Скобёлкин Глеб ФИб-1301  
Лабораторная работа №1

№1.  
Последовательный поиск. Массив: [2, 4, 0, 1, 7, 5]. Найти 7.

|  |  |  |
| --- | --- | --- |
| i | Arr[i] | found |
| 0 | 2 | false |
| 1 | 4 | false |
| 2 | 0 | false |
| 3 | 1 | false |
| 4 | 7 | true |

Алгоритм С. Уоршалла. Массив: 0 1  
 1 0

|  |  |  |  |
| --- | --- | --- | --- |
| m | i | j | Arr[i][j] |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

№2.

#include <iostream>

#include <ctime>

#define SIZE 10

#define RANGE rand() % 10 - 5

void fillArray(int arr[]);

void output(int arr[]);

void linearSearch(int arr[], int key);

int main()

{

int arr[SIZE];

int key, ans;

fillArray(arr);

output(arr);

std::cout << "Enter the key: ";

std::cin >> key;

linearSearch(arr, key);

system("PAUSE");

return 0;

}

void fillArray(int arr[]) {

srand(time(NULL));

for (int i = 0; i < SIZE; ++i) {

arr[i] = RANGE;

}

}

void output(int arr[]) {

for (int i = 0; i < SIZE; ++i)

std::cout << arr[i] << '\t';

std::cout << std::endl;

}

void linearSearch(int arr[], int key) {

bool found = false;

for (int i = 0; i < SIZE; ++i)

if (arr[i] == key) {

std::cout << key << " found (#" << i << ")" << std::endl;

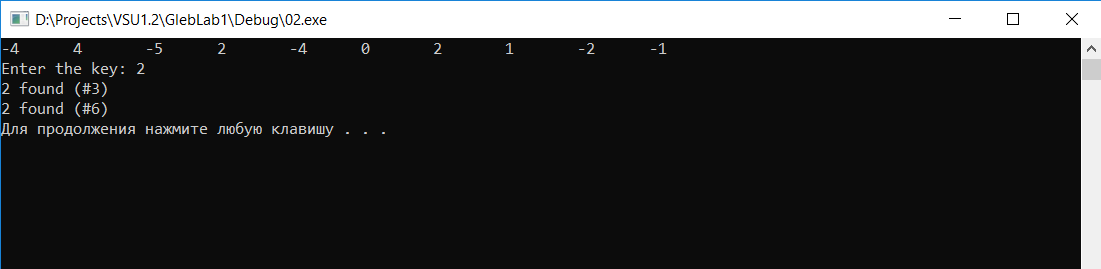
found = true;

}

if (!found)

std::cout << key << " not found!" << std::endl;

}

Временная оценка работы алгоритма НЕ ИЗМЕНИТСЯ.

№3.  
#include <iostream>

#include <ctime>

#define SIZE 10

#define RANGE rand() % 10 - 5

void fillArray(int arr[]);

void output(int arr[]);

void maxElements(int arr[]);

int recMaxElem(int arr[], int max, int i);

int main()

{

int arr[SIZE];

fillArray(arr);

output(arr);

maxElements(arr);

int i = SIZE - 1;

int max = arr[i];

std::cout << "Max element is " << recMaxElem(arr, max, i) << std::endl;

system("PAUSE");

return 0;

}

void fillArray(int arr[]) {

srand(time(NULL));

for (int i = 0; i < SIZE; ++i) {

arr[i] = RANGE;

}

}

void output(int arr[]) {

for (int i = 0; i < SIZE; ++i)

std::cout << arr[i] << '\t';

std::cout << std::endl;

}

void maxElements(int arr[]) {

int max = arr[0];

for (int i = 1; i < SIZE; ++i)

if (arr[i] > max)

max = arr[i];

std::cout << "Max element is " << max << std::endl;

for (int i = 0; i < SIZE; ++i)

if (arr[i] == max)

std::cout << "Max element found in #" << i << std::endl;

}

int recMaxElem(int arr[], int max, int i) {

if (i == 0)

return max;

else {

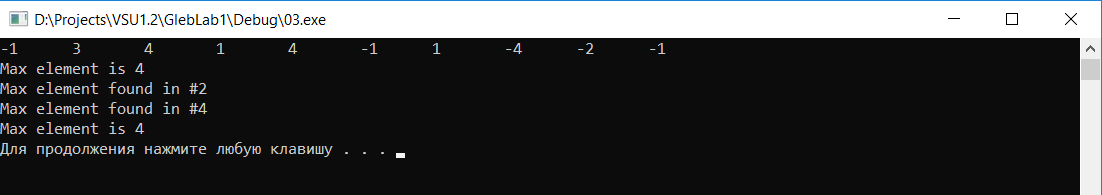
if (max < arr[i])

max = arr[i];

recMaxElem(arr, max, i - 1);

}

}



№4.  
#include <iostream>

#include <ctime>

#define SIZE 9

#define RANGE rand() % 10 - 5

void fillArray(int arr[]);

void output(int arr[]);

int recursiveSearch(int arr[], int key, int left, int right);

int main()

{

int arr[SIZE];

int key;

int left = 0;

int right = SIZE;

int ans;

fillArray(arr);

output(arr);

std::cout << "Enter the key: ";

std::cin >> key;

ans = recursiveSearch(arr, key, left, right);

if (ans == -1)

std::cout << key << " not found!" << std::endl;

else

std::cout << key << " found in element #" << ans << std::endl;

system("PAUSE");

return 0;

}

void fillArray(int arr[]) {

srand(time(NULL));

int t = 0;

for (int i = 0; i < SIZE; ++i) {

arr[i] = RANGE + t;

t += 10; // чтобы элементы шли в порядке возрастания

}

}

void output(int arr[]) {

for (int i = 0; i < SIZE; ++i)

std::cout << arr[i] << '\t';

std::cout << std::endl;

}

int recursiveSearch(int arr[], int key, int left, int right) {

int middle = (left + right) / 2;

if (arr[middle] == key)

return middle;

else if (middle == left || middle == right)

return -1;

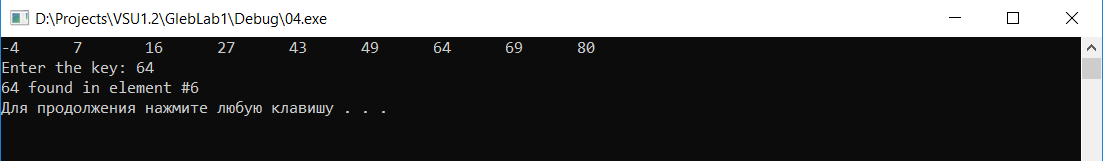
if (key < arr[middle])

recursiveSearch(arr, key, left, middle);

else

recursiveSearch(arr, key, middle, right);

}



№5.  
#include <iostream>

#include <cmath>

int main()

