Дисциплина «Искусственный интеллект»

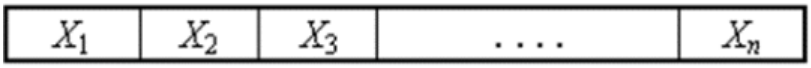
Рабочая тетрадь № 6

Теоретический материал – Эволюционные методы

Деревья решений являются одним из наиболее эффективных Эволюционные методы

Эволюционные методы относятся к числу эффективных средств решения задач оптимизации и структурного синтеза проектных решений. Они основаны на использовании принципов оптимального приспособления организмов в живой природе к условиям окружающей среды. К числу эволюционных относятся методы генетические, колонии муравьев, поведения толпы. Наиболее развиты и востребованы в настоящее время генетические алгоритмы. По мере развития техники и технологий растет доля сложных задач проектирования и управления, для решения которых применение традиционных методов проблематично. Поэтому все большее внимание уделяется применению методов искусственного интеллекта. Генетические алгоритмы Для применения ГА необходимо:

1. выделить совокупность свойств объекта, характеризуемых внутренними параметрами и влияющих на его полезность, т.е. выделить множество управляемых параметровX=(x_1,x_2,…,x_n) среди x_i могут быть величины различных типов (real, integer, Boolean, enumeration). Наличие нечисловых величин (enumeration) обусловливает возможность решения задач не только параметрической, но и структурной оптимизации;
2. сформулировать количественную оценку полезности вариантов объекта — функцию полезности F. Если в исходном виде задача многокритериальна, то такая формулировка означает выбор скалярного (обобщенного) критерия;
3. представить вектор X в форме хромосомы — записи следующего вида:



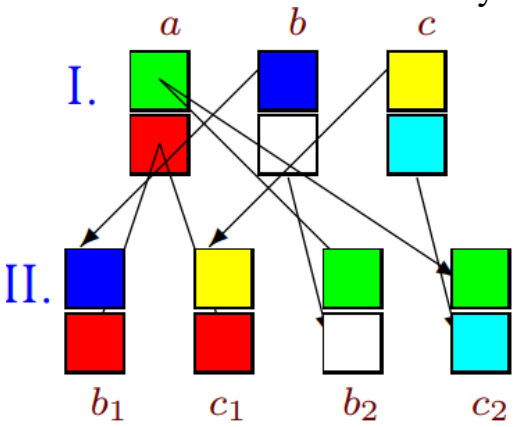
Этапы генетического алгоритма могут быть представлены в следующем виде:

```
for (k=0; k<G; k++)
{ for (j=0; j<N; j++)
  { Выбор родительской пары хромосом;
    Кроссовер;
    Мутации;
    Оценка функции полезности F потомков;
    Селекция;
  }
  Замена текущего поколения новым;
}
```

1.1.1 Пример

Задача:

Пусть дана начальная популяция из четырех хромосом с двумя генами x и y. Показатель качества хромосомы оценивается функцией Z. При равном качестве хромосом предпочтение отдается хромосоме с большим номером. На каждом этапе хромосома a с высшим качеством порождает четыре новых хромосомы b_1, c_1, b_2, c_2, обмениваясь генами с двумя хромосомами b и с более низкого качества по указанной схеме:



Последняя хромосома (с низшим качеством) выбывает из популяции. Найти максимальный показатель качества хромосомы в популяции и общее качество популяции после четырех этапов эволюции.

Потребуется несколько функций для реализации алгоритма. Напишем их.

Решение:

Начнем с функции оценки качества хромосомы qZ(x,y):

```
# функция качества хромосомы
def qZ(x, y):
    return (x - 3 * y + 1) / (3* x ** 2 + 3 * y ** 2 + 1)
```

Далее, оценим суммарное качество хромосом:

```
# сумма качества хромосом
def qSumZ(Z):
    return sum(Z)
```

И запрограммируем представленную выше схему обмена хромосомами:

```
def exchangeScheme(oldX, oldY, sortedId):
    X = [0 for i in range(4)]
    Y = [0 for i in range(4)]

    X[2] = oldX[sortedId[2]]
    X[3] = oldX[sortedId[2]]

    X[0] = oldX[sortedId[0]]

    X[1] = oldX[sortedId[1]]

    Y[0] = oldY[sortedId[2]]
    Y[1] = oldY[sortedId[2]]

    Y[2] = oldY[sortedId[0]]

    Y[3] = oldY[sortedId[1]]

    return X, Y
```

