I

Consider the instance of the binary knapsack problem given by:
$p = (10, 10, 15, 10, 10)$, $w = (45, 40, 80, 5, 10)$ and $c = 95$.

I.1 Apply the Greedy heuristic to it. Which solution do you get?

First divide the profit p by the weight w of each item to compute the so-called efficiency: $\left(\frac{2}{9}, \frac{1}{4}, \frac{3}{16}, 2, 1\right)$. 0.1

Sorting the variables in non-increasing value of efficiency leads to the sequence $4, 5, 2, 1, 3$. 0.1

When we assemble a solution by taking the items in the previous sequence we find that 1 is the first that does not fit, meaning that the greedy solution consists of variables $4, 5, 2$ and can also be denoted by $x = (0, 1, 0, 1, 1)$. 0.1

The value of the greedy solution is $10 + 10 + 10 = 30$. 0.1

The greedy solution occupies $40 + 5 + 10 = 55$ of the available capacity of 95. 0.1

I.2 Apply the modified Greedy heuristic to it (the heuristic that becomes an approximation guaranteed to deviate at most twice from the optimum). Which solution do you get?

Reasoning exactly as in the previous question one concludes that the first item that does not fit while following the 'Greedy sequence' is item 1. 0.2

This means that the modified Greedy solution will be either the greedy solution $x = (0, 1, 0, 1, 1)$ of the solution consisting only of item 1 i.e. $y = (1, 0, 0, 0, 0)$. 0.1

Since the value of x is 30 and the value of y is 10 (smaller) the modified Greedy solution equals the Greedy solution. 0.2

I.3 Solve the fractional version of the problem. What can you say about the optimal value?
The same item as in the previous questions is important to this question as well: the (optimal) fractional solution is equal to the Greedy solution x with the addition of the fraction of item 1 that still fits the knapsack. This fraction is $\frac{95-55}{45} = \frac{8}{9}$. 0.4

The value of this solution is $30 + 10 \times \frac{8}{9} = \frac{350}{9}$. 0.3

The optimal value is therefore between 30 and $\frac{350}{9} = 38.(8)$ 0.3

NOTE: this is not asked but as curiosity the optimum is 35 given by $(0, 0, 1, 1, 1)$.

II

II.1 The Nearest Addition heuristic can be applied to the TSP. At each step a subtour is expanded by the addition of one node until this subtour becomes a tour visiting all vertices. The heuristic can be described as follows:

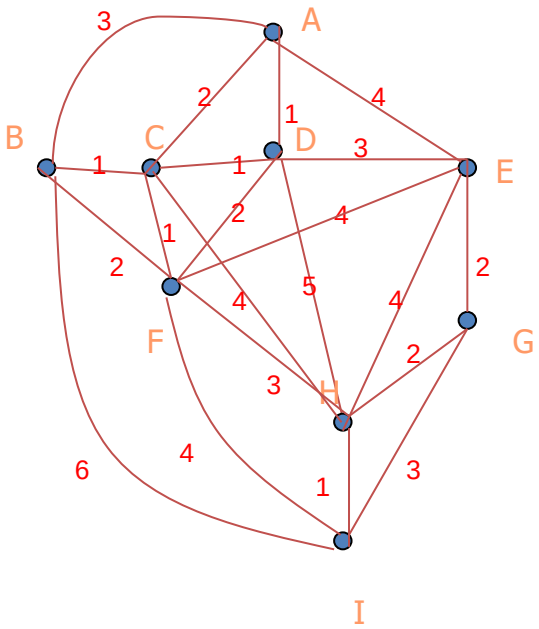
Initialization: start at one node (can be chosen arbitrarily) and initialize subtour T to include that node.

Iteration: until T does not include all nodes find nodes i in T and node j not in T which minimize the addition cost c_{ij} and add j to T after or before i depending on which gives the cheapest increase in cost $c_{ij} + c_{jk} - c_{ik}$ being k the node that was previously either the successor of the predecessor of i.

Determine the order of the running time of this algorithm.

The initialization takes some (content) time. 0.1 Furthermore, the iteration repeats $O(n)$ times 0.1 a search for a minimum value involving two nodes (i, j). If done with a nested pair of loops, then this step is $O(n^2)$. 0.1 Note that by keeping an auxiliary vector where per node i we store the node j that is the closest to i we may find the minimum (i, j) in $O(n)$ time. Then one needs to update the closest node to node i which can again be done in $O(n)$ time. The total effort is therefore $O(n^2)$ if one uses the auxiliary vector as described or $O(n^3)$ if not 0.2 NOTE: because this type of questions are perceived as difficult both alternative answers even if summarily justified were considered correct.

