Лабораторная работа МОДЕЛИ ВЫЧИСЛИТЕЛЬНЫХ АЛГОРИТМОВ

Цель работы:

- изучение технологии математического моделирования вычислительных алгоритмов;
- моделирование вычислительного алгоритма для оценки его трудоемкости;
- реализация математической модели на ЭВМ.

1.Теоретическая часть.

Вычислительный процесс, реализуемый в информационной системе, представляется в виде алгоритма - правила, определяющего последовательность действий над исходными данными, приводящего к получению искомых результатов. Реализация алгоритма состоит из последовательности периодов обработки информации и обращения к файлам, которые следуют в порядке, определяемом программой. Длительность вышеназванных периодов определяется трудоемкостью работ - количеством операций, выполняемых процессором; количеством данных, передаваемых при обращении к файлам. При этом должно быть оговорено быстродействие устройств системы, используемых для выполнения соответствующих работ. Таким образом, трудоемкость - одна из основных характеристик алгоритма, обусловленная его вычислительной сложностью и определяющая его потребность в ресурсах системы.

Точная оценка трудоемкости алгоритма обычно апостериорна и возможна после получения машинной программы. Потому, в целях прогноза требуемых ресурсов обычно используют априорные приближенные оценки, которые находятся путем анализа задачи, порождающей вычислительный процесс.

Будем под трудоемкостью алгоритма понимать количество вычислительной работы, требуемой для реализации алгоритма. В таком случае оценкой трудоемкости может быть количество операций, выполняемых во

время процессорной обработки и вводе/выводе данных в процессе реализации алгоритма. Если при этом известна производительность устройств обработки и ввода/вывода, то можно получить интегрированную временную оценку трудоемкости.

Каждая реализация алгоритма случайна вследствие того, что исходные данные представляют собой, в общем случае, случайную выборку из множества исходных данных, к которым применим алгоритм. Поэтому полная характеристика трудоемкости предполагает описание количества операций, выполняемых за одну реализацию алгоритма, случайными величинами в форме закона распределения числа операций в реализации. Получение таких сведений - чрезвычайно сложный процесс. В практических целях трудоемкость алгоритма обычно оценивают на уровне средних оценок, например, математическими ожиданиями числа выполняемых операций.

Определим следующую совокупность параметров, используемых для определения трудоемкости алгоритма:

θ - среднее количество процессорных операций, выполняемых за одну реализацию алгоритма (при одном прогоне программы);

$N_1,...,N_H$ - среднее количество обращений к файлам $F_1,...,F_H$, соответственно, за одну реализацию алгоритма;

$\theta_1,...,\theta_H$ - среднее количество данных (байтов), передаваемое за одно обращение к файлам $F_1,...,F_H$, соответственно.

Значение θ характеризует трудоемкость обработки данных (счета), а значения $N_1,...,N_H$, $\theta_1,...,\theta_H$ - трудоемкость процесса ввода/вывода данных.

Математическая модель вычислительного процесса должна отражать его свойства, существенные, в данном случае, для определения трудоемкости.

Будем рассматривать вычислительный процесс как последовательность этапов счета и ввода/вывода данных при обращении к файлам $F_1,...,F_H$. Состояние

процесса, соответствующее этапу счета, обозначим S_0, а состояния, соответствующие обращениям к файлам $F_1,...,F_H$ - символами $S_1,...,S_H$. Кроме того, окончание вычислительного процесса рассматривается как переход процесса в состояние S_{H+1}, поглощающее вычислительный процесс.

Переходя во временное пространство, вычислительный процесс можно описать последовательностью состояний $S_{t_0},S_{t_1},...,S_{t_M}$, в которые процесс попадает в моменты времени $t_0,t_1,...,t_M$, причем $S_{t_i} \in \{S_0,S_1,...,S_H\}$, $i=\overline{0,M-1}$; а $S_{t_M}=S_{H+1}$.

Марковская модель вычислительного процесса

Принимая допущение об отсутствии последействия, получают наиболее простую модель вычислительного процесса - марковскую модель, описывающую его как марковский случайный процесс, определяемый множеством состояний $\{S_0,...,S_{H+1}\}$, матрицей вероятностей переходов $P = \left[p_{ij}\right]$, $i,j = \overline{0,H+1}$, и распределением вероятностей $(a_0,...,a_{H+1})$ состояний $S_1,...,S_H,S_{H+1}$ в момент времени $t = 0$.

