ЛАБОРАТОРНАЯ РАБОТА № 5.
“ПОИСК КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ”

1 ЦЕЛЬ РАБОТЫ

Целью работы является изучение алгоритмов поиска кратчайших путей на графах на примере метода динамического программирования

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Основные определения

Путем (или *ориентированным маршрутом*) ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей. Так на рисунке 5 последовательности дуг $\mu_1=\{a_6,a_5, a_9, a_8, a_4\}$, $\mu_2=\{a_1,a_6, a_5, a_9\}$, $\mu_3=\{a_1,a_6, a_5, a_9, a_{10}, a_6, a_4\}$являются путями.

Ориентированной цепью (*орцепью*) называется такой путь, в котором каждая дуга используется не больше одного раза. Так, например, приведенные выше пути μ_1 и μ_2 являются орцепями, а путь μ_3 не является таким, поскольку дуга a_6 в нем используется дважды.

Маршрут есть неориентированный "двойник" пути, и это понятие рассматривается в тех случаях, когда можно пренебречь направленностью дуг в графе.

Таким образом, маршрут есть последовательность ребер a_1, a_2,...,a_q, в которой каждое ребро a_i, за исключением, возможно, первого и последнего ребер, связано с ребрами a_{i-1} и a_{i+1} своими двумя концевыми вершинами.

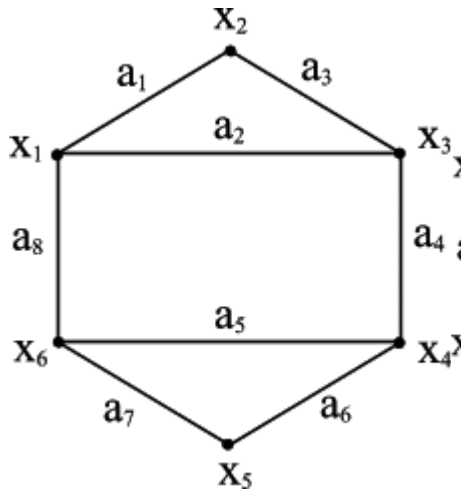

Рисунок 1 – Пример графа

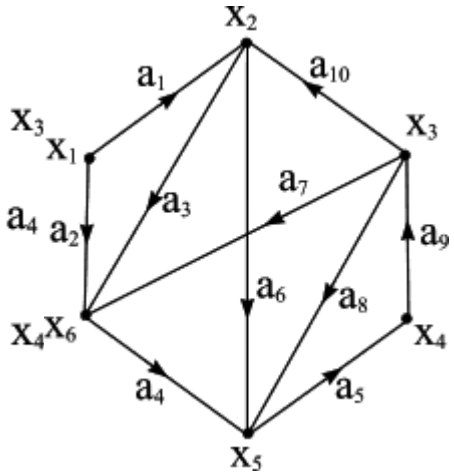

Рисунок 2 – Пример графа

Последовательности дуг на рисунке2

$\mu_4=\{a_2,a_4, a_8, a_{10}\}$, $\mu_5=\{a_2,a_7, a_8, a_4, a_3\}$ и$\mu_6=\{a_{10},a_7, a_4, a_8, a_7, a_2\}$

являются маршрутами.

Контуром (простой цепью) называется такой путь (маршрут), в котором каждая вершина используется не более одного раза. Например, путь μ_2 является контуром, а пути μ_1 и μ_3 - нет. Очевидно, что контур является также цепью, но обратное утверждение неверно. Например, путь μ_1 является цепью, но не контуром, путь μ_2 является цепью и контуром, а путь μ_3 не является ни цепью, ни контуром.

Аналогично определяется простая цепь в неориентированных графах. Так, например, маршрут μ_4 есть простая цепь, маршрут μ_5 - цепь, а маршрут μ_6 не является цепью.

Путь или маршрут можно изображать также последовательностью вершин. Например, путь μ_1 можно представить также: $\mu_1=\{x_2, x_5, x_4, x_3, x_5, x_6\}$ и такое представление часто оказывается более полезным в тех случаях, когда осуществляется поиск контуров или простых цепей.

