Topic two : Truck overview (Slides)

↳ Topic one & two

- imp Definitions :-

⊛ Intermodalism : moving freight by 2 or more modes of transportation.

⊛ TOFC : Trailer on Flat Car (truck & train)

⊛ COFC : Container on Flat Car (sea & train)

Single or double

terms of intermodalism transportation modes

⊛ FTL : "Full truck load", the entire Capacity of the truck is dedicated to one Shipper (Customer)

→ So wether you fill it or not, it's dedicated to the Shipper from origin to destination.

⊛ LTL : "less than truck load", the Space inside the trailer (truck) is divided potentially among multiple Shippers.

Scanned with CamScanner

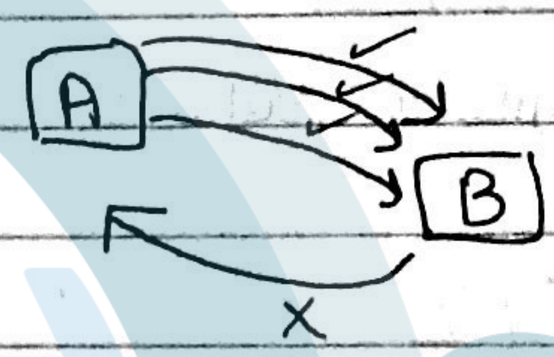
* Hub-to-Hub: where there's no direct shipping between, stops in between are called Spokes
↳ origin & Destination



* Tracking industry trends :-

- 3PL → Third-Party logistics شركة خارجية تقوم بالشحن اللوجستية والاستشارية.

* Flow imbalance
where no shipments From B, only recieving.



KAIZEN TEAM

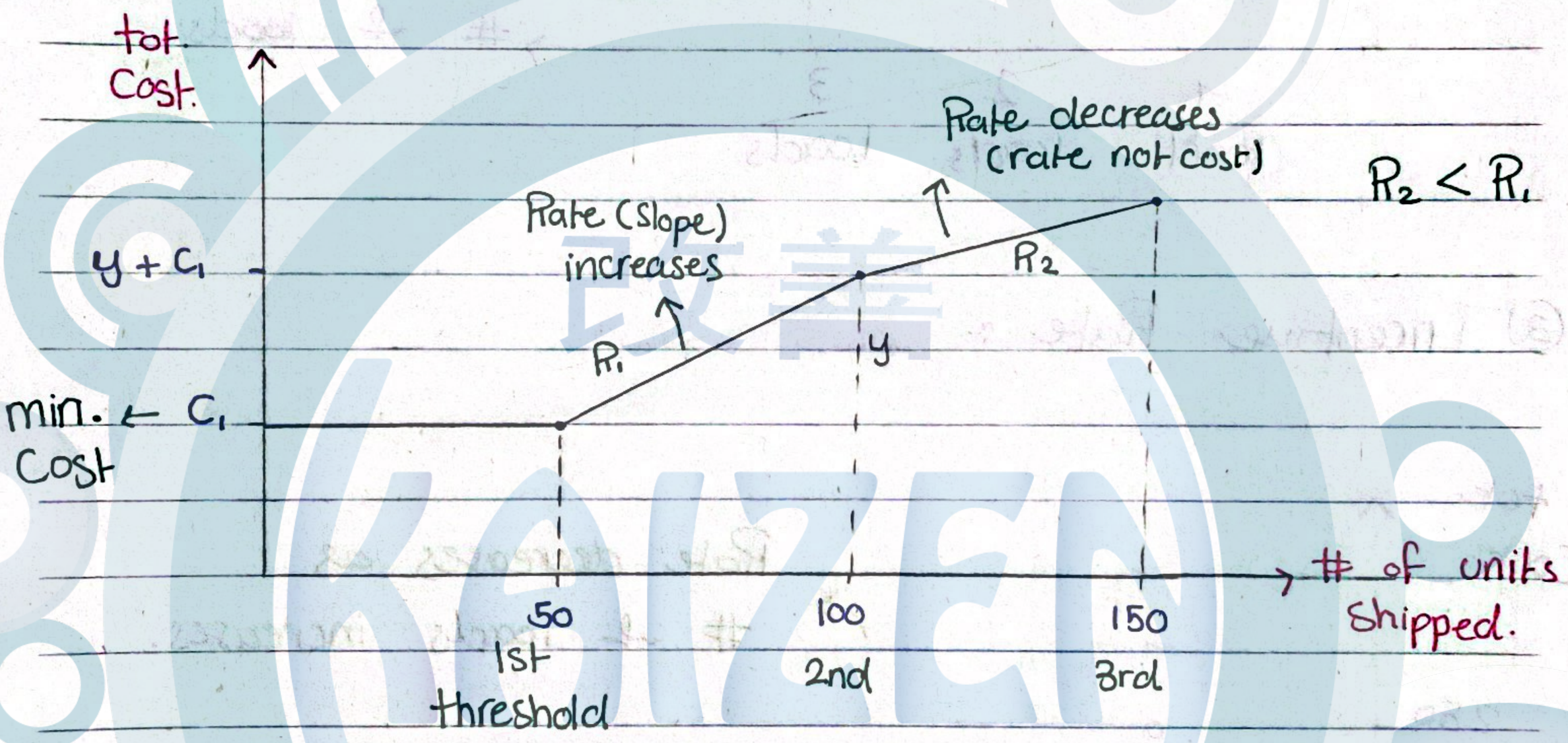
Scanned with CamScanner

Topic Three : Transportation Rates and Costs.

