ребро (a_i, y) и нет других ребер между y и вершинами из W_{i-1}.

Если удовлетворяет хотя бы одному из них, то полагается $W_i = W_{i-1}$; в противном случае W_i получается из W_{i-1} выбрасыванием вершины a_i.

После выполнения n шагов множество W_n искомое.

б) Свойство «независимости» подмножества $W \subseteq A$: никакие две вершины из W не соединены в графе ребром.

Алгоритм построения максимального независимого подмножества состоит в выполнении не более n шагов. На первом шаге выбирается вершина a_1 и включается в W, в графе помечаются a_1 и все те вершины, которые соединены с ней ребрами. На i-ом шаге проверяется, имеются ли в графе непомеченные вершины. Если нет, то процесс построения W закончен. В противном случае выбирается непомеченная вершина x_k (с минимальным номером), включается в W, помечаются в графе a_k и все вершины, которые соединены с ней ребром. После этого делается переход к $(i + 1)$ –му шагу.

в) Свойство «независимости» подмножества $W \subseteq V$: никакие два ребра из W не имеют в графе общей вершины.

Алгоритм построения максимального независимого подмножества состоит в выполнении не более m шагов. На первом шаге выбирается ребро V_1 и включается в W, в графе помечаются V_1 и все те ребра, которые имеют общую вершину с V_1. На i-м шаге проверяется, имеются ли в графе не помеченные ребра. Если нет, то процесс построения W закончен. В противном случае выбирается непомеченное ребро V_k и все ребра, имеющие с ним общие вершины. После этого делается переход к $(i +1)$-му шагу.

г) Свойство «покрываемости» подмножества $W \subseteq V$: любая вершина графа является концом хотя бы одного ребра из W.

Алгоритм построения минимального покрывающего подмножества состоит из m шагов. На i-ом шаге из W_{i-1} строится W (при $i = 1$ $W_0 = V$): если ребро $V_i = (a_s, a_q)$ соединяет вершины a_s и a_q и в W_{i-1} есть другие ребра с

концами и в вершине a_s, и в вершине a_q, то W_i получается из W_{i-1} удалением ребра V_i; в противном случае $W_i = W_{i-1}$. искомым множеством будет W_m .

д) Свойство «покрываемости» подмножества $W \subseteq A$: любое ребро графа имеет концом некоторую вершину из W .

Алгоритм построения минимального покрывающего подмножества состоит из n шагов. На i-ом шаге из W_{i-1} строится W_i (при $i = 1$ $W_0 = A$): если для вершины a_i найдется такое ребро (a_i, b), что $b \notin W_{i-1}$, то $W_i = W_{i-1}$; в противном случае W_i получается из W_{i-1} выбрасыванием вершины a_i . Искомым будет W_n .

е) Свойство «полноты» подмножества $W \subseteq A$: любая пара различных вершин из W соединена в графе ребром.

Программа должна построить полное подмножество, исходя из двух концевых вершин некоторого ребра (номер ребра вводится): $V_j = (a, b)$. Алгоритм построения полного максимального подмножества W , содержащего вершины a и b, состоит в выполнении шагов. Пусть до i-го шага построено W_{i-1} (при $i = 1$ $W_0 = \{a, b\}$). Если находится в $A \backslash W_{i-1}$ такая вершина X, что в графе есть ребра между X и всякой вершиной из W_{i-1} , то W_{i-1} получается из W_i добавлением вершины X (и переход к следующему шагу); в противном случае W_{i-1} – искомое (и алгоритм заканчивается).

III. Упорядочивание вершин и ребер. Перед началом выполнения алгоритма множество вершин (в задачах IIа, б, д, е) или множество ребер (в задачах IIв, г) следует упорядочить согласно следующему правилу. Обозначим через $\varphi(a)$ количество ребер, имеющих концом вершину a; через $\psi(v)$ количество ребер, имеющих концевую вершину с ребром v.

Порядок вершин (их нумерация) должен быть таким, чтобы для всякого $i = 1, 2, \ldots, n-1$:

а) $\varphi(a_i) \le \varphi(a_{i+1})$

либо

б) $\varphi(a_i) \ge \varphi(a_{i+1})$

Порядок ребер (их нумерация) должен быть таким, чтобы для всякого $j = 1, 2, \ldots, m-1$:

а) $\psi(v_i) \ge \psi(v_{i+1})$

либо

б) $\psi(v_i) \le \psi(v_{i+1})$

1.5 *Поиск путей в схеме дорог*.

Под схемой дорог понимается совокупность пунктов $x_1, x_2, \ldots, x_n$ и дорог между некоторыми из них. Каждой дороге, обозначаемой (x_i, x_j), соответствует число $\mu(x_i, x_j)$, называемое расстоянием между пунктами x_i и x_j. Путем в схеме дорог между пунктами X и Y называется цепочка дорог $P_{x,y} = \{(x, a_1), (a_1, a_2), \ldots, (a_{k-1}, y)\}$, в которой дорога из схемы встречается не более одного раза. Длиною пути p считается число k, а расстояние пути – величина $\mu(P_{x,y}) = \Sigma(a_i, a_{i+1})$, если $x = a_0$ и $y = a_k$. Считается, что если между пунктами X, Y, Z имеются дороги, то

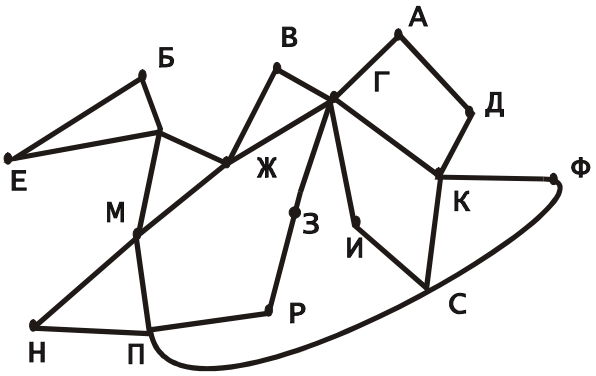
$$\mu(x, y) \le \mu(x, z) + \mu(z, y).$$

Задание. Составить программу, которая в заданной схеме дорог находит путь требуемого вида.

