UCI University of California, Irvine

EECS 221 Adv App Algorithms

Final Report

Project Part 6

Student ID: 14563990

Student netID: yuanzhl4

Student Name: Yuanzhe Li

Table of Contents

1.Introduction 2

2.Algorithm 2

2.1 Nearest neighbor algorithm 2

2.2 Branch and bound algorithm 3

3.Language and Tools 4

3.1 Python language 4

3.2 Library 4

4.Batch processing 4

5.GUI 8

7. Appendix: 10

7.1 Machine Info 10

7.2 Code structure 10

UCI University of California, Irvine

1.Introduction

This report Is the final report of EECS 221 advanced application algorithm. It is a project course. Student make their own program based on an application scenario. In this case, we build a warehouse system to process orders. The problem is how to pick up items with order list. It is very interesting project since Amazon already use intelligent warehouse system to cut costs. It is very useful.

Under this warehouse scenario, two algorithms are used to solve orders problem, visiting each item once and pick them up from starting point to destination point, liking a TSP problem. One algorithm is nearest neighbor algorithm, another algorithm is branch and bound algorithm.

Program could process one order and output a path, or process batch of orders and store them in a file. Order could be re-visited from the processed output file.

CMD and GUI are provided.

A GUI is created to visualize optimal order path. In my project, I implemented GUI via Tkinter library. It could implement different function depending on which function users choose. To process with different algorithm, process order considering weight and process batch of orders.

In the project part 6, I finished preferred requirements. Build a GUI interface, show current path via typing new order and choosing algorithm and weight. And I implemented file batching process. It could process whole file with two algorithms and store cost and optimal orders results in result file. User could type in the number of order and check path visually in GUI. Effort will be calculated if weight is considered. If items weights are not considered, the effort will reduce to the length of path.

2.Algorithm

Two algorithms are used. One algorithm is nearest neighbor algorithm, another algorithm is branch and bound algorithm.

2.1 Nearest neighbor algorithm

Nearest neighbor algorithm aims to find shortest path from different items. First, a nearest matrix is built. It contains each distance between each item. Every item will be a start item once, and find its nearest neighbor, adding up path cost. Then, reduce matrix, and find nearest neighbor' nearest neighbor. Adding cost from start point to this start item and from end item to end point (destination). Finally, compare all shortest path cost depending on different start point, and pick the smallest one as the shortest path.

If taking weight into consideration, weight cost will be calculated in each sub-process when reducing distance matrix. Total weight in car will increase. Then, each shortest path cost will be re-calculated with weight. Compare and choose the smallest one.

2.2 Branch and bound algorithm

Branch and bound is a branch cut algorithm in my opinion. On the worst case, it is the brute force. Each leaf will be extended if the cost is lower than upper bound. Once we get a solution, many branches could be cut. If no leaf is extended, every node has the possibility to be extended, which is the reason why it cost so much time. But it is better than brute force and it could get optimal result. So, it is also a good algorithm.

In my program, I implemented this branch and bound algorithm by store each node in a heap. Heap will return a smallest cost leaf every time. Extended leaf will be discard. Heap only store leaf node at that time. So, it saves space. Each node contains information of cost, order, reduced distance matrix etc.

3.Language and Tools

3.1 Python language

I implemented my program with Python 2.7 language. So, if you want to run my program, you could use command: python warehouse.py

3.2 Library

import os

import csv

import re

import time

import copy

import heapq

import sys

import Tkinter

4.Batch processing

warehouse-orders-v02-tabbed.txt could be upload into program and output a file named warehouseOutput.txt, which contains all orders information: start and end point, original orders, new orders with two algorithms, cost with two algorithms. Order could be checked via this warehouseOutput.txt from GUI or CMD.