{

double a, b;

for (int N = 0; N < 100000; ++N) {

a = N;

b = log2(N);

if (b \* 10 - a > -0.2 && b \* 10 - a < 0.2)

std::cout << N << '(' << 10 << ')' << std::endl;

else if (b \* 100 - a > -0.1 && b \* 100 - a < 0.1)

std::cout << N << '(' << 100 << ')' << std::endl;

else if (b \* 1000 - a > -0.2 && b \* 1000 - a < 0.2)

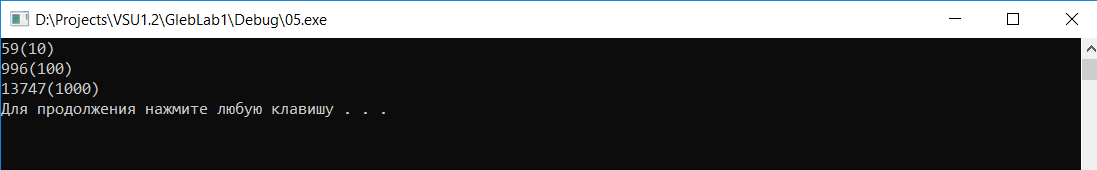
std::cout << N << '(' << 1000 << ')' << std::endl;

}

system("PAUSE");

return 0;

}



Поскольку log2(59) = 5,883; log2(996) = 9,96; log2(13747) = 13,747

№6.  
#include <iostream>

#include <ctime>

#define SIZE 4

#define RANGE rand() % 10

void fillArray();

void output(int(&arr)[SIZE][SIZE]);

void solve();

void way(long i, long j);

int a[SIZE][SIZE];

int b[SIZE][SIZE];

int main()

{

fillArray();

output(a);

solve();

output(b);

way(SIZE - 1, SIZE - 1);

std::cout << std::endl << std::endl;

system("PAUSE");

return 0;

}

void fillArray() {

srand(time(NULL));

for (int i = 0; i < SIZE; ++i)

for (int j = 0; j < SIZE; ++j)

a[i][j] = RANGE;

}

void output(int(&arr)[SIZE][SIZE]) {

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

std::cout << arr[i][j] << ' ';

if (arr[i][j] < 10)

std::cout << ' ';

}

std::cout << std::endl;

}

std::cout << std::endl;

}

void solve() {

b[0][0] = a[0][0];

for (long i = 1; i < SIZE; ++i)

b[i][0] = b[i - 1][0] + a[i][0];

for (long j = 1; j < SIZE; ++j)

b[0][j] = b[0][j - 1] + a[0][j];

for (long i = 1; i < SIZE; ++i)

for (long j = 1; j < SIZE; ++j) {

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i][j - 1];

}

}

void way(long i, long j) {

if (i == 0 && j == 0) {

std::cout << i << ' ' << j << "; ";

return;

}

else if (i == 0 && j > 0)

way(i, j - 1);

else if (i > 0 && j == 0)

way(i - 1, j);

else if (b[i][j] - a[i][j] == b[i - 1][j])

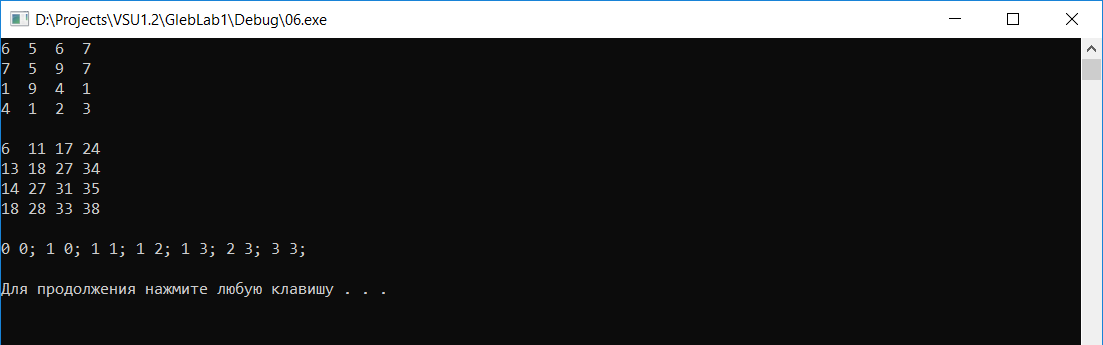
way(i - 1, j);

else

way(i, j - 1);

std::cout << i << ' ' << j << "; ";

}



Проверил, всё работает

№7.  
#include <iostream>

#include <ctime>

#define SIZE 4

#define RANGE rand() % 10

void fillArray();

void output(int(&arr)[SIZE][SIZE]);

void modi1();

void modi2();

void modi3();

void modi4();

void way(long i, long j);

int a[SIZE][SIZE];

int b[SIZE][SIZE];

int choice;

int main()

{

fillArray();

output(a);

std::cout << "Select the modification (1 - 4): ";

std::cin >> choice;

std::cout << std::endl;

switch (choice) {

case 1:

modi1();

break;

case 2:

modi2();

break;

case 3:

modi3();

break;

case 4:

modi4();

break;

default:

std::cout << "\*\*\* ERROR \*\*\* " << std::endl;

system("PAUSE");

exit(1);

}

output(b);

way(SIZE - 1, SIZE - 1);

std::cout << std::endl << std::endl;

system("PAUSE");

return 0;

}

void fillArray() {

//srand(time(NULL));

for (int i = 0; i < SIZE; ++i)

for (int j = 0; j < SIZE; ++j)

a[i][j] = RANGE;

}

void output(int(&arr)[SIZE][SIZE]) {

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

std::cout << arr[i][j] << ' ';

if (arr[i][j] < 10)

std::cout << ' ';

}

std::cout << std::endl;

}

std::cout << std::endl;

}

Первая модификация:

void modi1() {

b[0][0] = a[0][0];

// первый столбец

for (long i = 1; i < SIZE; ++i)

b[i][0] = b[i - 1][0] + a[i][0];

// первая строка

for (long j = 1; j < SIZE; ++j)

b[0][j] = b[0][j - 1] + a[0][j];

for (long i = 1; i < SIZE; ++i)

for (long j = 1; j < SIZE; ++j) {

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1] && b[i - 1][j] > b[i - 1][j - 1])

b[i][j] += b[i - 1][j];

else if (b[i - 1][j - 1] > b[i][j - 1] && b[i - 1][j - 1] > b[i - 1][j])

b[i][j] += b[i - 1][j - 1];

else

b[i][j] += b[i][j - 1];

}

}

Вторая модификация:

void modi2() {

b[0][0] = a[0][0];

// первая строка

for (long j = 1; j < SIZE; ++j)

b[0][j] = b[0][j - 1] + a[0][j];

for (long i = 1; i < SIZE; ++i) {

long j = 0;

// первый столбец

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i - 1][j + 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i - 1][j + 1];

for (j = 1; j < SIZE - 1; ++j) {

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1] && b[i - 1][j] > b[i - 1][j + 1])

b[i][j] += b[i - 1][j];

else if (b[i][j - 1] > b[i - 1][j] && b[i][j - 1] > b[i - 1][j + 1])

b[i][j] += b[i][j - 1];

else

b[i][j] += b[i - 1][j + 1];