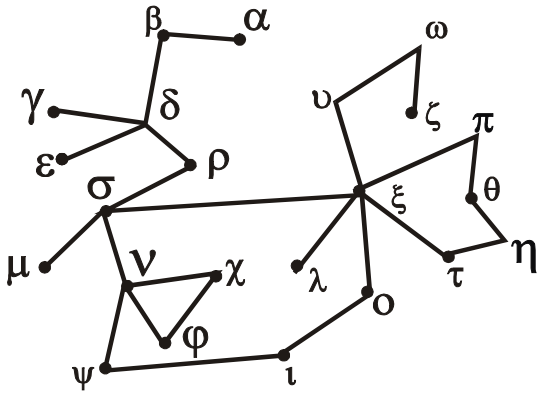
Исходные данные

I. Схема дорог.

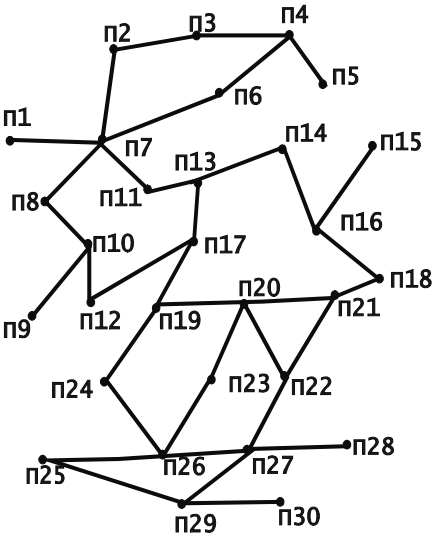
а)



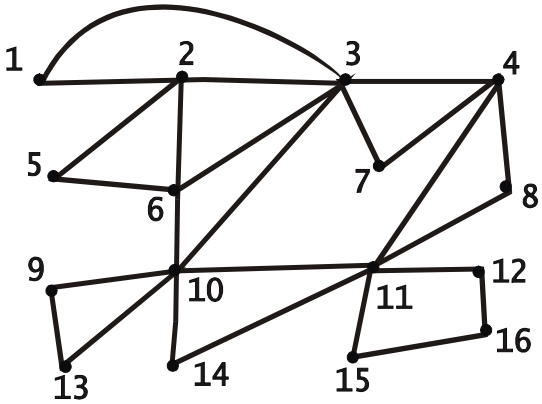
б)



в)



г)



Расстояния между пунктами выбираются самостоятельно.

II. Задачи по выбору путей.

а) По заданным (произвольно) двум пунктам X и Y (названия пунктов вводятся в программу) найти путь $P_{x,y}$ с минимальным расстоянием.

Алгоритм решения задачи. Пусть z – максимальное расстояние между пунктами; приписываем пункту X индекс-число 0, остальным пунктам – число $n \cdot z$. Совокупность этих индексов обозначим через $M_0(x_i)$ и введем строку $T_0(x_i) = \Lambda$, сопоставляющую каждому пункту пустой символ ($i = 1, 2, \ldots, n$). На j-м шаге алгоритма из $M_{j-1}(x_i)$ и $T_{j-1}(x_i)$ строятся новые строки $M_j(x_i)$ и $T_j(x_i)$, $i = 1, \ldots, n$: если в схеме дорог есть пара пунктов u и v такая, что $\mu(u,v) < M_{j-1}(u) - M_{j-1}(v)$, то

$$M_j(x_i) = \begin{cases} M_{j-1}(u) + \mu(u,v), & \text{если } x_i = v, \\ M_{j-1}(x_i), & \text{если } x_i \neq v; \end{cases}$$

$$T_j(x_i) = \begin{cases} u, & \text{если } x_i = v, \\ T_{j-1}(x_i), & \text{если } x_i \neq v \end{cases}$$

(и переход к следующему шагу); если же пары пунктов u и v нет, то алгоритм заканчивается. Пусть $M(x_i)$ и $T(x_i)$ – строки, полученные на последнем шаге алгоритма. Тогда $M(y)$ – расстояние требуемого пути, а пункты этого пути y, $T(y)$, $T(T(y))$, $T(T(T(y)))$, … X.

б) По заданным (произвольно) двум пунктам X и Y (названия пунктов вводятся в программу) найти путь $P_{x,y}$ минимальной длины и определить его расстояние.

Алгоритм решения задачи. Отнесем пункт X к ярусу 0. На i-ом шаге алгоритма всякий пункт a относится к ярусу i, если он еще не отнесен ни к какому ярусу и имеется дорога, связывающая его с пунктом яруса i-1; алгоритм прекращается, как только пункт становится отнесенным к некоторому ярусу. Ярус пункта y и будет требуемой длиной пути. Используя ярусы пунктов, найти затем некоторый путь и его расстояние.

в) По заданному пункту X определить все те пункты y, для которых $Dл(P_{x,y}) = \max_a(\min(P_{x,a}))$, где минимум берется по всем путям из X в a. Найти некоторый путь из X в один из x этих пунктов и вычислить его расстояние.

Алгоритм решения задачи. Припишем пункту X ранг 0. На i-шаге алгоритма определяется, имеются ли в схеме дорого пункты без рангов. Если имеются, то ранг i присваивается всем тем пунктам, которые до этого не имели ранга, но связываются дорогой с пунктом ранга $i-1$ (переход к следующему шагу). Если не имеются, то алгоритм прекращается и искомым будут все пункты максимального ранга. Используя ранги, построить некоторый путь из X в один из пунктов максимального яруса.

г) Путь, содержащий все дороги схемы, называется эйлеровым.

Алгоритм построения эйлерова пути. Выбирается любой пункт X и строится какой-нибудь путь из X в x. Если он оказался эйлеровым, то алгоритм закончен. В противном случае из схемы убираются те дороги (но не пункты), которые оказались в построенном пути. В оставшейся схеме выбирается один из пунктов y (построенного ранее пути), из которого есть дорога. Из него строится какой-нибудь путь, заканчивающийся в нем – и этот путь «вкладывается» в ранее построенный. Этот процесс продолжается до тех пор, пока не будет построен требуемый путь, либо не окажется пути из пункта в него же.

Примечание. Эйлеров путь имеется в пунктах 1а) и 1г) и его нет в схемах 1б) и 1в).

д) Для произвольно заданного пункта X найти все пути длины 3, начинающиеся в X, и вычислить расстояние каждого из них.

Алгоритм построения. Все пути из X длины 1 – это просто дороги, исходящие из X. Все пункты, к которым ведут дороги, назовем пунктами первого шага. Затем из каждого пункта первого шага по исходящим из них дорогам придем к пунктам второго шага – построив все пути длины 2. Затем из пунктов второго шага по исходящим из них дорогам придем к пунктам третьего шага, построив все пути длины 3. Заметим, что один и тот же пункт может оказаться пунктом и первого, и второго, и третьего шага.

е) Для произвольно заданного пункта X определить тот путь, который получается, если из него двигаться по дорогам, выбирая в каждом пункте нехоженую дорогу с минимальным расстоянием.

ж) Для произвольно заданного пункта X определить тот путь, который получится, если начать из X двигаться по дорогам, выбирая в каждом пункте самую длинную нехоженую дорогу.

з) Путник, не зная пути, собирается выйти из пункта X, чтобы достигнуть пункта y (названия этих пунктов вводятся в программу). Определить расстояние, которое он покроет в своем движении, если будет следовать следующим правилам. Перед началом движения для путника все дорого зеленые и все пункты – *новые*, кроме пункта X, который *старый*:

- если путник приходит в *новый* пункт a (по зеленой дороге), то но «перекрашивает» ее в желтый цвет и пункт «превращается» в *старый*. Начинает искать зеленую дорогу из пункта a (то же делает в пункте X): если находит, то двигается по ней; если не находит, то возвращается по только что пройденной дороге в предыдущий пункт, перекрашивая вторично дорогу из желтой в красную;

- если путник приходит в *старый* пункт по зеленой дороге, то по пути он перекрашивает ее в желтый цвет, затем возвращается по ней обратно в предыдущий пункт;

- если путник приходит в *старый* пункт b по желтой дороге, то он перекрашивает дорогу в красную; затем ищет из пункта b зеленую дорогу: если находит, то движется по ней; если не находит, то движется по (единственной) желтой дороге, исходящей из b.