Иногда дугам графа G сопоставляются (приписываются) числа - дуге (x_i,x_j) ставится в соответствие некоторое число c_{ij}, называемое *весом*, или *длиной*, или *стоимостью* (*ценой*) дуги. Тогда граф G называется *взвешенным*. Иногда веса (числа v_i) приписываются вершинам x_i графа.

При рассмотрении пути μ, представленного последовательностью дуг $(a_1, a_2,...,a_q)$, за его *вес* (или *длину*, или *стоимость*) принимается число $L(\mu)$, равное сумме весов всех дуг, входящих в μ, т. е.

$$L(\mu) = \sum_{(x_i,x_j)\in\mu} c_{ij} \qquad (1)$$

Таким образом, когда слова "длина", "стоимость", "цена" и "вес" применяются к дугам, то они эквивалентны по содержанию, и в каждом конкретном случае выбирается такое слово, которое ближе подходит по смыслу.

Длиной (или *мощностью*) пути и называется число дуг, входящих в него.

2.2 Задача о кратчайшем пути

Пусть дан граф $G=(X,\Gamma)$, дугам которого приписаны веса (стоимости), задаваемые матрицей$C=[c_{ij}]$. *Задача о кратчайшем пути* состоит в нахождении кратчайшего пути от заданной начальной вершины (*истока*) s до заданной конечной вершины (*стока*) t, при условии, что такой путь существует:

Найти $\mu(s,t)$ при$L(\mu)\to min$, $s,t\in X$, $t\in R(s)$,

где $R(s)$ - множество, достижимое из вершины s.

В общем случае элементы c_{ij} матрицы весовC могут быть положительными, отрицательными или нулями. Единственное ограничение состоит в том, чтобы в G не было циклов с суммарным отрицательным

весом. Отсюда следует, что дуги (ребра) графа G не должны иметь отрицательные веса.

Почти все методы, позволяющие решить задачу о кратчайшем (s-t)-пути, дают также (в процессе решения) и все кратчайшие пути от s к $x_i(\forall x_i \in X)$. Таким образом, они позволяют решить задачу с небольшими дополнительными вычислительными затратами.

Допускается, что матрица весовC не удовлетворяет условию треугольника, т. е. не обязательно $c_{ij} \leq c_{ik} + c_{kj}$ для всех i, j и k.

Если в графе G дуга (x_i, x_j) отсутствует, то ее вес полагается равным ∞.

Ряд задач, например, задачи нахождения в графах путей с максимальной надежностью и с максимальной пропускной способностью, связаны с задачей о кратчайшем пути, хотя в них характеристика пути (скажем, вес) является не суммой, а некоторой другой функцией характеристик (весов) дуг, образующих путь. Такие задачи можно переформулировать как задачи о кратчайшем пути и решать их соответствующим образом.

Существует множество методов решения данной задачи, отличающиеся областью применимости и трудоемкостью (Дейкстры, Флойда, динамического программирования). Среди них большое распространение получили частные алгоритмы, применяющиеся при решении частных задач, и имеющие меньшую трудоемкость. Эти частные случаи встречаются на практике довольно часто (например, когда c_{ij} являются расстояниями), так что рассмотрение этих специальных алгоритмов оправдано.

На практике задачу кратчайшего пути часто требуется решать для класса ориентированных ациклических графов. Такая задача успешно решается с помощью метода динамического программирования.

2.3.1 Алгоритм Дейкстры

Данный алгоритм предложил датский исследователь *Дейкстра* в 1959 году.

Пусть требуется найти кратчайшее расстояние из вершины s в вершину f. Изначально необходимо создать массив d[], в котором будет храниться для каждой вершины кратчайшее расстояние, за которое можно попасть в нее из исходной. При созданиид[s] = 0, а все остальные элементы равны бесконечности (на практике выбирают очень большое число). Кроме того, потребуется массив used[], чтобы хранить бит для каждой вершины, определяющий помечена она или еще нет. Изначально все вершины являются непомеченными.

