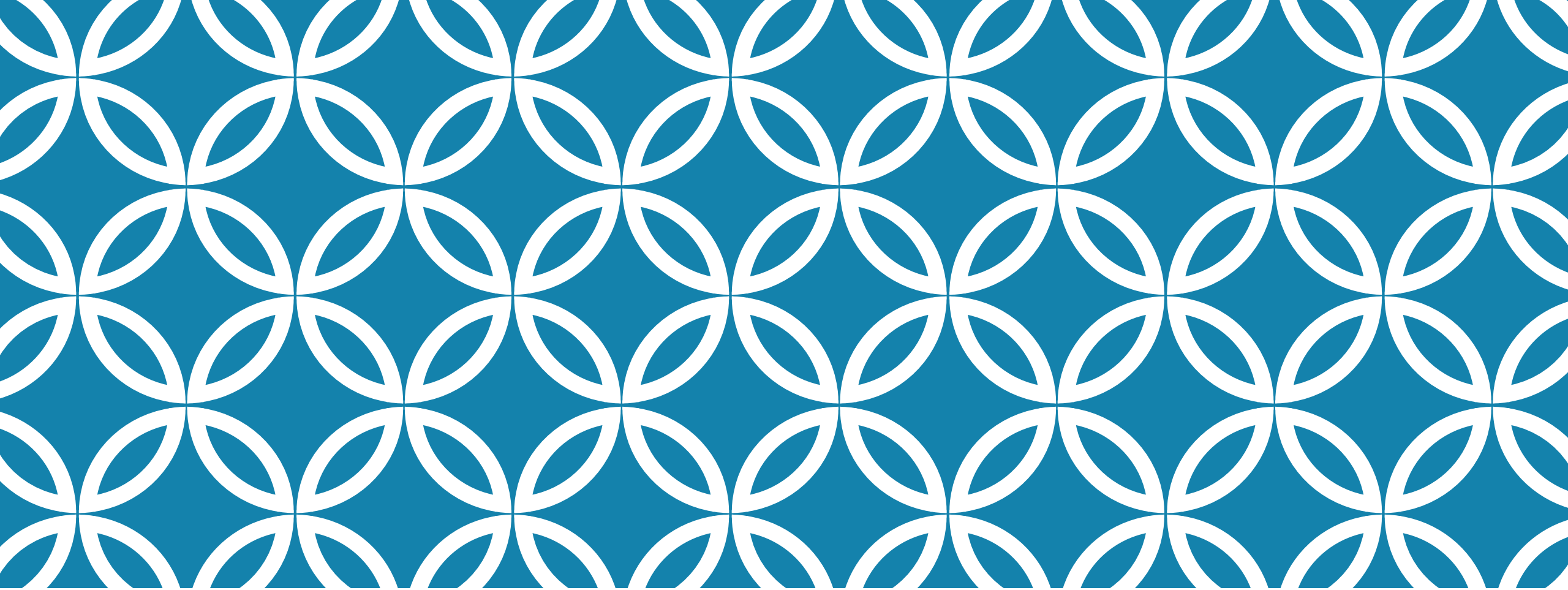
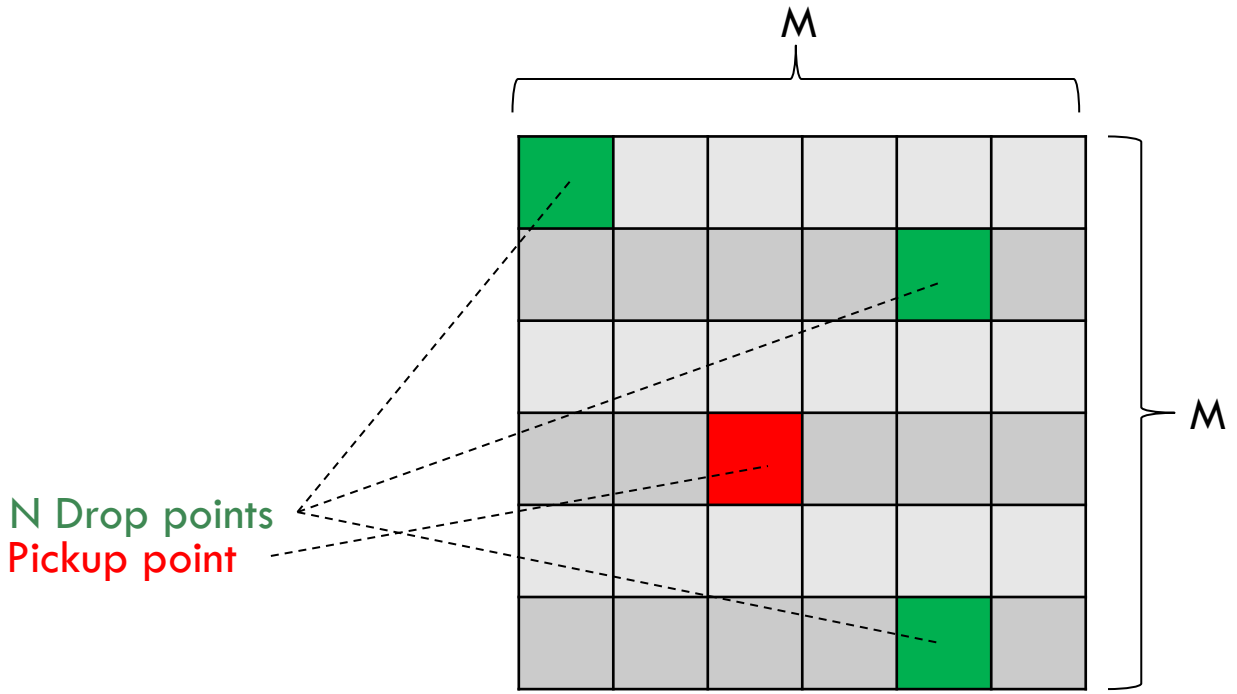


