МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Практична робота №2

з курсу “Штучний інтелект”

Варіант 15

Виконала: студент 4 курсу

групи КА-41

Пилипенко Д.С.

Київ – 2017р.

**Завдання:** Існує транспортна мережа між містами СНД. Мережа наведена у вигляді таблиці

зв’язків між містами (див. додаток 1). Зв’язки є двосторонніми, тобто передбачають рух у

двох напрямах. Потрібно побудувати маршрут проїзду із одного міста в інше. Відома

топологія зв’язків між містами. Виконати:

1) пошук у ширину;

2) пошук у глибину;

3) пошук з обмеженням глибини;

4) пошук у глибину з ітераційним збільшенням глибини;

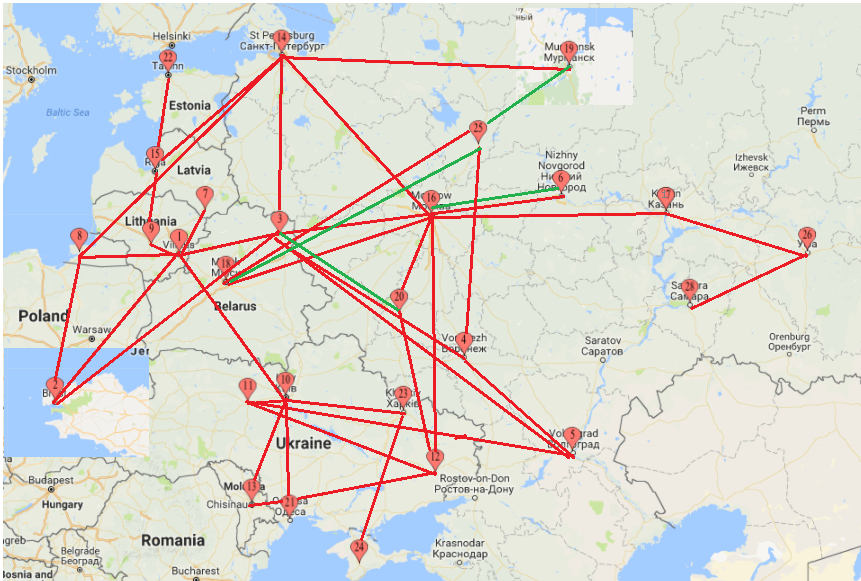
5) двонаправлений пошук.

Зобразити рух по дереву пошуку на його графі та вказати складність кожного виду

пошуку. Зробити висновки.

|  |  |  |
| --- | --- | --- |
| Номер варіанту | Пункт відправлення | Пункт призначення |
| 15 | Санкт-Петербург | Одеса |

**Відповідь:** Візуалізація графу:



Позначення міст через числа для зручності:

Вільнюс - 1

Брест -2

Вітебськ - 3

Вороніж - 4

Волгоград - 5

Нижній Новгород - 6

Даугавпілс - 7

Калінінград - 8

Каунас - 9

Київ – 10

Житомир - 11

Донецьк - 12

Кишинів -13

Санкт-Петербург - 14

Рига – 15

Москва - 16

Казань - 17

Мінськ - 18

Мурманськ - 19

Орел - 20

Одеса - 21

Таллінн -22

Харків - 23

Сімферополь - 24

Ярославль - 25

Уфа - 26

Самара – 27

**Програмна реалізація:**

import numpy as np

coordinates = [(1 ,2 ), (3 ,2 ), (3 ,1 ),(4 ,3 ), (5 ,4 ), (5 ,3 ),(3 ,6 ),

(1 ,7 ), (8 ,2 ),(8 ,1 ), (9 ,1 ), (10 ,1 ),(10 ,11 ), (11 ,12 ),

(11 ,5 ),(13 ,10 ), (13 ,12 ), (14 ,3 ),(14 ,8 ), (14 ,15 ), (16 ,17 ),

(16 ,18 ), (16 ,12 ), (16 ,6 ),(16 ,14 ), (19 ,14 ), (19 ,18 ),(20 ,3 ),

(20 ,12 ), (20 ,16 ),(21 ,10 ), (15 ,9 ), (22 ,15 ),(23 ,10 ), (23 ,24 ),

(25 ,4 ),(25 ,18 ), (26 ,17 ), (26 ,27 )]