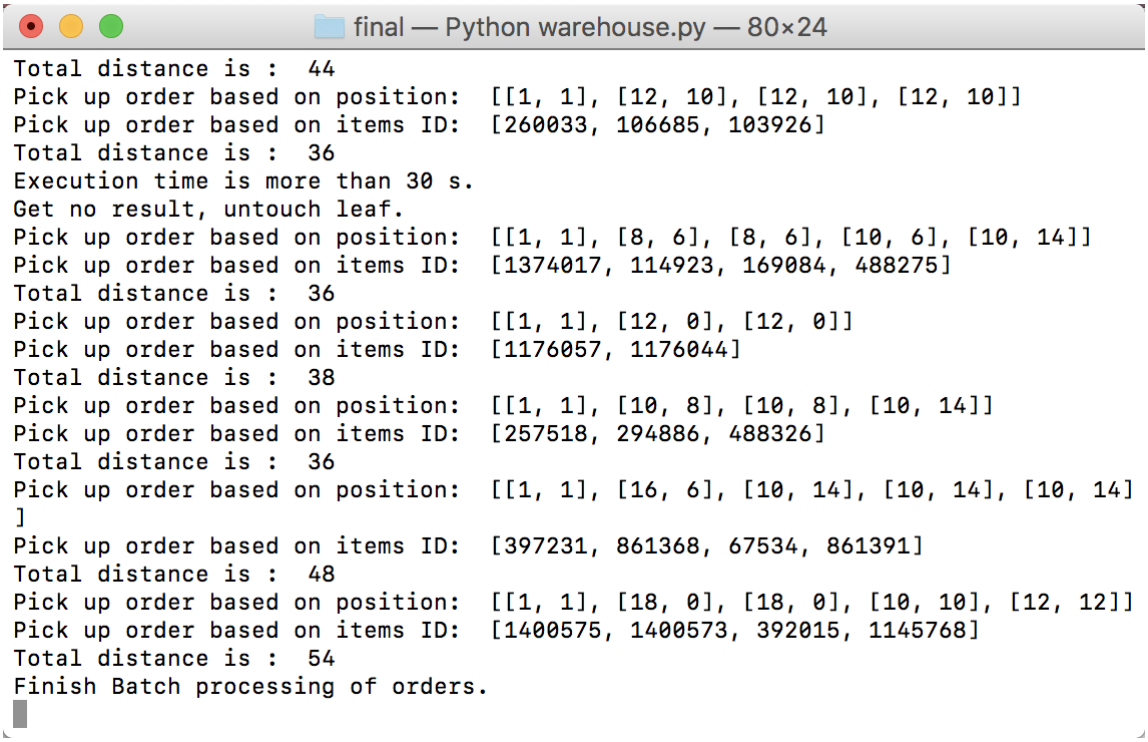

Figure 1, Processing result on CMD

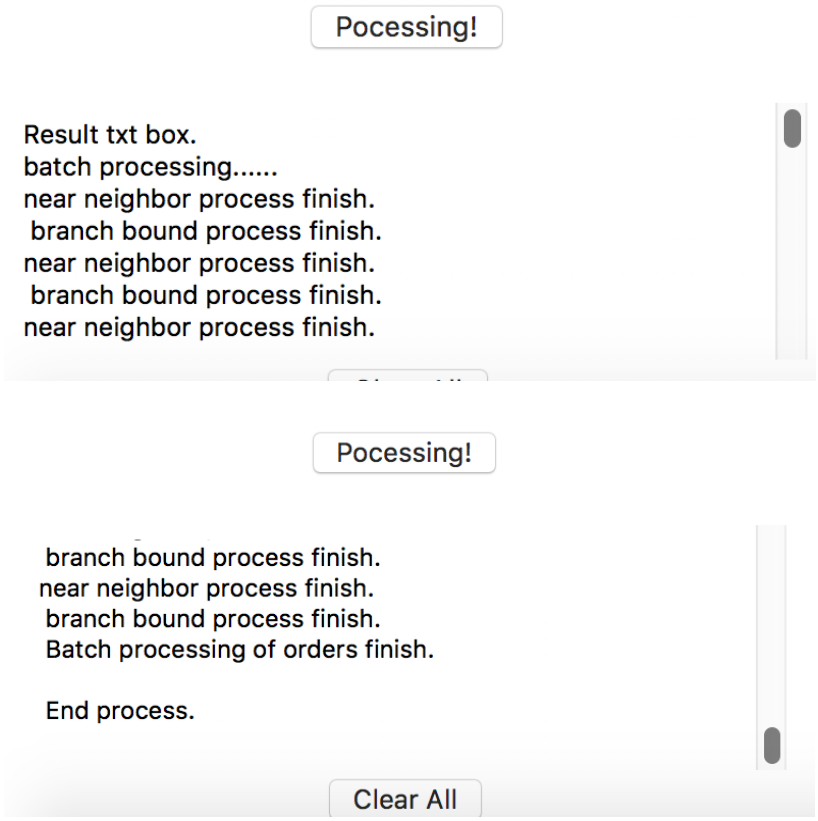

Figure 2, Processing result on GUI

