**РУСЕНСКИ УНИВЕРСИТЕТ “АНГЕЛ КЪНЧЕВ”**

**КУРСОВА РАБОТА**

**по Изкуствен интелект**

Студент:

Факултетен номер:

Група:

Специалност:

Курс:

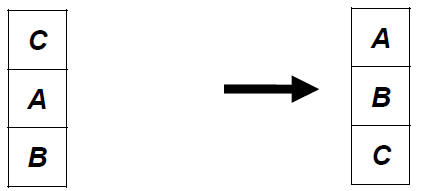
**Изготвил:**

**Дата: Проверил:**

# Задание.

**“Подредба на кубчета” (Block-world problem):**

Три кубчета A, B и C са подредени едно върху друго така както е показано на фиг.1. Целта е да се направи план за извършване на преподреждането им.



Фигура 1 Подредба на кубчета

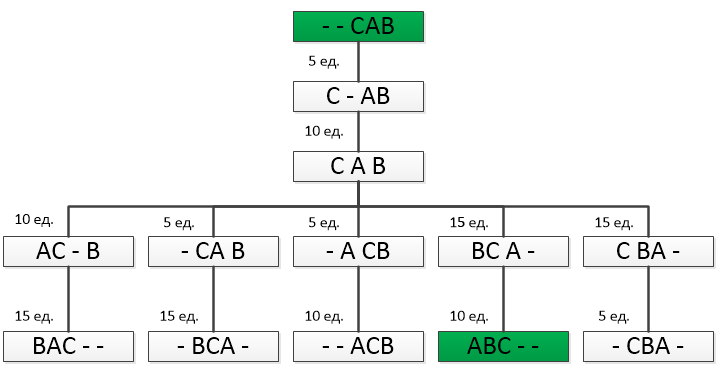
В даден момент е разрешено местенето само на едно кубче. Може да бъде премествано кубче, върху което няма друго такова. Кубчето може да бъде поставяно върху масата или върху друго такова. Нека кубчетата са с различна тежест, т.е. разходите за местенето им са различни.

Алгоритъм: търсене с връщане [*Backtracking*].

1. **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, α2 и α3 са трите места, където могат да се сложат кубчетата.
* **Начално състояние**: (–,–, CAB).
* **Финално състояние**: (ABC,–, –).
* **Оператори и разходи на стрелките**:
* **A** – прехвърляне на кубчето А (10 ед.);
* **B** – прехвърляне на кубчето B (15 ед.);
* **C** – прехвърляне на кубчето C (5 ед.);

# Граф на ПС.



# Програмна реализация.

## Псевдо-код на алгоритъма.

void опит(стъпка i)

{

/\* разширяване \*/

for (k = 0; k < n; k++)

if (k-тият\_кандидат\_е\_приемлив) {

регистрация\_на\_кандидата;

опит(k);

премахване\_на\_регистрацията;

if (k == GOAL) върхът\_е\_намерен;

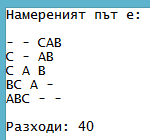
}

}

## Описание на разработениете модули.

* **Вид на функцията**: backtracking(unsigned i)
* **Действие**: Функцията служи за намиране на път между два върха в един граф, описан чрез матрица на съседство, чрез *DFS with Backtracking* алгоритъма;
* **Параметри**: i – номерът на началния връх;
* **Използвани функции**: рекурсия
* **Вид на функцията**: int main(void)
* **Действие**: Главна функция
* **Параметри**: няма;
* **Използвани функции**: backtracking ().

## Тестови примери.



Фигура 2 Тестов пример

## Код на програмата.

#include <stdlib.h>

#include <iostream>

using namespace std;

#define MAXN 20 /\* Максимален брой върхове в графа \*/

const unsigned n = 13; /\* Брой върхове в графа \*/

const unsigned ev = 11; /\* Краен връх \*/

unsigned sv; /\* Начален връх \*/

/\* Матрица на теглата на графа \*/

const char A[MAXN][MAXN] = {

{ 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, //0 - - CAB

{ 5, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, //1 C - AB

{ 0, 10, 0, 10, 5, 5, 15, 15, 0, 0, 0, 0, 0 }, //2 C A B

{ 0, 0, 10, 0, 0, 0, 0, 0, 15, 0, 0, 0, 0 }, //3 AC - B

{ 0, 0, 5, 0, 0, 0, 0, 0, 0, 15, 0, 0, 0 }, //4 - CA B

{ 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 10, 0, 0 }, //5 - A CB

{ 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 10, 0 }, //6 BC A -

{ 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5 }, //7 C BA -

{ 0, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0, 0 }, //8 BAC - -

{ 0, 0, 0, 0, 15, 0, 0, 0, 0, 0, 0, 0, 0 }, //9 - BCA -

{ 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0 }, //10 - - ACB

{ 0, 0, 0, 0, 0, 0, 10, 0, 0, 0, 0, 0, 0 }, //11 ABC - -

{ 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0 } //12 - CBA -

};

const string States[n] = {

"- - CAB", "C - AB",

"C A B", "AC - B",

"- CA B", "- A CB",

"BC A -", "C BA -",

"BAC - -", "- BCA -",

"- - ACB", "ABC - -",

"- CBA -"

};

unsigned vertex[MAXN], savePath[MAXN];

char used[MAXN];

int maxLen, tempLen, si, ti;

void backtracking(unsigned i) {

unsigned j, k;

for (k = 0; k < n; k++) {

if (!used[k]) { /\* ако върхът k не участва в пътя до момента \*/

/\* ако върхът, който добавяме, е съседен на последния от пътя \*/

if (A[i][k] > 0) {

tempLen += A[i][k];

used[k] = 1; /\* маркираме k като участващ в пътя \*/

vertex[ti++] = k; /\* добавяме върха k към пътя \*/

backtracking(k);

used[k] = 0; /\* връщане от рекурсията \*/

tempLen -= A[i][k];

ti--;

if (k == ev) { /\* върхът е намерен \*/

maxLen = tempLen + A[i][k];

for (j = 0; j <= ti; j++)

savePath[j] = vertex[j];

si = ti;

}

}

}

}

}

int main(void) {

setlocale(LC\_ALL, "BGR");

unsigned i, ch;

do {

cout << endl << "[1]. ABC";

cout << " [2]. ACB" << endl;

cout << "[3]. BAC";

cout << " [4]. BCA" << endl;

cout << "[5]. CAB";

cout << " [6]. CBA" << endl;

cout << "[0]. Изход" << endl;

cout << "Изберете начален връх: ";

cin >> ch;

system("cls");

switch(ch) {

case 1: sv = 11; break;

case 2: sv = 10; break;

case 3: sv = 8; break;

case 4: sv = 9; break;

case 5: sv = 0; break;

case 6: sv = 12; break;

case 0: ch = 0; break;

default: ch = -1; break;

}

if (ch != 0 && ch != -1) {

if (sv == ev) {

cout << "Кубчетата са подредени!" << endl;

}

else {

maxLen = 0; tempLen = 0; si = 0; ti = 1;

for (i = 0; i < n; i++) {

used[i] = 0;

vertex[i] = 0;

}

used[sv] = 1; vertex[0] = sv;

backtracking(sv);

cout << "Намереният път е:" << endl << endl;

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

cout << States[savePath[i]] << endl;

cout << endl << "Разходи: " << maxLen << endl;

}

}

} while(ch != 0);

return 0;

}

# Творческа задача.

Така както е представен графът има само едно.