ALGORITHMS' TERM PROJECT

Samyak Chakrabarty
(19EC10084),
Kaizer Rahman
(19IE10044)

PROBLEM STATEMENT



Q

There is a land map in the form of an M*M grid. Each square of the grid represents a location and there are straight paths connecting any two squares. A pickup location is specified by its coordinates (x_s,y_s), from where the items are collected. There are N drop locations specified by (x_i,y_i) where the items are supposed to be delivered. The Delivery Truck has a capacity C and each location requires item of a specific mass q_i to be delivered there. Our aim is to find the optimal path for the truck to travel to minimize distance travelled by it.

INPUT FORMAT :-

- The first line contains two integers N and C.
- The second line contains 2 integers x_s, y_s.
- The following N lines contain 3 integers x_i, y_i, q_i for the i^{th} pickup point.

OUTPUT FORMAT :-

- An array of destinations revealing the tour path followed by the truck.
- The minimum distance travelled by the truck.

EXAMPLE INPUT :-

Write N and C: 4 10
Write Pickup Point Coordinate: 1 1
Write Delivery Locations and Weights:
0 0 4
2 2 6
0 2 5
2 0 5

EXAMPLE OUTPUT :-

Optimal Tour: 0 1 3 0 2 4 0
Distance: 11.312

ABSTRACT

This problem is a generalization of the travelling salesman problem (TSP). It is also known as the Vehicle Routing Problem (VRP) or the Truck Dispatch Problem (TDP). We have divided our problem in two parts – a) where we find the approximately best path for the truck, b) where we divide the trip into multiple trips owing to the capacity constraint of the vehicle. The first part is 2-approximate solution using a Minimum Spanning Tree to find the Hamiltonian Cycle. The second part is also solved 2-approximately using greedy approach. The full algorithm is quite rigorous and time taking. Such problems are usually solved using other methods like genetic algorithms. However here we provide a heuristic, quite good approximation of the problem in less time and space complexity. This algorithm can be made more accurate using Christofides's Algorithm of 1.5-approximation, or other PTAS (Polynomial Time Approximate Solution) methods.