Элементы p_{ij} матрицы P определяют вероятности перехода процесса из состояния S_i в состояние S_j. Матрица P - стохастическая, для которой $\sum_j p_{ij}=1$.

Вероятности a_j определяют первое возможное состояние процесса - S_{t_0}.

Примем следующую концептуальную модель вычислительного процесса: процесс начинается с состояния S_0, т.е. программа начинает выполняться с этапа счета. Этап ввода/вывода может быть инициирован только процессором, т.е. следовать только за этапом счета. Это одновременно означает, что после

каждого этапа ввода/вывода следует этап счета. В таком случае вероятности начальных состояний и матрица вероятностей переходов примут вид:

$$(a_0, a_1, a_2, ..., a_{H+1}) = (1\ ,\ 0\ ,\ 0\ ,\ ...\ ,\ 0\ );$$

$$P = \| p_{ij} \| = \begin{vmatrix} 0 & p_{0,1} & p_{0,2} & \cdots & p_{0,H} & p_{0,H+1} \\ 1 & 0 & 0 & ... & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & 0 & ... & 0 & 0 \\ 0 & 0 & 0 & ... & 0 & 1 \end{vmatrix}, \quad i, j = \overline{0,H+1}\ .$$

Значения вероятностей $p_{0,1},...,p_{0,H+1}$ предопределяют ход вычислительного процесса и зависят от параметров трудоемкости алгоритма. Эти значения находятся из следующих рассуждений.

Трудоемкость алгоритма определяется, в частности, средним числом $N_1,...,N_H$ обращений к файлам $F_1,...,F_H$. Тогда среднее число переходов из состояния S_0 в состояния $S_1,...,S_H$ должно быть $N_1 + N_2 + \cdots + N_H$. Один раз процесс переходит из состояния S_0 в поглощающее состояние S_{H+1}. Поэтому среднее число этапов счета в реализации алгоритма будет равно

$$N = \sum_{h=1}^{H} N_h + 1\ .$$

Значение $p_{0,h}$ определяет долю переходов в состояние S_h по отношению ко всевозможным переходам из состояния S_0 в состояния $S_1,...,S_H,S_{H+1}$. Следовательно,

$$p_{0,h} = \frac{N_h}{N} \quad \text{и} \quad p_{0,H+1} = \frac{1}{N}.$$

Трудоемкость каждого этапа будем рассматривать как случайную величину T_h с математическим ожиданием θ_h , h = 0 , 1 , ... , H . В частности, средняя трудоемкость этапа счета есть величина

$$\theta_0 = \frac{\theta}{N}\ ,$$

где θ - среднее количество процессорных операций, выполняемых за одну реализацию алгоритма.

Таким образом, моделью вычислительного процесса является марковская цепь с (H+2) состояниями, начальным состоянием S_0 и матрицей вероятностей переходов - P. Реализация вычислительного процесса - случайная последовательность состояний $S_{t_0}, S_{t_1}, S_{t_2}, ..., S_{t_M}$, изменение которых происходит в соответствии с матрицей вероятностей переходов.

С состояниями $S_0, S_1, ..., S_H$ связано определенное количество работы $\theta_0, \theta_1, ..., \theta_H$, характеризуемое значениями случайных величин $T_0, T_1, ..., T_H$, соответственно. Значение $T_h^{(j)}$ можно рассматривать как j -ое значение случайной величины T_h (h = 0 , ... , H), математическим ожиданием которой является θ_h.

Оценка трудоемкости алгоритмов методом теории марковских цепей

Все операторы алгоритма подразделяются на функциональные, перехода, ввода/вывода.

Функциональный оператор - задает совокупность вычислительных операций.

Оператор перехода - задает правило выбора одного из возможных путей развития вычислительного процесса, соответствующего текущим значениям данных, отношения между которыми представляются предикатами.

Оператор ввода/вывода - задает обращение к определенному файлу с целью передачи некоторого количества данных.

Первые два типа операторов задают совокупность вычислительных операций над данными и относятся к классу операторов, называемых основными.

Совокупность операторов алгоритма и связей между ними наглядно представляются графом алгоритма, вершины которого соответствуют

операторам алгоритма, а дуги отображают связи между операторами. Среди вершин графа выделяют начальную, конечную и операторные.