и) В схеме дорог каждый пункт связан дорогой с любым другим пунктом. Требуется построить такой путь, который проходит через каждый пункт, причем точно по одному разу. Алгоритм построения пути.

На первом шаге выбираются два пункта с самой короткой дорогой между ними, пусть ими будут пункты X_i и X_j . Затем выбирается пункт x_k так, чтобы $\mu(x_k, x_i) = \min \mu(x_l, x_i)$.

Образуется замкнутый путь

$$P_1 = \{(x_i, x_k), (x_k, x_j), (x_j, x_i)\}.$$

В общем случае, пусть к i -ому шагу алгоритма построен замкнутый путь P_{i-1}. Выбирается пункт $x_l \notin P_{i-1}$ такой, что $\mu(x_l, x_m) = \min \mu(y, z)$, где минимум берется по всем пунктам $y \notin P_{i-1}$ и всем пунктам $z \in P_{i-1}$. Новый путь P_i получается из P_{i-1} заменой дороги (x_m, x_l) из P_{i-1} на новые две (x_m, x_l) (x_l, x_r). Число шагов алгоритма n –2. Программа должна вывести построенный путь и его расстояние.

к) Задача формулируется аналогично предыдущей. Отличие только в описании произвольного шага алгоритма (i-го шага с $i \geq 2$). Пусть к i -ому шагу

построен замкнутый путь P_{i-1} . Выбирается произвольный пункт $x_l \notin P_{i-1}$. Отыскивается такая дорога в пути P_{i-1} – дорога (x_s, x_r), чтобы была минимальна величина $\mu(x_s, x_l) + \mu(x_l, x_r) - \mu(x_s, x_r)$, где минимум берется по всем дорогам из пути P_{i-1}. Строится новый путь P_i путем замены одной дороги (x_s, x_l) на пару дорог (x_s, x_l) (x_l, x_r).

1.6 *Формулы с двуместными операциями*.

Для записи формул используются: символы двуместных операций $\mathfrak{R} = \{\alpha_1, \ldots, \alpha_n\}$, символы переменных и величин $X = \{x_1, \ldots, x_m\}$ и круглые скобки « (» и «) ». Формула определяется индуктивно (в определениях $\theta \in \mathfrak{R}$ и $a, b \in x$).

Определение 1 формулы в записи со скобками. Выражение вида $(a)\ \theta\ (b)$ называется формулой (элементарной); если A и B – формулы или символы из X, то выражение (A) θ (B) называется формулой.

Определение 2 формулы в бесскобочной записи. Выражение вида θab называется формулой (элементарной); если A и B – формулы или символы из X, то выражение θAB называется формулой.

Задание. Для заданной системы формул составить программу по решению одной из задач над их записями.

Исходные данные

I. Система формул определяется символами:

а) $\mathfrak{R} = \{+, -, \times, /\}$ – знаки арифметических операций;

$X = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ – цифры.

б) $\mathfrak{R} = \{+, -, \times, /\}$ – знаки арифметических операций;

$X = \{x, y, z, t, u, v, w\}$ – символы переменных.

в) $\mathfrak{R} = \{\vee, \wedge, \supset, \equiv, \uparrow, \oplus\}$ – знаки логических операций;

$X = \{u, л\}$ – символы логических величин.

г) $\mathfrak{R} = \{\vee, \wedge, \supset, \equiv, \uparrow, \oplus\}$ – знаки логических операций;

$X = \{\alpha, \beta, \gamma, \delta, \varepsilon, \mu, \nu, \lambda\}$ – символы логических переменных.

д) $\mathfrak{R} = \{f, g, h, \varphi, \psi\}$ – символы двуместных операций;

$X = \{0, 1\ p, q, r, s\}$ – символы переменных и чисел.

е) $\mathfrak{R} = \{F, G, H\}$, $X = \{f, g, h, 0, 3, 6, 9\}$.

II. Задача над записями формул.

а) Проверка на правильность записи формулы в записи со скобками. Программа вводит строку символов произвольной длины $\sigma_1, \ldots, \sigma_l$. Проверяет, содержит ли она символы только из $\Re$, X и скобки. Если есть иной символ, то сообщает об этом, прекращая дальнейшую проверку. Если других символов нет, то создает новую строку $\rho_1, \rho_2, \ldots, \rho_l$, в которой $\rho_i = 0$, если $\sigma_i =$ "("; $\rho_i = 1$, если $\sigma_i =$ ")"; $\rho_i = 2$, если $\sigma_i \in \Re$; $\rho_i = 3$, если $\sigma_i \in X$. Запись формулы будет правильной только тогда, когда строку $\rho_1, \rho_2, \ldots, \rho_l$ можно превратить в одну цифру 3 заменами в ней комбинаций 0312031 на 3.

б) Проверка на правильность записи формулы в бесскобочной записи. Программа вводит строку символов произвольной длины $\sigma_1, \ldots, \sigma_l$. Если в ней есть символы, не входящие в $\Re$ или X, то печатается сообщение и анализ прекращается. Если других символов нет, то создается строка $\rho_1, \rho_2, \ldots, \rho_l$, где $\rho_i = 0$, если $\sigma_i \in \Re$, $\rho_i = 1$, если $\sigma_i \in X$. Запись формулы будет правильной только тогда, когда эту строку можно превратить в цифру 1 заменами в ней комбинаций 011 на 1.

в) Перевод формулы из записи со скобками в эквивалентную бесскобочную запись. Программа составляется в предположении, что вводимая строка символов представляет собой правильную запись формулы со скобками. В результате выполнения программы получается эквивалентная бесскобочная запись формулы, которая выводится на печать. Алгоритм перевода состоит в том, что в исходной строке символов отыскивается часть строки вида $(a_1,\ldots,a_{l_1})\theta(b_1,\ldots,b_{l_2})$, в которой $a_1,\ldots,a_{l_1}\ b_1,\ldots,b_{l_2}$ не содержит скобок, и эта часть заменяется на $\theta a_1,\ldots,a_{l_1}\ b_1,\ldots,b_{l_2}$. Формула превратится в искомую, когда в ней исчезнут все скобки.

г) Перевод формулы из записи со скобками в эквивалентную бесскобочную запись. Программа вводит строку символов – правильную запись формулы со скобками. Перевод формулы в бесскобочную запись производится согласно следующему алгоритму. На каждом шаге отыскивается первая слева такая операция θ, слева от которой число открывающих скобок равно числу закрывающих

$$c_1\ldots c_{k_1}(a_1\ldots a_{l_1})\theta(b_1\ldots b_{l_2})d_{1\ldots d_{k_2}},$$

где среди $c_1, \ldots, c_{k1}$ нет скобок, а среди $a_1, \ldots, a_{l1}$ они могут быть. Справа отыскивается такая скобка (между b_{l_2} и d_1), чтобы между θ и d_1 было одинаковое количество открывающих и закрывающих скобок. Тогда эта строка переводит в новую $c_1 \ldots c_{k_1} \theta\, a_1 \ldots a_{l_1}\, b_1 \ldots b_{l_2}\, d_1 \ldots d_{k_2}$.