SOLUTION TO THE PROBLEM

A two part solution of the Truck dispatching problem using heuristic approach

FIRST PART ALGORITHM

In this part we have a function that takes in input a map in form of a graph and returns an order of points that approximately forms the most efficient Hamiltonian Cycle, i.e., a tour plan for the Truck assuming C is infinite. The algorithm is as follows:

- Approx_TSP(G):
  - Compute a minimum spanning tree T for G.
  - Select an arbitrary vertex u ∈ T as the root.
  - Initialize an empty tour Γ.
  - Inorder_Traversal(u, Γ)
  - Append u to Γ
  - return Γ
- Inorder_Traversal(u, Γ):
  - Append u to Γ.
  - for each child v of u
    - do Inorder_Traversal(v)

PROOF THAT THIS APPROACH IS 2-APPROXIMATE:

*A 2-approximate solution is one such that Cost(Γ) ≤ 2*Cost(OPT) where Γ is the approximate solution and OPT is the optimal solution.*

Let Γ be an optimal tour for G. By deleting an edge from Γ we obtain a path Γ' and since all edge lengths are non-negative we have length(Γ') < length(Γ) = OPT. Because a path is (a special case of) a tree, a minimum spanning tree is at least as short as Γ' . Hence, **MST ≤ length(Γ') ≤ OPT.**

Now let us assume a path Γ" that is defined by the truck going from the root to every node following the MST and returning. Therefore **Γ" = 2*MST ≤ 2*OPT.** Hence our solution is definitely ≤ 2*OPT.

COMPLEXITY ANALYSIS OF PART 1:

A naïve brute force approach to find the optimal solution of this problem shall give a O(n!) time complexity and O(n) space complexity where n is the no. of delivery locations. A dynamic programming approach of the same will give a time complexity of $O(2^n*n^2)$ and space O(n!). Both of these approaches are unfeasible.

Our approximate approach is of the same time complexity as Prim's Algorithm for Minimum spanning tree, i.e., $O(n^2)$. Our Space Complexity is $O(n^2)$ because we need a matrix to store the MST.

- **$T(n) = O(n^2)$**
- **$S(n) = O(n^2)$**

SECOND PART ALGORITHM

In this part we take a 'weights' array ordered as per the tour from first part of algorithm and divide it into multiple tours as an Online Loading problem. By Online Loading, we mean that we load the Trucks as we traverse the array. In other words, we rearrange the weights to keep it as close to the original order as possible. In this way we ensure that minimum distance is travelled in each tour.

This can be done using a First-Fit algorithm of memory allocation problem. The approach is to maintain an AVL tree of previously loaded bins and check each bin from last to first to find where the new item can fit. If the item can't be fit anywhere, then a new bin is created.

This approach is not optimal, it gives a 1.7-approximate solution to the problem but is much faster and space efficient.

The algorithm is as follows:

- Approx_FFA(C,S,T): (C is the capacity, S is the output array of part 1, T is root of AVL tree)

```
x = T
While(x.key > C – S_i.w):
        x = x.left
If(x != NULL)
        Update_key(x, key+ S_i.w)
        Balance_Tree(x)
else:
        insert(S_i.w)
S = S – {S_i}
Approx_FFA(C,S,T)
```

COMPLEXITY ANALYSIS OF PART 2:

The time complexity of FFMA (First Fit Memory Allocation) is given as O(n*logn). In each iteration, the AVL tree has to be traversed which gives a time complexity of at most O(logn). Hence for n iterations the complexity comes out to be O(n*logn). An additional O(n) space is required to store the AVL Tree.