Граф алгоритма является корректным, если:

- имеются только одна начальная и только одна конечная вершины;
- для каждой вершины (кроме начальной) существует по крайней мере один путь, ведущий в эту вершину из начальной;
- для каждой вершины (кроме конечной) существует по крайней мере один путь, ведущий из этой вершины в конечную;
- разные выходы из одной вершины ведут к разным вершинам;
- при любых значениях логических условий (предикатов) существует путь из начальной вершины в конечную, причем любому фиксированному набору значений условий соответствует только один такой путь.

Номера вершин графа обозначим 0,1, ... ,k , где 0 - начальная, k - конечная вершина графа. Номера 1,2, ... ,k-1 идентифицируют операторы алгоритма.

Таким образом, граф алгоритма дает наглядное представление структуры алгоритма, определяя множество операторов $V = \{v_1, \dots, v_{k-1}\}$ и дуг $D = \{(i, j)\}$; $i = 0,...,k-1$; $j = 1,...,k$, связывающих операторы.

Для оценки трудоемкости алгоритма обозначим множество основных операторов: $S_0 = \{v_{\alpha_1}, ..., v_{\alpha_{m_0}}\}$, где $\alpha \in \{1,2,...,k-1\}$; $m_0 \in I = \overline{1, k-1}$.

Множество операторов ввода/вывода: $S_h = \{v_{\beta_1}, ..., v_{\beta_{m_h}}\}$, где $\beta \in \{1,2,...,k-1\}$; $m_h \in I = \overline{1, H}$. Каждый из операторов $v_{\beta_{m_h}}$ задает обращение к файлу F_h.

Обозначим k_α - среднее количество операций, составляющих оператор v_α;

ℓ_β - среднее количество данных, передаваемых при выполнении оператора v_β.

Переходы между операторами v_i и v_j рассматриваем как случайные события и характеризуем вероятностями p_{ij}, т.е. каждая дуга (i, j) графа алгоритма помечается числом p_{ij}.

Так как вычислительный процесс не может приостановиться в вершине v_i, то с вероятностью, равной 1, произойдет переход к какой-либо вершине графа алгоритма. Поэтому вероятности переходов должны отвечать условию:

$$\sum_{j=1}^{k} p_{ij} = 1, \qquad i = 0,1, \ldots ,k-1 .$$

Если за оператором i непременно выполняется оператор j, то $p_{ij}=1$.

Пусть $n_1, \ldots, n_{k-1}$ - среднее число обращений к операторам $v_1, \ldots, v_{k-1}$ за один прогон алгоритма. Тогда трудоемкость алгоритма характеризуется следующим набором соотношений.

- Среднее число процессорных операций, выполняемых при одном прогоне алгоритма (трудоемкость алгоритма по основным операторам):

$$\theta_{OCH} = \sum_{v_i \in S_0} n_i \cdot k_i \ . \qquad (1)$$

- Среднее число обращений к файлу F_h при одном прогоне алгоритма:

$$N_h = \sum_{v_i \in S_h} n_i , \qquad h = 1, \ldots ,H . \qquad (2)$$

- Среднее количество данных, передаваемое при одном обращении к файлу F_h:

$$\theta_h = \frac{1}{N_h} \sum_{v_i \in S_h} n_i \cdot \ell_i , \qquad h =1, \ldots ,H . \qquad (3)$$

- Среднее количество данных, передаваемых при обращениях к файлу F_h за один прогон алгоритма (трудоемкость алгоритма по h - вводу/выводу):

$$\theta_{B/B}^{(h)} = \theta_h \cdot N_h = \sum_{v_i \in S_h} n_i \cdot \ell_i , \qquad h =1, \ldots ,H . \qquad (4)$$

- Среднее количество данных, передаваемых при обращениях к файлам за один прогон алгоритма (трудоемкость алгоритма по вводу/выводу):

$$\theta_{B/B} = \sum_{h} \theta_{B/B}^{(h)} , \qquad h = 1, \ldots, H . \qquad (5)$$

В приведенных выражениях суммирование выполняется по всем вершинам графа, относящимся к классу основных операторов - S_0 или к классу операторов ввода/вывода - S_h, обращающихся к файлу F_h.

Таким образом, для оценки трудоемкости алгоритма необходимо определить среднее число обращений $n_1, \ldots, n_{k-1}$ к операторам. Для этого примем ряд допущений, идеализирующих модель.