x, y = [i[0] for i in coordinates], [i[1] for i in coordinates]

maxi = max(max(x), max(y))

image = np.zeros((maxi + 1, maxi + 1))

for i in range(len(coordinates)):

image[y[i], x[i]] = 1

image[x[i], y[i]] = 1

#Поиск в ширину

def bfs(st, en):

lvl = []

sl = []

lvl.append([st])

sl.append([st])

l = 0

goin=True

while goin:

le = []

les = []

for el in lvl[l]:

for i in range(maxi):

if (image[el,i]==1) and (not(i in le)):

le.append(i)

les.append(str(el) + "->" + str(i))

lvl.append(le)

sl.append(les)

l += 1

goin = not(en in le)

if l>10:

goin = False

print("Searched item is on the " + str(l) + " level")

print("Here are the levels:")

print(sl)

print("\nTask 1:Поиск в ширину")

bfs(14,21)

#Поиск в глубину

def dfsns(st,en):

lvl = [] #lvl is now the pool of visited leaves

sl = []

l=0

el = st

lvl.append(st)

goin = True

while goin: #building chain

l += 1

m = True

s=2

while m: #searching next element of chain

for i in range(maxi):

if (image[el,i]==1) and (not(i in lvl)) and m:

sl.append(str(el) + "->" + str(i))

el = i

lvl.append(i)

m = False

if m:

el = lvl[-s]

s += 1

if (el == en) or (l > 100):

goin = False

print("Searched item is on the " + str(l) + " position")

print("Here is the path:")

print(sl)

print("\nTask 2:Поиск в глубину")

dfsns(14,21)

#Поиск в глубину с остановкой

def dfsn(st, en, z, p):

lvl = []

sl = []

g = False

lvl.append([st])

sl.append([st])

l = 0

goin = True

while goin:

le = []

les = []

for el in lvl[l]:

for i in range(maxi):

if (image[el, i] == 1) and (not(i in le)):

le.append(i)

les.append(str(el) + "->" + str(i))

lvl.append(le)

sl.append(les)

l += 1

goin = not(en in le)

if not(goin):

print("searched item is on the "+str(l)+" level")

g = True

if l > z:

goin = False

if p:

print("here are the levels:")

print(sl)

return g, sl

print("\nTask 3:Поиск в глубину с лимитом")

dfsn(14, 21, 3, True)

#Поиск в глубину с итеративной остановкой

def dfsin(st, en):

lvl = [] #lvl is now the pool of visited leaves

sl = []

m = False

l = 1

while not(m):

g, sl = dfsn(st, en, l, False)

if g:

m = True

print("Here is the path:")

print(sl)

else:

l+=1

print("\nTask 4:Поиск в глубину с итеративной остановкой")

dfsin(14,21)

#Двосторонний поиск

def bfsds(st,en):

lvl1 = []

lvl2 = []

fa1=[]

fa2=[]

sl1 = []

sl2 = []

lvl1.append([st])

sl1.append([st])

fa1.append(st)

lvl2.append([en])

sl2.append([en])

fa2.append(en)

l=0

goin=True

while goin:

le1 = []

les1 = []

le2 = []

les2 = []

for el in lvl1[l]:

for i in range(maxi):

if (image[el,i]==1) and (not(i in le1)):

fa1.append(i)

le1.append(i)

les1.append(str(el) + "->" + str(i))

lvl1.append(le1)

sl1.append(les1)

for el in lvl2[l]:

for i in range(maxi):

if (image[el,i]==1) and (not(i in le2)):

fa2.append(i)

le2.append(i)

les2.append(str(el) + "->" + str(i))

lvl2.append(le2)

sl2.append(les2)

l+=1

goin=not(bool(set(fa1) & set(fa2)))

if l>10:

goin = False

print("Searched item is on the " + str(l) + " level")

print("Element(s) in between:")

print(list(set(fa1).intersection(fa2)))

print("Here are the levels:")

print(sl1, sl2)

print("\nTask 5:Двосторонний поиск")

bfsds(14, 21)