Отсортируем массив качества наших потомков и выделим полученные индексы:

```
def sorting(Z):
    sortedId = sorted(range(len(Z)), key = lambda k: Z[k])

    return sortedId
```

Напишем функцию для шага эволюции:

```
# шаг эволюции
def evoStep(X, Y, Z):
    _, minId = min((value, id) for (id, value) in enumerate(Z))
    X = X[:]
    Y = Y[:]
    Z = Z[:]

    X.pop(minId)
    Y.pop(minId)
    Z.pop(minId)

    return X, Y, Z
```

Произведем эволюционные изменения, в соответствии с задачей - 4 шага:

```python
# шаги эволюции (конечная функция), по умолчанию 4 шага
def evoSteps(X, Y, stepsNum = 4):
    results = []

    for i in range(4):
        arrZ = [qZ(x, Y[i]) for i, x in enumerate(X)]

        X, Y, Z = evoStep(X, Y, arrZ)

        X, Y = exchangeScheme(X, Y, sorting(Z))

        results.append([X, Y, qSumZ(arrZ), arrZ])

    return X, Y, results
```

Теперь, когда мы подготовились к решению задачи, написав все необходимые функции для реализации генетического алгоритма (оценки качества хромосом, сортировки потомков и эволюционных шагов), решим задачу в числах. Пусть даны следующие массивы хромосом X и Y:

x	-2	-1	0	1
y	-2	-1	0	1

Запишем их в требуемом виде и воспользуемся написанной функцией evoSteps.

```python
# объявление массивов хромосом
X = [-2, -1, 0, 1]
Y = [-2, -1, 0, 1]
```

```python
# Реализация алгоритма
results = evoSteps(X, Y)
```

Теперь, выведем полученные значения для показателя качества хромосомы в популяции и общее качество популяции после четырех этапов эволюции. Для этого, воспользуемся циклом по значениям переменной results.

```python
for i in range(len(results[2])):
    print(f'max_{i + 1}_step: {results[2][i][2]}')

qualityArrZ = []
for i in range(len(results[2])):
    qualityArrZ += results[2][i][3]

print(f'max Z:      {max(qualityArrZ)}')
```

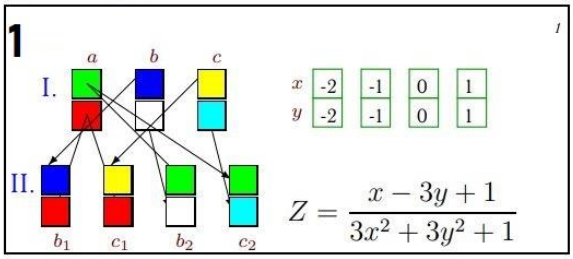
Ответ:

```
max_1_step: 1.4857142857142858
max_2_step: 1.4615384615384615
max_3_step: 2.967032967032967
max_4_step: 3.5384615384615383
max Z:      1.0
```

Задание:

Выполните по вариантам соответственно реализацию генетического алгоритма в соответствии с приложенными начальными данными.