Шаги алгоритма выполняются до тех пор, пока в строке не останется скобок. Полученная строка выводится на печать.

д) Перевод формулы из бесскобочной записи в запись со скобками. Программа вводит строку символов – правильную бесскобочную запись формулы. Алгоритм перевода основан на том, что всякая формула в бесскобочной записи содержит l символов из X и $l-1$ символов из $\Re$. Поэтому на первом шаге алгоритма в строке выделяются две части $\theta a_1 \ldots a_r\, b_1 \ldots b_s$, где $r \geq 1$, $s \geq 1$, такие, чтобы строки $a_1 \ldots a_r$ и $b_1 \ldots b_s$ были строками формул; образуется новая строка $(a_1 \ldots a_r)\ \theta\ (b_1 \ldots b_s)$. На каждом следующем шаге отыскивается символ $\theta' \in \Re$, стоящий после скобки « (». Если такого символа нет, то алгоритм заканчивается, и программа выводит образованную запись со скобками. Если такой символ находится, то выделяется часть строки, начинающаяся с θ', над которой выполняются те же операции, что на первом шаге.

е) Перевод формулы из бесскобочной записи в запись со скобками. Вводится строка символов, представляющих правильную запись без скобок. Алгоритм перевода основан на том, что всякая формула в записи со скобками имеет одинаковое число открывающих и закрывающих скобок. На первом шаге все символы из X заключаются в круглые скобки. На каждом последующем отыскивается подстрока вида $\theta\ (a_1 \ldots a_l)\ (b_1 \ldots b_k)$, чтобы среди $a_1 \ldots a_l$ (и среди $b_1 \ldots b_k$) было поровну скобок открывающих и закрывающих. Она заменяется на новую подстроку $(a_1 \ldots a_l)\theta\ (b_1 \ldots b_k)$. Алгоритм заканчивается, когда исходная строка превратится в новую – начинающуюся со скобки « (», а не знака из $\Re$.

1.7 *Алгоритм преобразования слов*. Набор $A = \{a_1, a_2, \ldots, a_k\}$ некоторых символов называется алфавитом, а сами символы $a_1, \ldots, a_n$ буквами. Последовательность букв называется словом в алфавите A. Множество всех возможных слов в A обозначается через A^*. Слово, не содержащее ни одной

буквы, называется пустым. Количество букв в слове называется его длиной *Дл*(*P*), $P \in A^*$.

Если слово $P \in A^*$ можно представить в виде трех слов *QRT*, где *Q* или *T* может быть пустым, тогда говорят, что слово *R* входит в слово *P*, или слово *P* содержит подслово *R*. Если к тому же слово *P* нельзя представить в виде *Q′RT′* с условием Дл (*Q′*) < Дл (*Q*), тогда говорят, что в представлении *QRT* = *P* слово *R* является первым вхождением в слово *P*.

Операция R →S преобразует слово *P* = *QRT* в слово u = *QST*. Мы говорим, что в слове *P* слово *S* заменяет первое вхождение слова *R*. Совокупность таких операций с указанием порядка их выполнения называется алгоритмом преобразования слов.

Задание. Составить программу, моделирующую работу заданного алгоритма преобразования слов и решающую при этом конкретную задачу *E*.

Исходные данные

I. Задание алгоритма.

а) Нормальные алгоритмы. Задана пронумерованная совокупность операций *i*) $R_i \to S_i$, i= 1, 2, …, *L*, причем в некоторых из них после стрелки может стоять точка $R_i \to .S_i$ (такая операция называется заключительной).

Каждый шаг алгоритма состоит в следующем. Пусть на предыдущем шаге получено слово *P* (на первом шаге слово исходное). Отыскивается первая по порядку операция, левое слово которой входит в *P*. Если такой операции нет, то алгоритм заканчивает работу и *P* является его результатом. Если есть, то правое слово из операции (*S*) подставляется в слово *P* вместо первого вхождения в него левого слова операции (*R*) – образуется новое слово *P ′*. При этом, если операция заключительная, то алгоритм заканчивается и *P ′* считается результатом; в противном случае к *P ′* применяется следующий шаг алгоритма.

Примеры нормальных алгоритмов.

а1) алгоритм умножения

A = {*, 1, ∨, ?}

1) * 11→ ∨*1

2) *1 → ∨

3) 1∨ → ∨1?

а2) алгоритм обращения слов

A = {*a*, *b*, 0, 1, +, *}

1) +* → ·

2) +*a* →0

3) +*b* → 1

4) $?\vee \to \vee ?$

5) $?1 \to 1?$

6) $\vee 1 \to \vee$

7) $\vee ? \to ?$

4) $0a \to a0$

5) $0b \to b0$

6) $0* \to *a$

7) $1a \to a1$

8) $1b \to b1$

9) $1* \to *b$

10) $\to +$

а3) алгоритм удвоения слова

$A = \{\alpha, \beta, x, y, z, 1\}$

1) $x\,\alpha \to \alpha x$

2) $x\beta \to \beta x$

3) $x1 \to 1\alpha$

4) $y\alpha \to \alpha y$

5) $y\beta \to \beta y$

6) $y1 \to 1\beta$

7) $z\alpha \to \alpha z x$

8) $z\beta \to \beta z y$

9) $z1 \to \cdot$

а4) $A = \{0, 1, \alpha, \beta, \gamma, \delta, \varepsilon\}$

1) $\alpha 0 \to \alpha$

2) $\alpha 1 \to 11\beta$

3) $\beta 0 \to \gamma$

4) $\beta 1 \to 101\beta$

5) $\gamma 0 \to \delta 00$

6) $\gamma 1 \to \varepsilon$

7) $\varepsilon \to \beta$

8) $1 \to 0\beta\alpha$

а5) $A = \{1, 2, 3, a, b, c\}$

1) $1\,a \to aa1$

2) $1b \to b2$

3) $2b \to b3$

4) $2a \to ac2$

5) $3a \to aac3$

6) $3b \to \cdot 1$

7) $\to 1$

а6) $A = \{а, и, л, п, *\}$

1) *лапа → пал*

2) *ла →*

3) *п → пила*

4) * а → пли*

5) *л → липа*

6) *и → ·

7) * → и

8) → *

б) Обобщенные нормальные алгоритмы. Задана пронумерованная совокупность операций с числами *i*) $R_i \to S_i\alpha_i$, причем $i = 1, 2, \ldots, L$ и $1 \le \alpha \le L+1$. На 1-ом шаге алгоритма к исходному слову применяется операция с номером 1. В общем случае пусть к слову Q применяется операция j на некотором шаге алгоритма. Это применение алгоритма состоит в следую-

щем: если R_j входит в Q, тогда первое вхождение слова R_j заменяется на S_j, получается новое слово Q′, к которому на следующем шаге применяется операция с номером α_j; если же R_j не входит в Q, тогда к Q (на следующем шаге) применяется операция с номером j +1. Процесс обрывается, когда требуется выполнить операцию с номером L +1.

