**Алгоритм для «Самый короткий путь»**

Условие:

Имеется граф, который хранит вершины. Есть вершина, в которой хранится имя магазина, список названий товаров, а также цена отдельного товара и его количество на складе.

Например:

class Vertex():  
 def \_\_init\_\_(self, name):  
 self.name = name  
 self.all\_products = {}  
 self.neighbors = {}

Имеется корзина со списком товаров и требуемым их количеством.

Например:

shop\_box = {'pizza' : 22, 'beer' : 2, 'beer2': 13}

Требуется найти самый короткий путь, такой что бы клиент купил все требуемые товары и прошёл минимальное расстояние.

Решение:

1. Создать переменные для хранение минимального пути (массив с названием вершин), и значением этого пути (число). Принять за значение минимального пути большое число, например 1000000.
2. Далее для каждого шага (step) от 1 до N-1, (где N это количество всех вершин) требуется обойти граф с помощью такого алгоритма:
3. Если переменная стоп равна 1 завершить алгоритм.
4. Взять в качестве начальной вершины, вершину Клиент. Поместить эту вершину в стек.
5. Пока стек не пустой:
   1. Взять верхнею вершину из стека.
   2. Ко временной переменной текущего расстояние прибавить расстояние из текущей вершины, до последней во временном списке пути.
   3. «Купить» в данной вершине максимальное количество товара, не превышающее максимальное количество товара этой вершины и не превышающее значение в переменной shop\_box.
   4. Внести данную вершину во временный список пути.
   5. Если куплены все товары, и расстояние меньше, чем минимальное, записываем данный вариант в результирующий и делаем текущее значение расстояние минимальным. Завершаем итерацию
   6. Если расстояние получилось больше, чем минимальное, завершаем итерацию.
   7. Если значение шага не превышает глубины поиска, то поместить всех соседей данной вершины, за исключением тех, что находятся во временном списке (или же кроме тех которые мы брали из стека)
6. Если минимальное расстояние меньше стартового (например, 1000000) и если при выполнении шага путь всех комбинаций оказался больше минимального, то переменную стоп сделать равной 1.
7. Повторить следующий шаг итерации.
8. По результату работы алгоритма получаем минимальный путь.

То что жёлтым спорно… «а» и «г» срезают много итераций.

**Алгоритм для «Самый выгодный по цене»**

Условие:

Имеется граф, который хранит вершины. Есть вершина, в которой хранится имя магазина, список названий товаров, а также цена отдельного товара и его количество на складе.

Например:

class Vertex():  
 def \_\_init\_\_(self, name):  
 self.name = name  
 self.all\_products = {}  
 self.neighbors = {}

Имеется корзина со списком товаров и требуемым их количеством.

Например:

shop\_box = {'pizza' : 22, 'beer' : 2, 'beer2': 13}

Требуется найти самый дешёвый способ купить продукты, такой что бы клиент купил все требуемые товары и заплатил минимальную сумму.

Решение:

1. Создать переменные для хранение минимального пути (массив с названием вершин), и значением этого пути – стоимостью покупки (число). Принять за значение минимального пути большое число, например 1000000.
2. Создаём N количество (равное количеству позиций, которые купил клиент) структур, в которые помещаем магазины с ценой на этот товар, и количество этих товаров.
3. Отсортируем магазины по цене товара.
4. Отсортируем магазины, у которых одна и та же цена на товар, по наличию товара.
5. Для каждого товара в переменной shop\_box:
   1. Пока текущее\_количество не равно нужному \_количеству:
      1. Вынимаем из структуры верхнее значение.
      2. Записываем в результат.
      3. Добавляем количество к текущему, не превышающее максимальное количество товара этого магазина и не превышающее значение в переменной нужное количество.

**Алгоритм для «Самый оптимальный»**

Условие:

Имеется граф, который хранит вершины. Есть вершина, в которой хранится имя магазина, список названий товаров, а также цена отдельного товара и его количество на складе.

Например:

class Vertex():  
 def \_\_init\_\_(self, name):  
 self.name = name  
 self.all\_products = {}  
 self.neighbors = {}

Имеется корзина со списком товаров и требуемым их количеством.

Например:

shop\_box = {'pizza' : 22, 'beer' : 2, 'beer2': 13}

Требуется найти самый оптимальный способ купить продукты, такой что бы клиент купил все требуемые товары, и комплексная оценка была наименьшей. Под комплексной оценкой понимаемся оценка, которая зависит от цены на товар и на расстояние до магазина, где это товар можно купить.

Решение:

1. Создать переменные для хранение минимального пути (массив с названием вершин), и значением этого пути (число, минимальная комплексная оценка). Принять за значение минимального пути большое число, например 1000000.
2. Далее для каждого шага (step) от 1 до N-1, (где N это количество всех вершин) требуется обойти граф с помощью такого алгоритма:
3. Если переменная стоп равна 1 завершить алгоритм.
4. Взять в качестве начальной вершины, вершину Клиент. Поместить эту вершину в стек.
5. Пока стек не пустой:
   1. Взять верхнею вершину из стека.
   2. Ко временной переменной текущей комплексной оценки прибавить комплексной оценку из текущей вершины, до последней во временном списке пути.
   3. «Купить» в данной вершине максимальное количество товара, не превышающее максимальное количество товара этой вершины и не превышающее значение в переменной shop\_box.
   4. Внести данную вершину во временный список пути.
   5. Если куплены все товары, и комплексная оценка меньше, чем минимальная, записываем данный вариант в результирующий и делаем текущее значение пути минимальным. Завершаем итерацию
   6. Если комплексная оценка получилось больше, чем минимальная, завершаем итерацию.
   7. Если значение шага не превышает глубины поиска, то поместить всех соседей данной вершины, за исключением тех, что находятся во временном списке (или же кроме тех которые мы брали из стека)
6. Если минимальный путь меньше стартового (например, 1000000) и если при выполнении шага путь всех комбинаций оказался больше минимального, то переменную стоп сделать равной 1.
7. Повторить следующий шаг итерации.
8. По результату работы алгоритма получаем минимальный путь.

То что жёлтым спорно… «а» и «г» срезают много итераций.