- **T(n) = O(n*logn)**
- **S(n) = O(n)**

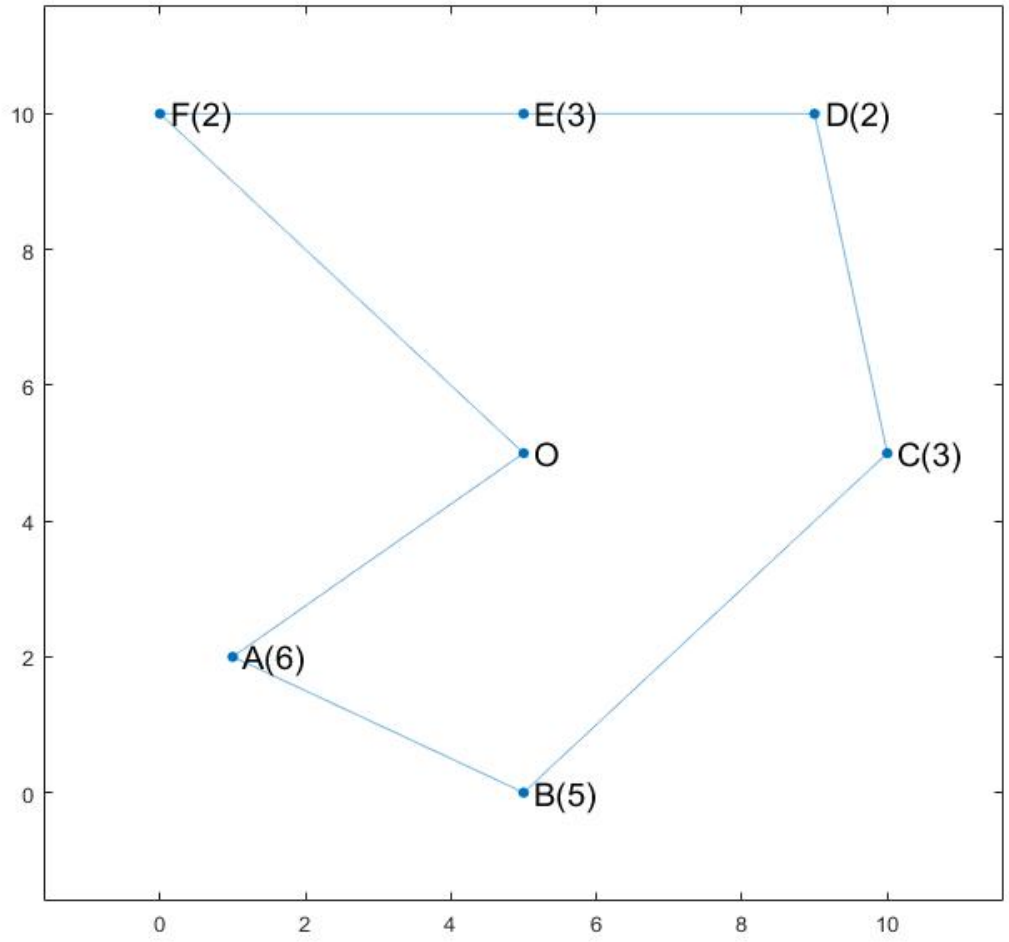
DEMONSTRATION OF THE SOLUTION

An example to show the working of our solution.

EXAMPLE:

Pickup Point: (5,5); Truck Capacity: 10;

Delivery Points and weights: (1,2,6), (5,0,5), (10,5,3), (9,10,2), (5,10,3), (0,10,2)



PART 1:

1. The graph connection all the points with each other is created.
2. The minimum spanning tree with root at O is drawn
3. In-order Traversal of the tree is performed to obtain the result of the first part and root is appended to it.

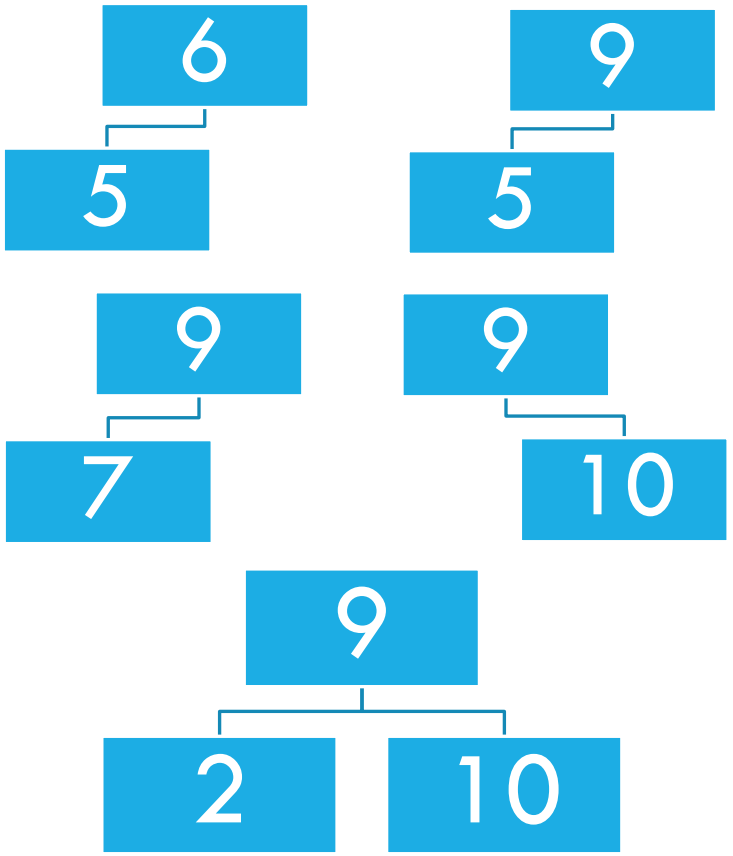
Result: O A B C D E F O

4. The Result is passed onto next function.

EXAMPLE:

