Yuan Qin
yqin7@uci.edu
Apr.15th,2018

Project Part 2 Report

1. Progress Summary

What has been done:

In this second session of the project, the path of the picking up robot is mapped into a graph with each node corresponding to the robot trail location node. I redesigned the shortest distance algorithm in part one and used Dijkstra to find the shortest path between different location node, which I hope make constructing following functionalities to be easier. Then arbitrary starting and dropping off point, together with an order of items can be entered as input.

Then total distance in order of original without optimization is calculated and displayed as benchmark. Optimized new order for a shortest distance is calculated in brute force and then moving tracks of the robot are displayed. Afterwards, the relevant information is written into a csv output file.

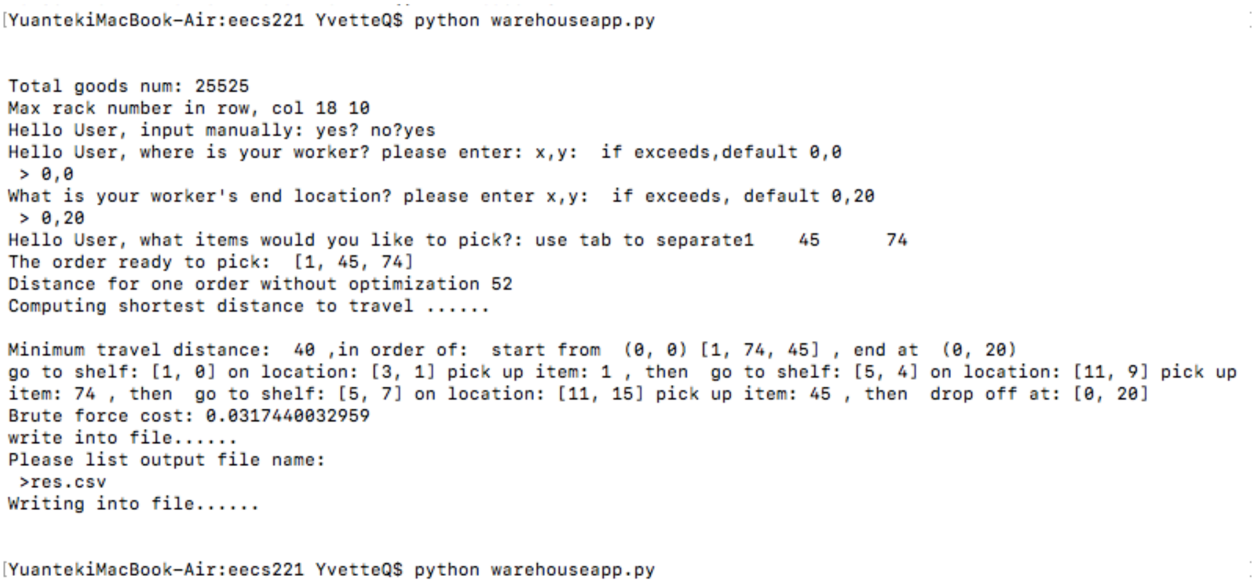
As for detailed moving trail, I made it in presentation of path node number and according coordinates, which can be printed out in the *graphtest.py* module, and it is stored in the *traversedpoint* array computed in the *pathgraph() f*unction. Due to its huge quantity, it is not displayed on screen except only for debugging.

To be finished**:**

1. Only enter one order manually for now. Reading orders from actual order file and computes it in a bunch at a time
2. Optimize algorithm to calculate the shortest tsp-like problem in shorter time instead of brute force.