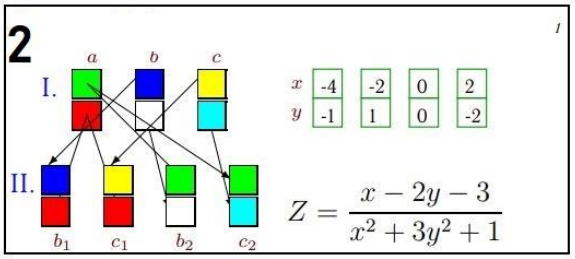
1



x	-2	-1	0	1
y	-2	-1	0	1

$$Z = \frac{x - 3y + 1}{3x^2 + 3y^2 + 1}$$

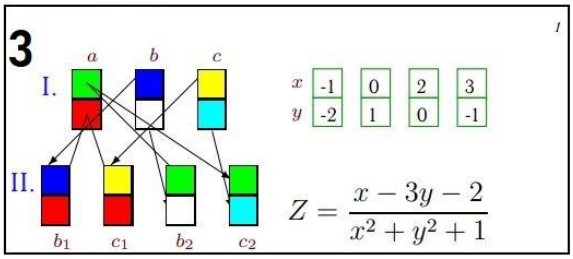
2



x	-4	-2	0	2
y	-1	1	0	-2

$$Z = \frac{x - 2y - 3}{x^2 + 3y^2 + 1}$$

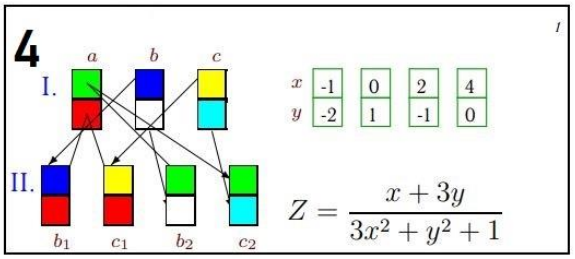
3



x	-1	0	2	3
y	-2	1	0	-1

$$Z = \frac{x - 3y - 2}{x^2 + y^2 + 1}$$

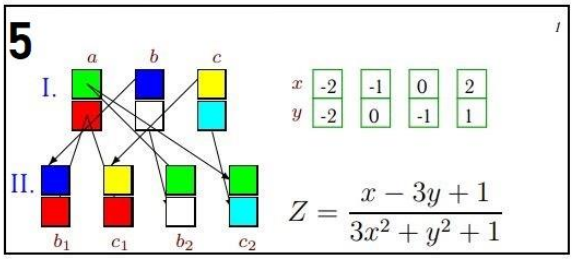
4



x	-1	0	2	4
y	-2	1	-1	0

$$Z = \frac{x + 3y}{3x^2 + y^2 + 1}$$

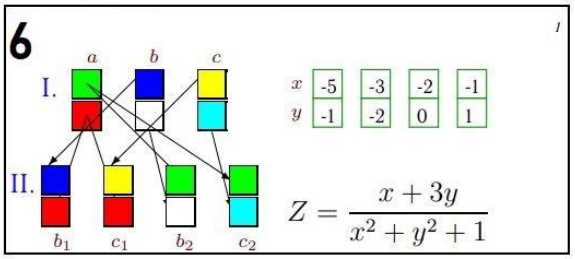
5



x	-2	-1	0	2
y	-2	0	-1	1

$$Z = \frac{x - 3y + 1}{3x^2 + y^2 + 1}$$

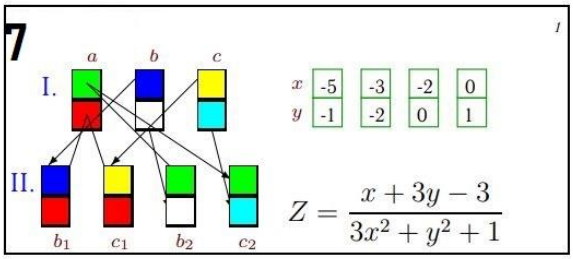
6



x	-5	-3	-2	-1
y	-1	-2	0	1

$$Z = \frac{x + 3y}{x^2 + y^2 + 1}$$

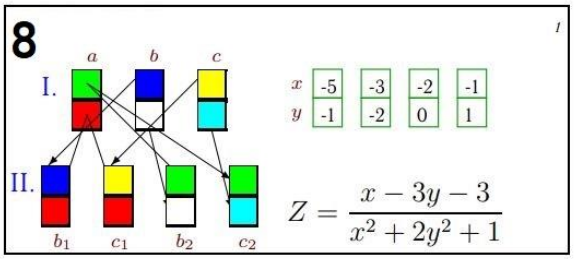
7



x	-5	-3	-2	0
y	-1	-2	0	1

$$Z = \frac{x + 3y - 3}{3x^2 + y^2 + 1}$$

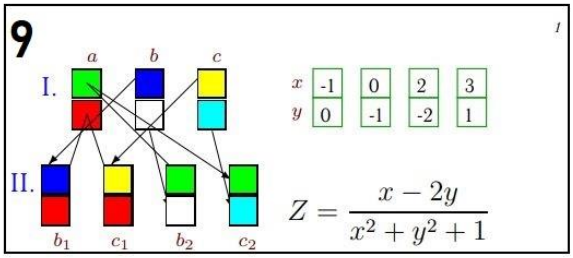
8



x	-5	-3	-2	-1
y	-1	-2	0	1

$$Z = \frac{x - 3y - 3}{x^2 + 2y^2 + 1}$$

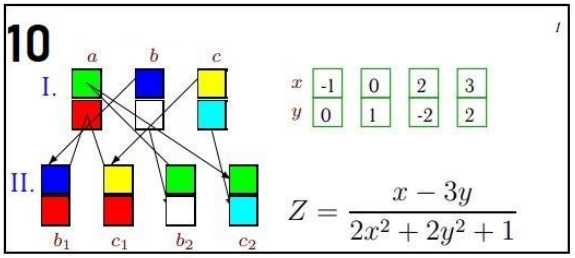
9



x	-1	0	2	3
y	0	-1	-2	1

$$Z = \frac{x - 2y}{x^2 + y^2 + 1}$$

10



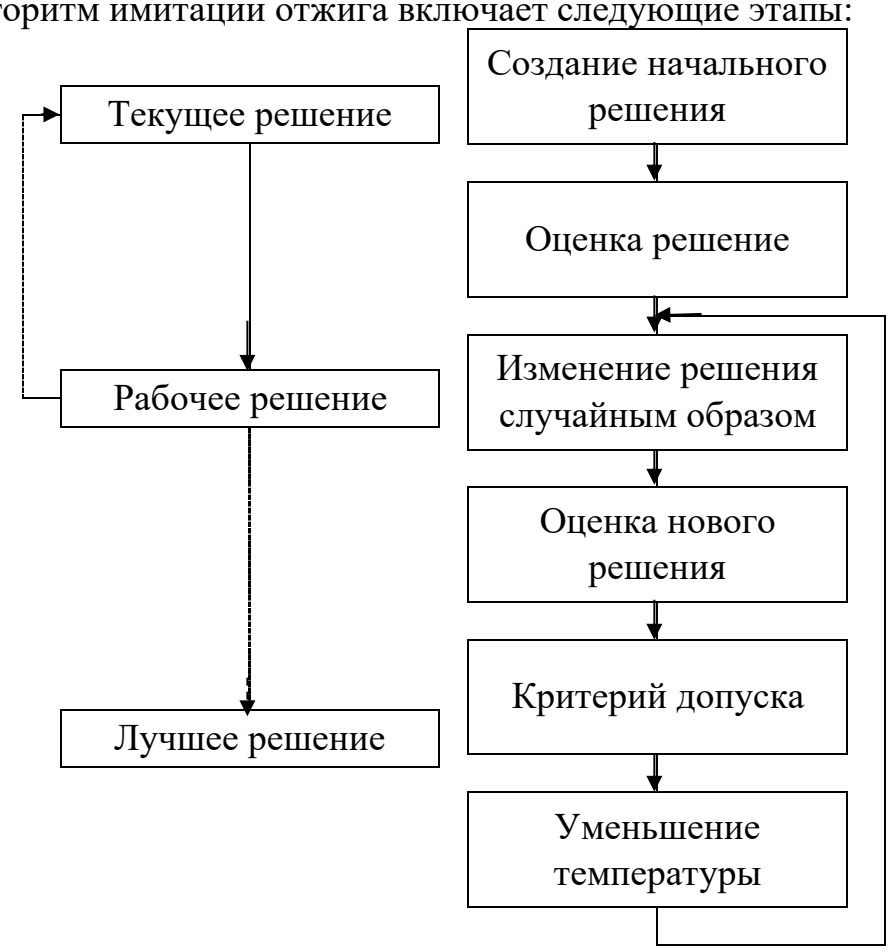
x	-1	0	2	3
y	0	1	-2	2

$$Z = \frac{x - 3y}{2x^2 + 2y^2 + 1}$$

1.2. Теоретический материал – Метод имитации отжига

Алгоритм отжига – это метод оптимизации, который называется отжигом, или симуляцией восстановления (Simulated annealing). Как ясно из названия, метод поиска моделирует процесс восстановления. Восстановление – это физический процесс, который заключается в нагреве и последующем контролируемом охлаждении субстанции. В результате получается прочная кристаллическая структура, которая отличается от структуры с дефектами, образующейся при быстром беспорядочном охлаждении. Структура здесь представляет собой кодированное решение, а температура используется для того, чтобы указать, как и когда будут приниматься новые решения.

Алгоритм имитации отжига включает следующие этапы:



Метод отжига может быть эффективным при решении задач различных классов, требующих оптимизации. Ниже приводится их краткий список:

1. создание пути;
2. реконструкция изображения;
3. назначение задач и планирование;
4. размещение сети;
5. глобальная маршрутизация;
6. обнаружение и распознавание визуальных объектов;
7. разработка специальных цифровых фильтров.

