1. Identification:

Nan Dai 11162942 nbd596

Solo

2. Implementation:

1.

The problem is given number of packages and vehicles, find the optimal solution to delivery every packages from their sours to destination. I have a Point class to represent the locations, each point has a x coordinate and a y coordinate, an ID that represent which package it belongs. And inside Point class, a getParrent method that can return the package it belongs. And a Package class to represent packages, each package has a sours, destination, currenPos, onCar, delivered and ID. Sours and destination are points, currentPos represent where the package is, it will updated after a vehicle pick it up, and binding with vehicle's current location, if the package is at its destination, unbinding with vehicle's current location. I also have a vehicle class to represent vehicles, each vehicle has a sours, destination, currentPos, hasPackage, packages, numberOfPackages, distanceTraveled, history, and ID. Sours and destination are both set the garage (I set the location of garage to be (0,0)). CurrentPos updated whenever it pickup or drop a package, hasPackage is a boolean value determine if there is any package in this vehicle, packages is a list, append when pickup package, remove when drop package. numOfPackages is just the length of packages. DistanceTraveled updated whenevery it moves. And history keep track of the path of this vehicle.

2.

The successor function first find the best position to go from current location, then compare the cost (by using heuristic function) of pick up the package with pick anyother packages, if the cost of pick up this package is smaller than other, then pick it up. If sending another vehicle to it reduce the estimate cost, send another vehicle. Recursively call untill reach the goal state. Because I'm removing each point from the Points after

delivered or picked a package,
so if the length of Points is 0, then I reach the goal state.

3.
The heuristic function I'm using is to estimate the cost of pick/drop each package.
For p in packages:
if theCostOfPick(p) is minimum:
minimum = theCostOfPick(p)
packageToPick = p
goPiCk(p)

And the cost of picking up the package is:
the vehicle's current traveled distance + distanceOf(car.currentPos, package.sours) +
distanceBacktoGarage()
where the distanceBacktoGrage compute the distance of dropping every package on
vehicle and then back to garage.
Travel through all minimum heuristic value doesn't mean it's the best solution, so
I decide to use hueristic on when to send another vehicle to next point. Whenever the
vehicle is willing to move to next point, it compares the cost of go back to garage
and send another vehicle to pick the package with the cost of let the current
vehicle pick the package and then go back to garage, if sending another car is
expensive, then it will do the work by it self.

When vehicle is 1, it always has the least total distance, but always has the max
distance of single vehicle traveled. So I can determine when to send another vehicle
by comparing the cost of sendAnotherCar and the cost of pickUpUsingCurrentCar.

I think the estimate is very precise. But I'm having problem implementing it, so I
used a easyToCompute version of it. But basically doing the same work.

And an example is:
Car1 picked up package1 at pick1, now it compares with cost.
If the cost of [[[[let car1 drop package1 at drop1 + distanceOf(garage, pick2) +
distanceOf(pick2,drop2) +
distanceOf(drop2,garage)]]]]

is smaller than [[[[let car1 to pick package2 at pick2 then drop package1 and
package2]]]]:
then send another car to package2, and let car1 delivery every package on it
and back to garage.

4.
I tried BFS at first place, it cannot find the optimal solution for some case, but
close.
It works fine when vehicle = 1, and I spend couple hours on figure out how to
change the algorithm so it will do the properly thing when vehicle is greater than
1,
but didn't worked.

Since I'm removing the point from points after pick/drop, and removing the package
from packages
after delivered it, so there is no repeated state.

Just limited depth when current vehicle is the last vehicle available.
The algorithm works better when the difference of vehicles and packages are smalle.

5.
It determine if need to send another vehicle by comparing the COST just described
in 3.
It's not like multiple vehicles are working at same time, it's more like make a
schedule
for it, so we would know how many cars should we use and where should they go.

Multiple packages per vehicle is easy to management, since the Vehicle class has a
field packages = [].

If number of vehicles times number of packages each vehicle can carry is smaller than
number of packages, then there is no way to complete it, exit with message.
Else,
from the garage, find the packages with nearest sours, send a vehicle to there,
then search for the next position to go, if next position is a

destination of a
        package, check if the package is on the vehicle, if true:
drop, else, search for
        another. If next position is a sours of a package, then
compute the distance for this
        vehicle go pick the package and delivery every package
currently on vehicle PLUS the
        longest distance, then compute the distance for this vehicle
delivery every package
        currently on vehicle PLUS the distance for send another
vehicle to the package and
        delivery the package then back to garage PLUS the longest
distance. Compare two
        results, if send another vehicle is more expensive than pick
the package using current
         vehicle, then let the current vehicle pick up. Once the
vehicle reach next position,
         call the search algorithm recursively.