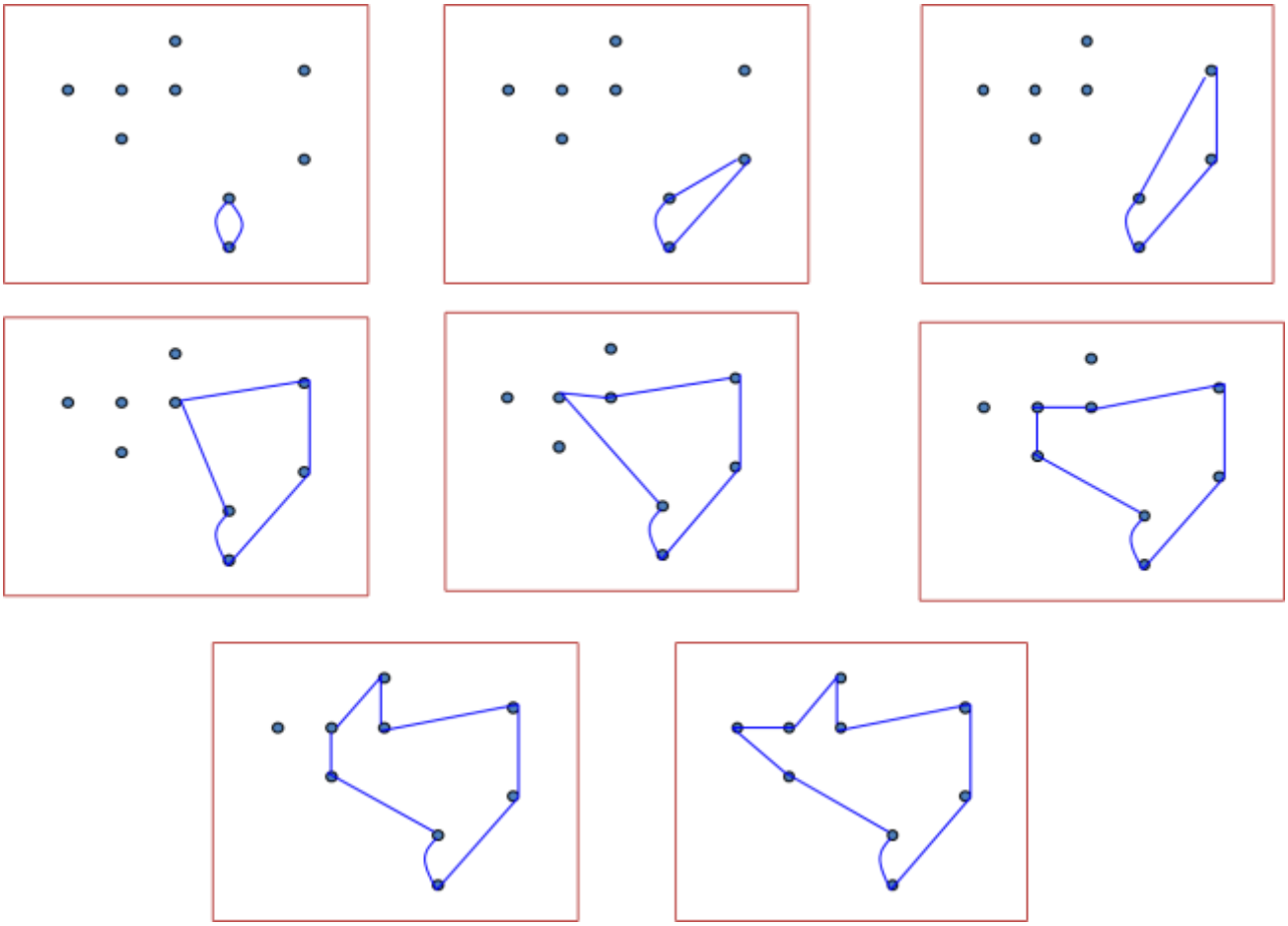
II.2 Consider the graph in the picture on the right and apply the nearest addition algorithm to it starting at node I.



You can consider the omitted edges as having cost equal to the length of shortest path connecting the corresponding nodes, but the given edges should suffice.

Note that when your subtour consists of only one node it plays the role of both i and k meaning that the insertion cost after finding the nearest addition node is $c_{ij} + c_{jk} - c_{ik} = c_{ij} + c_{ji} - 0$ in that case.

Either a 'graphical' solution and/or all computations (if correct) is ok.