}

// последний столбец

j = SIZE - 1;

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i][j - 1];

}

}

Третья модификация:

void modi3() {

b[0][0] = a[0][0];

// первый столбец

for (long i = 1; i < SIZE; ++i)

b[i][0] = b[i - 1][0] + a[i][0];

for (int j = 1; j < SIZE; ++j) {

long i = 0;

// первая строка

b[i][j] = a[i][j];

if (b[i][j - 1] > b[i + 1][j - 1])

b[i][j] += b[i][j - 1];

else

b[i][j] += b[i + 1][j - 1];

for (i = 1; i < SIZE - 1; ++i) {

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1] && b[i - 1][j] > b[i + 1][j - 1])

b[i][j] += b[i - 1][j];

else if (b[i][j - 1] > b[i - 1][j] && b[i][j - 1] > b[i + 1][j - 1])

b[i][j] += b[i][j - 1];

else

b[i][j] += b[i + 1][j - 1];

}

// последняя строка

i = SIZE - 1;

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i][j - 1];

}

}

Четвертая модификация:

void modi4() {

b[0][0] = a[0][0];

// первая строка

for (long j = 1; j < SIZE; ++j)

b[0][j] = b[0][j - 1] + a[0][j];

for (long i = 1; i < SIZE; ++i) {

long j = 0;

// первый столбец

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i - 1][j + 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i - 1][j + 1];

for (j = 1; j < SIZE - 1; ++j) {

b[i][j] = a[i][j];

if (b[i - 1][j] > b[i][j - 1] && b[i - 1][j] > b[i - 1][j + 1] && b[i - 1][j] > b[i - 1][j - 1])

b[i][j] += b[i - 1][j];

else if (b[i][j - 1] > b[i - 1][j - 1] && b[i][j - 1] > b[i - 1][j + 1])

b[i][j] += b[i][j - 1];

else if (b[i - 1][j - 1] > b[i - 1][j + 1])

b[i][j] += b[i - 1][j - 1];

else

b[i][j] += b[i - 1][j + 1];

}

// последний столбец

j = SIZE - 1;

b[i][j] = a[i][j];

if (b[i][j - 1] > b[i - 1][j] && b[i][j - 1] > b[i - 1][j - 1])

b[i][j] += b[i][j - 1];

else if (b[i - 1][j] > b[i - 1][j - 1])

b[i][j] += b[i - 1][j];

else

b[i][j] += b[i - 1][j - 1];

}

}

Вычисление пути

void way(long i, long j) {

if (i == 0 && j == 0) {

std::cout << i << ' ' << j << "; ";

return;

}

else if (b[i][j] - a[i][j] == b[i - 1][j] && i != 0)

way(i - 1, j);

else if (b[i][j] - a[i][j] == b[i][j - 1] && j != 0)

way(i, j - 1);

else if (b[i][j] - a[i][j] == b[i - 1][j - 1] && i != 0 && j != 0 && (choice == 1 || choice == 4))

way(i - 1, j - 1);

else if (b[i][j] - a[i][j] == b[i - 1][j + 1] && i != 0 && j != SIZE - 1 && (choice == 2 || choice == 4))

way(i - 1, j + 1);

else if (b[i][j] - a[i][j] == b[i + 1][j - 1] && i != SIZE - 1 && j != 0 && choice == 3)

way(i + 1, j - 1);

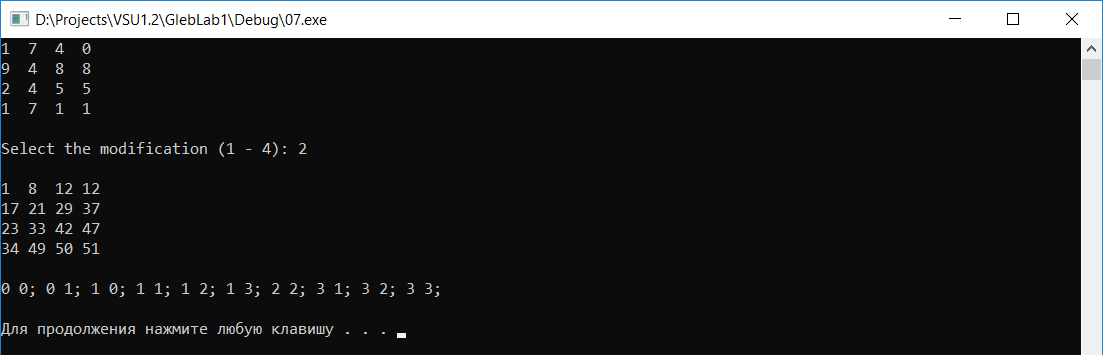
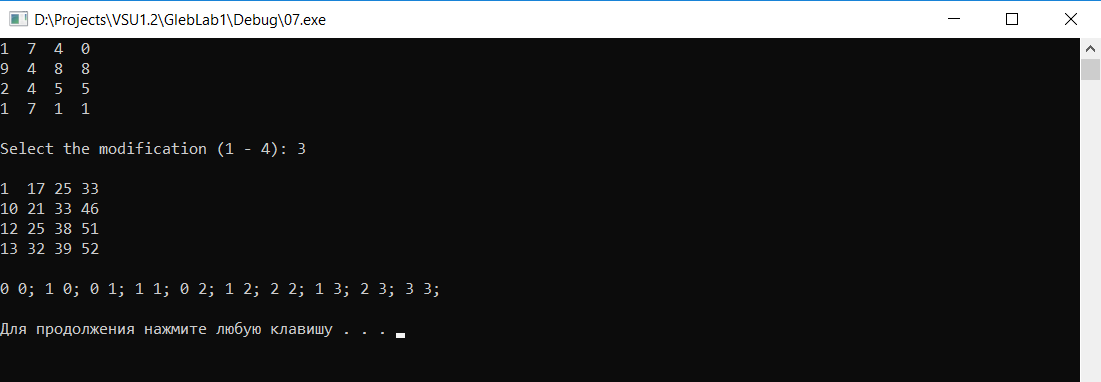
else if (i == 0 && j > 0)

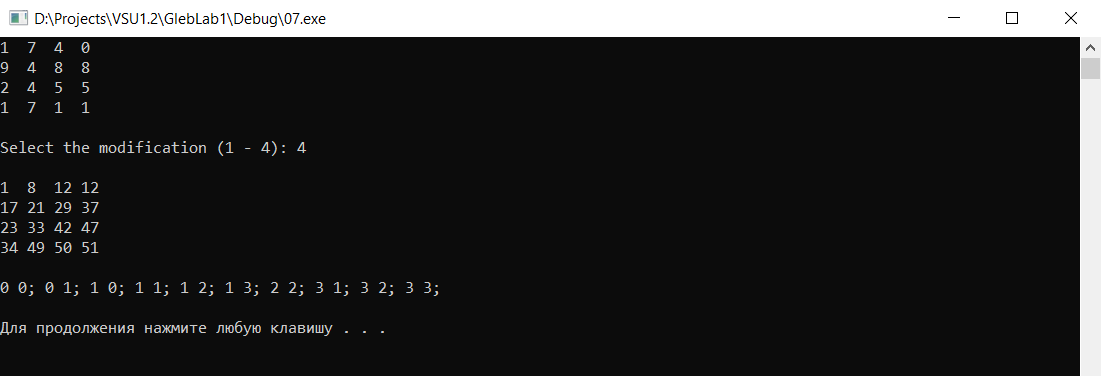
way(i, j - 1);