3. Results:
        a) search nodes are 3N+M, N packages 2N Points and M vehicles.
        b) depth of search in the search tree is 2N. It searchs all
other points to find the
        next position.
        c) time is actually quick, when vehicle = 10, packages = 500,
it terminated under 3
        seconds.
        d) the maximum size of queue would be 3N+M
==========
==========
How many vehicles: 1
How many packages:2
How many packages can each vehicle carry?1
Number of vehicles * vehicle carry limit must >= numer of packages!

Process finished with exit code 1
==========
==========
How many vehicles: 1
How many packages:1
How many packages can each vehicle carry?1
0 's picking x and y pos: 34 29
0 's dropping x and y pos: 44 0

Start:

Picked up package number 0

```
Dropping package number  0                    Number of deliveried
package:  1
FINISHED!!!
Vehicle used: 1
totaldistanceforall: 119.3635292433646
longest: 119.3635292433646
The vehicle #  1 traveled:  119.3635292433646
The vehicle # 1 's Path:
0 ->
0 ->
Time used:  0.00037407875061035156
====================================
====================================
How many vehicles: 2
How many packages:3
How many packages can each vehicle carry?5
0 's  picking x and y pos:  42 41
0 's  dropping x and y pos:  43 21
1 's  picking x and y pos:  16 34
1 's  dropping x and y pos:  29 12
2 's  picking x and y pos:  49 49
2 's  dropping x and y pos:  14 50
------------------------------------------------------------------------
---
Start:

Picked up package number  1
Picked up package number  2
Picked up package number  0
Dropping package number  0                    Number of deliveried
package:  1
Dropping package number  1                    Number of deliveried
package:  2
Dropping package number  2                    Number of deliveried
package:  3
FINISHED!!!
Vehicle used: 1
totaldistanceforall: 213.90058993584393
longest: 213.90058993584393
The vehicle #  1 traveled:  213.90058993584393
The vehicle # 1 's Path:
1 ->
2 ->
0 ->
0 ->
1 ->
2 ->
Time used:  0.0006289482116699219

====================================
```

```
==================================
How many vehicles: 2
How many packages:20
How many packages can each vehicle carry?10
0 's  picking x and y pos:  36 30
0 's  dropping x and y pos:  27 23
1 's  picking x and y pos:  15 4
1 's  dropping x and y pos:  15 1
2 's  picking x and y pos:  50 16
2 's  dropping x and y pos:  48 12
3 's  picking x and y pos:  14 19
3 's  dropping x and y pos:  10 42
4 's  picking x and y pos:  50 45
4 's  dropping x and y pos:  36 27
5 's  picking x and y pos:  41 38
5 's  dropping x and y pos:  42 21
6 's  picking x and y pos:  35 42
6 's  dropping x and y pos:  0 13
7 's  picking x and y pos:  10 21
7 's  dropping x and y pos:  50 38
8 's  picking x and y pos:  12 20
8 's  dropping x and y pos:  14 39
9 's  picking x and y pos:  37 30
9 's  dropping x and y pos:  18 43
10 's  picking x and y pos:  0 10
10 's  dropping x and y pos:  28 0
11 's  picking x and y pos:  36 31
11 's  dropping x and y pos:  3 34
12 's  picking x and y pos:  0 41
12 's  dropping x and y pos:  39 23
13 's  picking x and y pos:  32 18
13 's  dropping x and y pos:  22 48
14 's  picking x and y pos:  30 28
14 's  dropping x and y pos:  13 26
15 's  picking x and y pos:  26 26
15 's  dropping x and y pos:  17 27
16 's  picking x and y pos:  26 13
16 's  dropping x and y pos:  2 40
17 's  picking x and y pos:  16 18
17 's  dropping x and y pos:  28 6
18 's  picking x and y pos:  36 31
18 's  dropping x and y pos:  31 1
19 's  picking x and y pos:  43 13
19 's  dropping x and y pos:  12 38
--------------------------------------------------------------------
---
Start:

Picked up package number  10
Picked up package number  6
```

```
Picked up package number  5
Picked up package number  11
Picked up package number  0
Picked up package number  9
Picked up package number  18
Picked up package number  4
Picked up package number  7
Need another vehicle!
Dropping package number  7                    Number of deliveried
package:  1
Dropping package number  7
Dropping package number  4                    Number of deliveried
package:  2
Dropping package number  5                    Number of deliveried
package:  3
Dropping package number  0                    Number of deliveried
package:  4
Dropping package number  9                    Number of deliveried
package:  5
Dropping package number  11                    Number of deliveried
package:  6
Dropping package number  6                    Number of deliveried
package:  7
Dropping package number  10                    Number of deliveried
package:  8
Dropping package number  18                    Number of deliveried
package:  9
-----------------------------------------------
 ----Vehicle # 1 back to garage, Vehicle # 2 is out for delivery
Active Another Vehicle!
Picked up package number  1
Dropping package number  1                    Number of deliveried
package:  10
Picked up package number  17
Picked up package number  3
Picked up package number  8
Picked up package number  14
Picked up package number  15
Dropping package number  15                    Number of deliveried
package:  11
Dropping package number  14                    Number of deliveried
package:  12
Picked up package number  19
Picked up package number  2
Dropping package number  2                    Number of deliveried
package:  13
Picked up package number  12
Picked up package number  16
Dropping package number  17                    Number of deliveried
package:  14
```

```
Picked up package number  13