Примеры обобщенных нормальных алгоритмов.

σ1) A = {a, b, 1, 2, 3}

1) $\rightarrow$ 1, 2
2) 1*a* $\rightarrow$ *aa* 1, 2
3) 1*b* $\rightarrow$ *b* 2, 4
4) 2*b* $\rightarrow$ *bb* 2, 4
5) 2*a* $\rightarrow$ 3, 6
6) 3*ba* $\rightarrow$ 1, 3
7) 3*ba* $\rightarrow$ *b*1, 2

σ2) A = {0, 1, α, β, γ}

1) 00 $\rightarrow$ α, 3
2) 11 $\rightarrow$ β, 4
3) α0 $\rightarrow$ 0 γ α, 2
4) β1 $\rightarrow$ 1 γ α, 1
5) γ $\rightarrow$ α α, 3
6) α1 $\rightarrow$ α β, 3
7) 0 $\rightarrow$ γ, 5

σ3) A = {1, +, *x*, ∨, ∧,*}

1) 111 $\rightarrow$ 1 *x*, 5
2) + $\rightarrow$ *x*∨ ∧*, 4
3) +1 $\rightarrow$ +, 3
4) ∨* $\rightarrow$, 7
5) ∨ ∧*x* $\rightarrow$∨, 2
6) *x* $\rightarrow$ ∨ +, 1

σ4) A = {0, 1, 2, 3, 4}

1) 0 $\rightarrow$ 44, 2
2) 13 $\rightarrow$ 31, 1
3) 111 $\rightarrow$ 3, 5
4) 41 $\rightarrow$ 000, 1
5) 2 $\rightarrow$ 4, 3
6) 22 $\rightarrow$ 2, 1

σ5) A = {α, β, γ, +}

1) α $\rightarrow$ +, 1
2) β $\rightarrow$ α β, 4
3) γ $\rightarrow$ α γ, 4
4) α+ $\rightarrow$ γ, 3
5) β+ $\rightarrow$ γγ, 8
6) γ+ $\rightarrow$ γγ+, 1
7) α β γ $\rightarrow$ ++, 2

σ6) A = {*а*, б, р, н}

1) н*а* $\rightarrow$ н*а*н*а*, 3
2) р*а*б $\rightarrow$ *а*р*а*, 4
3) б*а* $\rightarrow$ б*а*р*а*н, 1
4) бр*а* $\rightarrow$ рр, 5
5) р*а*н*а* $\rightarrow$ б*а*р*а*б*а*н, 2
6) б*а*р $\rightarrow$ *а*, 1
7) р $\rightarrow$ *а*рб*а*, 4
8) *а* $\rightarrow$ б*а*н*а*н, 6

II. Задача Е, решаемая программой.

Программа вводит произвольное слово в алфавите A в качестве исходного для работы алгоритма и число N. В результате выполнения алгоритма получается результирующее слово (либо в момент останова алгоритма, либо после N шагов), которое программа выводит. Вместе со словом выводится величина M (через P_i обозначено слово, получающееся после i-го шага алгоритма):

а) M = max Дл(P_i);

б) M = max ($x_1(P_i) + x_2(P_i) + x_3(P_i)$), где $x_j(P)$ – количество вхождений буквы $a_j \in A$ в слово P;

в) M – количество выполненных операций с номерами 1, 3, 4;

г) M – количество слов среди $P, P_1, P_2, \ldots, P_n$, в которые не входят буквы a_1 и a_4;

д) Дл ($P_м$) = max Дл(P_i).

1.8 *Построение циклической структуры подстановки.*

Подстановкой f называется отображение конечного множества $A = \{a_1, a_2, \ldots, a_n\}$ на себя, изображаемое либо с помощью выражения, либо двумя строчками

$$f = \begin{pmatrix} a_1 a_2 \ldots a_n \\ a_{i_1} a_{i_2} \ldots a_{i_n} \end{pmatrix}, \quad \text{т.е. } f(a_m) = a_{i_m},$$

в которой все индексы $i_1, i_2, \ldots, i_n$ различны, либо совокупностью циклов $f = (a_{j_1}, \ldots a_{j_r}) \ldots (a_{k_1}, \ldots, a_{k_{r_s}})$. В один цикл включаются элементы в следующем порядке $a_{j_1}, a_{j_2} = f(a_{j_1}), a_{j_3} = f(a_{j_2}), \ldots a_{j_r} = f(a_{j_{r-1}})$, причем $f(a_{j_r}) = a_{j_1}$. Например, если $f = \begin{pmatrix} a_1\ a_2\ a_3\ a_4\ a_5\ a_6 \\ a_3\ a_6\ a_4\ a_1\ a_5\ a_2 \end{pmatrix}$, то $f = (a_1, a_3, a_4)\ (a_2, a_6)\ (a_5)$. Числа $r_1, r_2, \ldots, r_s$ называются циклической структурой подстановки f.

Если заданы две подстановки f и g, то их произведением является новая подстановка $h = f \cdot g$, которая определяется как $h(a) = g(f(a))$ для всякого $a \in A$.

Задание. По двум заданным подстановкам f_0 и f_1 на множестве A и последовательности $\mu_1, \mu_2, \ldots, \mu_q$ из нулей и единиц построить подстановку

$h = f_{\mu_1} \cdot f_{\mu_2} \cdot \ldots \cdot f_{\mu_q}$, вычислить ее циклическую структуру и напечатать. Последовательность $\mu_1, \mu_2, \ldots, \mu_q$ вводится в программу.

Исходные данные

I. Множество A = {1, 2, 3, 4, … , 18, 19, 20}.

Подстановки f_0 и f_1 задаются с помощью выражений.

а) $f_0(k) = \begin{cases} k+1, & \text{если } k < 20, \\ 1, & \text{если } k = 20; \end{cases}$ $f_1(k) = \begin{cases} 2k, & \text{если } 1 \le k \le 10, \\ 2k-21, & \text{если } 11 \le k \le 20; \end{cases}$

б) $f_0(k) = \begin{cases} k+5, & \text{если } 1 \le k \le 15, \\ k-15, & \text{если } 16 \le k \le 20; \end{cases}$ $f_1(k) = \begin{cases} k+1, & \text{если } k \text{ нечетно}, \\ k-1, & \text{если } k \text{ четно}; \end{cases}$

в) $f_0(k) = \begin{cases} k+1, & \text{если } 1 \le k \le 10, \\ 1, & \text{если } k = 11, \\ k, & \text{если } 12 \le k \le 20; \end{cases}$ $f_1(k) = \begin{cases} k-1, & \text{если } 10 \le k \le 20, \\ 20, & \text{если } k = 9, \\ k, & \text{если } 1 \le k \le 8; \end{cases}$

