**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

**КУРСОВА РАБОТА**

**№1**

**по Изкуствен интелект**

Студент:

Факултетен номер:

Група:

Специалност:

Курс:

**Изготвил:**

**Дата: Проверил:**

# Задание.

“Фермер, вълк, коза и зелка” (Farmer, wolf, goat and cabbage problem):

Фермер с вълк, коза и зелка се намират на левия бряг на една река, която искат да прекосят. Целта на задачата е всички да бъдат прехвърлени на десния бряг на реката като се използва лодка. Ограниченията са следните:

1. Лодката е двуместна, като само фермерът може да гребе.
2. На кой да е бряг, вълкът не трябва да бъде оставен сам (без фермера) с козата и козата - сама със зелето.

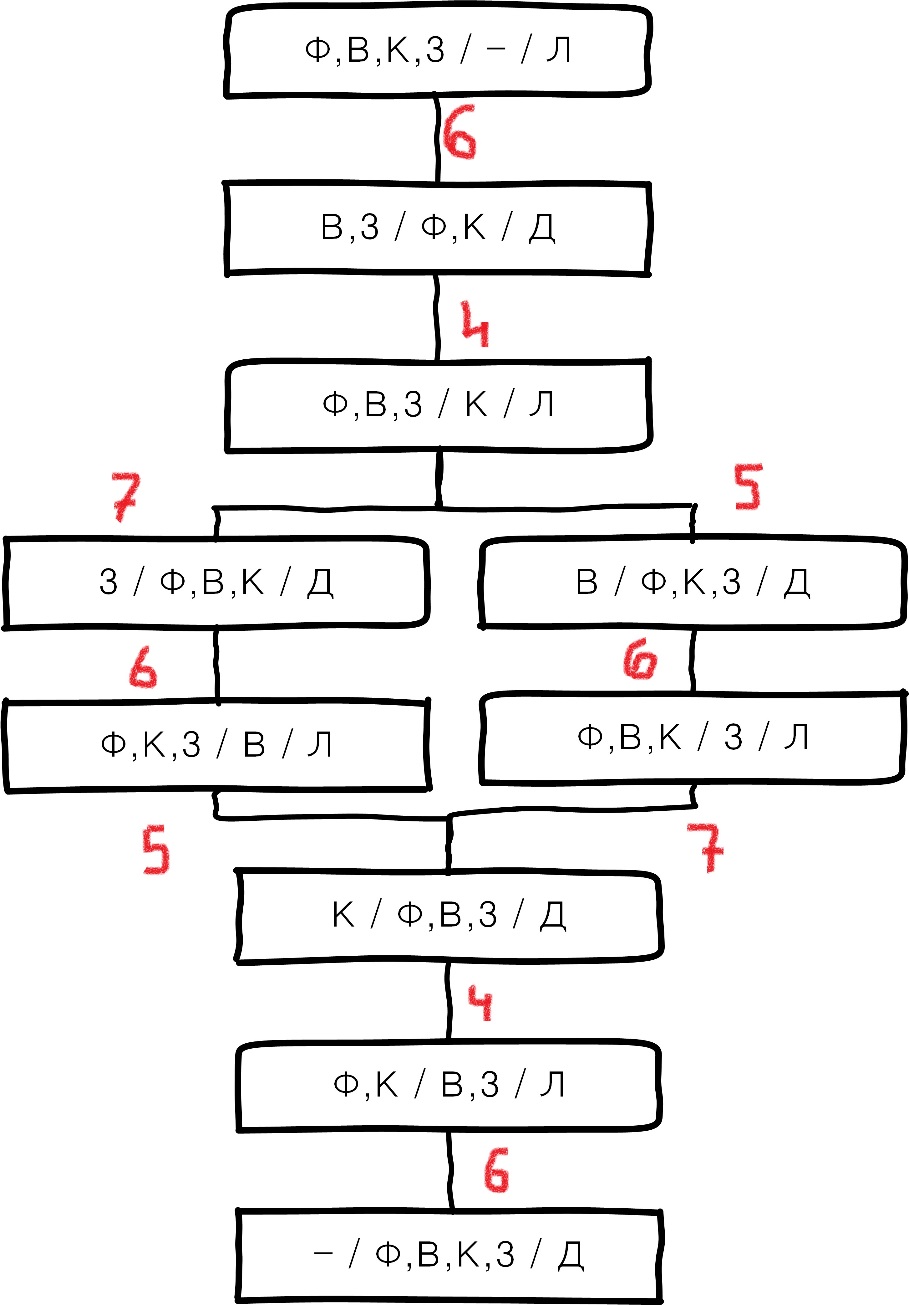
Нека разходите за превозването на различните пасажери са различни.

Алгоритъм: Depth first search (DFS).