2. Program running

A. Terminal output with 3 items to check correctness



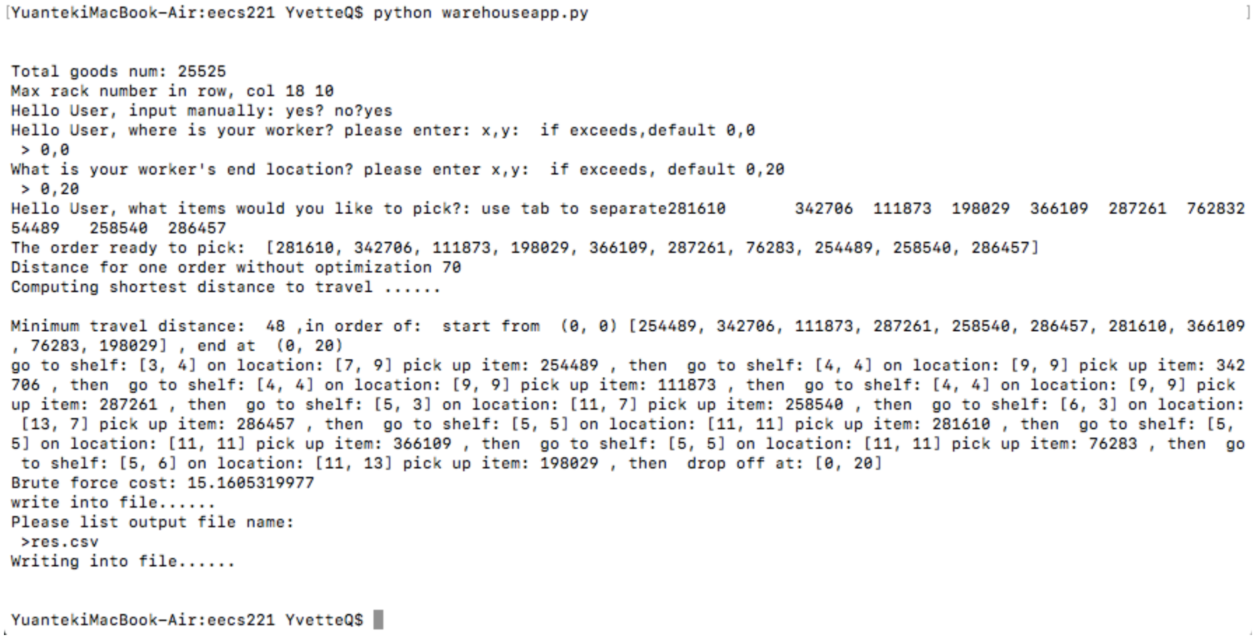
```
[YuantekiMacBook-Air:eecs221 YvetteQ$ python warehouseapp.py


Total goods num: 25525
Max rack number in row, col 18 10
Hello User, input manually: yes? no?yes
Hello User, where is your worker? please enter: x,y:  if exceeds,default 0,0
 > 0,0
What is your worker's end location? please enter x,y:  if exceeds, default 0,20
 > 0,20
Hello User, what items would you like to pick?: use tab to separate1    45      74
The order ready to pick:  [1, 45, 74]
Distance for one order without optimization 52
Computing shortest distance to travel ......

Minimum travel distance:  40 ,in order of:  start from  (0, 0) [1, 74, 45] , end at  (0, 20)
go to shelf: [1, 0] on location: [3, 1] pick up item: 1 , then  go to shelf: [5, 4] on location: [11, 9] pick up
item: 74 , then  go to shelf: [5, 7] on location: [11, 15] pick up item: 45 , then  drop off at: [0, 20]
Brute force cost: 0.0317440032959
write into file......
Please list output file name:
 >res.csv
Writing into file......


[YuantekiMacBook-Air:eecs221 YvetteQ$ python warehouseapp.py
```

B. Terminal output of optimized 10 parts with time cost approx.： 15 seconds
Brute force for one order under 11 items are worth waiting for.



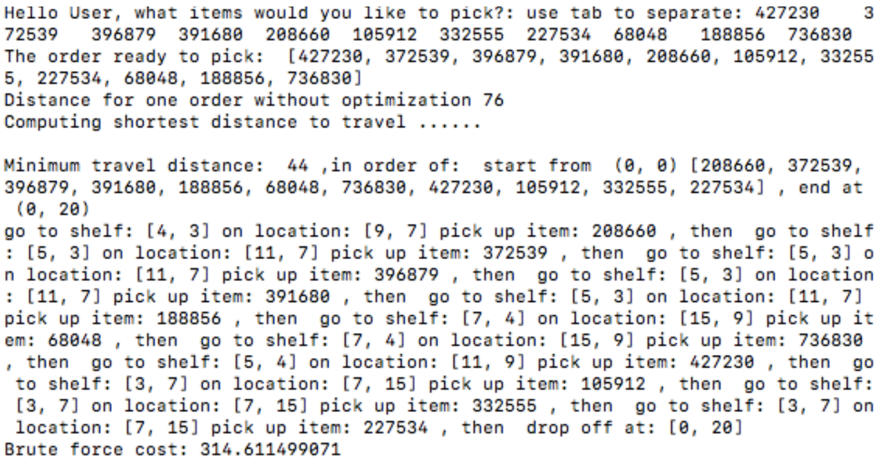
```
YuantekiMacBook-Air:eecs221 YvetteQ$ python warehouseapp.py


Total goods num: 25525
Max rack number in row, col 18 10
Hello User, input manually: yes? no?yes
Hello User, where is your worker? please enter: x,y:  if exceeds,default 0,0
 > 0,0
What is your worker's end location? please enter x,y:  if exceeds, default 0,20
 > 0,20
Hello User, what items would you like to pick?: use tab to separate281610       342706  111873  198029  366109  287261  762832
54489   258540  286457
The order ready to pick:  [281610, 342706, 111873, 198029, 366109, 287261, 76283, 254489, 258540, 286457]
Distance for one order without optimization 70
Computing shortest distance to travel ......

Minimum travel distance:  48 ,in order of:  start from  (0, 0) [254489, 342706, 111873, 287261, 258540, 286457, 281610, 366109
, 76283, 198029] , end at  (0, 20)
go to shelf: [3, 4] on location: [7, 9] pick up item: 254489 , then  go to shelf: [4, 4] on location: [9, 9] pick up item: 342
706 , then  go to shelf: [4, 4] on location: [9, 9] pick up item: 111873 , then  go to shelf: [4, 4] on location: [9, 9] pick
up item: 287261 , then  go to shelf: [5, 3] on location: [11, 7] pick up item: 258540 , then  go to shelf: [6, 3] on location:
 [13, 7] pick up item: 286457 , then  go to shelf: [5, 5] on location: [11, 11] pick up item: 281610 , then  go to shelf: [5,
5] on location: [11, 11] pick up item: 366109 , then  go to shelf: [5, 5] on location: [11, 11] pick up item: 76283 , then  go
 to shelf: [5, 6] on location: [11, 13] pick up item: 198029 , then  drop off at: [0, 20]
Brute force cost: 15.1605319977
write into file......
Please list output file name:
 >res.csv
Writing into file......


YuantekiMacBook-Air:eecs221 YvetteQ$ 
```