Будем считать, что p_{ij} постоянны и после выполнения оператора v_i (i =1,...,k-1) переход к следующему оператору определяется только распределением вероятностей p_{ij}, т.е. не зависит от хода вычислительного процесса в прошлом - до перехода к оператору v_i. В таком случае процесс выполнения алгоритма можно считать марковским процессом с k состояниями $S_1,...,S_k$, соответствующими пребыванию процесса в вершинах $v_1,...,v_k$ графа алгоритма.

Состояния $S_1,...,S_{k-1}$ - невозвратные. Состояние S_k - поглощающее. Начальным является состояние S_i, определенное дугой (0, i), выходящей из вершины v_0 графа. Для упрощения обозначений примем i = 1, т.е. начальным состоянием процесса является состояние S_1, соответствующее вершине v_1 (всегда можно перенумеровать вершины графа так, чтобы дуга (0, i) стала дугой (0, 1)). В результате принятых условий, граф алгоритма можно рассматривать в качестве графа марковской цепи.

Среднее число $n_1, \ldots, n_{k-1}$ пребываний марковского процесса в невозвратных состояниях $S_1,...,S_{k-1}$ (т.е. искомое среднее число $n_1, \ldots, n_{k-1}$

обращений к операторам $v_1,...,v_{k-1}$) определяется корнями системы линейных алгебраических уравнений:

$$n_i = \delta_{1i} + \sum_{j=1}^{k-1} p_{ji} \cdot n_j \ , \qquad i = 1, \ldots ,k\text{-}1 \ .$$

Здесь δ_{1i} - символ Кронекера : $\delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$.

К вышеприведенной записи приводят следующие рассуждения. Значение n_i будет равно, по крайней мере, 1, если процесс начинается из состояния с номером i = 1, что определяется значением $\delta_{11} = 1$. В остальных случаях процесс попадает в состояние S_i только из какого-либо другого состояния S_j, j = 1,...,k-1 с вероятностью p_{ji}.

Если процесс находится в состоянии j n_j раз и $p_{ji} \neq 0$, то из этого состояния он попадает в состояние i в среднем $p_{ji} \cdot n_j$ раз. Суммированием значений $p_{ji} \cdot n_j$ по всем j находится число попаданий процесса в состояние i из всех других состояний j.

Каноническая запись системы уравнений имеет вид:

$$\begin{cases} n_1 = 1 & +p_{11}n_1 & +p_{21}n_2 & +\ldots & +p_{k-1,1}n_{k-1} \\ n_2 = & p_{12}n_1 & +p_{22}n_2 & +\ldots & +p_{k-1,2}n_{k-1} \\ \ldots & & \cdots & & \\ n_{k-1} = & p_{1,k-1}n_1 & +p_{2,k-1}n_2 & +\ldots & +p_{k-1,k-1}n_{k-1} \end{cases} .$$

После преобразований получим:

$$\begin{cases} (p_{11}-1)n_1 + p_{21}n_2 + \ldots + p_{k-1,1}n_{k-1} = -1 \\ p_{12}n_1 + (p_{22}-1)n_2 + \ldots + p_{k-1,2}n_{k-1} = 0 \\ \ldots \\ p_{1,k-1}n_1 + p_{2,k-1}n_2 + \ldots + (p_{k-1,k-1}-1)n_{k-1} = 0 \end{cases}.$$

Пример оценки трудоемкости алгоритма по методу марковских цепей

Пусть для некоторой конкретной реализации граф алгоритма имеет вид:

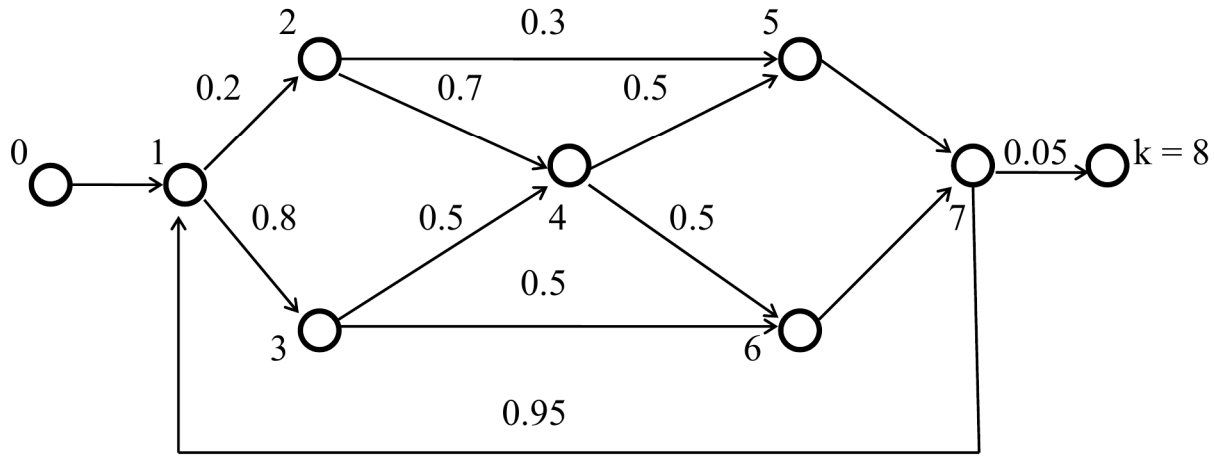

Рис.1.

Имеем систему из семи линейных алгебраических уравнений:

$$\begin{cases} -n_1 & & & & & & +0.95n_7 & = -1 \\ 0.2n_1 & -n_2 & & & & & & = 0 \\ 0.8n_1 & & -n_3 & & & & & = 0 \\ & 0.7n_2 & +0.5n_3 & -n_4 & & & & = 0 \\ & 0.3n_2 & & +0.5n_4 & -n_5 & & & = 0 \\ & & 0.5n_3 & +0.5n_4 & & -n_6 & & = 0 \\ & & & & n_5 & +n_6 & -n_7 & = 0 \end{cases}.$$

Решение системы уравнений определяет среднее число попаданий вычислительного процесса в состояния $S_1,\ldots,S_7$:

$n_1 = 20,\quad n_3 = 16,\quad n_5 = 6.6,\quad n_7 = 20.$

$n_2 = 4,\quad n_4 = 10.8,\quad n_6 = 13.4,$

• Пусть все операторы алгоритма - основные, а количество операций - k_i, порождаемых оператором v_i, постоянно и равно 1. Тогда трудоемкость алгоритма будет равна $\theta_{ОСН} = \sum_{i=1}^{7} k_i n_i = 20 + 4 + ... + 20 = 90.8$ операций.

• • Если, к примеру, операторы v_4 и v_7 являются операторами ввода/вывода, а количество операций, порождаемых каждым из основных операторов, равно $k_1 = 500$, $k_2 = 500$, $k_3 = 50$, $k_5 = 300$, $k_6 = 20$, соответственно, то трудоемкость по основным операторам составит величину

$\theta_{ОСН}$ = 500*20 + 500*4 + 50*16 + 300*6.6 + 20*13.4 = 15 048 операций.

Пусть при обращении из оператора v_4 к файлу F_4 передается $\ell_4 = 1800$ байт данных, а при обращении из оператора v_7 к файлу F_7 передается ℓ_7 = 2540 байт. Тогда трудоемкость по вводу/выводу определяется, как $\theta_{В/В} = \sum_{4,7} \theta_{В/В}^{(h)} = \theta_{В/В}^{(4)} + \theta_{В/В}^{(7)}$. В свою очередь: $\theta_{В/В}^{(4)} = \sum_{v_4 \in S_4} n_i \ell_i = n_4 \ell_4 = 10.8*1800 = 19440$ байт.

Аналогично, $\theta_{В/В}^{(7)} = n_7 \ell_7 = 20*2540 = 50800$ байт. Тогда $\theta_{В/В}$ = 19440 + 50800 = 70240 байт.

Интегрированная оценка трудоемкости в этом случае составляет величину:

$$\theta = \theta_{ОСН} / B_{ОСН} + \theta_{В/В} / B_{В/В} \quad [\text{ед.времени}] . \qquad (6)$$

Здесь $B_{ОСН}$ - быстродействие процессора по основным операциям; $B_{В/В}$ - быстродействие соответствующих устройств ввода/вывода.

• • • Если операторы v_4 и v_7 обращаются к одному и тому же файлу F_{47}, то в этом случае трудоемкость по вводу/выводу определяется, как $\theta_{В/В} = \theta_{В/В}^{(47)}$.