As in the pictures above we have:

Initialization: $T = (I, I)$ 0.2

Iteration 1: adding H is the nearest at cost 1 with insertion 1+1=2, leading to $T = (I, H, I)$ 0.1

Iteration 2: adding G after H is the nearest at cost 2 with insertion 2+3-1=4, $T = (I, H, G, I)$ 0.1

Iteration 3: E before G is the nearest at cost 2 with insertion 4+2-2=4, $T = (I, H, E, G, I)$ 0.1

Iteration 4: D before E is the nearest at cost 3 and insertion 5+3-4=4, $T = (I, H, D, E, G, I)$ 0.1

Iteration 5: C before D is the nearest at cost $1$and insertion 4+1-5=0, $T = (I, H, C, D, E, G, I)$ 0.1

Iteration 6: F before C is the nearest at cost 1 and insertion 3+1-4=0, $T = (I, H, F, C, D, E, G, I)$ 0.1

Iteration 7: A before D is the nearest at cost 1 and insertion 2+1-1=2,
$T = (I, H, F, C, A, D, E, G, I)$ 0.1

Iteration 8: B before C is the nearest at cost 1 and insertion 2+1-1=2,
$T = (I, H, F, B, C, A, D, E, G, I)$ 0.1

The total cost is the sum of the insertion costs: 18. 0.5 The same value can be found by following the tour $(I, H, F, B, C, A, D, E, G, I)$. NOTE: only the pictures without solution or cost penalization of 0.5

II.3 Prove that if the TSP instance is metric then the Nearest Addition heuristic can never produce a solution with a value higher than twice the optimum (i.e. it yields a 1-approximation)**.**

Hint: remember the proof that the algorithm of Prim is optimal for the Minimum Spanning Tree.

It is enough to give a (correct) idea of the proof. Prim may also start with any node and at each iteration adds the shortest edge connecting the (growing) tree with a not yet connected node. This leads to the minimum spanning tree. 0.2

We may conclude that the expanding tour includes the minimum spanning tree, plus one extra edge. 0.2

By the triangle inequality $c_{jk} \leq c_{ji} + c_{ik}$ leading to $c_{jk} + c_{ji} \leq 2c_{ji} + c_{ik}$ and by the symmetry of the graph $c_{ij} + c_{jk} - c_{ik} \leq 2c_{ij}$ meaning that the increase in the cost of the tour remains lower than twice the increase in the cost of the tree that leads to the minimum spanning tree. 0.4

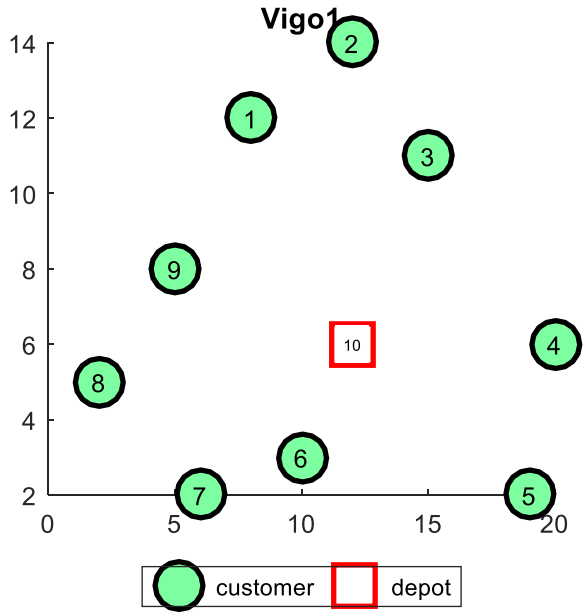
Therefore, the tour found has a cost not higher than twice the minimum spanning tree plus one extra edge which is a lower bound for the optimum of the TSP. 0.2

II.4 Show that the instance above is metric. What can you say about the optimal value with respect to the value of the solution found in **II.2**?

For each edge present there is no shortest way to connect its vertices than the edge, this can be exhaustively verified but such a statement and one example such as ‘CA costs 2 while the cheapest alternative is CD + DA also with cost 2’ suffices. 0.3 Since the solution found in II.2 cannot have a higher value than twice the optimum, the optimum cannot be less than half of the value of the solution found in II.2. So $9 \leq v(TSP) \leq 18$. 0.2

III

Consider your well-known Vigo1 instance of the Capacitated vehicle routing problem as depicted on the right. The demand vector is $q = (5,3,2,3,6,2,2,3,2,0)$ and the vehicle capacity is $Q = 10$. The (rounded) distances are:



	1	2	3	4	5	6	7	8	9	10
1	0	4	7	13	15	9	10	9	5	7
2	4	0	4	11	14	11	13	13	9	8
3	7	4	0	7	10	9	13	14	10	6
4	13	11	7	0	4	10	15	18	15	8
5	15	14	10	4	0	9	13	17	15	8
6	9	11	9	10	9	0	4	8	7	4
7	10	13	13	15	13	4	0	5	6	7
8	9	13	14	18	17	8	5	0	4	10
9	5	9	10	15	15	7	6	4	0	7
10	7	8	6	8	8	4	7	10	7	0

The angles of the position vector of each customer, in degrees, are:
$\alpha = (123.7, 90.0, 59.0, 0.0, -29.7, -123.7, -146.3, -174.3, 164.1)$.

