**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа № 4.1

По дисциплине: Дискретная математика

Тема: «Маршруты»

Выполнил: Енин Г. И.

Проверил: Рязанов Ю.Д.

Белгород 2018

**Цель занятия:** изучить основные понятия теории графов, способы задания графов, научиться программно реализовывать алгоритмы получения и анализа маршрутов в графах.

**Задания**

1. Представить графы G1 и G2 (см.”Варианты заданий”, п.а) матрицей смежности, матрицей инцидентности, диаграммой.

2. Определить, являются ли последовательности вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

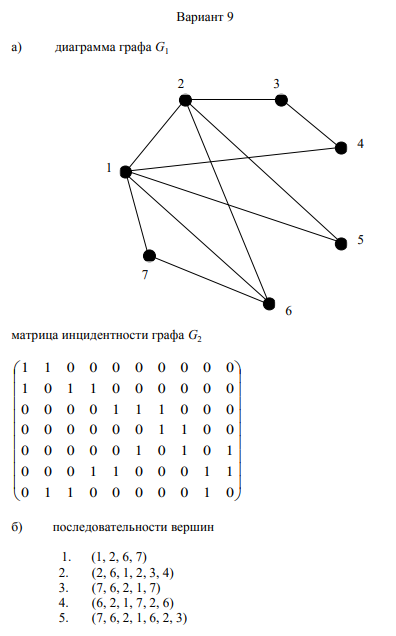
3. Написать программу, определяющую, является ли заданная последовательность вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

4. Написать программу, получающую все маршруты заданной длины, выходящие из заданной вершины. Использовать программу для получения всех маршрутов заданной длины в графах G1 и G2 (см.”Варианты заданий”, п.а).

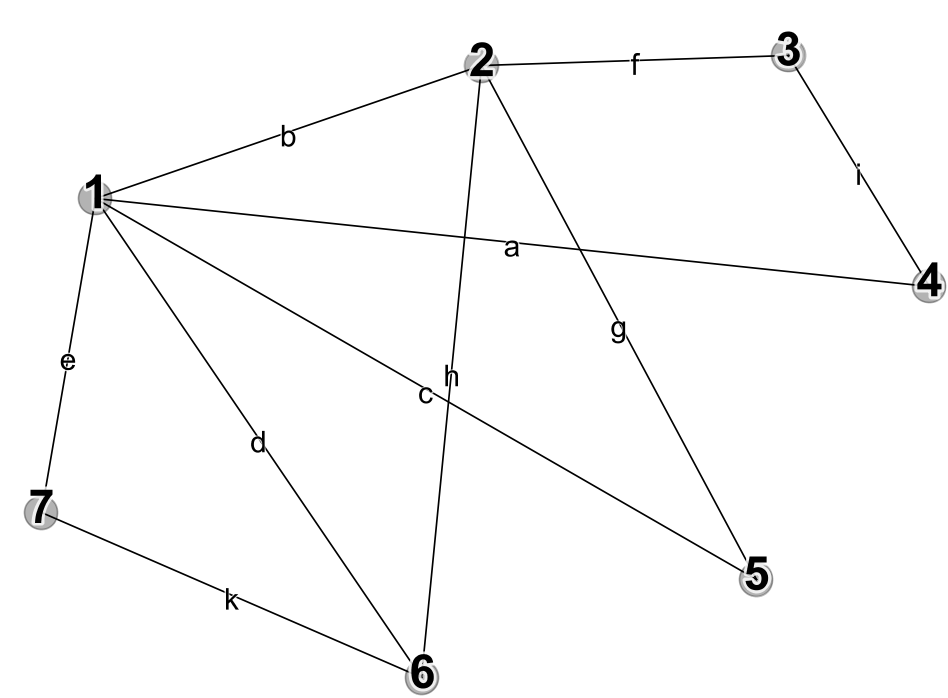
5. Написать программу, определяющую количество маршрутов заданной длины между каждой парой вершин графа. Использовать программу для определения количества маршрутов заданной длины между каждой парой вершин в графах G1 и G2 (см.”Варианты заданий”, п.а).

6. Написать программу, определяющую все маршруты заданной длины между заданной парой вершин графа. Использовать программу для определения всех маршрутов заданной длины между заданной парой вершин в графах G1 и G2 (см.”Варианты заданий”, п.а).