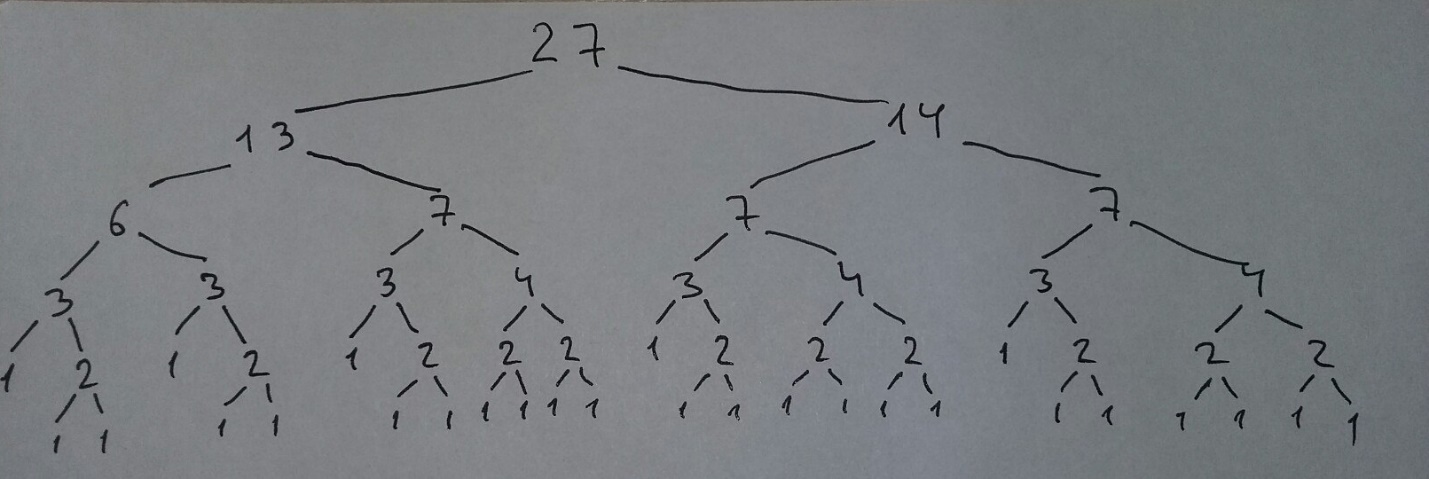
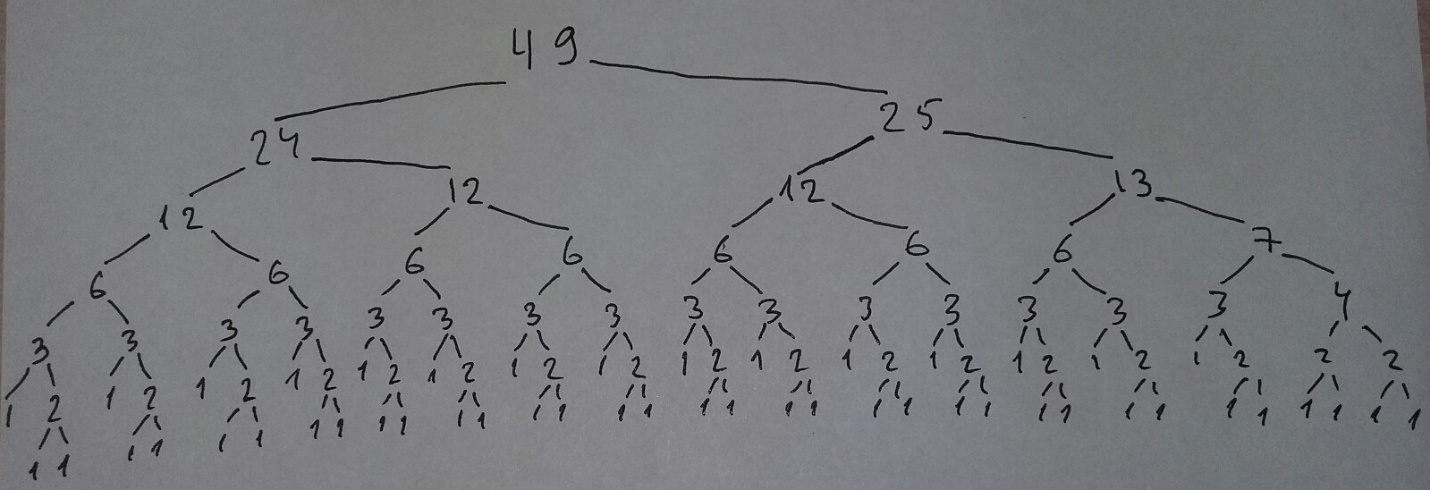
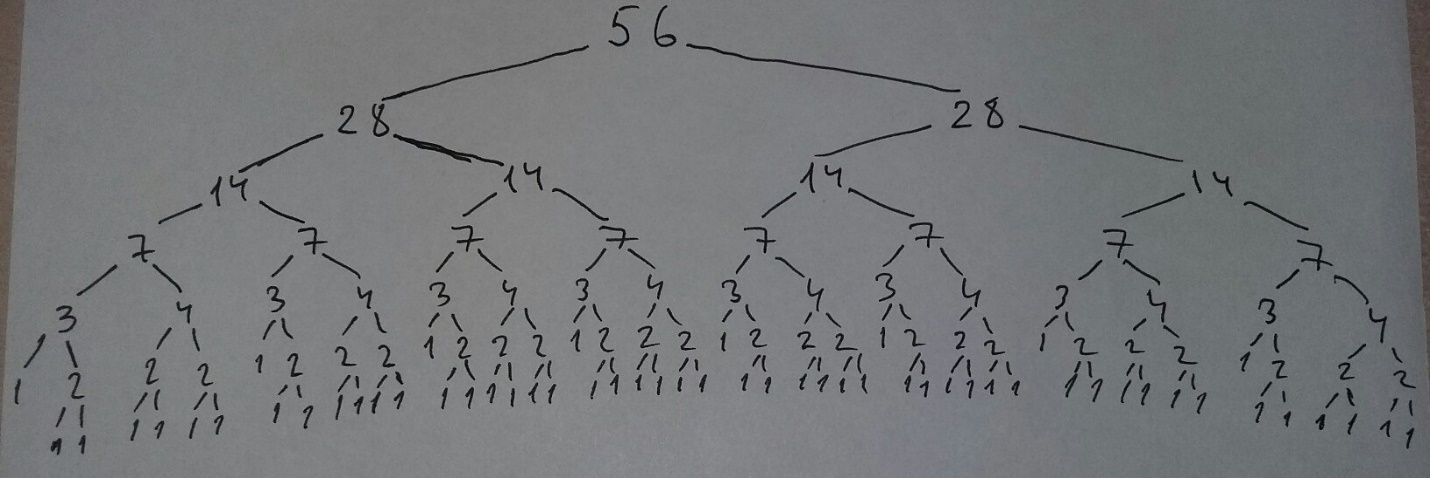
else