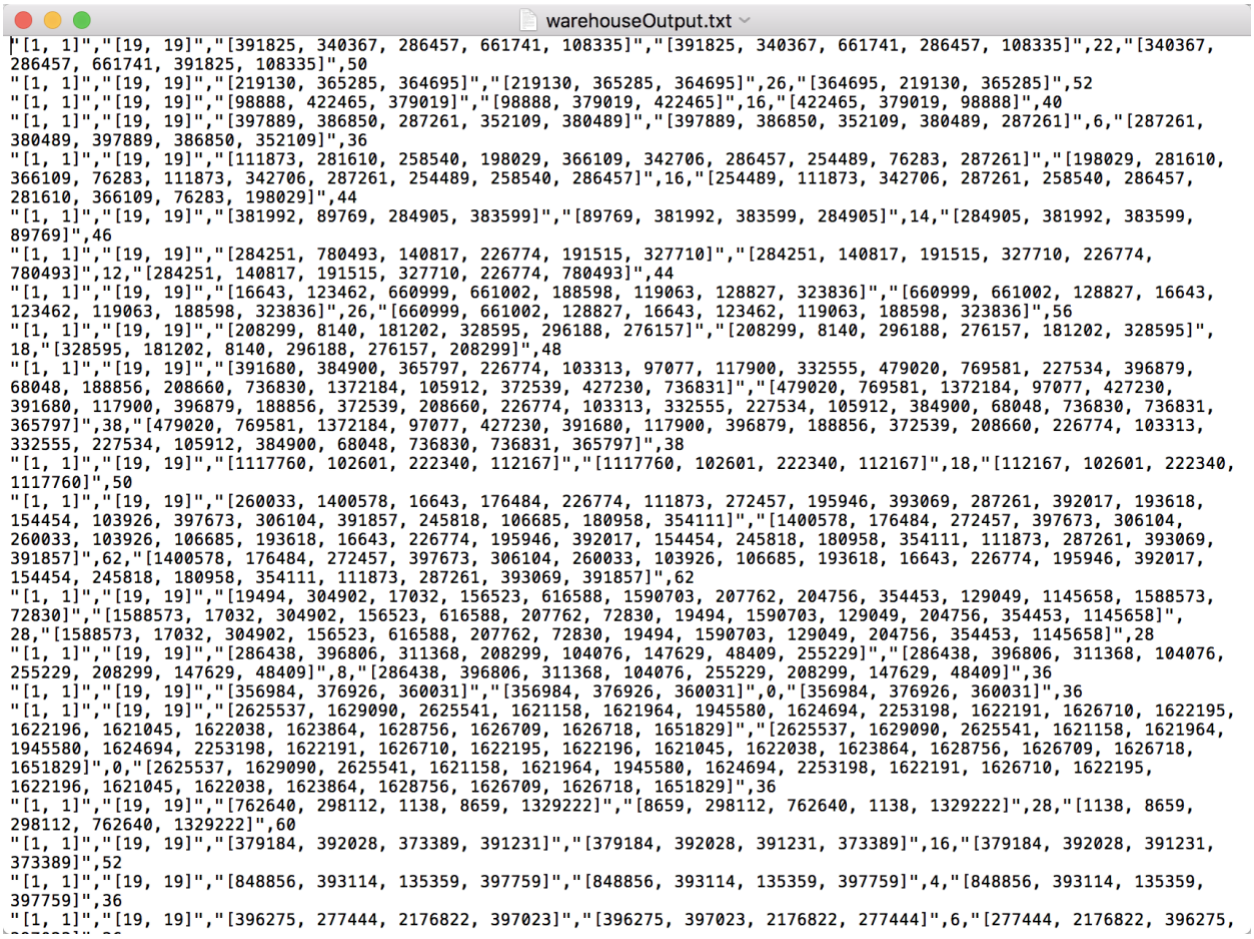

Figure 3, Result file warehouseOutput.txt

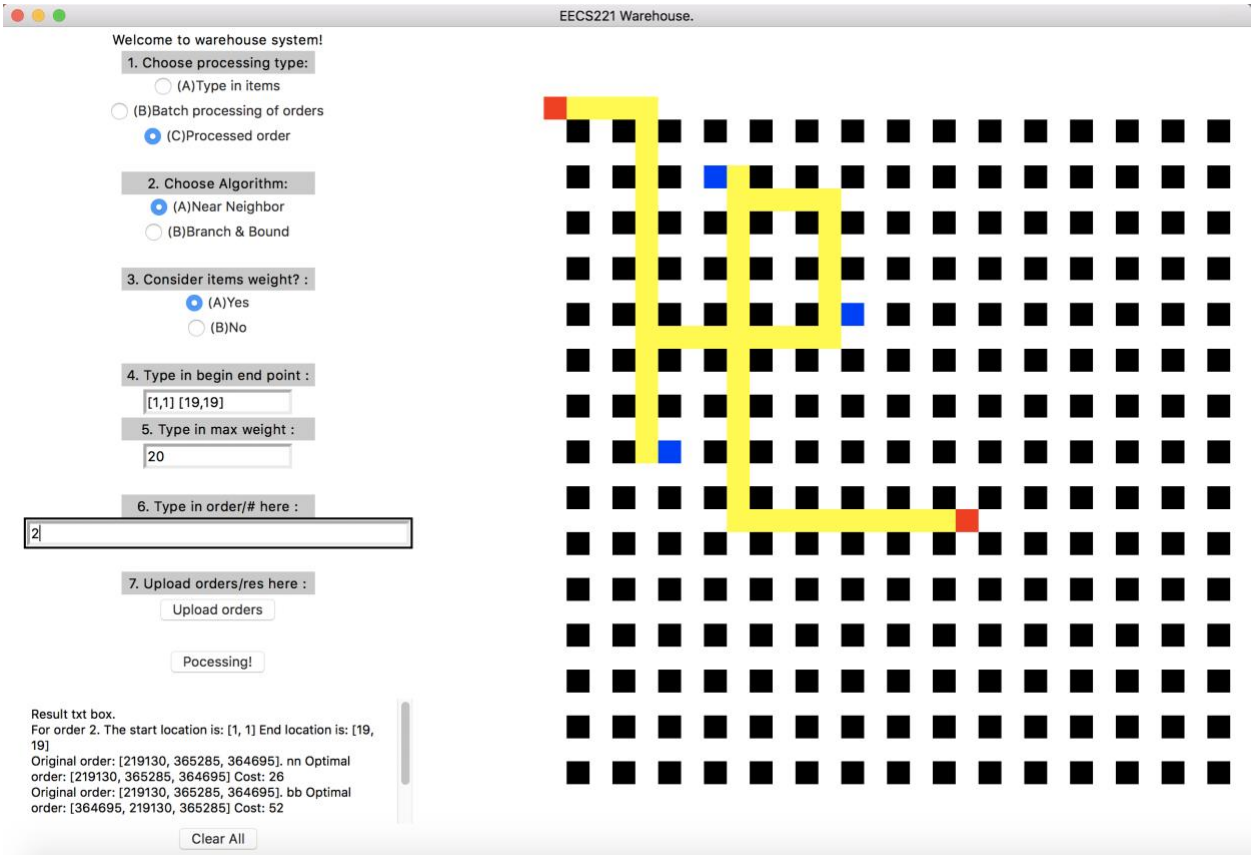

Figure 3, Read one result from processed warehouseOutput.txt

Because it considers weight of items, it visits the left below item first, rather than nearest one.