Сам алгоритм состоит из n (n – количество вершин) итераций, на очередной итерации выбирается такая непомеченная вершина, что кратчайшее расстояние от исходной до нее минимально.То есть вершина v такая, что d[v] = min из всех вершин, которые еще не помечены (на первой итерации будет выбрана вершина s, что логично). А далее просматриваются

все ребра, выходящие из вершины v, и производятся так называемые *релаксации*. Таким образом, если есть ребро v, to с весом len, то будет предпринята попытка улучшить значение d[to] = min(d[to], d[v] + len). Действительно, выбирается минимальное значение между тем, что уже было определено и новым способом добраться до вершины to. На этом текущая итерация заканчивается. После окончания n итераций в массиве d будут лежать ответы - расстояния для каждой вершины. То есть в вершину f мы сможем добраться за d[f] единиц расстояния.

Стоит заметить, что если до некоторых вершин добраться невозможно, то значение d для них не изменится, то есть останется равным бесконечности (тому числу, которым мы инициализировали массив).

```
void solve() {
	// n - количество вершин
	// g[n][n] - матрица смежности, g[i][j] = 0, если текущего ребра нет
	// d[n] - массив ответов
	// s - стартовая вершина
	// used - массив для пометок вершин
	int i,j, // i-я итерация, j - для поиска минимальной
		v, // минимальная вершина
		to, // ребро из вершины v в to
		len; // длины len
	d[s] = 0;
	for(i = 0; i < n; i++) {
		v = -1;
		for(j = 0; j < n; j++)  // поиск вершины с минимальным d[j]
			if(!used[j] && (v == -1 || d[j] < d[v]))
				v = j;
		used[v] = true; // пометка вершины
		for(to = 0; to < n; to++) {
			if(g[v][to]) {
				len = g[v][to];
				if(d[v] + len < d[to]) {
					d[to] = d[v] + len;
				}
			}
		}
	}
}
```

2.3.2Метод динамического программирования

Прямая итерация. Пусть вершины пронумерованы так, что дуга(x_i,x_j)всегда ориентирована от вершины x_i к вершине x_j, имеющей

больший номер. Для ациклического графа такая нумерация всегда возможна и производится очень легко. При этом начальная вершина *s* получает номер *1*, а конечная *t* - номер *n*.

Пусть $\lambda(x_i)$– пометка вершины x_i, равная длине кратчайшего пути от *1* до x_j, *s*– начальная вершина (источник), *t*– конечная вершина (сток).

Шаг 1. Положить $\lambda(S) = 0$, $\lambda(x_i) = \infty$ для всех вершин $x_i \in X/s; i=1;$

Шаг 2. *i=i+1.* Присвоим вершине x_j пометку $\lambda(x_j)$, – равную длине кратчайшего пути от *1* до x_j, используя для этого соотношение

$$\lambda(x_j) = \min(\lambda(x_i) + c_{ij}) \quad (2)$$

Шаг 3. Повторить п.2. до тех пор, пока последняя вершина *n* не получит пометку $\lambda(t)$.

Необходимо отметить, что если вершина x_j помечена, то пометки $\lambda(x_i)$известны для всех вершин $x_i \in \Gamma^{-1}(x_j)$, так как в соответствии со способом нумерации это означает, что $x_i < x_j$ и, следовательно, вершины x_i уже помечены в процессе применения алгоритма.

Пометка $\lambda(t)$ равна длине самого короткого пути от *s* до *t*. Сами дуги, образующие путь, могут быть найдены способом последовательного возвращения. А именно дуга(x_i, x_j),согласно (2), принадлежит пути тогда и только тогда, когда

$$\lambda(x_j) = \lambda(x_i) + c_{ij} \quad (3)$$

Обратная итерация: начиная с вершины *t,* имеющей номер *n*, полагаем на каждом шаге x_jравной такой вершине (скажем, x_j^*), для которой выполняется соотношение (3), и так продолжаем до тех пор, пока не будет достигнута начальная вершина (т.е. пока не будет $x_j^* \equiv s$).

Совершенно очевидно, что пометка $\lambda(x_j)$ вершины x_jдает длину кратчайшего пути *μ* от *s* до x_j.