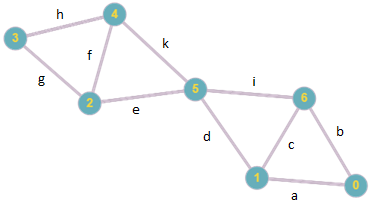
7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для получения всех простые максимальных цепей, выходящих из заданной вершины в графах G1 и G2 (см.”Варианты заданий”, п.а).

****

**1.**

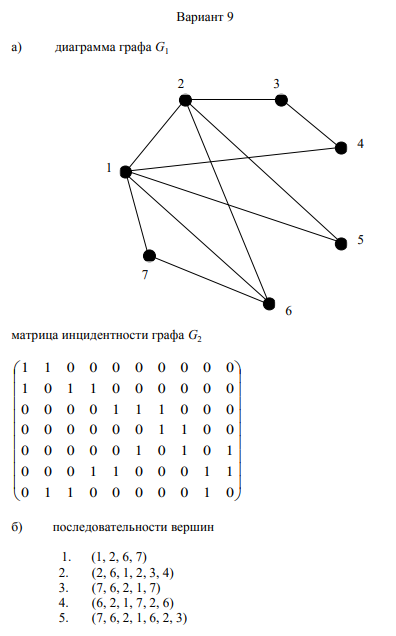
**G1**

|  |  |
| --- | --- |
| **матрица смежности** | **матрица инцидентности** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **1** |  | **1** | **1** | **1** | **1** | | **1** |  | **1** |  | **1** | **1** |  | |  | **1** |  | **1** |  |  |  | | **1** |  | **1** |  |  |  |  | | **1** | **1** |  |  |  |  |  | | **1** | **1** |  |  |  |  | **1** | | **1** |  |  |  |  | **1** |  | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **k** | | **1** | **1** | **1** | **1** | **1** | **1** |  |  |  |  |  | | **2** |  | **1** |  |  |  | **1** | **1** | **1** |  |  | | **3** |  |  |  |  |  | **1** |  |  | **1** |  | | **4** | **1** |  |  |  |  |  |  |  | **1** |  | | **5** |  |  | **1** |  |  |  | **1** |  |  |  | | **6** |  |  |  |  |  |  |  | **1** |  | **1** | |

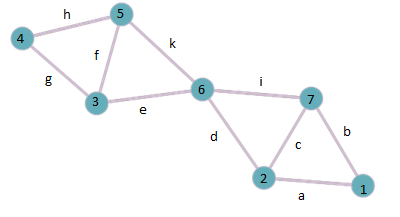
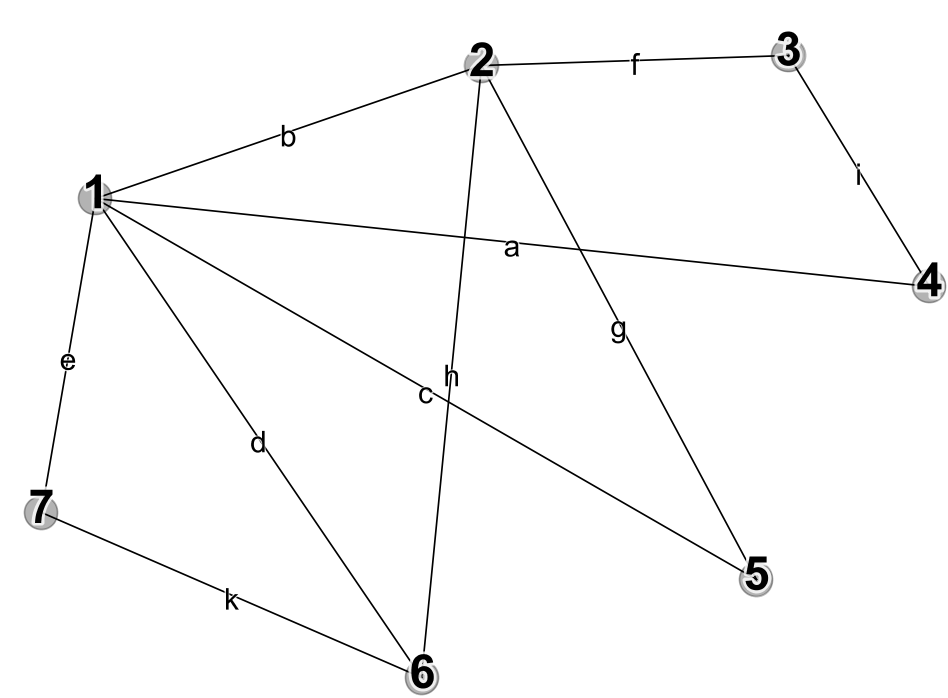
** G2**

