**Разбор задания**

Найти наиболее выгодную последовательность обменов для конвертации заданной валюты s в заданную валюту t.

\*Получается, что нужно проводить попытки конвертации до тех пор, пока конвертация не начнёт уменьшать сумму денег.

Наиболее выгодный обмен = где купить удаётся больше.

Нужно найти кратчайший путь.

и сказать, по какому пути он был найден. (используй метод bfs\_parent).

-- Для этого я буду использовать:

При нахождении кратчайшего пути нужно перемножать, а не складывать значения.

# Граф ориентированный, т.к. обмены происходят в одну сторону

#

# вершины - валюты

# ребра - курсы обмена

#

'''

Мой план по решению этой КТ:

1) Разобраться с тем, как реализован Алгоритм дейкстры кодом от selfedu

2) Переписать граф моей задачи на формат, пригодный для этого кода

Результат: Я получу строку таблицы, которая будет соответствовать результату самой дешевой конвертации заданной валюты во все остальные валюты.

-- Далее полученные данные нужно снова проверить, ведь может оказаться так, что последующая конвертация будет выгодней.

--

'''

from collections import deque

# С данным графом обмен туда обратно не даст больше денег.

currencies\_exchange = {

    'Rub': {'Dollar':0.0096, 'Euro':0.0087, 'GPB':0.0072},

    'Dollar':{'Rub':100.9, 'Euro':0.9, 'GPB':0.75},

    'Euro':{'Rub':110.5, 'Dollar':1.1, 'GPB':0.83},

    'GPB':{'Rub':135, 'Dollar':0.7, 'Euro':0.80}

}

# В основе алгоритма - алгоритм Дейкстры

def most\_beneficial\_deal (currency, G):

    exhange\_costs = {} # Стоимости конвертации

    exhange\_costs[currency] = 1 # Конвертация валюты саму в себя без потерь

    queue = deque()

    queue.append(currency)

    while queue:

        cur\_currency = queue.popleft()

        for neighbor in G[cur\_currency]:

            if (neighbor not in exhange\_costs

                or exhange\_costs[cur\_currency] \* G[cur\_currency][neighbor] < exhange\_costs[neighbor]):

                exhange\_costs[neighbor] = exhange\_costs[cur\_currency] \* G[cur\_currency][neighbor]

                queue.append(neighbor)

    return exhange\_costs

print(most\_beneficial\_deal('Rub', currencies\_exchange))