г) $f_0(k) = \begin{cases} k+10, & \text{если } 1 \le k \le 10, \\ 21-k, & \text{если } 11 \le k \le 20; \end{cases}$ $f_1(k) = \begin{cases} 20-k, & \text{если } 1 \le k \le 19, \\ 20, & \text{если } k = 20; \end{cases}$

д) $f_0(k) = \begin{cases} 20-k, & \text{если } 1 \le k \le 19, \\ 20, & \text{если } k = 20; \end{cases}$ $f_1(k) = \begin{cases} k-1, & \text{если } 2 \le k \le 20, \\ 1, & \text{если } k = 20; \end{cases}$

е) $f_0(k) = \begin{cases} 4k, & \text{если } 1 \le k \le 5, \\ 4k-22, & \text{если } 6 \le k \le 10, \\ 4k-43, & \text{если } 11 \le k \le 15, \\ 4k-61, & \text{если } 16 \le k \le 20; \end{cases}$ $f_1(k) = \begin{cases} k+2, & \text{если } 1 \le k \le 18, \\ 1, & \text{если } k = 19, \\ 2, & \text{если } k = 20; \end{cases}$

ж) $f_0(k) = \begin{cases} 2k, & \text{если } k \text{ нечетно}, k \le 9, \\ 2k-1, & \text{если } k \text{ четно}, k \ge 10, \\ 2k-20, & \text{если } k \text{ четно}, 12 \le k, \\ 2k-21, & \text{если } k \text{ нечетно}, 11 \le k; \end{cases}$ $f_1(k) = \begin{cases} k-3, & \text{если } 4 \le k \le 20, \\ 18, & \text{если } k = 3, \\ 19, & \text{если } k = 2, \\ 20, & \text{если } k = 1; \end{cases}$

з) $f_0(k) = \begin{cases} 3k, & \text{если } 1 \le k \le 6, \\ 3k-20, & \text{если } 7 \le k \le 12, \\ 3k-40, & \text{если } 13 \le k \le 20; \end{cases}$ $f_1(k) = \begin{cases} k+1, & \text{если } k \text{ нечетно}, \\ k-1, & \text{если } k \text{ четно}. \end{cases}$

II. Множество $A = \{\alpha, \beta, \gamma, \delta, a, b, c, d, 0, 1, 2, 3, +, -, /, \times\}$

Подстановки f_0 и f_1 задаются строками: верхняя строка постоянна, поэтому приводится только нижняя

а) α β γ δ *a* *b* *c* *d* 0 1 2 3 + − / ×

f_0 = β *a* 3 − 0 δ 1 α 2 / + *b* γ × *d* *c*

f_1 = *a* *c* γ β 0 + − δ *b* α 1 × *d* / 2 3

б) f_0 = 0 *a* × *d* δ 1 β 2 + α 3 / *c* − γ *b*

f_1 = − *a* 3 + 0 *b* α *c* δ / 2 × γ α 1 β

в) f_0 = *c* / *a* *b* 2 1 *d* 3 δ α 0 + β γ − ×

f_1 = γ *c* 0 δ *b* × − α *d* 1 β + 2 3 *a* /

г) f_0 = *d* + 3 *c* 1 γ *a* 2 / × β 0 δ α − *b*

f_1 = α 3 *c* 2 β *a* + − 0 *b* δ *d* × 1 γ /

д) f_0 = + *c* δ / − × α β *d* *a* 3 0 1 γ *b* 2

f_1 = 3 *b* + *a* / 2 × γ 1 *c* − α δ 0 *d* β

е) f_0 = / 0 × β γ *a* 1 α 3 *d* + *c* *b* − 2 *b*

f_1 = β γ δ α 1 2 3 0 *d* *c* *a* *b* − / × +

ж) f_0 = *b* − 0 γ *c* β δ + / *a* 3 × α 2 *d* 1

f_1 = 0 1 2 3 δ γ β α − + / × *a* *b* *c* *d*

з) f_0 = γ / δ *a* 0 − + α 2 3 β α *b* × *c* 1

f_1 = *a* 1 *b* 0 *c* 3 *d* 2 δ + γ / α × β −

III. Множество $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}\}$.

Подстановки f_0 и f_1 задаются совокупностью циклов

а) $f_0 = (a_1, a_5, a_7, a_{15}, a_{11})(a_2, a_{13}, a_6, a_3, a_{10})(a_3, a_8, a_9, a_5, a_{14})$,

$f_1 = (a_1, a_2, a_3)(a_4, a_8)(a_5, a_{15})(a_6)(a_7, a_{10})(a_9, a_{11})(a_{12}, a_{13}, a_{14})$;

б) $f_0 = (a_1, a_6, a_2, a_{14})(a_3, a_{11}, a_8)(a_5)(a_4, a_7, a_{13}, a_{15}, a_{12})(a_9, a_{10})$,

$f_1 = (a_1, a_2, a_{11}, a_7, a_9, a_5, a_3, a_{15})(a_4, a_6, a_8, a_{10}, a_{12}, a_{13}, a_{14})$;

в) $f_0 = (a_1, a_{13}, a_4)(a_2, a_8, a_7, a_{14}, a_{15})(a_3, a_6, a_5, a_9)(a_{10})(a_{11})(a_{12})$,

$f_1 = (a_1, a_8, a_6, a_{10}, a_2, a_{12})(a_4, a_{11})(a_3)(a_5, a_{15}, a_{13})(a_7)(a_9, a_{14})$;

г) $f_0 = (a_1, a_8)(a_2, a_{14})(a_3, a_7, a_9)(a_4, a_{13}, a_{15}, a_5)(a_6)(a_{10}, a_{11})(a_{12})$,

$f_1 = (a_1, a_3, a_6)(a_8, a_{14}, a_{12})(a_9, a_4, a_{10}, a_{15}, a_5, a_2, a_7)(a_{11})(a_{13})$;

д) $f_0 = (a_1, a_{10}, a_{15}, a_2, a_{11}, a_4, a_3, a_9, a_8, a_7, a_{13}, a_{12}, a_5, a_6, a_{14})$,

$f_1 = (a_1)(a_2, a_4, a_6)(a_3, a_5)(a_7, a_8, a_9, a_{10})(a_{11})(a_{12}, a_{13})(a_{14}, a_{15})$;

е) $f_0 = (a_1, a_{11}, a_8, a_4, a_5, a_{15})(a_2, a_7, a_{13}, a_6, a_3, a_{14}, a_{10})(a_{12}, a_9)$,

$f_1 = (a_1)(a_2)(a_3, a_4)(a_5, a_6, a_7)(a_8, a_9, a_{10}, a_{11})(a_{12}, a_{13}, a_{14}, a_{15})$;

ж) $f_0 = (a_1, a_2, a_3, a_4, a_5, a_6)(a_7, a_8, a_9)(a_{10}, a_{11})(a_{12}, a_{13}, a_{14}, a_{15})$,

$f_1 = (a_1)(a_2, a_3)(a_4, a_5, a_{10}, a_8)(a_6, a_{11}, a_{15})(a_7, a_9, a_{12}, a_{13})$;

з) $f_0 = (a_1, a_2, a_3)(a_4, a_5, a_6)(a_7)(a_8, a_9)(a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15})$,

$f_1 = (a_1, a_5, a_7, a_{10}, a_{13}, a_{15})(a_2, a_8, a_{14})(a_3)(a_4)(a_6, a_9)(a_{11}, a_{12})$.