Поскольку метод отжига представляет собой процесс генерации случайных чисел, поиск решения с использованием данного алгоритма может занять значительное время. В некоторых случаях алгоритм вообще не находит решение или выбирает не самое оптимальное. Алгоритм отжига как способ выполнения процедур поиска и оптимизации. Данный метод является аналогом процесса нагревания тела до состояния плавления с последующим постепенным охлаждением. При высоких температурах поиск ведется по всему диапазону. При снижении температуры диапазон поиска уменьшается до небольшой области вокруг текущего решения.

Рассмотрим решение задачи поиска оптимального маршрута на графе методом имитации отжига Для этого, представим формальную постановку задачи и рассмотрим пример, который иллюстрирует алгоритм решения.

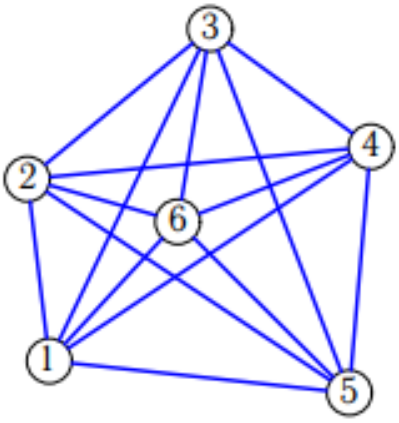
Итак, необходимо Найти длину гамильтонова цикла S_4 в полном графе K_6 после четырех циклов решения задачи методом отжига. Даны расстояния $L_{i,j}$ между вершинами. Даны также: начальная последовательность вершин L_0, последовательность замен вершин Z и выпавшие при этом вероятности перехода $P_k, k = 1, \ldots, 4$.

Переход на худшее ($\Delta S_k = S_k - S_{k-1} > 0$) решение допустим, если $P_* = 100$где снижение температуры происходит по закону $T_{k+1} = 0.5T_k$ от $T_1 = 100$.

1.2.1 Пример

Задача:

Итак, начальные условия задачи представляют собой следующий граф с расстояниями между ребрами:



Ребро	$L_{i,j}$
$1-2$	20
$1-3$	40
$1-4$	42
$1-5$	33
$1-6$	21
$2-3$	26
$2-4$	38
$2-5$	42
$2-6$	17
$3-4$	22
$3-5$	43
$3-6$	21
$4-5$	27
$4-6$	22
$5-6$	26

$\mathbf{V} = [1, 4, 5, 2, 6, 3, 1].$

$\mathbf{Z} = [V_3 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_6],$
$[V_5 \rightleftarrows V_2],\ [V_6 \rightleftarrows V_2].$

$\mathbf{P} = 49, 54, 43, 54.$

Решение:

Рассмотрим решение с применением Python.

Импортируем библиотеки:

```
import networkx as nx
from math import e
```

Далее, опишем массив длин ребер, последовательности прохождение вершин на маршруте и их замены, значения P, а также начальную температуру:

```
distances = [(1, 2, 20),
             (1, 3, 40),
             (1, 4, 42),
             (1, 5, 33),
             (1, 6, 21),
             (2, 3, 26),
             (2, 4, 38),
             (2, 5, 42),
             (2, 6, 17),
             (3, 4, 22),
             (3, 5, 43),
             (3, 6, 21),
             (4, 5, 27),
             (4, 6, 22),
             (5, 6, 26)] # длины рёбер

V = [1, 4, 5, 2, 6, 3, 1] # последовательность прохождения маршрута
Z = [(3, 4),
     (4, 6),
     (5, 2),
     (6, 2)] # последовательность замен вершин
P = [49, 54, 43, 54] # случайные числа, выпавшие в процессе счёта

T = 100 # начальная температура
```