2.4 Алгоритмтопологической сортировки

В некоторых случаях исходный граф является ациклическим, но имеет неправильную нумерацию – содержит дуги(x_j, x_i),ориентированные от вершины x_j к вершине x_i, имеющей меньший номер (*j>i*). Для успешного нахождения кратчайшего пути с помощью метода динамического программирования к такому графу сначала применяется алгоритм топологической сортировки вершин.

Алгоритм топологической сортировки вершин очень простой. Он позволяет не только правильно перенумеровать вершины графа, но и определить его ацикличность.

Шаг 1. Положить *i=n*, где *n* – число вершин графа *G.*

Шаг 2. В графе определяется вершина x_k, для которой выполняется условие $|\Gamma(x_k)|=\varnothing$ (т.е., вершина, из которой не выходит ни одна дуга). Вершина x_kполучает порядковый номер *i*(перенумеруется) и исключается из

дальнейшего рассмотрения вместе со всеми входящими в нее инцидентными дугами. *i=i–1.*

Шаг 3. Повторять п.2. до тех пор, пока не будет выполнено одно из условий:

1)*i=1* – достигнута начальная вершина. Вершины графа получили правильную нумерацию.

2) Невозможно определить вершину, для которой выполнялось бы условие $|\Gamma(x_k)|=\varnothing$. В графе имеется цикл.

В последнем случае алгоритм динамического программирования неприменим. Для поиска кратчайших путей на таком графе необходимо использовать более эффективные методы, например, алгоритм Дейкстры.

2.5 Контрольный пример

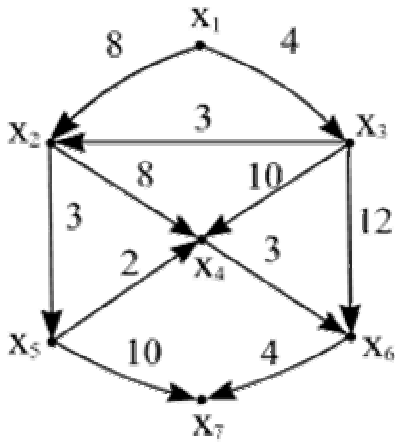

Рисунок 3 – Начальный вид графа

Для графа, изображенного на рисунке 3, определим кратчайший путь между вершинами x_1 и x_7, используя метод динамического программирования.

Так как граф содержит дуги (x_3 ,x_2) и (x_5 ,x_4), имеющие неправильную нумерацию (от большего к меньшему), необходимо перенумеровать вершины графа, применяя алгоритм топологической сортировки вершин.

В нашем случае из графа будут последовательно исключаться вершины x_7, x_6 , x_4, x_5, x_2, x_3, x_1. Соответственно, вершины графа получат новую нумерацию, и граф будет иметь вид, представленный на рисунке 8 (старая нумерация вершин сохранена в скобках).

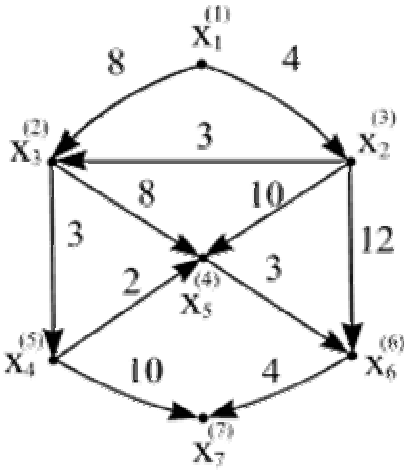

Рисунок 4 – Итог перенумерации

На первом шаге оценка $\lambda(x_1)=0$.Осуществляется переход к вершине x_2. Множество $\Gamma^{-1}(x_2)$ включает только одну вершину x_1.Следовательно, оценка для вершины x_2 определяемая по формуле (2), будет $\lambda(x_2)=min\{0+4\}=4$. Переходим к вершине x_3. Для вершины x_3 множество $\Gamma^{-1}(x_3)=\{x_1, x_2\}$. В этом случае, оценка будет выбираться как минимальная из двух возможных: $\lambda(x_3)=min\{0+8, 4+3\}=7$.Для вершины x_4 оценка определяется снова однозначно: $\lambda(x_4)=min\{7+3\}=10$. Однако после перехода к вершине x_5необходимо рассматривать сразу три входящих дуги (x_2, x_5), (x_3, x_5), (x_4, x_5).По формуле (2) определяется оценка для вершины x_5:

$\lambda(x_5)=min\{4+10, 7+8, 10+2\}=12.$

Далее, аналогичным образом вершина x_6 получает оценку $\lambda(x_6)=min\{4+12, 12+3\}=15$ и, наконец, вершина x_7получает оценку $\lambda(x_7)=min\{10+10, 15+4\}=19$.