IV. Множество A состоит из 16 двоичных наборов длины 4, т.е. из наборов 0000, 0001, … , 1111. Подстановка на A задается системой булевых функций: если $a_i = x_1 x_2 x_3 x_4$, то $f(a_i) = y_1 y_2 y_3 y_4$. В выражениях сумма берется по модулю 2 (операция отрицания эквивалентности), а произведение – как конъюнкция.

а) f_0: $y_1 = x_1 + x_2 \cdot x_3 \cdot x_4$ $\quad$ f_1: $y_1 = x_1 + \bar{x}_2 \cdot \bar{x}_3$

$y_2 = x_2 + x_3 \cdot x_4$ $\quad$ $y_2 = x_2 + x_3 \cdot x_4 + 1$

$y_3 = x_3 + x_4$ $\quad$ $y_3 = x_3$

$y_4 = x_4 + 1$ $\quad$ $y_4 = x_4 + 1$

б) f_0: $y_1 = x_4 \oplus x_1 \cdot \bar{x}_2 \cdot x_3 + x_2$ $\quad$ f_1: $y_1 = x_4 \oplus x_1 \cdot \bar{x}_2 \cdot x_3 + \bar{x}_2$

$y_2 = x_1$ $\quad$ $y_2 = x_1$

$y_3 = x_2$ $\quad$ $y_3 = x_2$

$y_4 = x_3$ $\quad$ $y_4 = x_3$

в) f_0: $y_1 = \bar{x}_1$ $\quad$ f_1: $y_1 = x_1 + \overline{x_2 x_3 x_4}$

$y_2 = \bar{x}_2$ $\quad$ $y_2 = x_2 + \overline{x_3 x_4}$

$y_3 = \bar{x}_3$ $\quad$ $y_3 = x_3 + \bar{x}_4$

$y_4 = \bar{x}_4$ $\quad$ $y_4 = \bar{x}_4$

г) f_0: $y_1 = \bar{x}_1$ $\quad$ f_1: $y_1 = x_1 + \bar{x}_3 \cdot \bar{x}_4$

$y_2 = x_2 + x_1$ $\quad$ $y_2 = x_2$

$y_3 = x_3 + \overline{x_1 x_2}$ $\quad$ $y_3 = x_3$

$y_4 = \bar{x}_4$ $\quad$ $y_4 = x_4$

1.9 *Решение краевой задачи методом Монте-Карло*.

Плоскость (x, y) делится сеткой с шагом h, и рассматриваются только точки сетки. Задается область с границей Γ. Точки границы обозначим через

(x_i, y_i), $i = 1, 2, \ldots, n$. В этих точках задана функция $f(x, y)$. Краевая задача формулируется так: определить значение функции $u(x, y)$ во внутренней точке области (x^*, y^*), которая удовлетворяет уравнению Лапласа $\Delta u = 0$ и принимает на границе Γ значения $u(x_i, y_i) = f(x_i, y_i)$.

Задача решается методом Монте-Карло. Из точки (x^*, y^*) организуется N блужданий по внутренним точкам области, (точкам сетки) до выхода на граничную точку. Пусть φ_i – количество блужданий с приходом в точку (x_i, y_i) границы Γ. Тогда искомое значение

$$u(x^*, y^*) = \frac{1}{N} \sum_{i=1}^{n} \varphi_i \cdot f(x_i, y_i).$$

Задание. Составить программу для заданных h, Γ, $f(x, y)$ и «способа блуждания», которая вводит координаты точки (x^*, y^*) и число N, вычисляет $u(x^*, y^*)$ и печатает это значение.

Исходные данные

I. Величина h, граница Γ.

а) $h = 0{,}1$, Г – эллипс $(x - 1)^2 + 2y^2 = 1$;

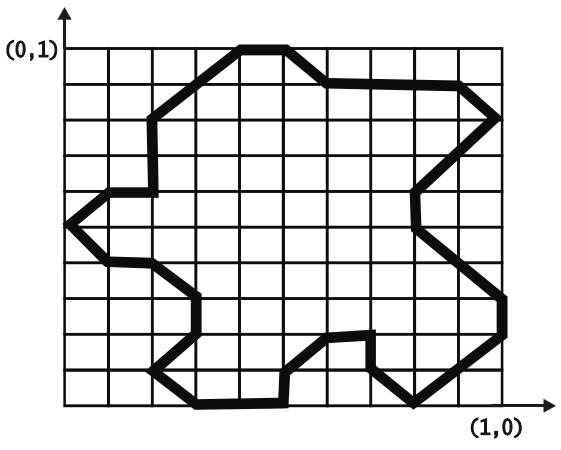
б) $h = 0{,}1$, Г – граница области $y^2 \geq x$, $0 \leq y \leq 1$;

в) $h = 0{,}2$, Г – граница области, ограниченной прямыми $y = 0$, $y = x$, $x + y = 4$;

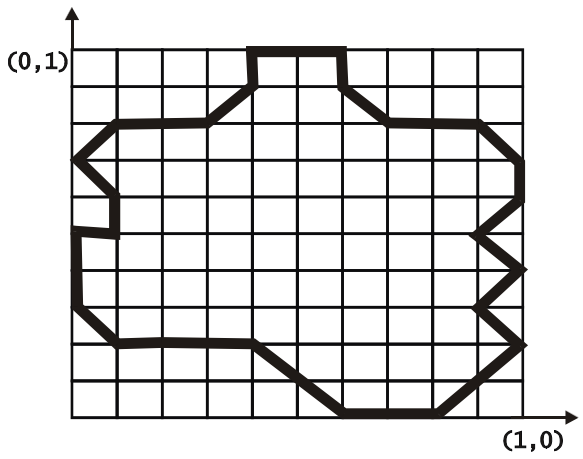
г) $h = 0{,}2$, Г – граница области $y \geq x - 2$, $y \geq x + 2$ $y \leq \sqrt{2x - x^2} + 2$;

д) $h = 1$, Г – граница области $x \geq y^2$, $x \leq 20 - y^2$, $y \geq 0$;

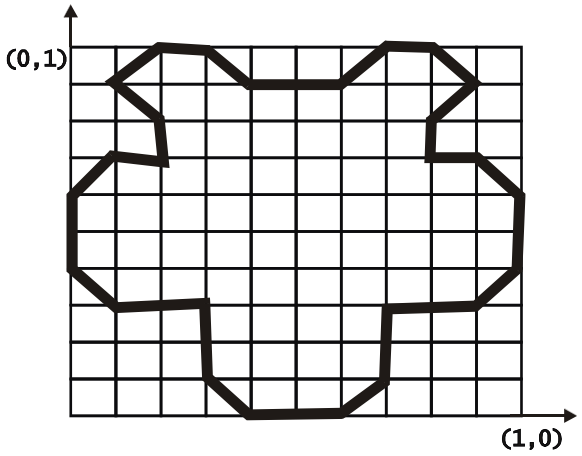
е)