Сразу находим: $\theta_{В/В}^{(47)} = \sum_{v_4, v_7 \in S_{47}} n_i \ell_i = n_4 \ell_4 + n_7 \ell_7 = 19440 + 50800 = 70240$ байт. Как видно, результат не отличается от предыдущего.

Оценка трудоемкости алгоритмов сетевым методом

Количество вычислений, проводимых при расчете трудоемкости алгоритмов, можно значительно сократить, если использовать сетевой подход к анализу трудоемкости. Однако, он применим к графам алгоритмов, несодержащих циклы. Поэтому, вначале рассматривается граф алгоритма без циклов, после чего метод обобщается на алгоритм с циклами. Это производится путем преобразования графа алгоритма с циклами в граф с эквивалентной трудоемкостью, но без циклов.

Для применения сетевого метода к оценке трудоемкости алгоритма вершины графа должны быть перенумерованы в порядке их следования: любая вершина должна иметь номер, больший любого номера предшествующих ей вершин. Нумерация вершин производится следующим образом.

Начальной вершине присваивается номер 0. Очередной номер i = 1 ,2 , ... присваивается вершине, в которую входят дуги от уже пронумерованных вершин с номерами, меньшими i. При этом, любым двум вершинам должны соответствовать разные номера. Такой порядок нумерации является результативным для любого графа без циклов.

Поскольку граф не содержит циклов, то $p_{ji} = 0$ для всех $j \geq i$. Тогда, при прогоне алгоритма, вершина 1 будет выполнена точно один раз, т.е. $n_1 = 1$.

Среднее число попаданий вычислительного процесса в вершину i будет равно:

$$n_i = \sum_{j=1}^{k-1} p_{ji} n_j \ , \quad i = 2,3, \ldots ,k\text{-}1 \ .$$

При описанном выше порядке нумерации вершин графа, на момент вычисления n_i значения $n_1,\ldots,n_{i-1}$ будут уже определены. Очевидно, суммирование следует проводить только для $j < i$, поскольку $p_{ji} = 0$ для $j \geq i$.

В качестве примера рассмотрим граф на рис.1, но исключим из рассмотрения дугу (7,1). В таком случае имеем граф без циклов,

удовлетворяющий порядку нумерации вершин. Среднее число обращений $n_1, n_2, ..., n_7$ к операторам алгоритма будет равно:

$n_1 = 1;$ $n_2 = p_{12}n_1 = 0.2*1 = 0.2;$ $n_3 = p_{13}n_1 = 0.8*1 = 0.8;$

$n_4 = p_{24}n_2 + p_{34}n_3 = 0.7*0.2 + 0.5*0.8 = 0.54;$

$n_5 = p_{25}n_2 + p_{45}n_4 = 0.3*0.2 + 0.5*0.54 = 0.33;$

$n_6 = p_{36}n_3 + p_{46}n_4 = 0.5*0.8 + 0.5*0.54 = 0.67;$

$n_7 = n_5 + n_6 = 0.33 + 0.67 = 1.$

Теперь рассмотрим случай алгоритма, содержащего циклы. Задача состоит в том, чтобы исключить циклы, заменив их операторами с эквивалентной трудоемкостью.

Все циклы делятся на ранги. К рангу 1 относятся циклы, несодержащие внутри себя ни одного цикла, к рангу 2 - циклы, содержащие в себе циклы ранга 1 и т.д. Для графа на рис.1 с учетом дуги (7,1) имеем цикл ранга 1.

Совокупность операторов, входящих в цикл, и связывающих их дуг, за исключением дуги, замыкающей цикл, называют телом цикла. Тело цикла ранга 1 является графом без циклов. Применяя к этому графу вышеописанную методику, можно определить значения n_i для операторов тела цикла. Тогда трудоемкость тела цикла С может быть определена как $\sum_{v_j \in C} k_j n_j$, где k_j - трудоемкость j-ого оператора тела цикла; суммирование ведется по всем вершинам v_j, содержащимся в цикле С.

Обозначим n_C - среднее число повторений цикла, равное числу выполнений тела цикла при одном прогоне алгоритма. Если вероятность перехода по дуге, замыкающей цикл, равна p_{kl}, тогда

$$n_C = 1 + p_{kl}n_C .$$

В этом выражении второе слагаемое представляет собой среднее число повторных выполнений тела цикла, которое можно трактовать как долю от общего числа выполнений цикла - n_C, обусловленную вероятностью p_{kl}.