# Представяне на задачата като задача за търсене в ПС.

* **Състояние** – Нека състоянието на тази задача да се представи чрез наредена тройка (α1, α2, α3), където α1 и α2 описват кои от героите са съответно на левия и десния бряг, а α3 показва къде се намира лодката. Ф – фермер, В – вълк, К – коза, З – зеле, Л – лодката е на левият бряг, Д – лодката е на десния бряг на реката.
* **Начално състояние** – (Ф,В,К,З / - / Л).
* **Финално състояние** – (- / Ф,В,К,З / Д).
* **Оператори и разходи на стрелките** –
* прехвърляне на фермера – 4 единици;
* прехвърляне на вълка – 3 единици;
* прехвърляне на козата – 2 единици;
* прехвърляне на зелето – 1 единици;

# Граф на ПС.



# Програмна реализация.

## Псевдо-код на алгоритъма.

path[n], count;

procedure **dfs(START, GOAL)**;

begin

if (START = GOAL) begin

print(path);

exit;

end

маркиране на посетения връх START;

path[count++] = START; {върхът START участва в пътя}

рекурсия за всички съседи на START, които не са маркирани;

размаркиране на посетения връх START

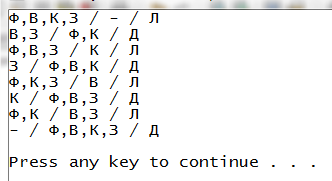
count--;

end; {dfs}

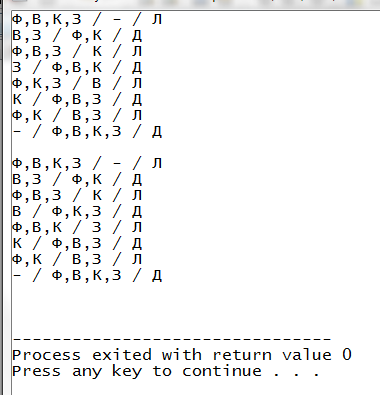
## Описание на разработениете модули.

* **Вид на функцията**: void DFS(unsigned i, unsigned j)
* **Действие**: Функцията служи за намиране на път между върховете i и j, чрез *Depth-first search* алгоритъма;
* **Параметри**: unsigned i, j – номерата съответно на началния и крайния връх;
* **Използвани функции**: рекурсия
* **Вид на функцията**: int main(void)
* **Действие**: Главна функция
* **Параметри**: няма;
* **Използвани функции**: DFS().

## Тестови примери.



Фигура 1 Намиране на 1 път



Фигура 2 Намиране на всички прости пътища

## Код на програмата.

**#include <iostream>**

**#include <vector>**

**#include <stdlib.h>**

**using namespace std;**

**const unsigned n = 10;**

**int graph[n][n] =**

**{**

**{ 0, 6, 0, 0, 0, 0, 0, 0, 0, 0 }, //0 | Ф,В,К,З / - / Л**

**{ 6, 0, 4, 0, 0, 0, 0, 0, 0, 0 }, //1 | В,З / Ф,К / Д**

**{ 0, 4, 0, 7, 5, 0, 0, 0, 0, 0 }, //2 | Ф,В,З / К / Л**

**{ 0, 0, 7, 0, 0, 6, 0, 0, 0, 0 }, //3 | З / Ф,В,К / Д**

**{ 0, 0, 5, 0, 0, 0, 6, 0, 0, 0 }, //4 | В / Ф,К,З / Д**

**{ 0, 0, 0, 6, 0, 0, 0, 5, 0, 0 }, //5 | Ф,К,З / В / Л**

**{ 0, 0, 0, 0, 6, 0, 0, 7, 0, 0 }, //6 | Ф,В,К / З / Л**

**{ 0, 0, 0, 0, 0, 5, 7, 0, 4, 0 }, //7 | К / Ф,В,З / Д**

**{ 0, 0, 0, 0, 0, 0, 0, 4, 0, 6 }, //8 | Ф,К / В,З / Л**

**{ 0, 0, 0, 0, 0, 0, 0, 0, 6, 0 } //9 | - / Ф,В,К,З / Д**

**};**

**string lblStates[n] =**

**{**

**"Ф,В,К,З / - / Л", "В,З / Ф,К / Д", "Ф,В,З / К / Л",**

**"З / Ф,В,К / Д", "В / Ф,К,З / Д", "Ф,К,З / В / Л",**

**"Ф,В,К / З / Л", "К / Ф,В,З / Д", "Ф,К / В,З / Л",**

**"- / Ф,В,К,З / Д"**

**};**

**char used[n];**

**unsigned path[n], count;**

**// Намира всички прости пътища между върховете i и j**

**void DFS(unsigned i, unsigned j)**

**{**

**unsigned k;**

**if (i == j) {**

**path[count] = j;**

**for (unsigned ind = 0; ind <= count; ind++)**

**printf("%s\n", lblStates[path[ind]].c\_str());**

**printf("\n");**

**system("pause");**

**exit(1);**

**}**

**// маркиране на посетения връх**

**used[i] = 1;**

**path[count++] = i;**

**for (k = 0; k < n; k++) // рекурсия за всички съседи на i**

**if (graph[i][k] && !used[k])**

**DFS(k, j);**

**// връщане: размаркиране на посетения връх**

**used[i] = 0;**

**count--;**

**}**

**int main(void) {**

**setlocale(LC\_ALL, "BGR");**

**unsigned sv = 0, ev = 9;**

**DFS(sv, ev);**

**cout << endl;**

**return 0;**

**}**

# Творческа задача.

Необходимо е да се премахне логиката за прекратяване на търсенето след като е намерен крайният връх:

if (i == j) {

path[count] = j;

for (unsigned ind = 0; ind <= count; ind++)

printf("%s\n", States[path[ind]].c\_str());

printf("\n");

~~system("pause");~~

~~exit(1);~~

return;

}