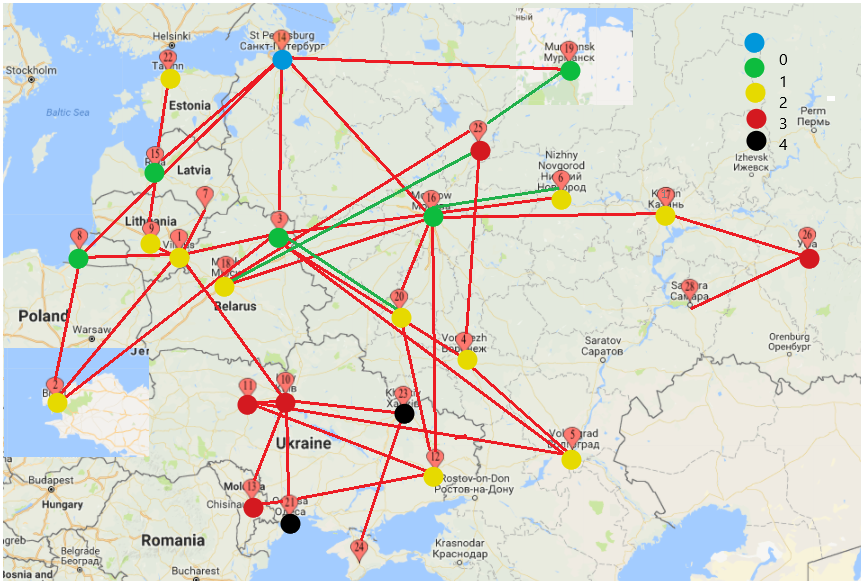
**Відповідь( вивід програми ):**

Task 1:Поиск в ширину

Searched item is on the 4 level

Here are the levels:

[[14], ['14->3', '14->8', '14->15', '14->16', '14->19'], ['3->1', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '15->9', '15->22', '16->12', '16->17', '16->18'], ['1->2', '1->3', '1->7', '1->8', '1->9', '1->10', '2->1', '4->5', '4->25', '5->4', '5->11', '6->16', '14->15', '14->19', '20->12', '12->13', '12->20', '17->26'], ['2->1', '2->3', '2->8', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '9->15', '10->11', '10->13', '10->21', '10->23', '1->7', '1->9', '1->10', '25->18', '4->25', '11->12', '16->17', '15->22', '12->16']]

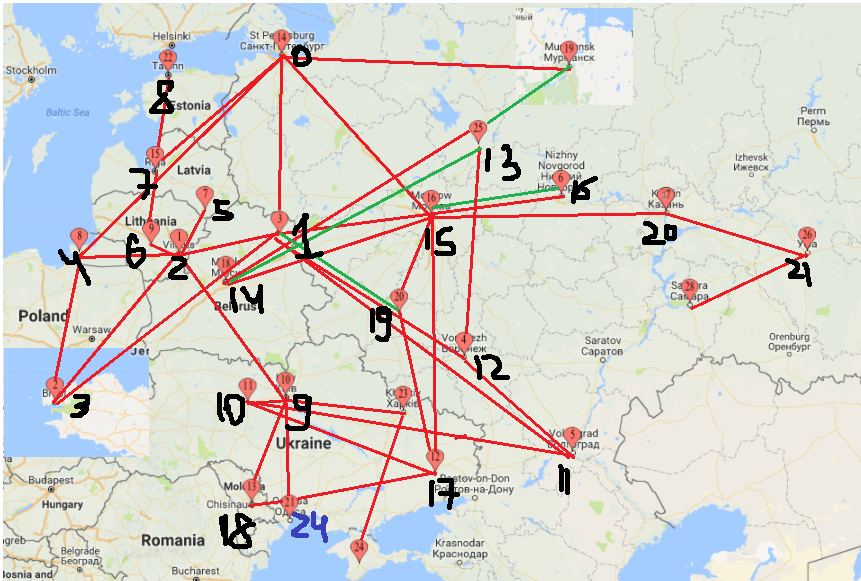


Task 2:Поиск в глубину

Searched item is on the 23 position

Here is the path:

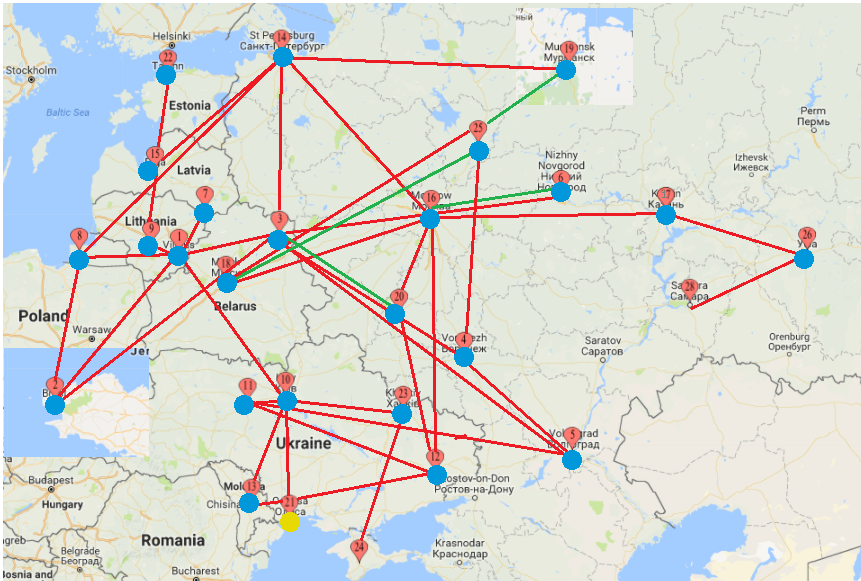
['14->3', '3->1', '1->2', '2->8', '1->7', '1->9', '9->15', '15->22', '1->10', '10->11', '11->5', '5->4', '4->25', '25->18', '18->16', '16->6', '16->12', '12->13', '12->20', '16->17', '17->26', '18->19', '10->21']



Task 3:Поиск в глубину с лимитом(3)

searched item is on the 4 level

here are the levels:

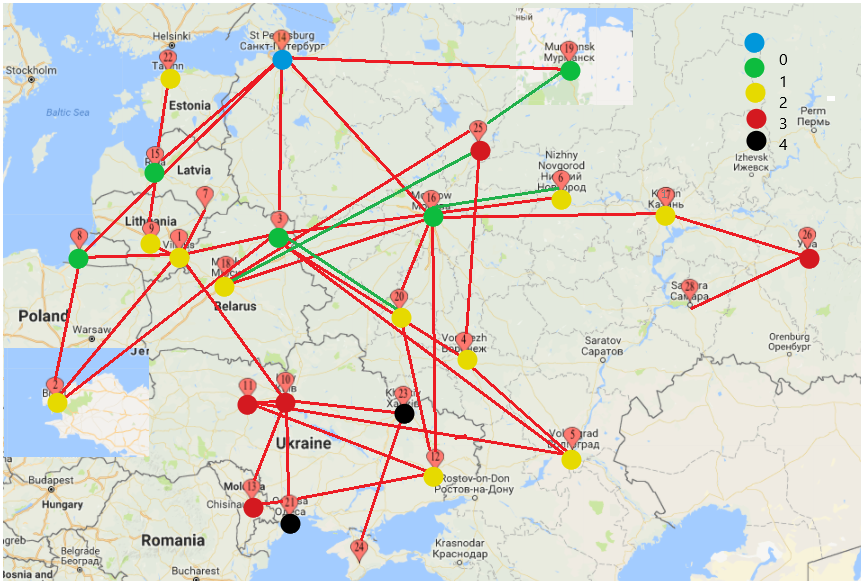
[[14], ['14->3', '14->8', '14->15', '14->16', '14->19'], ['3->1', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '15->9', '15->22', '16->12', '16->17', '16->18'], ['1->2', '1->3', '1->7', '1->8', '1->9', '1->10', '2->1', '4->5', '4->25', '5->4', '5->11', '6->16', '14->15', '14->19', '20->12', '12->13', '12->20', '17->26'], ['2->1', '2->3', '2->8', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '9->15', '10->11', '10->13', '10->21', '10->23', '1->7', '1->9', '1->10', '25->18', '4->25', '11->12', '16->17', '15->22', '12->16']]