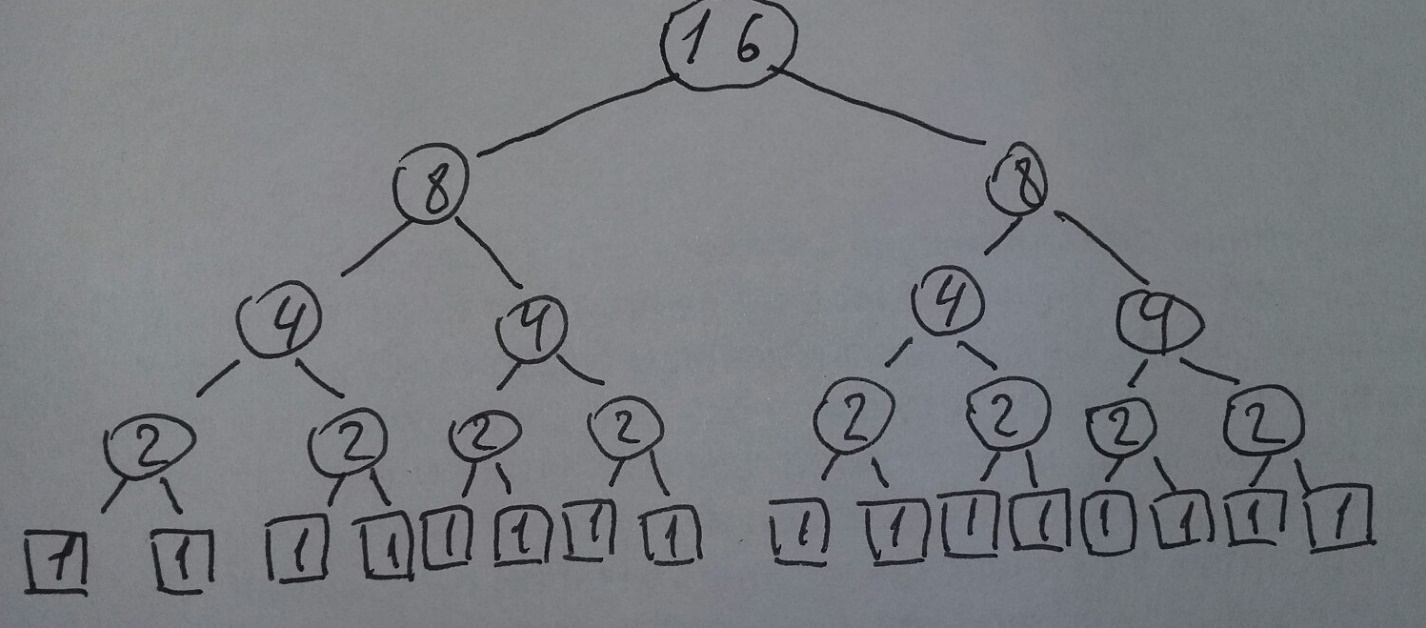
way(i - 1, j);

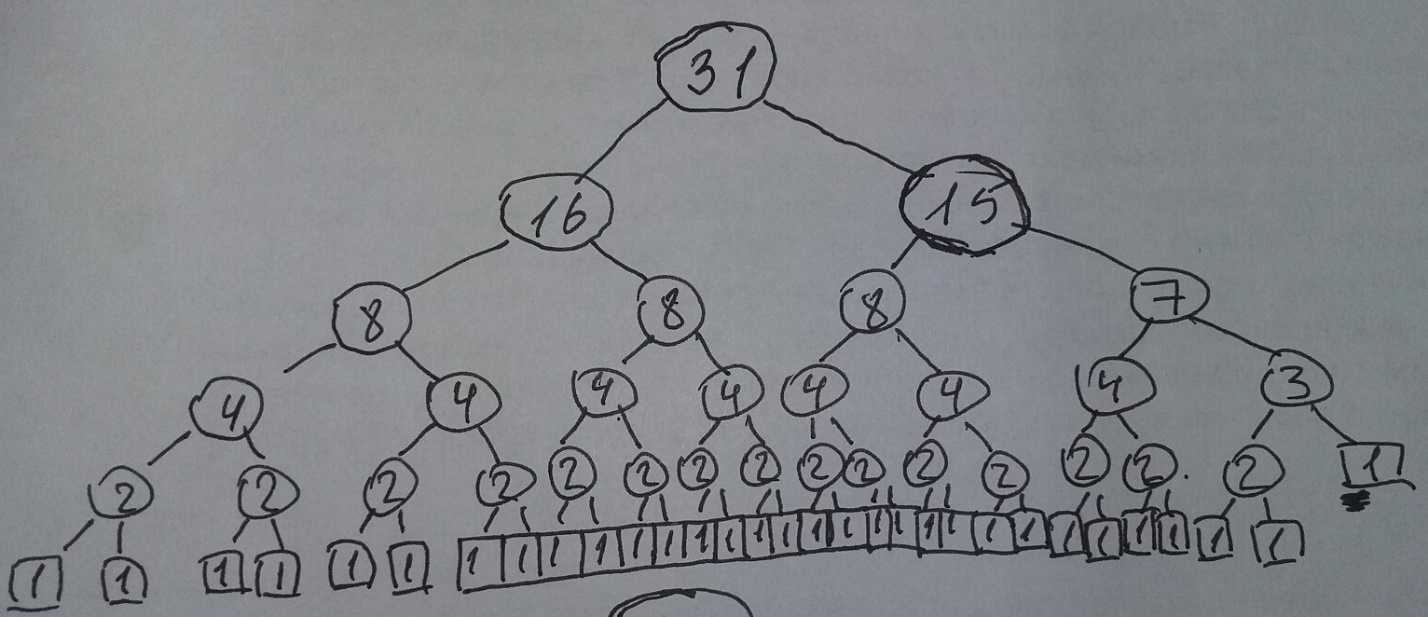
std::cout << i << ' ' << j << "; ";