Таким образом, конечная вершина x_7 пути $\mu(x_1,x_7)$достигнута, длина пути равна $L(\mu)=19.$

С помощью выражения (3) последовательно определяются вершины, которые входят в кратчайший путь. Перемещаясь от конечной вершины x_7, выбирается последовательность вершин, для которой выражение (3) принимает значение «истинно»: $x_6, x_5, x_4, x_3, x_2, x_1$, т.е. кратчайший путь проходит последовательно через все вершины графа.

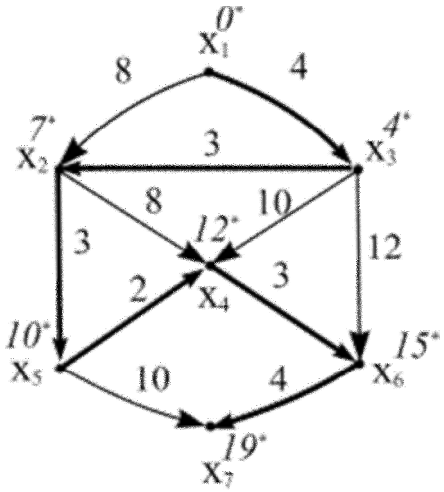

Рисунок 5 - Результат

Действительно, легко убедиться в истинности выражений:

$\lambda(x_7) = \lambda(x_6) + c_{67}$, (*19=15+4*);

$\lambda(x_6) = \lambda(x_5) + c_{56}$, (*15=12+3*);

$\lambda(x_5) = \lambda(x_4) + c_{45}$,(*12=10+2*);

$\lambda(x_4) = \lambda(x_3) + c_{34}$, (*10=7+3*);

$\lambda(x_3) = \lambda(x_2) + c_{23}$,(*7=4+3*);

$\lambda(x_2) = \lambda(x_1) + c_{12}$,(*4=0+4*);

После проведения перенумерации вершин графа на исходную, окончательно определяется кратчайший путь:

$\mu(x_1,x_7) = \{x_1,x_3,x_2,x_5,x_4,x_6,x_7\}$

Задача решена.

Результат решения в виде выделенного пути изображен на рисунке 5. Курсивом «со звездочкой» отмечены значения пометок вершин $\lambda(x_i)$.

2.6. Пример программы

```
constint INF = 1000000000; // бесконечность
bool used[100] = {0};   // массив для пометок
```

```
int top[100] = {0};     // топологический список
int g[100][100] = {0};  // матрица смежности
int n;                                  // количество вершин
int l;                                  // l-я вершина для добавления
int s;                                  // стартовая вершина
int f;                                  // конечная вершина
intd[100] = {0};              // массивответов

intdfs(intv) {
        if(used[v])
                return 0;
        used[v] = true;
        for(int to = 0; to < n; to++)
                if(g[v][to])
                        dfs(to);
        top[l++] = v; // добавление вершины v в отсортированный список
}

int topSort() {
        l = 0; // номер добавляемой вершины в отсортированный список
        for(int i = 0; i < n; i++)
                dfs(i); // запустить пробежку из всех вершин
        reverse(top, top+l); // развернутьмассив
        return 0;
}

int solve() {
        int i, j;
        for(i = 0; i < n; i++)
                d[i] = INF;
        d[s] = 0;
        for(i = 1; i < n; i++)
                for(j = 0; j < i; j++)
                        if(g[top[j]][top[i]])
                                d[top[i]] = min(d[top[i]], d[top[j]] + g[top[j]][top[i]]);
        return 0;
}
```

3. ХОД РАБОТЫ

1.Получить задание у преподавателя в виде исходного ориентированного графа.