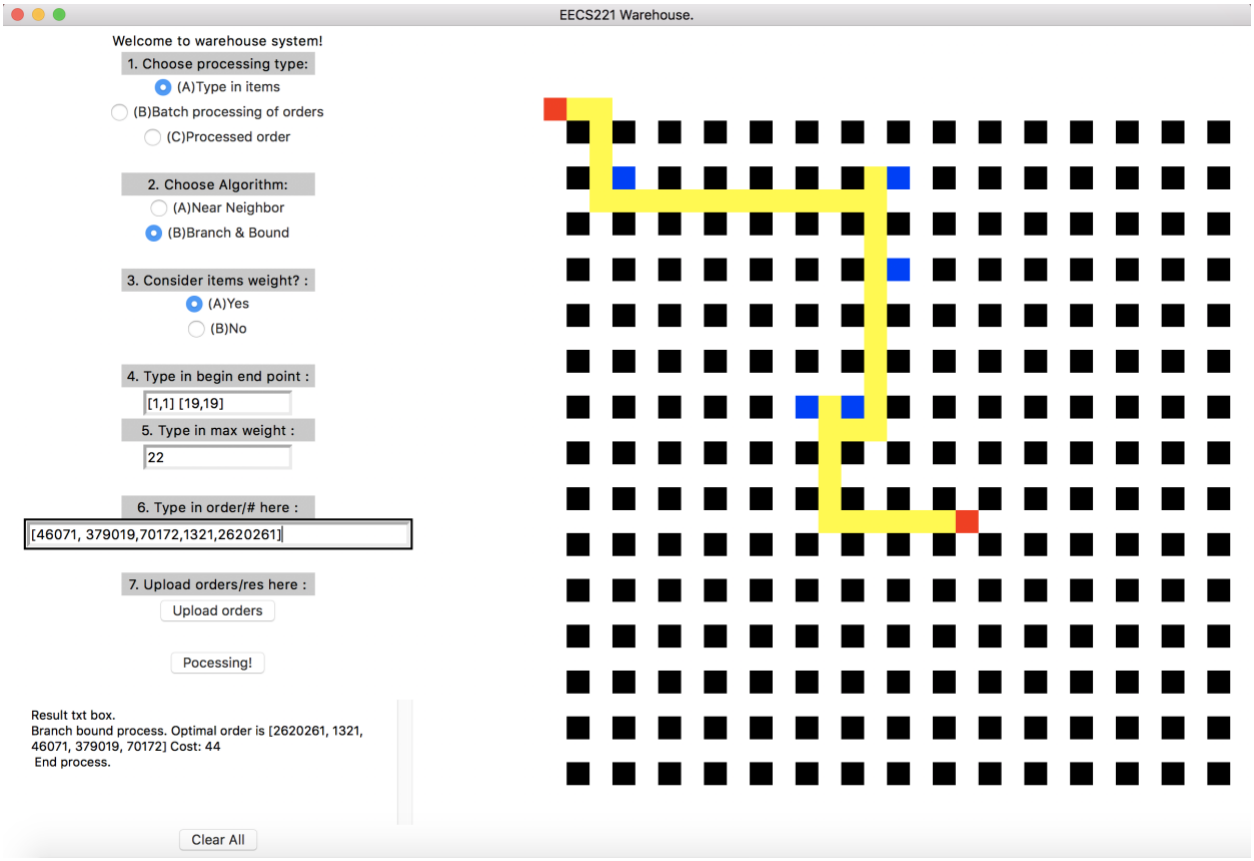

Figure 4, Processing one order by typing in

Choose processing type is to choose modes: "Type in items ", "Batch processing of orders" and "Processed order" three modes. Type in items is to typing in order. Batch processing of orders is batching processing order file, it takes long time. Once output file is created, processed order mode could be used to check result via order number.

Choose algorithm is to choose which algorithm is used in "type in items" mode and "Processed order" mode. "Batch processing of orders" will atomically process orders with two algorithms.

Consider items weight will choose weight is considered or not.

Type in begin and end point is to tell grogram where is the car's start and end point.

Type in max weight means the capacity of the car.

UCI University of California, Irvine

Type in order/# here is a place to type in order in mode in “type in items” and choose the order in output file with the number in result file.

Upload orders/res here is a place to upload file path. In order to set file path when processing batch orders or load warehosueOutput.txt

Processing! is a begin button.

There is a **text area** to present all order and result information.

Clear all button could clear text area.

On the right side is the **path map.** Red nodes are start and end point, blue nodes are items shelves. Yellow path is the optimal path.

7. Appendix:

7.1 Machine Info

OS: macOS Sierra

Processor: 2GHz Intel Core i5

Memory: 8GB

7.2 Code structure

warehouse.py is the main code. When executing warehouse.py, it will import and run branchBound.py, nearNeighbor.py and path.py program.

What's more, warehouse-grid.csv is used to locate each item.

item-dimensions-tabbed.txt is used to find items' weights.

warehouse-orders-v01.csv is a csv file containing orders, which program processes.

warehouse-orders-v02-tabbed.txt is big size orders.