Программное решение 1ых задач

Курс подготовки к ЕГЭ по информатике 2026

Вступление

Идея решения в переборе всех возможных перестановок вершин графа, и проверки, что в таблице на каждое ребро есть значение. Перед тем как приступить к задаче подразумевается, что вы, само собой, знаете основы программирования, но помимо основ также рекомендую ознакомиться с:

— Библиотека itertools;

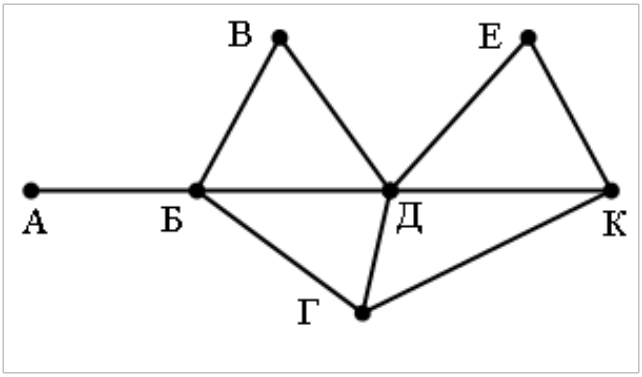
— Словарь;

— all/any.

ФИПИ А6ВА45

На рисунке справа схема дорог Н-ского района изображена в виде графа, в таблице содержатся сведения о протяжённости каждой из этих дорог (в километрах).

		Номер пункта						
		П1	П2	П3	П4	П5	П6	П7
Номер пункта	П1					13		8
	П2						14	9
	П3					16		
	П4					10	17	4
	П5	13		16	10			11
	П6		14		17			15
	П7	8	9		4	11	15	



Так как таблицу и схему рисовали независимо друг от друга, то нумерация населённых пунктов в таблице никак не связана с буквенными обозначениями на графе. Определите, какова протяжённость дороги из пункта Е в пункт К. В ответе запишите целое число – так, как оно указано в таблице.

Решение

Программное решение данной задачи является переборным. Перебираются все перестановки букв с графа. Порядок очередной буквы ставится в соответствии к номеру пункта в таблице. Проверяется, что все пересечения с графа существуют в таблице.

Приведем несколько примеров для задачи выше. Пусть изначальное состояние алфавита следующее.

АБВГДЕК

Подразумевается, что состояние таблицы соответствующее этому состоянию алфавита следующее.

Номер пункта								
		А	Б	В	Г	Д	Е	К
Номер пункта	А					13		8
	Б						14	9
	В					16		
	Г					10	17	4
	Д	13		16	10			11
	Е		14		17			15
	К	8	9		4	11	15	

Очевидно, что это неподходящая перестановка. Например, на рисунке пункты Б и В пересекаются, в таблице - нет.

Допустим состояние алфавита следующее.

АБВГДКЕ

Тогда таблица выглядит как.

Номер пункта								
		А	Б	В	Г	Д	К	Е
Номер пункта	А					13		8
	Б						14	9
	В					16		
	Г					10	17	4
	Д	13		16	10			11
	К		14		17			15
	Е	8	9		4	11	15	

Старая проблема осталась. Тоже не подходит. Возьмем перестановку, где Б и В пересекаются.

АДВГБЕК

Номер пункта								
		А	Д	В	Г	Б	Е	К
Номер пункта	А					13		8
	Д						14	9
	В					16		
	Г					10	17	4
	Б	13		16	10			11
	Е		14		17			15
	К	8	9		4	11	15	

Здесь уже проблема с пересечением, например, В и Д. Продолжать перебор всех перестановок вручную - дело гиблое, а вот делегировать этот процесс скрипту на python вполне рабочее решение. Для начала переберем все состояния алфавита.

```
1 from itertools import *
2 alph="АБВГДЕК"
3 for v in permutations(alph):
4     print(v)
```

Чтобы сопоставить строку/столбец таблицы с номером буквы создадим словарь, где ключем будет буква, а значением - соответствующий ей индекс.

```
from itertools import *
alph="АБВГДЕК"
for v in permutations(alph):
    dic={}
    for i in range(len(v)):
        dic[v[i]]=i
```

Далее можно составить матрицу, которая будет соответствовать таблице. На месте, где дорога отсутствует, ставим значение равное -1.

```
matr=[[-1,-1,-1,-1,13,-1, 8],
      [-1,-1,-1,-1,-1,14, 9],
      [-1,-1,-1,-1,16,-1,-1],
      [-1,-1,-1,-1,10,17, 4],
      [13,-1,16,10,-1,-1,11],
      [-1,14,-1,17,-1,-1,15],
      [ 8, 9,-1, 4,11,15,-1]]
```

А также перечислим пересечения между пунктами в виде пар букв.

```
pairs=["АБ","БВ","БГ","БД","ВД","ГД","ГК","ДЕ","ДК","ЕК"]
```