Dropping package number  12                    Number of deliveried
package:  15
Dropping package number  8                    Number of deliveried
package:  16
Dropping package number  19                    Number of deliveried
package:  17
Dropping package number  3                    Number of deliveried
package:  18
Dropping package number  16                    Number of deliveried
package:  19
Dropping package number  13                    Number of deliveried
package:  20
FINISHED!!!
Vehicle used: 2
totaldistanceforall: 717.6274291482662
longest: 363.95753926762274
The vehicle #  1 traveled:  353.6698898806434
The vehicle # 1 's Path:
10 ->
6 ->
5 ->
11 ->
0 ->
9 ->
18 ->
4 ->
7 ->
7 ->
4 ->
5 ->
0 ->
9 ->
11 ->
6 ->
10 ->
18 ->
The vehicle #  2 traveled:  363.95753926762274
The vehicle # 2 's Path:
1 ->
1 ->
17 ->
3 ->
8 ->
14 ->
15 ->
15 ->
14 ->
19 ->
2 ->
```

```
2 ->
12 ->
16 ->
17 ->
13 ->
12 ->
8 ->
19 ->
3 ->
16 ->
13 ->
Time used:  0.0075719356536865234

====================================
====================================
How many vehicles: 5
How many packages:20
How many packages can each vehicle carry?50
0 's  picking x and y pos:  13 11
0 's  dropping x and y pos:  3 37
1 's  picking x and y pos:  38 2
1 's  dropping x and y pos:  22 34
2 's  picking x and y pos:  49 46
2 's  dropping x and y pos:  49 20
3 's  picking x and y pos:  49 18
3 's  dropping x and y pos:  8 12
4 's  picking x and y pos:  34 34
4 's  dropping x and y pos:  15 47
5 's  picking x and y pos:  45 15
5 's  dropping x and y pos:  8 12
6 's  picking x and y pos:  7 19
6 's  dropping x and y pos:  7 3
7 's  picking x and y pos:  12 5
7 's  dropping x and y pos:  6 8
8 's  picking x and y pos:  42 29
8 's  dropping x and y pos:  13 18
9 's  picking x and y pos:  24 13
9 's  dropping x and y pos:  28 2
10 's  picking x and y pos:  21 4
10 's  dropping x and y pos:  39 36
11 's  picking x and y pos:  25 29
11 's  dropping x and y pos:  32 23
12 's  picking x and y pos:  30 31
12 's  dropping x and y pos:  30 13
13 's  picking x and y pos:  41 28
13 's  dropping x and y pos:  35 26
14 's  picking x and y pos:  7 3
14 's  dropping x and y pos:  11 9
15 's  picking x and y pos:  6 11
15 's  dropping x and y pos:  35 43
```

```
16 's  picking x and y pos:  2 7
16 's  dropping x and y pos:  29 6
17 's  picking x and y pos:  15 37
17 's  dropping x and y pos:  29 30
18 's  picking x and y pos:  18 26
18 's  dropping x and y pos:  29 42
19 's  picking x and y pos:  43 31
19 's  dropping x and y pos:  30 38
------------------------------------------------------------------------
---
Start:

Picked up package number  16
Need another vehicle!
Dropping package number  16                    Number of deliveried
package:  1
Dropping package number  16
-----------------------------------------------
 ----Vehicle # 1 back to garage, Vehicle # 2 is out for delivery
Active Another Vehicle!
Picked up package number  6
Picked up package number  8
Picked up package number  13
Picked up package number  19
Picked up package number  10
Need another vehicle!
Dropping package number  10                    Number of deliveried
package:  2
Dropping package number  10
Dropping package number  19                    Number of deliveried
package:  3
Dropping package number  13                    Number of deliveried
package:  4
Dropping package number  8                    Number of deliveried
package:  5
Dropping package number  6                    Number of deliveried
package:  6
-----------------------------------------------
 ----Vehicle # 2 back to garage, Vehicle # 3 is out for delivery
Active Another Vehicle!
Picked up package number  14
Picked up package number  7
Dropping package number  14                    Number of deliveried
package:  7
Picked up package number  0
Picked up package number  3
Picked up package number  2
Picked up package number  15
Dropping package number  3                    Number of deliveried
package:  8
```

```
Dropping package number  7                    Number of deliveried
package:  9
Picked up package number  5
Dropping package number  2                    Number of deliveried
package:  10
Picked up package number  11
Picked up package number  17
Picked up package number  1
Need another vehicle!
Dropping package number  0                    Number of deliveried
package:  11
Dropping package number  0
Dropping package number  1                    Number of deliveried
package:  12
Dropping package number  17                    Number of deliveried
package:  13
Dropping package number  11                    Number of deliveried
package:  14
Dropping package number  15                    Number of deliveried
package:  15
Dropping package number  5                    Number of deliveried
package:  16
-----------------------------------------------
 ----Vehicle # 3 back to garage, Vehicle # 4 is out for delivery
Active Another Vehicle!
Picked up package number  9
Picked up package number  12
Picked up package number  4
Picked up package number  18
Dropping package number  12                    Number of deliveried
package:  17
Dropping package number  9                    Number of deliveried
package:  18
Dropping package number  18                    Number of deliveried
package:  19
Dropping package number  4                    Number of deliveried
package:  20
FINISHED!!!
Vehicle used: 4
totaldistanceforall: 902.0586825298012
longest: 433.2904266825692
The vehicle #  1 traveled:  63.91280785141481
The vehicle # 1 's Path:
16 ->
16 ->
The vehicle #  2 traveled:  202.6122518574635
The vehicle # 2 's Path:
6 ->
8 ->
13 ->
```

