Структура графа:

self.fictive\_start\_top = -100 # шифр фиктивной стартовой вершины  
self.fictive\_end\_top = -101 # шифр фиктивной конечной вершины  
self.graph\_table = {'arc\_start': [],  
 'arc\_end': [],  
 'weight': [],  
 'is\_visited': [], } # таблица, содержащая дуги и веса графа  
self.struct\_graph\_table = {'arc\_start': [],  
 'arc\_end': [],  
 'weight': [],  
 'is\_visited': [], } # упоряденная таблица, содержащая дуги и веса графа  
self.first\_top = None # первая вершина графа  
self.last\_top = None # последняя вершина графа  
self.ways\_num = 0 # количество путей (работ) графа  
self.struct\_graph\_ways\_num = 0 # количество путей (работ) упорядоченного графа  
self.current\_way = [] # текущий путь для вывода всех полных путей

0) инициализация и получение исходного графа из файла происходит в конструкторе \_\_init\_\_(self, file\_in)

1) метод print\_row(self, row\_index, sorted\_graph=False)печатет строку таблицу по индексу

2) метод print\_graph(self, sorted\_graph=False)печатает всю таблицу графа, используя функцию print\_row. Есть параметр функции, который указывает какой граф выводить (упорядоченный или нет).

3) метод add\_row\_in\_graph\_table(self, start, end, weight, is\_visited=False, adding\_is\_in\_sort=False)добавляет строку в таблицу графа. Есть параметр функции, который указывает в какой граф добавлять (упорядоченный или нет). При добавлении метод увеличивает счётчик количества строк соотвествующего графа на 1.

4) метод def copy\_row\_to\_struct\_graph\_table(self, index)копирует строку по индексу из неупорядоченного графа и добавляет её в конец упорядоченного. Метод вызывает метод add\_row\_in\_graph\_table.

5) метод delete\_row\_from\_graph\_table(self, index)удаляет строку из неупорядоченного графа. Уменьшает счётчик количества строк в неупорядоченном графе на 1.

6) метод search\_first\_top(self)ищет начальную вершину СГ.

Проходит во внешнем цикле по всем строкам таблицы путей графа  
Во внутреннем цикле проверяется, не является ли стартовой (в неё входит дуга, кроме петель) текущая вершина. Если найдена стартовая вершина (и не посещенная), проверяется, первая ли она такая. Если первая, то происходит запоминание её шифра и переход к следующей вершине. Если стартовая вершина уже существует, то происходит добавление фиктивной вершины (добавляется дополнительная запись в таблицу СГ.

7) метод delete\_top\_loops(self, index) проверяет строку на наличие петли и удаляет строку, если петля есть. Возвращает счётчик цикла: если было удаление, то возвращает 0, чтобы запустить цикл сначала, если не было удаления, то возвращает тот же индекс цикла и ничего не изменяется. (так как нет goto)

8) метод check\_duplication(self, i, j)проверяет наличие дублирования строки (с одинаковым и разным весом). При разных весах предлагает выбрать, какую работу удалить. При одинаковом удаляет автоматически вторую (разницы какую удалять нет). Метод аналогично возвращает счётчик цикла для запуска цикла сначала. (так как нет goto).

9) метод search\_last\_top(self)работает аналогичну методу с поиском первой вершины. В нём происходит поиск последней вершины. Если вершина не одна, создается фиктивная конечная вершина.

10) метод optimize\_graph(self) вызывает методы delete\_top\_loops, check\_duplication, search\_last\_top и управляет порядком их вызова. Когда нужн повторно запустить метод, когда нет он понимает по возвращаемым значениям из этих функций (так как нет goto).

11) метод struct\_graph(self)упорядочивает граф, используя очередь (проходит граф в ширину). Новый упорядоченный граф хранится другом объекте.

12) метод recursive\_search(self)рекурсивно проходит по графу.

Если последний элемент в текущем пути равен последнему элементу в графе, запускается цикл печати пути, удаляется последняя вершина из текущего пути и происходит выход из функции.

Иначе происходит вход в цикл( i от 0 до количества строк в таблице графа). Если стартовая вершина i-ой работы равна последней работе в текущем пути, то добавляем в текущий путь конечную вершину i-ой работы и рекурсивно запускаем recursive\_search.

После этого цикла удаляется последняя вершина из текущего пути.

13) метод search\_full\_ways(self)находит все полные пути графа, вызывая рекурсивный метод recursive\_search(self). Перед этим вызовом находится первая строка, в которой хранится первая вершина. Добавляет эту вершину в текущий путь.

14) код основой программы

if \_\_name\_\_ == "\_\_main\_\_":  
 in\_file = "input.csv"  
 graph = Graph(in\_file)  
 graph.print\_graph()  
 graph.search\_first\_top()  
 graph.optimize\_graph()  
 graph.print\_graph()  
 graph.struct\_graph()  
 graph.print\_graph(sorted\_graph=True)  
 graph.search\_full\_ways()