Тогда мы можем реализовать перебор всех пар и проверить, что для каждой из них существует пересечение внутри нашей матрицы.

```python
from itertools import *

matr=[[-1,-1,-1,-1,13,-1, 8],
      [-1,-1,-1,-1,-1,14, 9],
      [-1,-1,-1,-1,16,-1,-1],
      [-1,-1,-1,-1,10,17, 4],
      [13,-1,16,10,-1,-1,11],
      [-1,14,-1,17,-1,-1,15],
      [ 8, 9,-1, 4,11,15,-1]]

pairs=["АБ","ББ","БГ","БД","ВД","ГД","ГК","ДЕ","ДК","ЕК"]

alph="АБВГДЕК"

for v in permutations(alph):
    dic={}
    for i in range(len(v)):
        dic[v[i]]=i
    b=True
    for p in pairs:
        if matr[dic[p[0]]][dic[p[1]]] == -1:
            b=False
    if b:
        print(matr[dic["Е"]][dic["К"]])
```

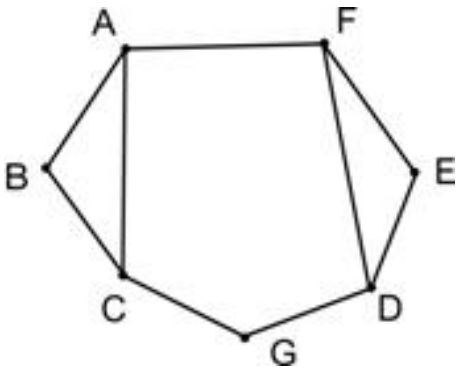
Ну и для любителей генераторов и all/any.

```python
from itertools import *

matr=[[-1,-1,-1,-1,13,-1, 8],
      [-1,-1,-1,-1,-1,14, 9],
      [-1,-1,-1,-1,16,-1,-1],
      [-1,-1,-1,-1,10,17, 4],
      [13,-1,16,10,-1,-1,11],
      [-1,14,-1,17,-1,-1,15],
      [ 8, 9,-1, 4,11,15,-1]]

pairs=["АБ","ББ","БГ","БД","ВД","ГД","ГК","ДЕ","ДК","ЕК"]

alph="АБВГДЕК"

for v in permutations(alph):
    dic={}
    for i in range(len(v)):
        dic[v[i]]=i
    if all([matr[dic[p[0]]][dic[p[1]]]>0 for p in pairs]):
        print(matr[dic["Е"]][dic["К"]])
```

ФИПИ 20495F

На рисунке слева изображена схема дорог Н-ского района, в таблице звёздочкой обозначено наличие дороги из одного населённого пункта в другой. Отсутствие звёздочки означает, что такой дороги нет.

Номер пункта								
		1	2	3	4	5	6	7
Номер пункта	1			*		*	*	
	2			*	*			*
	3	*	*					
	4		*			*		*
	5	*			*		*	
	6	*				*		
	7		*		*			



Каждому населённому пункту на схеме соответствует его номер в таблице, но неизвестно, какой именно номер. Определите, какие номера населённых пунктов в таблице могут соответствовать населённым пунктам A и F на схеме. В ответе запишите эти два номера в возрастающем порядке без пробелов и знаков препинания.

Решение

Решение будет выглядеть аналогично. В матрице там где звездочка присутствует поставим 1, а там где нет, поставим 0. И выводить будем не длину пересечения, которую всё равно не знаем, а номера соответствующих пунктов. Не забываем прибавить к ним 1, потому что нумерация в python начинается с 0.

```
from itertools import *

matr=[[0,0,1,0,1,1,0],
      [0,0,1,1,0,0,1],
      [1,1,0,0,0,0,0],
      [0,1,0,0,1,0,1],
      [1,0,0,1,0,1,0],
      [1,0,0,0,1,0,0],
      [0,1,0,1,0,0,0]]

pairs=["AB","AC","AF","BC","CG","DE","DF","DG","EF"]

alph="ABCDEFG"

for v in permutations(alph):
    dic={}
    for i in range(len(v)):
        dic[v[i]]=i
    b=True
    for p in pairs:
        if matr[dic[p[0]]][dic[p[1]]] == 0:
            b=False
    if b:
        print(dic["A"]+1,dic["F"]+1)
```

А также версия с all.

```
from itertools import *

matr=[[0,0,1,0,1,1,0],
      [0,0,1,1,0,0,1],
      [1,1,0,0,0,0,0],
      [0,1,0,0,1,0,1],
      [1,0,0,1,0,1,0],
      [1,0,0,0,1,0,0],
      [0,1,0,1,0,0,0]]

pairs=["AB","AC","AF","BC","CG","DE","DF","DG","EF"]

alph="ABCDEFG"

for v in permutations(alph):
    dic={}
    for i in range(len(v)):
        dic[v[i]]=i
    if all([matr[dic[p[0]]][dic[p[1]]]>0 for p in pairs]):
        print(dic["A"]+1,dic["F"]+1)
```