2.Составить структурную схему программы, определяющей кратчайший путь на графе от заданной начальной вершины ***s***до заданной конечной вершины ***t***

с помощью метода динамического программирования.

3.Составить структурную схему программы, реализующей алгоритм топологической сортировки с произвольной нумерацией вершин графа.

4.Создать программу, реализующую метод динамического программирования и алгоритм топологической сортировки вершин. Исходный граф задается в виде матрицы смежности, вводимой построчно с помощью консоли. Указание: для определения вершин, входящих во множество $\Gamma^{-1}(x_i)$ используйте *j*-йстолбец матрицы смежности.

5.Создать программу, которая использует приведенный в данной работе алгоритм Дейкстры для заданного графа.

6.Сравнить время выполнения двух алгоритмов.

4. ВАРИАНТЫ ЗАДАНИЙ

Пояснения к вариантам.

1. Обозначения: «откуда», «куда» – узлы выхода и входа ребер ориентированного графа; цифры в таблице – веса ребер.

2. Для неориентированного графа элементы таблицы следует отобразить относительно главной диагонали

3. Пример ориентированного графа, соответствующего по варианту 1, дан ниже: X_i – узлы (вершины) графа, цифры на соединительных линиях – веса соответствующих рёбер.

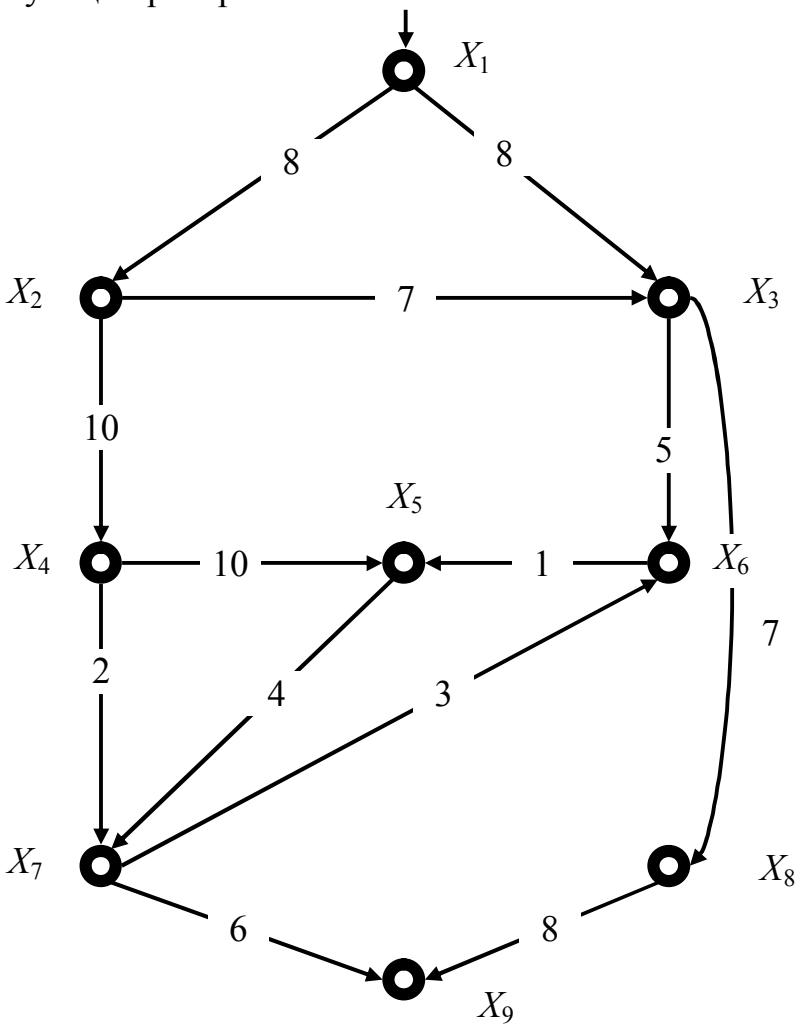

Рисунок Б.1 – Ориентированный граф к варианту 1

Вариант 1

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		8	8						
2			7	10					
3						5		7	
4					10		2		
5							4		
6					1				
7						3			6
8									5