III.1 Apply the sweep algorithm to the Vigo1 instance. To this end, start at the angle -180.0. and take the customers in increasing order of their angles in steps (buckets) of 45 degrees, breaking ties within the same bucket by considering the demand in increasing order. (**Note:** This corresponds to the criteria `+(45)angle, +(0)quantity` applied to the function SweepSequence of your code repository.)
Which solution do you obtain?

Let us first take the angles and assign each customer to the corresponding bucket of 45 degrees:
$(123.7, 90.0, 59.0, 0.0, -29.7, -123.7, -146.3, -174.3, 164.1)$ becomes $(90, 90, 45, 0, -45, -135, -180, -180, 135)$. 0.3

90	5	1
90	3	2
45	2	3
0	3	4
-45	6	5
-135	2	6
-180	2	7
-180	3	8
135	2	9

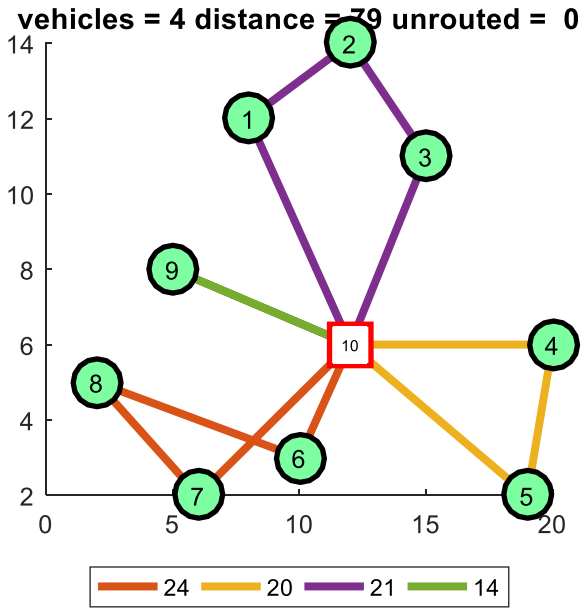
We can see that we obtain ties. The ties will be broken by the quantity in increasing order. We get the information as on the tables on the right. Sorting in increasing values of the first column and for equal values increasing values of the second column leads to the table below:

-180	2	7
-180	3	8
-135	2	6
-45	6	5
0	3	4
45	2	3
90	3	2
90	5	1
135	2	9

This means that the sweep algorithm forms routes by visiting the customers in the order $(7, 8, 6, 5, 4, 3, 2, 1, 9)$. 0.3

Considering the demands of each node we see that $(7,8,6)$ has total demand of 7 which fits in the capacity of 10 but the next node 5 with demand 6 needs to start a new route. This route becomes $(5,4)$ with demand 9 and node 3 starts a new route which becomes $(3,2,1)$with demand 10 forcing node 9 into a new route. 0.6

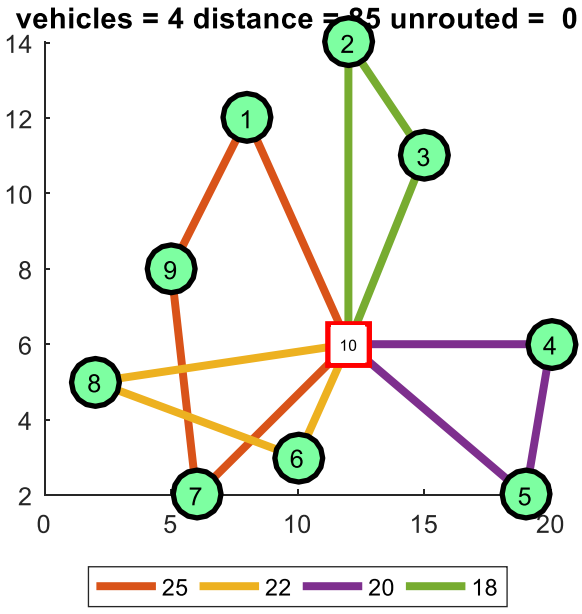
The solution obtained can be seen on the right. The length of each of the 4 routes is indicated and the total distance travelled is 79. 0.3



III.2 Consider the solution on the right and apply to it an iteration of the Ruin and Recreate algorithm.