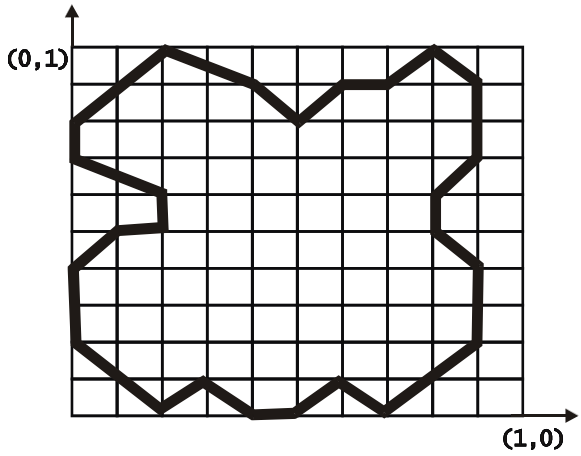
ж)



з)



и)



II. «Способ» блуждания. Из любой точки (x, y) области на шаге блуждания можно перейти:

а) в одну из четырех соседних точек $(x + h, y)$, $(x – h, y)$, $(x, y + h)$, $(x, y – h)$;

б) в одну из четырех соседних точек $(x + h, y + h)$, $(x – h, y + h)$, $(x + h, y – h)$, $(x – h, y – h)$;

в) в одну из восьми соседних точек $(x + h, y)$, $(x + h, y – h)$, $(x + h, y + h)$, $(x – h, y)$, $(x – h, y – h)$, $(x – h, y + h)$, $(x, y – h)$, $(x, y + h)$;

III. Способы выбора соседней точки при блуждании.

а) Задается булевский массив T_1, …, T_{11}. В начальный момент в него помещается любой набор булевских значений, кроме

$$T_1 = \ldots = T_{11} = false.$$

На каждом шаге блуждания производится преобразование элементов массива по закону

$l_1 := \neg (T_9 \equiv T_{11})$; $l_2 := \neg (T_8 \equiv T_{10}$; $l_3 := \neg (T_7 \equiv T_9)$;

$T_{11} := T_9$; $T_{10} := T_7$; . . .; $T_4 := T_1$; $T_3 := l_1$; $T_2 := l_2$; $T_1 := l_3$.

Направление движения определяется после этого значениями элементов T_{10}, T_{11} (в случае II а, б) или T_9, T_{10}, T_{11} (случай II в). Новое блуждание из точки (x^*, y^*) начинается с тех значений массива T, которые оказались после окончания предыдущего блуждания.

б) Определим последовательность чисел z_0, z_1, z_2, ... (когда число z_0 задано), которая образуется по закону

$$z_{i+1} = \{(z_i + 0{,}714983)^2 \cdot 1024\},$$

где $\{a\}$ обозначает дробную часть числа a. Каждое число последовательности лежит в пределах от 0 до 1. Отрезок (0, 1) делится на четыре (или восемь) равных частей и направление i-го шага блуждания определяется той частью, в которую попадает число z_i.

в) Определим массив T_1, …, T_{12}, элементами которого служат целые числа 0 или 1 (в начальный момент в него помещается любой набор чисел). На каждом шаге блуждания производится преобразование массива: образуется целое число

$$t = \sum_{i=1}^{12} T_i \cdot 2^{i-1};$$

оно преобразуется в новое $u = 7t + 11$ (mod 4087), двоичное представление которого помещается в T. Направление движения определяется содержимым элементов T_3, T_7 (для II а, б) или T_2, T_7, T_{10} (для случая II в).

1.10 *Программирование игр*.

1^0) Составить программу, анализирующую шахматную позицию. Программа вводит произвольную позицию, которая содержит фигуры из заданного списка, и после анализа выдает на печать результат.

Исходные данные

I. Позиция содержит

а) короля, ферзя, две ладьи и коня белых, короля и пешки черных;

б) короля, ферзя, двух коней и слона белых, короля и пешки черных;

в) короля, ферзя, ладью, коня и пешку белых; короля и пешки черных;

г) короля, две ладьи, слона, две пешки белых; короля, пешки и ладью черных;

д) короля, ладью, двух слонов, три пешки белых; короля, пешку, двух коней черных.

II. Анализ позиции состоит в том, чтобы выяснить

а) находится ли король черных под шахом;

б) объявлен ли в этой позиции мат черному королю;

в) могут ли белые одним ходом бить черную пешку.

2^0) Составить программу, имитирующую игру в крестики и нолики игроков: умного и играющего наугад. Программа сообщает ход поединка и результат игры. Умный игрок не должен проигрывать партнеру.

Исходные данные

I. Программа составляется для игры, в которой первый ход всегда делает

а) умный игрок,

б) играющий наугад.

II. Играющий наугад выбирает поле случайно для установки в него своего значка (× или 0). Способ выбора поля – аналогичен выбору соседней точки при блуждании в предыдущей задаче.

3^0) Составить программу, имитирующую игру в «морской бой». Игра ведется в одну сторону: результатом игры является количество ударов, произведенных по уничтожению всех кораблей. Программа выводит всю последовательность ударов и общее количество. Игра должна вестись в «обычной манере»: при попадании в корабль стремиться его уничтожить; не бить в поля, прилегающие к сбитым кораблям; вести учет сбитым и т.д. Корабли только линейные и не касаются друг друга.

Исходные данные

I. а) Поле игры 8×8. На поле 1 «трехпалубный», два «двухпалубных» и три «однопалубных»корабля.

б) Поле игры 10×10. На поле 1 «четырехпалубный», 2 «трехпалубных», 3 «двухпалубных» и 4 «однопалубных» корабля.

в) Поле игры 12×12. На поле 1 «пятипалубный», 2 «трехпалубных» и 5 «однопалубных» корабля.

II. Способ выбора клетки поля для удара – случайный (аналогично способу блуждания в предыдущей задаче).

4^0) Составить программу поиска выхода из лабиринта. Лабиринт предполагается прямоугольным, т.е. совокупностью комнат с 4 стенами, некоторые из которых глухие, а некоторые с дверьми. Начальная ситуация – «игрок» помещается в какую-нибудь комнату, затем он начинает ходить по лабиринту и искать выход «на волю».

Алгоритм поиска состоит в следующем. Игрок помечает те комнаты и двери, которые он проходит. Если он пришел в некоторую новую комнату, то он ее помечает и выбирает одну из дверей в другие комнаты: если таких дверей не находит, то возвращается обратно, «заколачивая» при этом пройденную дверь; если дверь есть, игрок помечает ее и идет через нее в соседнюю комнату (или «на волю»). Если игрок пришел в помеченную комнату, то он возвращается через только что пройденную дверь, «заколачивая» ее. После любого «заколачивания» игрок пытается найти «незаколоченную» и непомеченную дверь: если находит, то двигается через нее, помечая; если не находит, то через помеченную дверь возвращается, «заколачивая» пройденную второй раз дверь.