|  |  |
| --- | --- |
| **матрица смежности** | **матрица инцидентности** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  | **1** |  |  |  |  | **1** | | **1** |  |  |  |  | **1** | **1** | |  |  |  | **1** | **1** | **1** |  | |  |  | **1** |  | **1** |  |  | |  |  | **1** | **1** |  | **1** |  | |  | **1** | **1** |  | **1** |  | **1** | | **1** | **1** |  |  |  | **1** |  | | |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **h** | **i** | **k** | | **0** | **1** | **1** |  |  |  |  |  |  |  |  | | **1** | **1** |  | **1** | **1** |  |  |  |  |  |  | | **2** |  |  |  |  | **1** | **1** | **1** |  |  |  | | **3** |  |  |  |  |  |  | **1** | **1** |  |  | | **4** |  |  |  |  |  | **1** |  | **1** |  | **1** | | **5** |  |  |  | **1** | **1** |  |  |  | **1** | **1** | | **6** |  | **1** | **1** |  |  |  |  |  | **1** |  | |

2. Определить, являются ли последовательности вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

****

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | | | | | **2** | | | | | **3** | | | | | **4** | | | | | **5** | | | | |
| **G1** | **+** | **+** | **+** |  |  | **+** | **+** |  |  |  | **+** |  |  | **+** | **+** |  |  |  |  |  | **+** |  |  |  |  |
| **G2** | **+** | **+** | **+** |  |  |  |  |  |  |  | **+** |  |  | **+** | **+** | **+** |  |  |  |  |  |  |  |  |  |

****

3. Написать программу, определяющую, является ли заданная последовательность вершин (см. ”Варианты заданий”, п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G1 и G2 (см.”Варианты заданий”, п.а).

#include **<stdio.h>**#include **<malloc.h>  
  
int** counter = 0;  
  
**char** compare(**int** \*w, **int** \*myway, **int** n) {  
 **for** (**int** i = 0; i < n; ++i) {  
 **if** (myway[i] - 1 != w[i]) **return** 0;  
 }  
 **return** 1;  
}  
  
**int** find\_equals\_numbers(**int** \*w, **int** n) {  
 **for** (**int** i = 0; i < n - 1; ++i) {  
 **for** (**int** j = i + 1; j < n; ++j) {  
 **if** (w[i] == w[j]) {  
 **return** j;  
 }  
 }  
 }  
 **return** 0;  
}  
  
**int** find\_equals\_groups(**int** \*w, **int** n) {  
 **for** (**int** k = 0; k < n; ++k) printf(**"%i "**, w[k] + 1);  
 puts(**""**);  
 **for** (**int** i = 0; i < n - 1; ++i) {  
 **for** (**int** j = i + 1; j < n - 2; ++j) {**if** ((w[i] == w[j + 1]) && (w[i + 1] == w[j + 2])) {  
 **return** 1;  
 }  
 }  
 }  
 **return** 0;  
}  
  
**void** checker(**int** \*\*a, **int** \*w, **int** \*myway, **int** n, **int** l, **int** i) {  
 **for** (**int** j = 0; j < n; ++j) {  
 **if** (a[w[i - 1]][j]) {  
 w[i] = j;  
 **if** (i == l - 1) {**if** (compare(w, myway, l)) {  
 **int** equals\_groups = find\_equals\_groups(w, l);  
 **int** equals\_numbers = find\_equals\_numbers(w, l);  
 printf(**"Way\n"**);  
 **if** (equals\_groups==0) {  
 **if**(equals\_numbers==l-1) {  
 puts(**"Loop"**);  
 }**else**{  
 puts(**"Chain"**);  
 }  
 }  
 **if** ((equals\_numbers==0)||(equals\_numbers==l-1)) {  
 **if**(equals\_numbers==l-1) {  
 puts(**"Easy Loop"**);  
 }**else**{  
 puts(**"Easy Chain"**);  
 }  
 }  
 }  
 } **else** {  
 checker(a, w, myway, n, l, i + 1);  
 }  
 }  
 }  
}  
  