Pickup Point: (5,5); Truck Capacity: 10;

Delivery Points and weights: (1,2,6), (5,0,5), (10,5,3), (9,10,2), (5,10,3), (0,10,2)



PART 2:

1. 6 is added to the first bin and a node is created in the AVL tree.
2. 5 cannot be added with 6 since 5+6 = 11 (>10). Hence a new node in the AVL tree is created.
3. 3 is added to 6 since 3+6=9(<10) and the tree is updated.
4. 2 is added to 5 since 2+5=7(<10).
5. 3 is added to 7 since 3+7=10.
6. 2 is added to a new node since 2+9=11(>10)

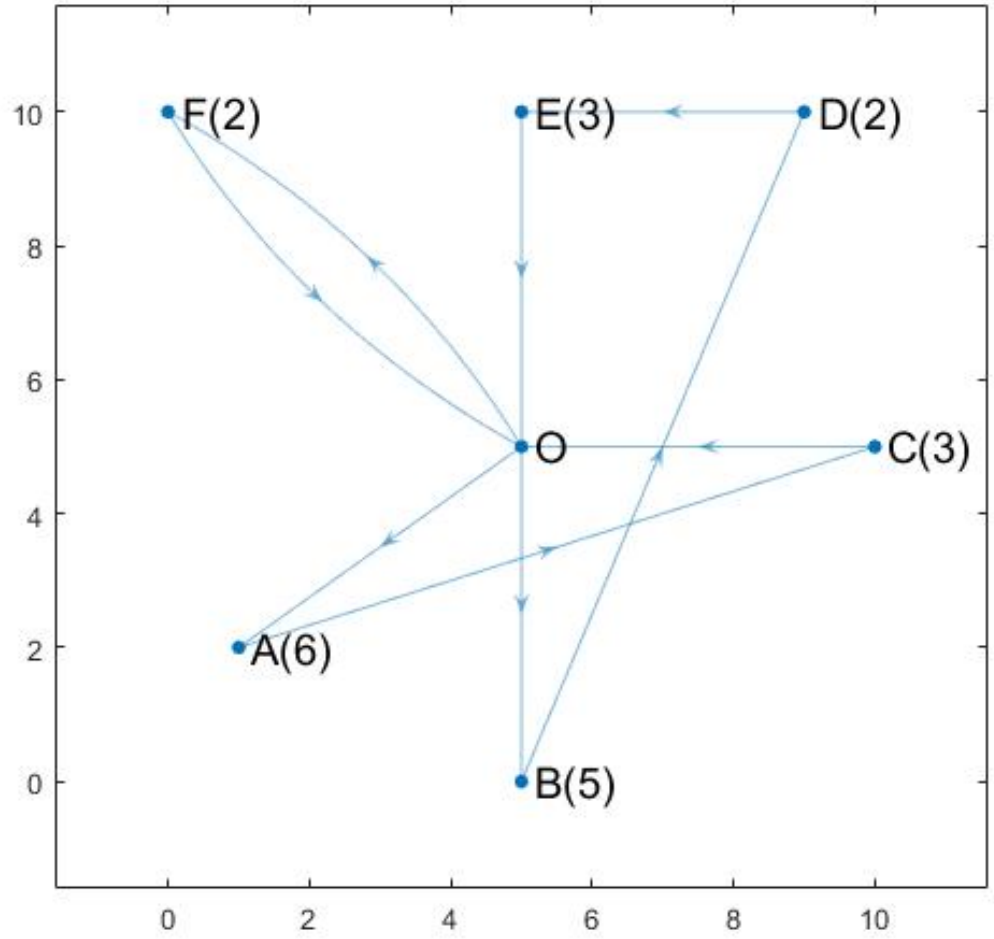
Thus we need three trips:

1. A, C
2. B, D, E
3. F

EXAMPLE:

Pickup Point: (5,5); Truck Capacity: 10;

Delivery Points and weights: (1,2,6), (5,0,5), (10,5,3), (9,10,2), (5,10,3), (0,10,2)



OUTPUT:

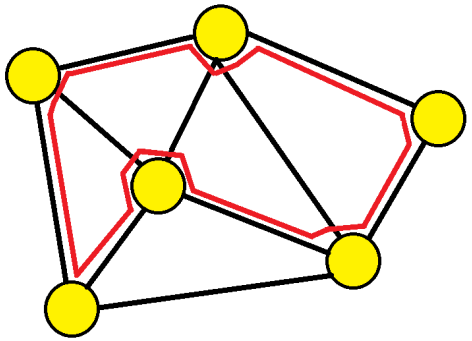
The output is traversed and in each step the distance is added and the node reached is printed.

Output Line :
Optimal Tour: 0 1 3 0 2 4 5 0 6 0
Distance: 58.39874

APPENDIX

- Hamiltonian Cycles
- Minimum Spanning Tree
- AVL Tree

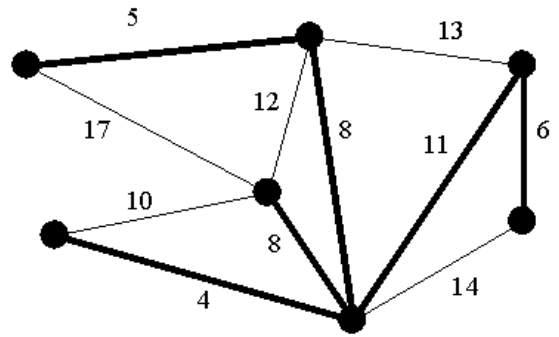
HAMILTONIAN CYCLE



A Hamiltonian path or traceable path is a path that visits each vertex of the graph exactly once. A graph that contains a Hamiltonian path is called a traceable graph. A graph is Hamiltonian-connected if for every pair of vertices there is a Hamiltonian path between the two vertices.

An Eulerian graph G (a connected graph in which every vertex has even degree) necessarily has an Euler tour, a closed walk passing through each edge of G exactly once. This tour corresponds to a Hamiltonian cycle in the line graph L(G), so the line graph of every Eulerian graph is Hamiltonian. Line graphs may have other Hamiltonian cycles that do not correspond to Euler tours, and in particular the line graph L(G) of every Hamiltonian graph G is itself Hamiltonian, regardless of whether the graph G is Eulerian.

MINIMUM SPANNING TREE



A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.

An algorithm to find MST in Prim's algorithm. The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree. And they must be connected with the minimum weight edge to make it a Minimum Spanning Tree.

A Graph G is input. The vertices of G are stored in V, a Tree T is initialised and a random vertex from V is shifted to T. The function prim_MST(G,V,T) outputs the minimum spanning tree.

- prim_MST(G,V,T):

  Find the minimum edge between V and T

  Add the vertex of this edge from V to T

  prim_MST(G,V,T)

AVL TREES

An AVL, also known as self balancing Binary Search Tree is a special kind of BST that obeys all properties of BST and also another property:

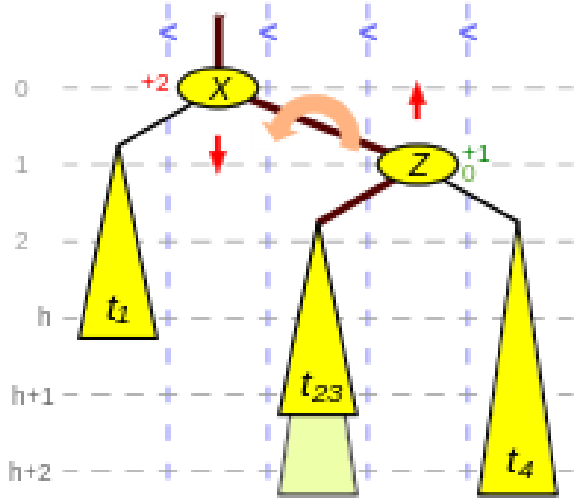
$$|\text{Depth(Left Tree)} - \text{Depth(Right Tree)}| < 2 \text{ at each node}$$