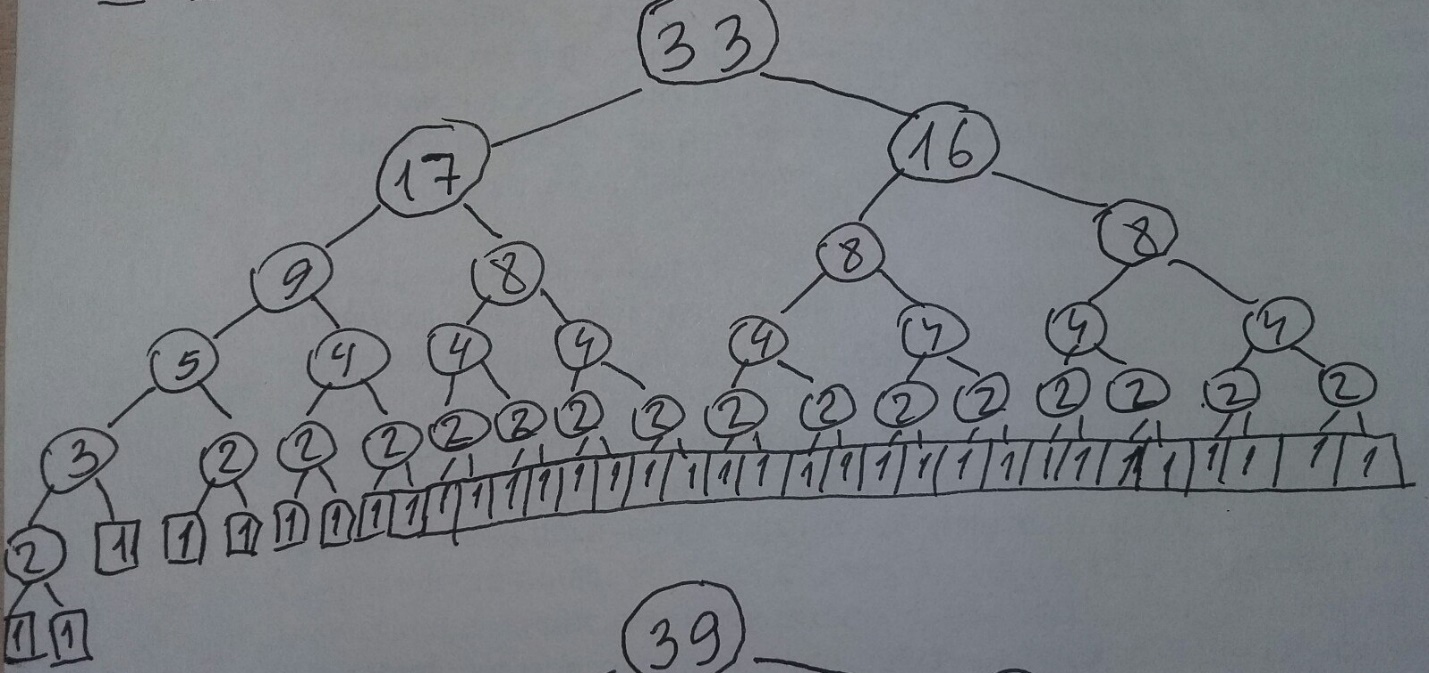
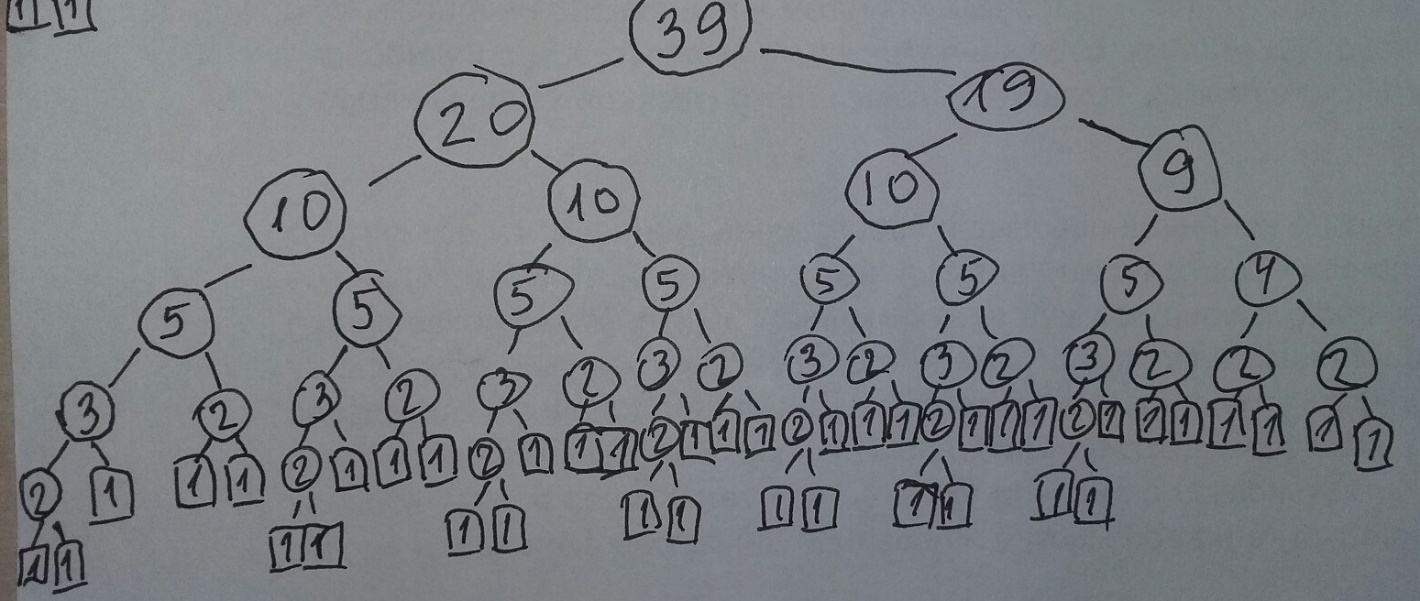
}  
  
  




№8.  
  
  


№9.  




  
  
  
  
№10.

Первая реализация:

#include <iostream>

#include <ctime>

#define SIZE 10

#define RANGE rand() % 10

void fillArray(int arr[]);

void output(int arr[]);

void merge(int arr[], int low, int high);

void mergeSort(int arr[], int left, int right);

int main()

{

int arr[SIZE];

fillArray(arr);

output(arr);

mergeSort(arr, 0, SIZE - 1);

output(arr);

system("PAUSE");

return 0;

}

void fillArray(int arr[]) {

srand(time(NULL));

for (int i = 0; i < SIZE; ++i)

arr[i] = RANGE;

}

void output(int arr[]) {

for (int i = 0; i < SIZE; ++i)

std::cout << arr[i] << ' ';

std::cout << std::endl;

}

void merge(int arr[], int low, int high) {

int i, j, t, m;

int arr2[SIZE];

i = low;

m = low + (high - low) / 2;

j = m + 1;

t = 0;

while (i <= m && j <= high) {

if (arr[i] <= arr[j])

arr2[t++] = arr[i++];

else

arr2[t++] = arr[j++];

}

while (i <= m)

arr2[t++] = arr[i++];

while (j <= high)

arr2[t++] = arr[j++];

for (i = 0; i < t; ++i)

arr[low + i] = arr2[i];

}

void mergeSort(int arr[] ,int left, int right) {

int t;

if (left >= right)

return;

if (right - left == 1) {

if (arr[right] < arr[left]) {

t = arr[left];

arr[left] = arr[right];

arr[right] = t; // на слайде arr[j] = arr[i] ???