Вариант 2

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		5	6						
2			3	8					
3					9	7		8	
4			3		2		6		
5							5		
6					7		10		
7									4
8									3

Вариант 3

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		2	1						
2			7	2					
3					2	3		6	
4			3				8		
5							4		
6					3				
7						2			10
8									1

Вариант 4

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		1	10						
2			3	3			7		
3					2	7		5	
4			2		4				
5							3		
6					2				
7						5			7
8									7

Вариант 5

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		4	10						
2			8						
3						8		3	
4					2		9		
5			6						
6				4	1			3	
7						1			5
8									7

Вариант 6

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		5	3						
2			4	2					
3						7	6	4	
4					10	5			
5							2	9	
6					7				
7						9			7
8									3

Вариант 7

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		9	4						
2			7	10					
3						4		6	
4			6		3				
5							1		
6					9			7	
7				1		8			6
8									4

Вариант 8

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		7	5						
2				3	9				
3		5			7	8		7	
4					7		4		
5							8		
6					3				
7						4			6
8									8

Вариант 9

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		9	1						
2			8	1					
3						4		3	
4			8		3		7		
5							6		
6					9			5	
7						6		1	4
8									1

Вариант 10

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		9	8						
2						5		1	
3		8							
4			9		1			4	
5							2		
6					8		7		
7				2					4
8									7

Вариант 11

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		8	6						
2			4						
3						5		8	
4					6		3		
5							10		
6				4	2		3		
7									9
8									5

Вариант 12

Откуда \ Куда	1	2	3	4	5	6	7	8	9
1		10	10						
2				1					
3				4		2		8	
4					3	4			
5							2		
6					9				
7				8					6
8									9

Вариант 13

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		7	3						
2			7		1		2		
3				1				4	
4					2		5		
5							10		
6					10				
7						5			6
8									4

Вариант 14

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		6	1						
2			8						
3						6			
4					3		1	1	
5							5		
6				2	4				
7						5			4
8									3

Вариант 15

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		5	6						
2			4	8					
3				2		7			
4					3		5		
5							2	5	
6					8			9	
7						5			1
8									5

Вариант 16

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		2	7						
2				8	3				
3								5	
4					4	1	3		
5							10	6	
6					6				
7						6			9
8									4

Вариант 17

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		3							
2				7	2				
3				1	2				
4					2		10	2	
5							8		
6					5				
7			3			2			10
8									6

Вариант 18

Куда / Откуда

	1	2	3	4	5	6	7	8	9
1		8	7						
2			3						
3						4		2	
4			8				3		
5							4		
6				8	2				
7						1			4
8									8

Вариант 19

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		9	7						
2			5	4	4				
3						5		8	
4						1			
5				4		9	7		
6								3	
7				5		4			3
8									6

Вариант 20

Куда

Откуда	1	2	3	4	5	6	7	8	9
1		4	8						
2			2	1	4				
3						8	5	4	
4					2	10			
5							3	8	
6					8				
7						5			2
8									6

5. СОДЕРЖАНИЕ ОТЧЁТА

1. Исходное задание и цель работы.
2.Структурная схема программы по п.2.
3. Структурная схема программы по п.3.
3. Распечатка текста программы по п.4.
4. Распечатка текста программы по п.5.
5. Контрольный пример и результаты машинного расчета.
6. Выводы по работе.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ

1.Дайте определение пути, маршрута, цепи, контура.
2.Какой граф называется взвешенным?
3.Как определяется длина пути графа?
4.Задача нахождения кратчайшего пути на графе.
5.Реализация метода динамического программирования для нахождения кратчайшего пути на графе.
6.Ограничения применения метода динамического программирования для нахождения кратчайшего пути на графе.
7.Что называется правильной нумерацией вершин графа?
8.Применение алгоритма топологической сортировки для перенумерации вершин графа.
9.Алгоритм Дейкстры. Сравнение с динамическим методом.