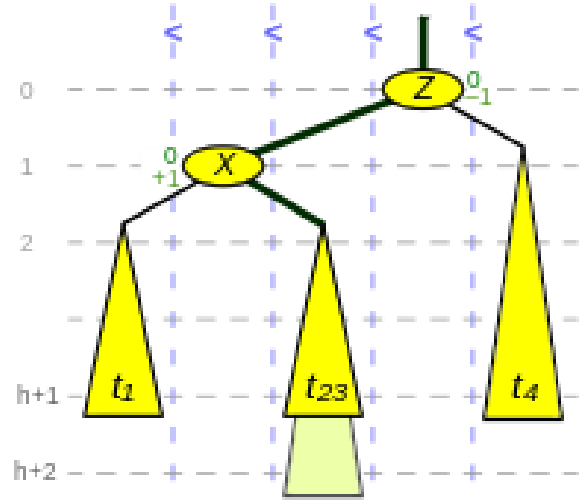
AVL trees have functions like find_min(), find_max(), insert(), delete(), update(), find().

AVL tree maintains their property by Rebalancing the tree after each insertion, deletion and updating. This rebalancing is done by single rotation and double rotation functions.

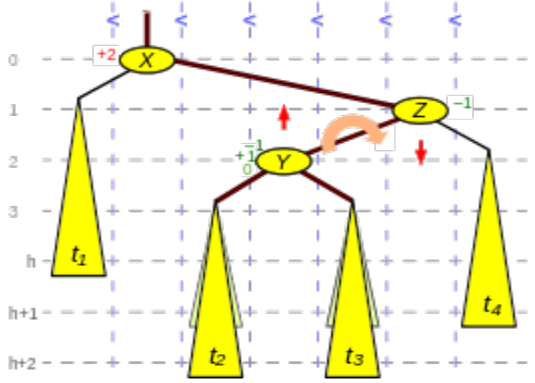
A pictorial representation of these rotations are given in the next slide.

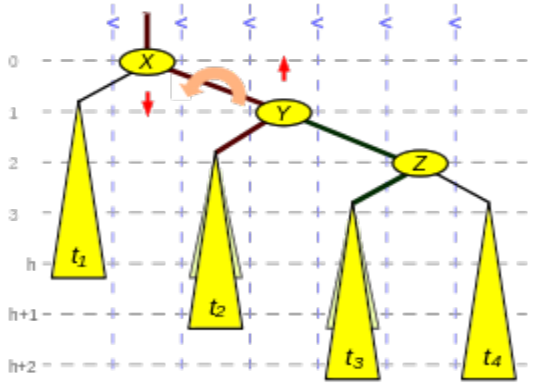
Single Rotation

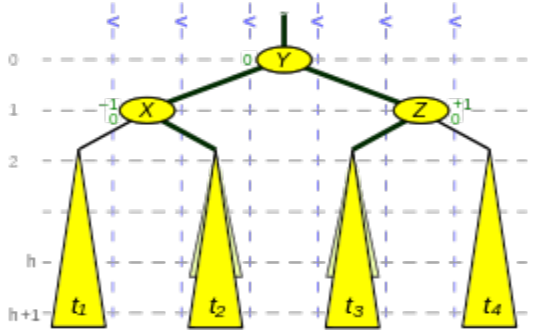




Double Rotation







REFERENCES

- *Near-optimal bin packing algorithms* by David Stifler
- *First Fit bin packing: A tight analysis* by György Dósa1 and Jiří Sgall
- *TU Eindhoven Advanced Algorithms (2IL45) - Course Notes*
- *The Truck Dispatching Problem* by G. B. Dantzig and J. H. Ramser
- *The Vehicle Routing Problem* by Paolo Toth Daniele Vigo
- *Programming, Games and Transportation Networks* by Claude Bergs and A. Ghouila
- *An Algorithm for the Organization of Information* by Adelson-Velsky, Georgy; Landis, Evgenii
- *Shortest Connection Networks And Some Generalizations.* by Prim, R.C.
- *Approximate algorithms for the traveling salesperson problem* by D. J. Rosenkrantz; R. E. Stearns;
- *AVL Trees* by Eric Alexander
- *Bin Packing Problem (Minimize number of used Bins)* – GeeksforGeeks
- *Travelling Salesman Problem | Set 2 (Approximate using MST)* – GeeksforGeeks