**int** main() {  
 **int** n, l;  
 scanf(**"%i %i"**, &n, &l); *// Ввод размера матрицы и длины маршрута* **int** \*\*a = malloc(n \* **sizeof**(**int** \*)), \*w = malloc(l \* **sizeof**(**int**)), \*myway = malloc(l \* **sizeof**(**int**));  
  
 **for** (**int** i = 0; i < n; ++i) a[i] = malloc(n \* **sizeof**(**int**));  
 **for** (**int** i = 0; i < l; ++i) scanf(**"%i"**, &myway[i]);  
 **for** (**int** i = 0; i < n; ++i) **for** (**int** j = 0; j < n; ++j) scanf(**"%i"**, &a[i][j]);  
 w[0] = myway[0] - 1;  
  
 checker(a, w, myway, n, l, 1);  
  
 **return** 0;  
}

4. Написать программу, получающую все маршруты заданной длины, выходящие из заданной вершины. Использовать программу для получения всех маршрутов заданной длины в графах G1 и G2 (см.”Варианты заданий”, п.а).

#include **<stdio.h>**#include **<malloc.h>  
int** counter=0;  
**void** ways(**int** \*\*a, **int** \*w, **int** n, **int** l, **int** i){  
 **for** (**int** j = 0; j < n; ++j) {  
 **if**(a[w[i-1]][j]){  
 w[i] = j;  
 **if**(i==l-1){  
 counter++;  
 **for** (**int** k = 0; k < l; ++k) printf(**"%i "**, w[k]+1); puts(**""**);  
 } **else** {  
 ways(a,w,n,l,i+1);  
 }  
 }  
 }  
}  
  
**int** main() {  
 **int** n,l;  
 scanf(**"%i %i"**, &n, &l); *// Ввод размера матрицы и длины маршрута* **int** \*\*a = malloc(n\* **sizeof**(**int**\*)), \*w = malloc(l\***sizeof**(**int**));  
 scanf(**"%i"**,&w[0]); w[0]--; // Ввод начальной вершины  
 **for** (**int** i = 0; i < n; ++i) a[i] = malloc(n\***sizeof**(**int**));  
 **for** (**int** i = 0; i < n; ++i) **for** (**int** j = 0; j < n; ++j) scanf(**"%i"**,&a[i][j]);  
  
 puts(**"All ways"**);  
 ways(a,w,n,l,1);  
 printf(**"Finded %i ways"**,counter);  
  
 **return** 0;  
}

5. Написать программу, определяющую количество маршрутов заданной длины между каждой парой вершин графа. Использовать программу для определения количества маршрутов заданной длины между каждой парой вершин в графах G1 и G2 (см.”Варианты заданий”, п.а).

#include **<stdio.h>**#include **<malloc.h>  
int** counter=0;  
**void** ways(**int** \*\*a, **int** \*w, **int** n, **int** l, **int** i){  
 **for** (**int** j = 0; j < n; ++j) {  
 **if**(a[w[i-1]][j]){  
 w[i] = j;  
 **if**(i==l-1){  
 counter++;  
 } **else** {  
 ways(a,w,n,l,i+1);  
 }  
 }  
 }  
 **if**((w[0]<l)&&(i==1)){  
 w[0]++;  
 ways(a,w,n,l,1);  
 }  
}  
  
**int** main() {  
 **int** n,l;  
 scanf(**"%i %i"**, &n, &l); // Ввод размера матрицы и длины маршрута  
 **int** \*\*a = malloc(n\* **sizeof**(**int**\*)), \*w = malloc(l\***sizeof**(**int**));  
 **for** (**int** i = 0; i < n; ++i) { a[i] = malloc(n\***sizeof**(**int**)); w[i] = 0; }  
 **for** (**int** i = 0; i < n; ++i) **for** (**int** j = 0; j < n; ++j) scanf(**"%i"**,&a[i][j]);  
  
 ways(a,w,n,l,1);  
 printf(**"Finded %i ways"**,counter);  
  
 **return** 0;  
}

6. Написать программу, определяющую все маршруты заданной длины между заданной парой вершин графа. Использовать программу для определения всех маршрутов заданной длины между заданной парой вершин в графах G1 и G2 (см.”Варианты заданий”, п.а).