Task 4:Поиск в глубину с итеративной остановкой

searched item is on the 4 level

Here is the path:

[[14], ['14->3', '14->8', '14->15', '14->16', '14->19'], ['3->1', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '15->9', '15->22', '16->12', '16->17', '16->18'], ['1->2', '1->3', '1->7', '1->8', '1->9', '1->10', '2->1', '4->5', '4->25', '5->4', '5->11', '6->16', '14->15', '14->19', '20->12', '12->13', '12->20', '17->26'], ['2->1', '2->3', '2->8', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '9->15', '10->11', '10->13', '10->21', '10->23', '1->7', '1->9', '1->10', '25->18', '4->25', '11->12', '16->17', '15->22', '12->16']]



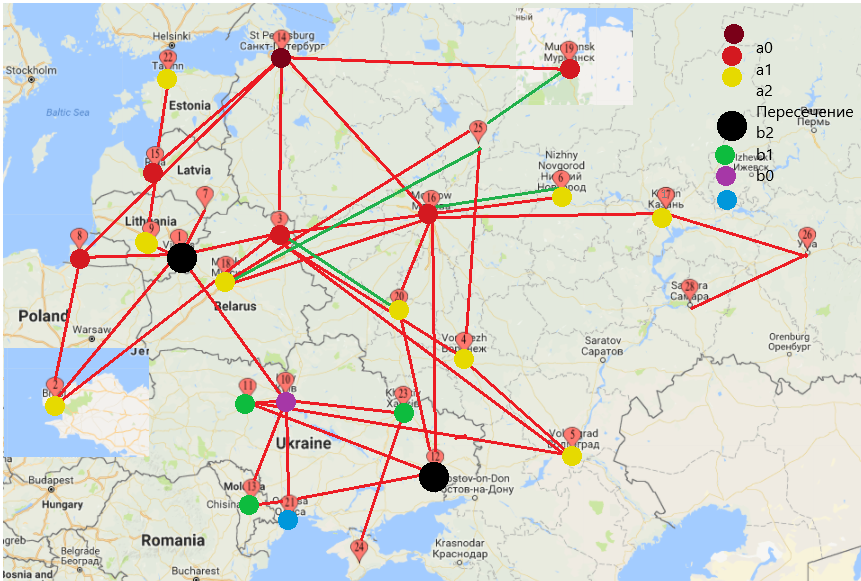
Task 5:Двосторонний поиск

Searched item is on the 2 level

Element(s) in between: [1]

Here are the levels:

[[14], ['14->3', '14->8', '14->15', '14->16', '14->19'], ['3->1', '3->2', '3->4', '3->5', '3->6', '3->14', '3->20', '15->9', '15->22', '16->12', '16->17', '16->18']] [[21], ['21->10'], ['10->1', '10->11', '10->13', '10->21', '10->23']]

****

**Оцінка складності стратегій неінформативного пошуку**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Стратегія | Повнота | Часова  складність | Витрати  пам’яті | Оптимальність |
| Пошук у ширину | Так | 729 | 729 | Так |
| Пошук у глибину | Ні | 243 | 78 | Ні |
| Пошук з обмеженням  глибини | Ні | 27 | 9 | Ні |
| Пошук з ітераційним  збільшенням глибини | Ні | 243 | 15 | Так |
| Двонаправлений пошук | Ні | 16 | 16 | Так |

B = 3– коефіцієнт розгалуження

D = 5 – глибина найбільш поверхового розв’язку

e = 3 – границя глибини

m = 5– максимальна глибина дерева

Висновок: Судячи з оцінок складності, найкращі показники у двонаправленому пошуку та пошуку з обмеженнями глибини. В ході виконання роботи я навчився будувати різні моделі пошуку на графі та дізнався який з них варто використовувати в різних випадках.