}

}

else {

mergeSort(arr, left, left + (right - left) / 2);

mergeSort(arr, left + (right - left) / 2 + 1, right);

merge(arr, left, right);

}

}



Вторая реализация (написал на Python):

def merge(B, C):

res = []

while B and C:

if B[-1] >= C[-1]:

res.append(B.pop())

else:

res.append(C.pop())

res.reverse()

res = B + C + res

print(res)

return res

def mergeSort(A):

if len(A) < 2:

return A

mid = len(A) // 2

B = mergeSort(A[:mid])

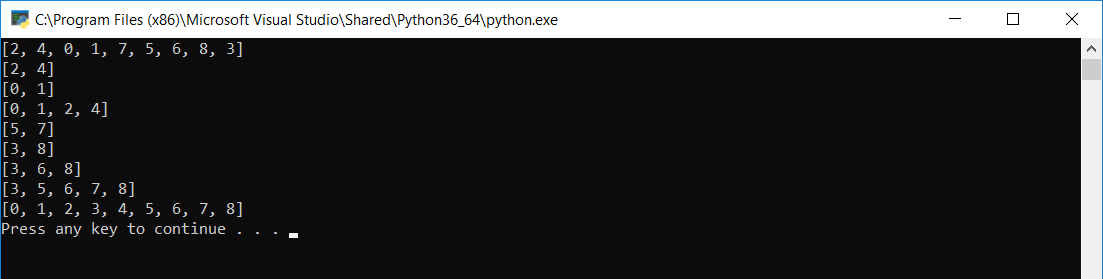
C = mergeSort(A[mid:])

return merge(B, C)

A = [2, 4, 0, 1, 7, 5, 6, 8, 3]

print(A)

mergeSort(A)



Сравнил

№11.  
#include <iostream>

#define SIZE 5

void output(int(&arr)[SIZE][SIZE]);

void solve(int(&arr)[SIZE][SIZE]);

int main()

{

int arr[SIZE][SIZE] = { {0, 1, 0, 0, 1}, {0, 0, 0, 1, 0}, {1, 0, 0, 0, 0}, {0, 1, 0, 0, 0}, {0, 0, 1, 0, 0} };

output(arr);

solve(arr);

output(arr);

system("PAUSE");

return 0;

}

void output(int(&arr)[SIZE][SIZE]) {

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j)

std::cout << arr[i][j] << ' ';

std::cout << std::endl;

}

std::cout << std::endl;

}

void solve(int(&arr)[SIZE][SIZE]) {

for (int m = 0; m < SIZE; ++m) {

for (int i = 0; i < SIZE; ++i) {

for (int j = 0; j < SIZE; ++j) {

arr[i][j] = arr[i][j] || (arr[i][m] && arr[m][j]);

std::cout << '\t' << arr[i][j] << std::endl;

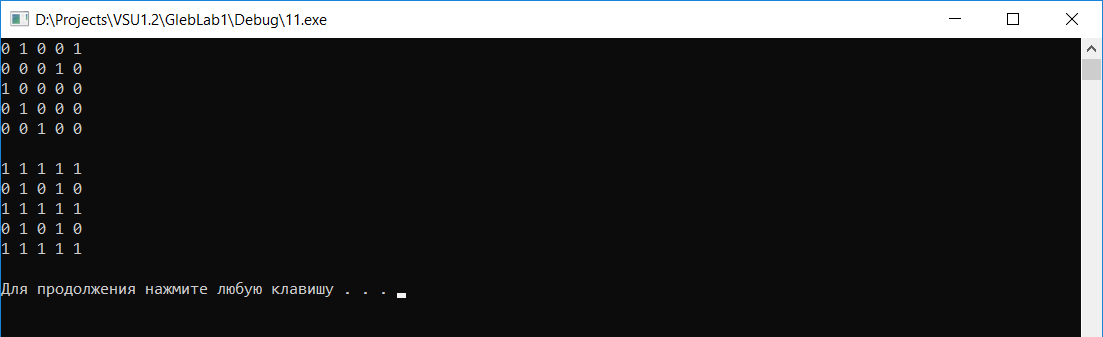
}

std::cout << std::endl;

}

}

}



№12.  
Первая реализация имеет экспоненциальную временную сложность, так как каждый рекурсивный вызов порождает два других.  
Вторая реализация имеет линейную временную сложность.  
Третья реализация имеет экспоненциальную временную сложность.  
Четвертая реализация имеет линейную временную сложность.

№13.

Вместо символов @ взято большое значение 99999 для того чтобы диагональные элементы не брались в рассчет

#include <iostream>

#define N 6

void output(int(&arr)[N][N]);

void solve(short v, short count, int cost);

int bestCost = 1000;

short bestWay[N];

short way[N] = { 0 };

bool nnew[N];

int arr[N][N] = {

{99999, 27, 43, 16, 30, 26},

{7, 99999, 16, 1, 30, 25},

{20, 13, 99999, 35, 5, 0},

{21, 16, 25, 99999, 18, 18},

{12, 46, 27, 48, 99999, 5},

{23, 5, 5, 9, 5, 99999} };

int main()

{

for (int i = 0; i < N; ++i)

nnew[i] = true;

output(arr);

solve(1, 1, 0);

std::cout << bestCost << std::endl;

for (int i = 0; i < N; ++i)

std::cout << bestWay[i] << '\t';

std::cout << std::endl;

system("PAUSE");

return 0;

}

void output(int(&arr)[N][N]) {

for (int i = 0; i < N; ++i) {

for (int j = 0; j < N; ++j)

std::cout << arr[i][j] << '\t';

std::cout << std::endl;

}

std::cout << std::endl;

}

void solve(short v, short count, int cost) {

nnew[v - 1] = false;

if (cost >= bestCost)

return;

if (count == N) {

cost += arr[v - 1][0];

way[count - 1] = v;

if (cost < bestCost) {

bestCost = cost;

for (int i = 0; i < N; ++i)

bestWay[i] = way[i];

return;

}

}

way[count - 1] = v;

for (int i = 0; i < N; ++i)

if (nnew[i]) {

nnew[i] = false;