#include **<stdio.h>**#include **<malloc.h>  
int** counter=0;  
**void** ways(**int** \*\*a, **int** \*w, **int** n, **int** l, **int** i, **int** target){  
 **for** (**int** j = 0; j < n; ++j) {  
 **if**(a[w[i-1]][j]){  
 w[i] = j;  
 **if**(i==l-1){  
 **if**(w[l-1]==target) {  
 counter++;  
 **for** (**int** k = 0; k < l; ++k) printf(**"%i "**, w[k] + 1);  
 puts(**""**);  
 }  
 } **else** {  
 ways(a,w,n,l,i+1,target);  
 }  
 }  
 }  
}  
  
**int** main() {  
 **int** n,l,from,target;  
 scanf(**"%i %i %i %i"**, &n, &l, &from, &target);  
 **int** \*\*a = malloc(n\* **sizeof**(**int**\*)), \*w = malloc(l\***sizeof**(**int**));  
 w[0] = from-1;  
 **for** (**int** i = 0; i < n; ++i) a[i] = malloc(n\***sizeof**(**int**));  
 **for** (**int** i = 0; i < n; ++i) **for** (**int** j = 0; j < n; ++j) scanf(**"%i"**,&a[i][j]);

puts(**"Pathways"**);  
 ways(a,w,n,l,1,target-1);  
 printf(**"Finded %i ways"**,counter);  
  
 **return** 0;  
}

7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для получения всех простые максимальных цепей, выходящих из заданной вершины в графах G1 и G2 (см.”Варианты заданий”, п.а).

#include **<stdio.h>**#include **<malloc.h>  
int** counter=0;  
  
**char** find(**int** \*w, **int** n){  
 **for** (**int** i = 0; i < n-1; ++i) {  
 **for** (**int** j = i+1; j < n; ++j) {**if**(w[i]==w[j]){**return** 1;  
 }  
 }  
 }**return** 0;  
}  
  
**void** chains(**int** \*\*a, **int** \*w, **int** n, **int** l, **int** i){  
 **for** (**int** j = 0; j < n; ++j) {  
 **if**(a[w[i-1]][j]){  
 w[i] = j;  
 **if**(i==l-1){  
 **if**(!find(w,l)){  
 counter++;

**for** (**int** k = 0; k < l; ++k) printf(**"%i "**, w[k] + 1);  
 puts(**""**);  
 }  
 } **else** {  
 chains(a,w,n,l,i+1);  
 }  
 }  
 }  
}  
  
**int** main() {  
 **int** n,l;  
 scanf(**"%i %i"**, &n, &l); *// Ввод размера матрицы и длины маршрута* **int** \*\*a = malloc(n\* **sizeof**(**int**\*)), \*w = malloc(l\***sizeof**(**int**));  
 scanf(**"%i"**,&w[0]); w[0]--;  
 **for** (**int** i = 0; i < n; ++i) a[i] = malloc(n\***sizeof**(**int**));  
 **for** (**int** i = 0; i < n; ++i) **for** (**int** j = 0; j < n; ++j) scanf(**"%i"**,&a[i][j]);  
  
 chains(a,w,n,l,1);  
 printf(**"Finded %i chains"**,counter);  
 **return** 0;  
}

**Выводы:**

**№3**

|  |  |
| --- | --- |
| 7 6  2 6 1 2 3 4  0 1 0 1 1 1 1  1 0 1 0 1 1 0  0 1 0 1 0 0 0  1 0 1 0 0 0 0  1 1 0 0 0 0 0  1 1 0 0 0 0 1  1 0 0 0 0 1 0  Way  Chain | 7 5  7 6 2 1 7  0 1 0 0 0 0 1  1 0 0 0 0 1 1  0 0 0 1 1 1 0  0 0 1 0 1 0 0  0 0 1 1 0 1 0  0 1 1 0 1 0 1  1 1 0 0 0 1 0  7 6 2 1 7  Way  Loop  Easy Loop |