19 ->
10 ->
10 ->
19 ->
13 ->
8 ->
6 ->
The vehicle # 3 traveled: 433.2904266825692
The vehicle # 3 's Path:
14 ->
7 ->
14 ->
0 ->
3 ->
2 ->
15 ->
3 ->
7 ->
5 ->
2 ->
11 ->
17 ->
1 ->
0 ->
1 ->
17 ->
11 ->
15 ->
5 ->
The vehicle # 4 traveled: 202.24319613835368
The vehicle # 4 's Path:
9 ->
12 ->
4 ->
18 ->
12 ->
9 ->
18 ->
4 ->
Time used: 0.009070873260498047

4. Discussion:

1. When M = 200, N = 5000, it tooks 83.3992919921875 seconds to complete. Here is the result:

Vehicle used: 16

totaldistanceforall: 80649.88954475618

longest: 11109.730838626607

Increasing dimension is just adding another field to Point class, but calculating the distance would be hard.

I didn't implement my idea fully, if I did, the run time might increase alot, but the results would be more precise. I was having problem implementing disBackToGarage().

It should be calculating the distance for delivery every package on that vehicle and back to garage, but I calculated the estimate distance (by finding the farthest drop point and calculate the distance from currentPos to that point PLUS distance from the point to garage).

It is really hard to add things to the data structure, function call functions, not orginzed welly.

2. It is very easy when vehicle is 1, but I didn't know it is so difficult to increasing the size of vehicle.

I have another idea, but I don't know how to implement it. Make the points adjacent, sours positions are adjacent to their drop position and other sours position, and drop positions are adjacent to any other position. Then then we have a map, so we can probably use a better heuristic to search the problem.