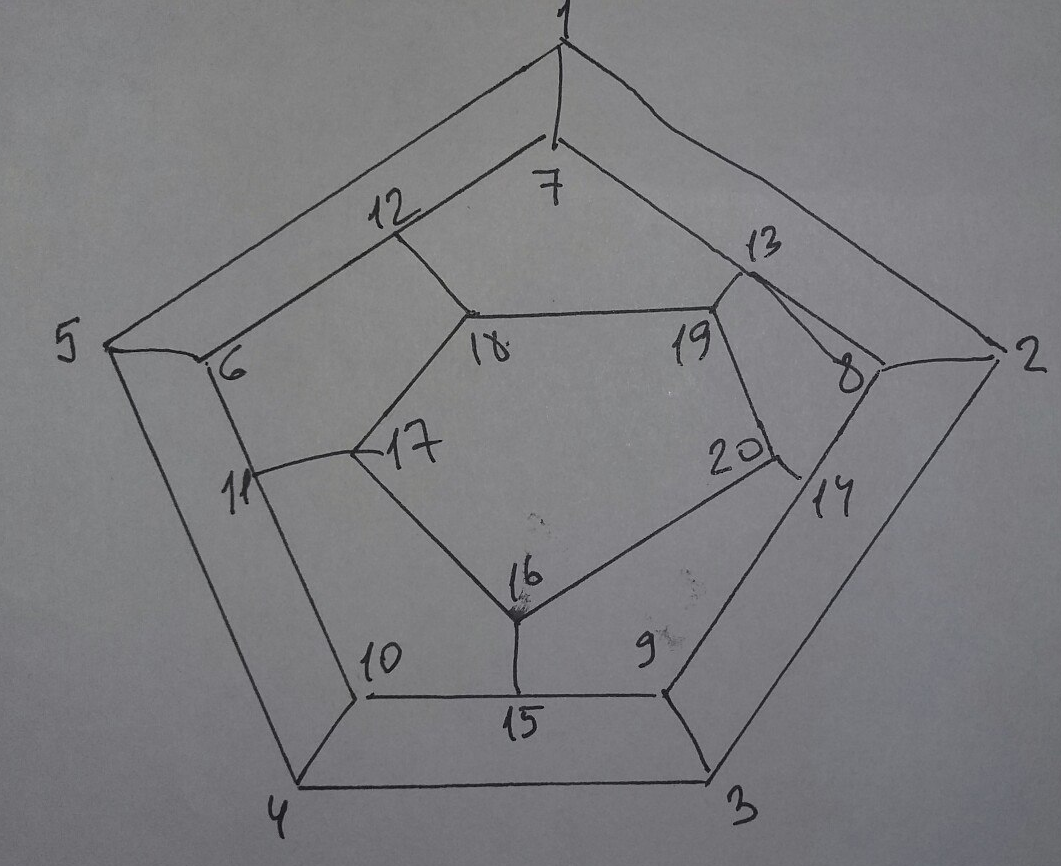
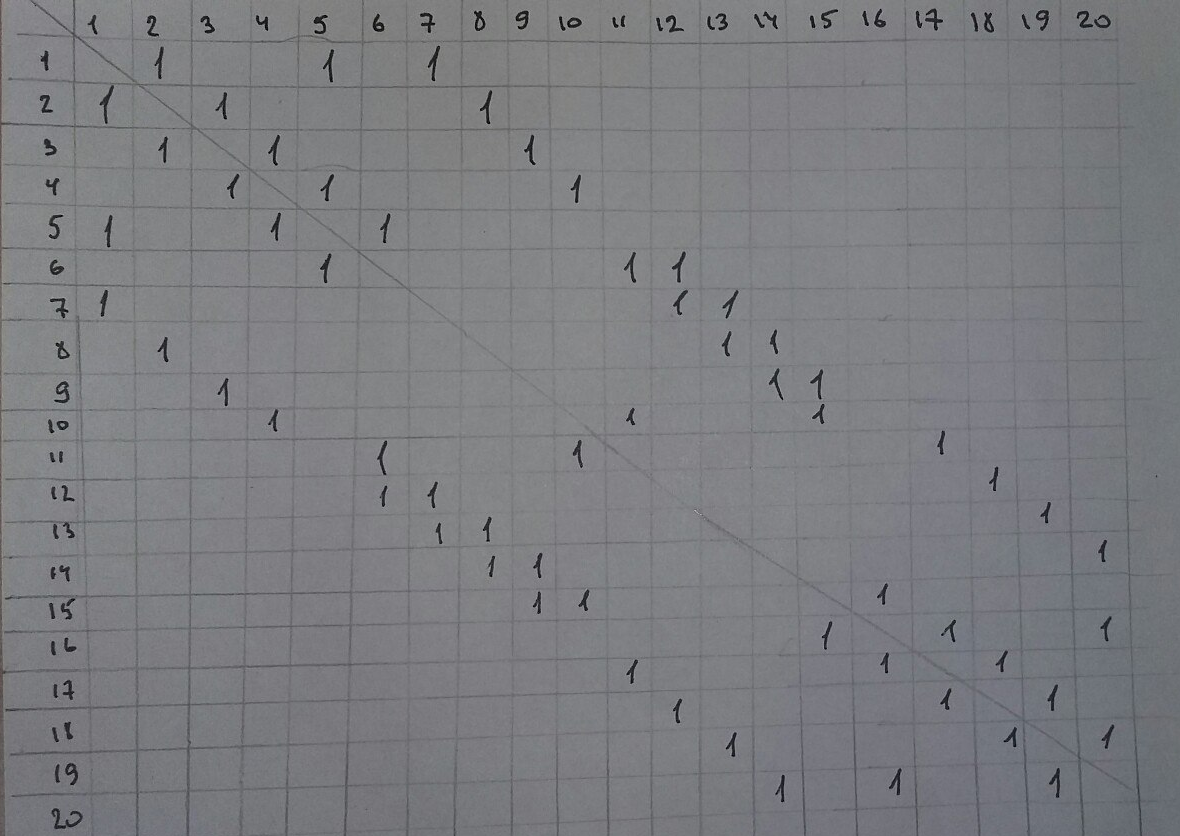
solve(i + 1, count + 1, cost + arr[v - 1][i]);

nnew[i] = true;

}

return;

}  


№14.   
  


Написал на Python (доделал, ответ получился верный):

def solve(v, count, cost):

global c

nnew[v - 1] = False

if cost > n:

return

if count == n:

cost += A[v - 1][0]

way[count - 1] = v

wayCheck = way[1:]

if A[t - 1][way[-1] - 1] != 9999 and check(wayCheck[::-1]):

store.append(wayCheck)

c += 1

print("WAY(", c, "):\t", way)

way[count - 1] = v

for i in range(n):

if nnew[i]:

nnew[i] = False

solve(i + 1, count + 1, cost + A[v - 1][i])

nnew[i] = True

return

def check(wayCheck):

have = False

for i in range(len(store)):

if wayCheck == store[i]:

have = True

if have == False:

return True

else:

return False

n = 20

A = [

[9999, 1, 9999, 9999, 1, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[1, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[9999, 1, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[9999, 9999, 1, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[1, 9999, 9999, 1, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999],

[9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999],

[9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999],

[9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999],

[9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999],

[9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999],

[9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 1, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 1, 9999, 9999, 1],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 1, 9999, 9999],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 1, 9999],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 9999, 9999, 9999, 1, 9999, 1],

[9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 9999, 1, 9999, 1, 9999, 9999, 1, 9999]

];

nnew = [True]\*n

way = [0]\*n

wayCheck = [0]\*n

store = []

c = 0

t = int(input("Enter the V: "))

for i in range(n):

for j in range(n):

if A[i][j] == 1:

print("1, ", end="")

else:

print("0, ", end="")

print()

print()

solve(t, 1, 0)

print('\n', c, '\n')