Отсюда получим, что

$$n_C = \frac{1}{1 - p_{kl}} \quad . \tag{7}$$

Тогда средняя трудоемкость цикла равна

$$k_C = n_C \sum_{v_j \in C} k_j n_j \, . \tag{8}$$

Теперь цикл С можно заменить оператором, имеющим трудоемкость k_C. Если граф содержит циклы, ранг которых выше 1, то последовательное применение процедуры эквивалентной замены приводит к графу без циклов, трудоемкость которого находится вышеописанным способом.

Отметим, что структура формулы (8) является обобщенной в том смысле, что дает возможность рассчитывать трудоемкость как по основным операторам тела цикла, так и по операторам ввода/вывода, используя соотношения (1), (4) и (5) в “циклической” форме:

$$k_{C\ OCH} = n_C \sum_{v_i \in S_0} n_i \cdot k_i \ ; \tag{1a}$$

$$\theta^{(h)}_{C\ B/B} = n_C \sum_{v_i \in S_h} n_i \cdot \ell_i \ ; \tag{4a}$$

$$\theta_{C\ B/B} = \sum_h \theta^{(h)}_{C\ B/B} \quad . \tag{5a}$$

Для рассмотренного выше примера, используя средние числа $n_1, \ldots, n_7$ обращений к операторам, полученные на основе сетевого метода, находим:

- $k_{C\ OCH} = \frac{1}{1 - 0.95} \sum_{i=1}^{7} n_i k_i = 20(1 + 0.2 + 0.8 + 0.54 + 0.33 + 0.67 + 1) = 90.8$ опер.

• • $k_{C\ \ ОСН} = \frac{1}{1-0.95} \sum_{v_i \in S_0} n_i k_i = 20(1*500 + 0.2*500 + 0.8*50 + 0.33*300 + 0.67*20) = 15048$ опер.

$$\theta_{C\ \ В/В}^{(4)} = \frac{1}{1-0.95} \sum_{v_4 \in S_4} n_i \cdot \ell_i = 20*0.54*1800 = 19440 \text{ байт.}$$

$$\theta_{C\ \ В/В}^{(7)} = \frac{1}{1-0.95} \sum_{v_7 \in S_7} n_i \cdot \ell_i = 20*1*2540 = 50800 \text{ байт.}$$

$$\theta_{C\ \ В/В} = \sum_{4,7} \theta_{C\ \ В/В}^{(h)} = 19440 + 50800 = 70240 \text{ байт.}$$

• • • $\theta_{C\ \ В/В} = \theta_{C\ \ В/В}^{(47)} = \frac{1}{1-0.95} \sum_{v_4, v_7 \in S_{47}} n_i \ell_i = 20(0.54*1800 + 1*2540) = 70240$ байт.

Таким образом, трудоемкость цикла в данном примере одновременно определяет трудоемкость алгоритма, поскольку граф алгоритма, по сути, является телом единственного цикла. Сравнивая полученные результаты с соответствующими результатами, полученными при использовании метода теории марковских цепей, видно, что результаты тождественны.

2. Порядок выполнения лабораторной работы

• Ознакомиться с содержанием лабораторной работы.

• Построить по таблице 1, в соответствии с вариантом задания, граф алгоритма.

• Построить математическую модель вычислительного процесса для оценки трудоемкости алгоритма по методу теории марковских цепей.

• Построить математическую модель вычислительного процесса для оценки трудоемкости алгоритма сетевым методом.

• Подготовить программу для расчета модельных характеристик трудоемкости на одном из языков высокого уровня

• Отладить программу и получить результаты расчетов.

• Подготовить в объектно-ориентированной среде разработки интерактивную форму для управления работой программы и визуализации полученных результатов.

• Подготовить отчет о выполненной работе.

3. Варианты заданий на выполнение лабораторной работы

А. Вариант графа алгоритма. Таблица 1.