'Ruin' it by removing the route that visits customers $1, 7, 9$ and 'recreate' it by means of cheapest insertion.

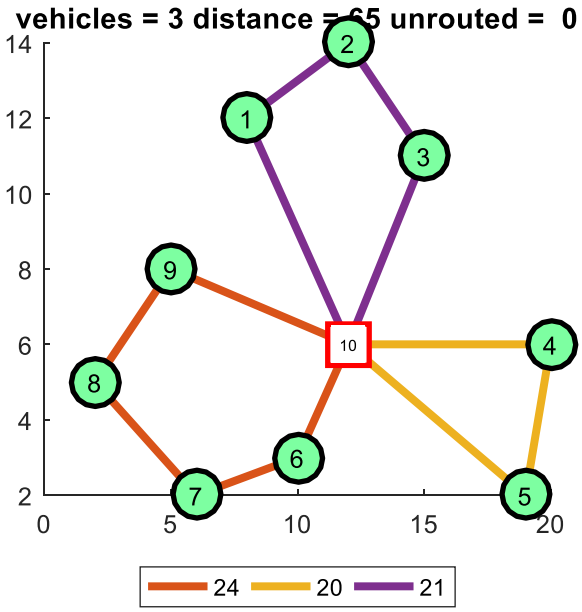
Which solution do you obtain?

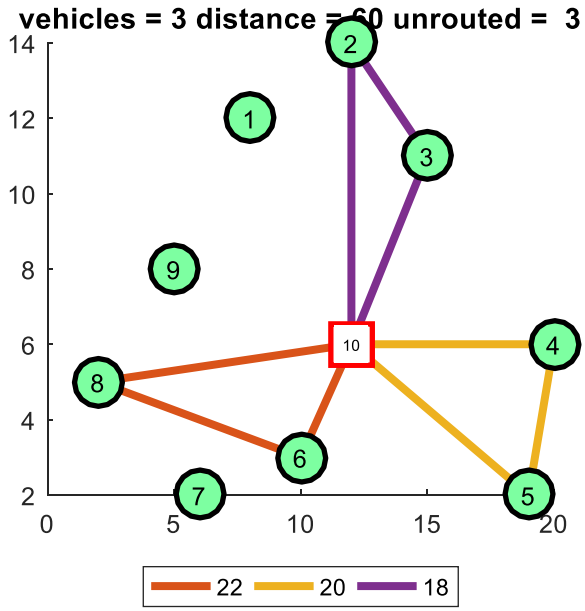


You may use the following pictures to represent the evolution of the Cheapest Insertion and/or report the detail of the computation

After removing the route indicated we obtain the partial solution as drawn on the right.

We can see that the cheapest insertion is going to add (maybe in a different order) 1 between depot and 2, 9 between depot and 8 and 7 between 8 and 6, provided that these are feasible.





A simple computation shows that all are feasible and therefore we obtain the solution on the left.

NOTE: visual arguments as here suffice. Of course complete computations are also ok…

III.3 Suppose now that in instance Vigo1 some of your customers are backhauls. How would you adapt the sweep algorithm to cope with this situation? Apply your algorithm to Vigo1 but now with 2 and 8 being backhauls. Start at angle 0 and sweep counterclockwise.

The most immediate way is to start sweeping at an arbitrary linehaul customer, and add customers while keeping the capacity constraint under control and respecting the precedence. This means that we after the linehaul capacity is reached only backhaul can be inserted, till the backhaul capacity is reached. The route is closed as soon as this is not possible. E.g. a linehaul is the next customer after the linehaul capacity is reached (similarly for backhauls) or a linehaul is to be inserted AFTER a backhaul. After closing the route if the next customer is a linehaul, we continue as above , if it is a backhaul an only-backhaul route is constructed. In this latter case a more sophisticated option is to construct a route with backhauls first, followed by linehauls only that can clearly reverted into a feasible one. The solution found should be:

$\alpha = (123.7, 90.0, 59.0, 0.0, -29.7, -123.7, -146.3, -174.3, 164.1).$

the first node is 8 which is a backhaul, the next is a linehaul so the route is Backhaul-first-Linehaul-second. Insert 9 and 1. The next is 2 which is a backhaul so the route is closed and 2 is used to initialize another Backhaul-first-Linehaul-second route. Next customers are 3 and 4. The customer 5 violates the linehaul capacity so a new route is started including customers 5, 6 and 8.