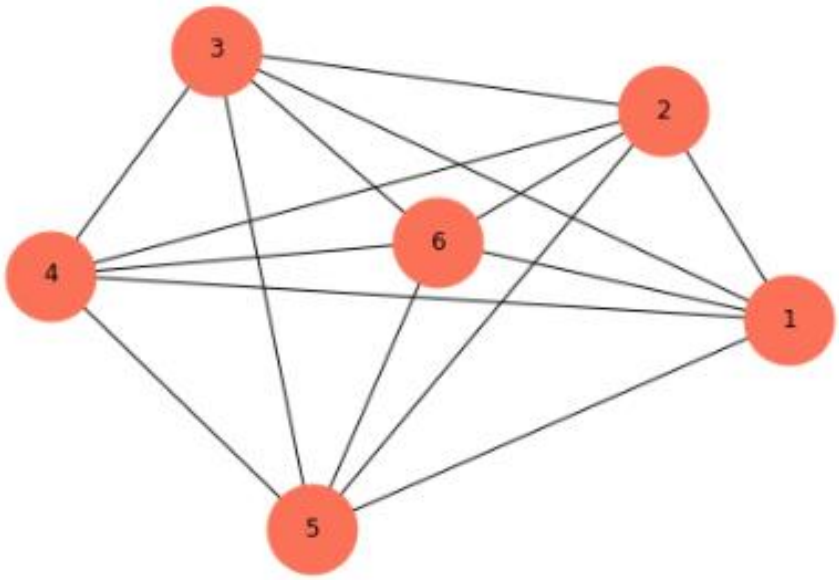
Запишем функции вероятности и изменения температуры:

```
# функция вероятности
def probability(delta, T):
    return 100 * e ** (-delta / T)

# функция изменения температуры
def reductTemp(prevT):
    nextT = 0.5 * prevT

    return nextT
```

Построим граф по заданным вершинам, как в начальном условии:

```
graph = nx.Graph() # создание пустого графа
graph.add_weighted_edges_from(distances) # добавление весов рёбер

# отрисовка графа с заданными вершинами
nx.draw_kamada_kawai(graph, node_color = '#fb7258', node_size = 2000, with_labels = True)
```

Ответ:



Теперь, напишем необходимые, согласно алгоритму выше, функции для работы метода имитации отжига. Вычислим длину ребра:

```
# вычисление длины ребра
def edgeLength(i, j, distances, roundTrip = True):
    if roundTrip:
        return max([(item[2] if (item[0] == i and item[1] == j) or (item[1] == i and item[0] == j) else -1)
                    for item in distances])
    else:
        return max([(item[2] if (item[0] == i and item[1] == j) else -1) for item in distances])
```

Вычислим длину маршрута:

```
# вычисление длины маршрута
def routeLength(V, distances):
    edges = []

    for i in range(len(V) - 1):
        edges.append(edgeLength(V[i], V[i + 1], distances))

    return sum(edges)
```

Запишем функцию для однократной перестановки в пути:

```
# одна перестановка в пути
def routeOneReplacement(arrV, Z, replacementByName = True):
    decrement = 1 if replacementByName else 0

    arrV[Z[0] - decrement], arrV[Z[1] - decrement] = arrV[Z[1] - decrement], arrV[Z[0] - decrement]

    return arrV
```

А теперь функцию, для реализации непосредственно самой перестановки:

```python
# перестановки в пути
def routeReplacement(V, Z):
    for z in Z:
        V = routeOneReplacement(V, z)
    return V
```

Теперь, опишем алгоритм выбора подходящего пути методом отжига:

```python
# выбор нужного пути методом отжига
def chooseRoute(distances, V, Z, T, P):
    sumLength = routeLength(V, distances) # нахождение длины пути
    arrSum = [sumLength] # массив сумм длин

    # циклы методом отжига
    for i in range(len(Z)):
        newV = routeOneReplacement(V[:], Z[i]) # новый маршрут после перестановки
        newS = routeLength(newV, distances) # длина нового маршрута
        arrSum.append(newS)
        deltaS = newS - sumLength # разница между длиной нового и старого маршрутов

        # в случае, если разница между длинами больше 0, то вычисляется вероятность
        if deltaS > 0:
            p = probability(deltaS, T) # подсчёт вероятности

            # если заданная вероятность попадает в интервал от 0 до p, то новый маршрут выбирается
            if p > P[i]:
                V = newV
                sumLength = newS
        else:
            V = newV
            sumLength = newS

        T = reductTemp(T) # вычисление температуры

    return V, arrSum
```

И нарисуем наш граф, отвечающий заданному маршруту:

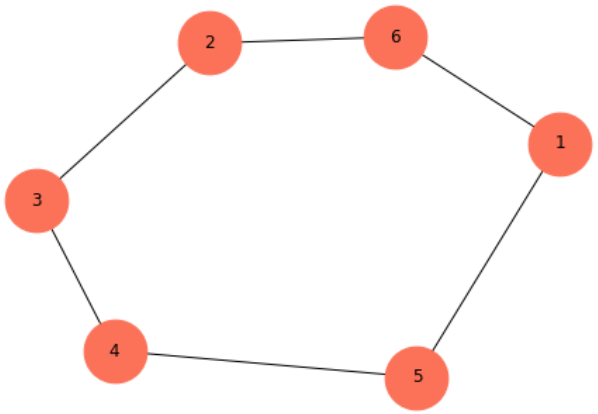
```python
# отрисовка графа по заданному маршруту
def drawRouteGraph(distances, bestRoute):
    newDistances = []
    # прохождение по вектору
    for i in range(len(bestRoute) - 1):
        for distance in distances:
            if distance[0] == bestRoute[i] and distance[1] == bestRoute[i + 1] #перенос, писать в строчку
            or distance[1] == bestRoute[i] and distance[0] == bestRoute[i + 1]:
                newDistances.append(distance)

    graph = nx.Graph() # создание пустого графа

    graph.add_weighted_edges_from(newDistances) # добавление весов рёбер
# отрисовка графа с заданными вершинами
    nx.draw_kamada_kawai(graph, node_color = '#fb7258', node_size = 2000, with_labels = True)
```

И, наконец, рассчитаем наилучший маршрут и его длину:

```python
bestRoute, arrLength = chooseRoute(distances, V, Z, T, P)

print(f'Лучший выбранный маршрут: {bestRoute}')
print(f'Длина лучшего выбранного маршрута: {routeLength(bestRoute, distances)}')
print(f'Длины всех рассмотренных маршрутов: {arrLength}')

drawRouteGraph(distances, bestRoute) # отрисовка лучшего маршрута
```

Ответ:

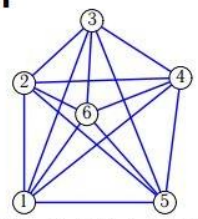
```
Лучший выбранный маршрут: [1, 6, 2, 3, 4, 5, 1]
Длина лучшего выбранного маршрута: 146
Длины всех рассмотренных маршрутов: [189, 209, 186, 146, 166]
```



Задание

Найти длину гамильтонова цикла S4 в полном графе K6 после четырех циклов решения задачи методом отжига по вариантам ниже.

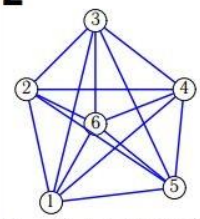
1



$\mathbf{V} = [1, 2, 3, 4, 5, 6, 1].$
$\mathbf{Z} = [V_3 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_6],$
$[V_5 \rightleftarrows V_6],\ [V_6 \rightleftarrows V_2].$
$\mathbf{P} = 90, 45, 43, 31.$

Ребро	$L_{i,j}$
$1-2$	26
$1-3$	42
$1-4$	44
$1-5$	31
$1-6$	24
$2-3$	20
$2-4$	34
$2-5$	40
$2-6$	15
$3-4$	23
$3-5$	43
$3-6$	20
$4-5$	27
$4-6$	22
$5-6$	26

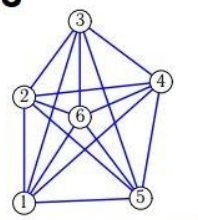
2



$\mathbf{V} = [1, 3, 5, 4, 6, 2, 1].$
$\mathbf{Z} = [V_3 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_6],$
$[V_5 \rightleftarrows V_6],\ [V_2 \rightleftarrows V_4].$
$\mathbf{P} = 41, 60, 85, 60.$

Ребро	$L_{i,j}$
$1-2$	25
$1-3$	41
$1-4$	38
$1-5$	27
$1-6$	20
$2-3$	21
$2-4$	34
$2-5$	39
$2-6$	17
$3-4$	24
$3-5$	40
$3-6$	22
$4-5$	21
$4-6$	21
$5-6$	22

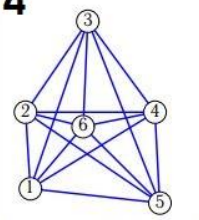
3



$\mathbf{V} = [1, 3, 4, 5, 6, 2, 1].$
$\mathbf{Z} = [V_4 \rightleftarrows V_5],\ [V_5 \rightleftarrows V_6],$
$[V_2 \rightleftarrows V_4],\ [V_6 \rightleftarrows V_2].$
$\mathbf{P} = 78, 24, 63, 17.$