Вариант	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
P_{12}	0.2	0.1	1	0.5	1	1	0.3	1	0.2	0.3	0.1	1	1	0.2
P_{13}	0.2	0.3		0.5			0.7		0.4	0.7	0.1			0.8
P_{14}	0.6	0.6							0.4		0.8			
P_{23}			0.1		0.3	0.6		0.2				0.2	0.5	
P_{24}			0.3	0.4	0.7	0.4	0.5	0.3		0.7		0.8	0.5	0.3
P_{25}	1	1	0.6	0.6			0.5	0.5	1	0.3	1			07
P_{34}					1							1		
P_{35}	0.1	0.3		0.3		0.1	0.1		0.1	0.6	0.2		0.2	0.2
P_{36}	0.9	0.3	1			0.9	0.2	1	0.2		0.8		0.8	0.2
P_{37}		0.4		0.7			0.7		0.7	0.4				0.6
P_{45}					0.5							0.3		
P_{46}	1		1	1	0.5	0.3	1	1		1	1	0.7	0.2	1
P_{47}		1				0.7			1				0.8	
P_{56}		1	1	1			1	1	1	1				1
P_{57}	1				1	1					1	1	1	
P_{61}			0.9					0.8						
P_{67}	1	1	0.1		1	1	1	0.2	1		1	1	1	1
P_{68}				1						1				
P_{71}	0.9	0.8					0.8		0.8		0.9			0.9
P_{72}					0.8							0.5		

P_{78}				1		1				1			1	
P_{81}				0.9		0.7				0.8			0.6	

Продолжение таблицы 1

Вариант	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
P_{12}	0.2	0.1	1	0.5	1	1	0.3	1	0.2	0.3	0.1	1	1	0.2
P_{13}	0.2	0.3		0.5			0.7		0.4	0.7	0.1			0.8
P_{14}	0.6	0.6							0.4		0.8			
P_{23}			0.1		0.3	0.6		0.2				0.2	0.5	
P_{24}			0.3	0.4	0.7	0.4	0.5	0.3		0.7		0.8	0.5	0.3
P_{25}	1	1	0.6	0.6			0.5	0.5	1	0.3	1			07
P_{34}					1							1		
P_{35}	0.1	0.3		0.3		0.1	0.1		0.1	0.6	0.2		0.2	0.2
P_{36}	0.9	0.3	1			0.9	0.2	1	0.2		0.8		0.8	0.2
P_{37}		0.4		0.7			0.7		0.7	0.4				0.6
P_{45}					0.5							0.3		
P_{46}	1		1	1	0.5	0.3	1	1		1	1	0.7	0.2	1
P_{47}		1				0.7			1				0.8	
P_{56}		1	1	1			1	1	1	1				1
P_{57}	1				1	1					1	1	1	
P_{61}			0.9				0.1	0.8			0.3			0.6
P_{67}	1	1	0.1		1	1	0.9	0.2	1		0.7	1	1	0.4
P_{68}				1						1				
P_{71}	0.8	0.6	0.7		0.3		0.8		0.8		0.9	0.2		0.9
P_{72}	0.1	0.2		0.3	0.5	0.2		0.7	0.1	0.6		0.5	0.2	
P_{78}				0.7		0.8				0.4			0.8	
P_{81}				0.9		0.7				0.8			0.6	

В. Тип и трудоемкость операторов. Таблица 2.

вариант	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	100	200	120	320	250	300	100	100	200	150	100	100	300	100
2	200	150	120	200	300	200	400	200	300	150	200	120	200	200
3	120	300	150	200	200	300	500	100	200	200	100	200	300	300
4	300	250	200	150	150	800	400	200	100	200	250	300	200	300
5	100	200	300	150	300	300	300	300	250	400	500	300	250	250
6	300	300	600	300	400	200	250	300	250	300	300	150	200	150
7	100	800	300	100	100	200	200	400	300	200	200	100	150	200
8	-	-	-	800	-	900	-	-	-	200	-	-	200	-

Продолжение таблицы 2

вариант	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
1	200	400	240	640	500	600	200	200	400	300	200	200	600	200
2	400	300	240	400	600	400	800	400	600	300	400	240	400	400
3	240	60	300	400	400	600	900	200	400	400	200	400	600	600
4	600	500	400	300	300	900	800	400	200	400	500	600	400	600
5	200	400	600	300	600	600	600	600	500	800	900	600	500	500
6	600	600	900	600	400	400	500	600	500	600	600	300	400	300
7	200	700	600	200	100	400	400	800	300	400	400	100	150	400
8	-	-	-	900	-	800	-	-	-	700	-	-	600	-

Примечание: Затемненная ячейка в таблице означает, что соответствующий оператор (строка таблицы) является оператором ввода/вывода.