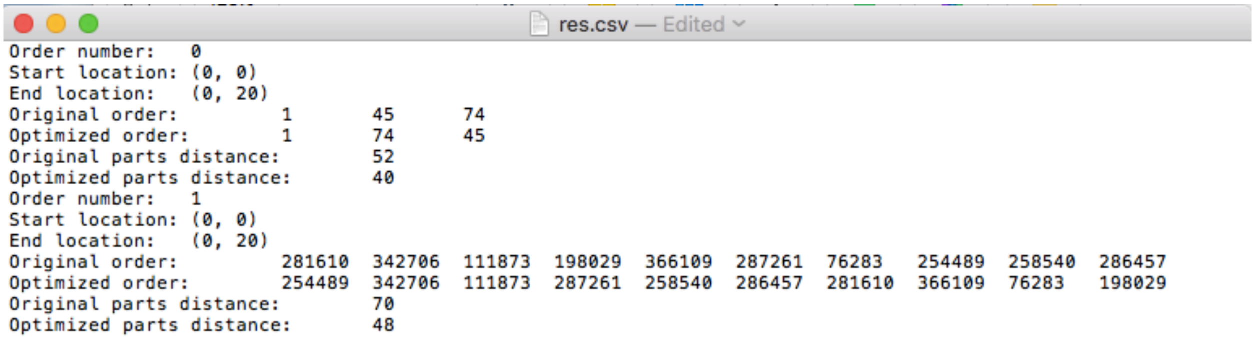
11 items:



```
Hello User, what items would you like to pick?: use tab to separate: 427230    3
72539   396879  391680  208660  105912  332555  227534  68048   188856  736830
The order ready to pick:  [427230, 372539, 396879, 391680, 208660, 105912, 33255
5, 227534, 68048, 188856, 736830]
Distance for one order without optimization 76
Computing shortest distance to travel ......

Minimum travel distance:  44 ,in order of:  start from  (0, 0) [208660, 372539,
396879, 391680, 188856, 68048, 736830, 427230, 105912, 332555, 227534] , end at
 (0, 20)
go to shelf: [4, 3] on location: [9, 7] pick up item: 208660 , then  go to shelf
: [5, 3] on location: [11, 7] pick up item: 372539 , then  go to shelf: [5, 3] o
n location: [11, 7] pick up item: 396879 , then  go to shelf: [5, 3] on location
: [11, 7] pick up item: 391680 , then  go to shelf: [5, 3] on location: [11, 7]
pick up item: 188856 , then  go to shelf: [7, 4] on location: [15, 9] pick up it
em: 68048 , then  go to shelf: [7, 4] on location: [15, 9] pick up item: 736830
, then  go to shelf: [5, 4] on location: [11, 9] pick up item: 427230 , then  go
 to shelf: [3, 7] on location: [7, 15] pick up item: 105912 , then  go to shelf:
 [3, 7] on location: [7, 15] pick up item: 332555 , then  go to shelf: [3, 7] on
 location: [7, 15] pick up item: 227534 , then  drop off at: [0, 20]
Brute force cost: 314.611499071
```

C. Output in file res.csv



```
Order number:   0
Start location: (0, 0)
End location:   (0, 20)
Original order:         1       45      74
Optimized order:        1       74      45
Original parts distance:        52
Optimized parts distance:       40
Order number:   1
Start location: (0, 0)
End location:   (0, 20)
Original order:         281610  342706  111873  198029  366109  287261  76283   254489  258540  286457
Optimized order:        254489  342706  111873  287261  258540  286457  281610  366109  76283   198029
Original parts distance:        70
Optimized parts distance:       48
```

3.

Command line input:

python warehouseapp.py

Then please follow the instrumentation prompted.