Ребро	$L_{i,j}$
$1-2$	23
$1-3$	42
$1-4$	40
$1-5$	25
$1-6$	22
$2-3$	20
$2-4$	30
$2-5$	34
$2-6$	13
$3-4$	22
$3-5$	41
$3-6$	21
$4-5$	26
$4-6$	19
$5-6$	22

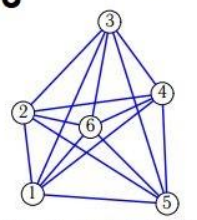
4



$\mathbf{V} = [1, 5, 2, 6, 3, 4, 1].$
$\mathbf{Z} = [V_3 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_5],$
$[V_5 \rightleftarrows V_2],\ [V_6 \rightleftarrows V_2].$
$\mathbf{P} = 78, 79, 25, 82.$

Ребро	$L_{i,j}$
$1-2$	17
$1-3$	39
$1-4$	32
$1-5$	28
$1-6$	18
$2-3$	24
$2-4$	28
$2-5$	35
$2-6$	13
$3-4$	25
$3-5$	43
$3-6$	23
$4-5$	20
$4-6$	16
$5-6$	24

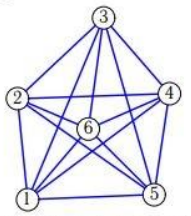
5



$\mathbf{V} = [1, 3, 4, 5, 6, 2, 1].$
$\mathbf{Z} = [V_2 \rightleftarrows V_4],\ [V_3 \rightleftarrows V_4],$
$[V_4 \rightleftarrows V_6],\ [V_5 \rightleftarrows V_6].$
$\mathbf{P} = 63, 49, 45, 53.$

Ребро	$L_{i,j}$
$1-2$	18
$1-3$	41
$1-4$	36
$1-5$	29
$1-6$	19
$2-3$	27
$2-4$	31
$2-5$	37
$2-6$	15
$3-4$	19
$3-5$	42
$3-6$	23
$4-5$	24
$4-6$	17
$5-6$	24

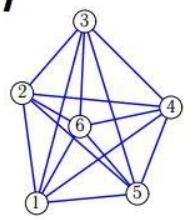
6



$\mathbf{V} = [1, 3, 4, 5, 6, 2, 1].$
$\mathbf{Z} = [V_2 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_6],$
$[V_3 \rightleftarrows V_5],\ [V_5 \rightleftarrows V_2].$
$\mathbf{P} = 51, 23, 29, 31.$

Ребро	$L_{i,j}$
$1-2$	22
$1-3$	43
$1-4$	39
$1-5$	28
$1-6$	20
$2-3$	26
$2-4$	33
$2-5$	36
$2-6$	17
$3-4$	22
$3-5$	40
$3-6$	24
$4-5$	22
$4-6$	19
$5-6$	20

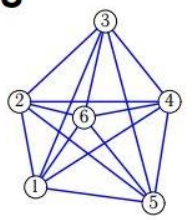
7



$\mathbf{V} = [1, 3, 4, 5, 6, 2, 1].$
$\mathbf{Z} = [V_3 \rightleftarrows V_4],\ [V_4 \rightleftarrows V_6],$
$[V_5 \rightleftarrows V_2],\ [V_6 \rightleftarrows V_2].$
$\mathbf{P} = 33, 82, 51, 76.$

Ребро	$L_{i,j}$
$1-2$	24
$1-3$	41
$1-4$	36
$1-5$	22
$1-6$	19
$2-3$	21
$2-4$	33
$2-5$	33
$2-6$	14
$3-4$	27
$3-5$	39
$3-6$	23
$4-5$	20
$4-6$	20
$5-6$	19

8



$\mathbf{V} = [1, 4, 2, 3, 5, 6, 1].$
$\mathbf{Z} = [V_5 \rightleftarrows V_2],\ [V_4 \rightleftarrows V_5],$
$[V_2 \rightleftarrows V_3],\ [V_3 \rightleftarrows V_4].$
$\mathbf{P} = 88, 54, 24, 64.$

Ребро	$L_{i,j}$
$1-2$	19
$1-3$	39
$1-4$	35
$1-5$	26
$1-6$	18
$2-3$	26
$2-4$	33
$2-5$	37
$2-6$	14
$3-4$	22
$3-5$	41
$3-6$	21
$4-5$	22
$4-6$	19
$5-6$	24

Задание*:

На решенном уже примере поэкспериментируйте с показателем S, проанализируйте результаты.