**№4**

|  |  |
| --- | --- |
| 7 3  0 1 0 1 1 1 1  1 0 1 0 1 1 0  0 1 0 1 0 0 0  1 0 1 0 0 0 0  1 1 0 0 0 0 0  1 1 0 0 0 0 1  1 0 0 0 0 1 0  All ways  1 2 1  1 2 3  1 2 5  1 2 6  1 4 1  1 4 3  1 5 1  1 5 2  1 6 1  1 6 2  1 6 7  1 7 1  1 7 6  2 1 2  2 1 4  2 1 5  2 1 6  2 1 7  2 3 2  2 3 4  2 5 1  2 5 2  2 6 1  2 6 2  2 6 7  3 2 1  3 2 3  3 2 5  3 2 6  3 4 1  3 4 3  4 1 2  4 1 4  4 1 5  4 1 6  4 1 7  4 3 2  4 3 4  Finded 38 ways | 7 3  0 1 0 0 0 0 1  1 0 0 0 0 1 1  0 0 0 1 1 1 0  0 0 1 0 1 0 0  0 0 1 1 0 1 0  0 1 1 0 1 0 1  1 1 0 0 0 1 0  All ways  1 2 1  1 2 6  1 2 7  1 7 1  1 7 2  1 7 6  2 1 2  2 1 7  2 6 2  2 6 3  2 6 5  2 6 7  2 7 1  2 7 2  2 7 6  3 4 3  3 4 5  3 5 3  3 5 4  3 5 6  3 6 2  3 6 3  3 6 5  3 6 7  4 3 4  4 3 5  4 3 6  4 5 3  4 5 4  4 5 6  Finded 30 ways |

**№5**

|  |  |
| --- | --- |
| 7 4  0 1 0 1 1 1 1  1 0 1 0 1 1 0  0 1 0 1 0 0 0  1 0 1 0 0 0 0  1 1 0 0 0 0 0  1 1 0 0 0 0 1  1 0 0 0 0 1 0  Finded 149 ways | 7 3  0 1 0 0 0 0 1  1 0 0 0 0 1 1  0 0 0 1 1 1 0  0 0 1 0 1 0 0  0 0 1 1 0 1 0  0 1 1 0 1 0 1  1 1 0 0 0 1 0  Finded 30 ways |

**№6**

|  |  |
| --- | --- |
| 7 4  3 5  0 1 0 1 1 1 1  1 0 1 0 1 1 0  0 1 0 1 0 0 0  1 0 1 0 0 0 0  1 1 0 0 0 0 0  1 1 0 0 0 0 1  1 0 0 0 0 1 0  Pathways  3 2 1 5  3 4 1 5  Finded 2 ways | 7 5  5 1  0 1 0 0 0 0 1  1 0 0 0 0 1 1  0 0 0 1 1 1 0  0 0 1 0 1 0 0  0 0 1 1 0 1 0  0 1 1 0 1 0 1  1 1 0 0 0 1 0  Pathways  5 3 6 2 1  5 3 6 7 1  5 6 2 7 1  5 6 7 2 1  Finded 4 ways |

**№7**

|  |  |
| --- | --- |
| 7 4  1  0 1 0 1 1 1 1  1 0 1 0 1 1 0  0 1 0 1 0 0 0  1 0 1 0 0 0 0  1 1 0 0 0 0 0  1 1 0 0 0 0 1  1 0 0 0 0 1 0  All ways  1 2 3 4  1 2 6 7  1 4 3 2  1 5 2 3  1 5 2 6  1 6 2 3  1 6 2 5  1 7 6 2  Finded 8 ways | 7 3  2  0 1 0 0 0 0 1  1 0 0 0 0 1 1  0 0 0 1 1 1 0  0 0 1 0 1 0 0  0 0 1 1 0 1 0  0 1 1 0 1 0 1  1 1 0 0 0 1 0  All ways  2 1 7  2 6 3  2 6 5  2 6 7  2 7 1  2 7 6  Finded 6 ways |