* EOS : Economic of Scale
→ the marginal cost decreases as the quantity shipped increase.

1 Rate Structure - Related to Shipment Size

① LTL , LCL :- Shipment less than truck / Container load:-



$$R = \begin{cases} \$20/\text{unit} & : x \leq 50 \\ \$10/\text{unit} & : 50 < x \leq 100 \\ \$7/\text{unit} & : 100 < x \leq 50 \\ \$5/\text{unit} & : x > 150 \end{cases}$$

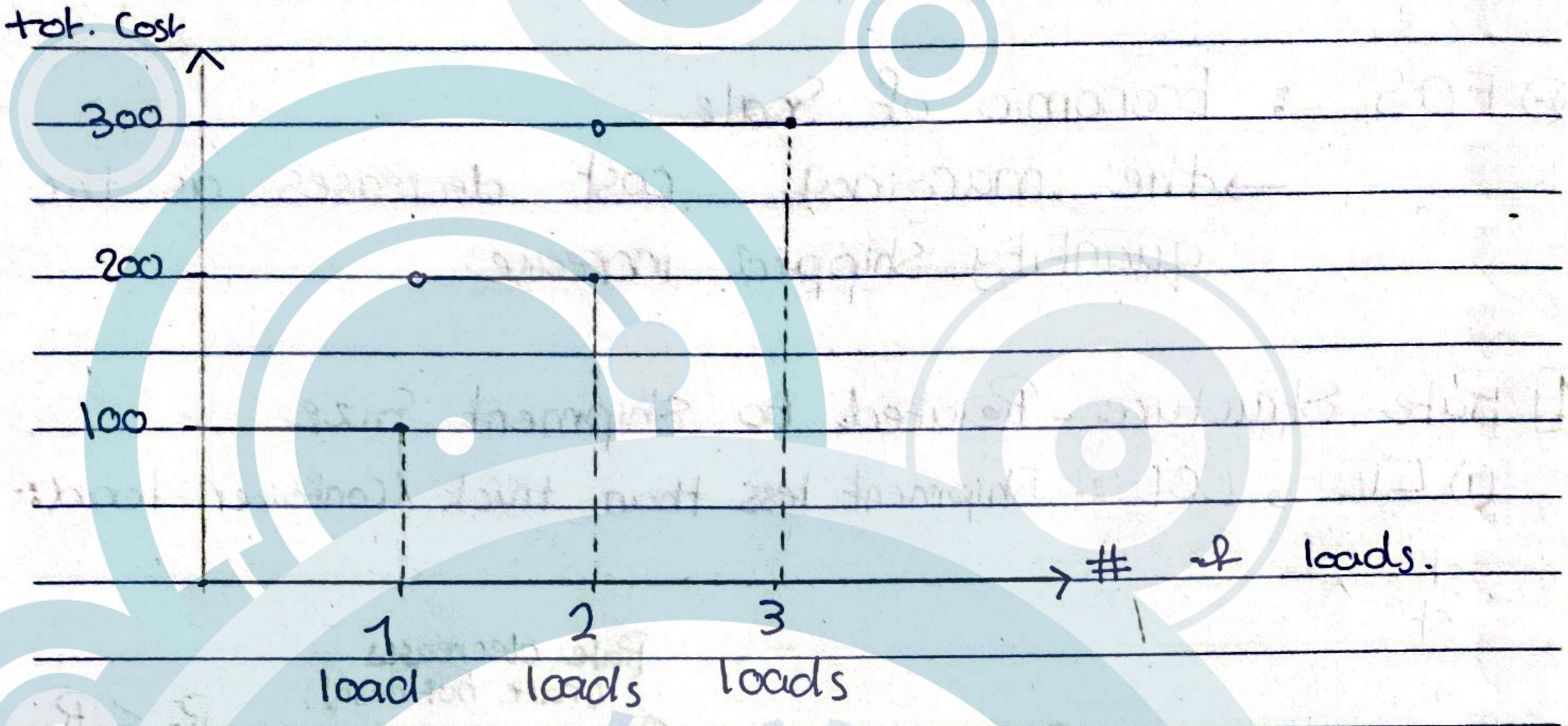
∴ tot. Cost = $20 + (10 * 50) + (7 * 50) + (5 * 20) = \$970$

↳ considering $x = 170$

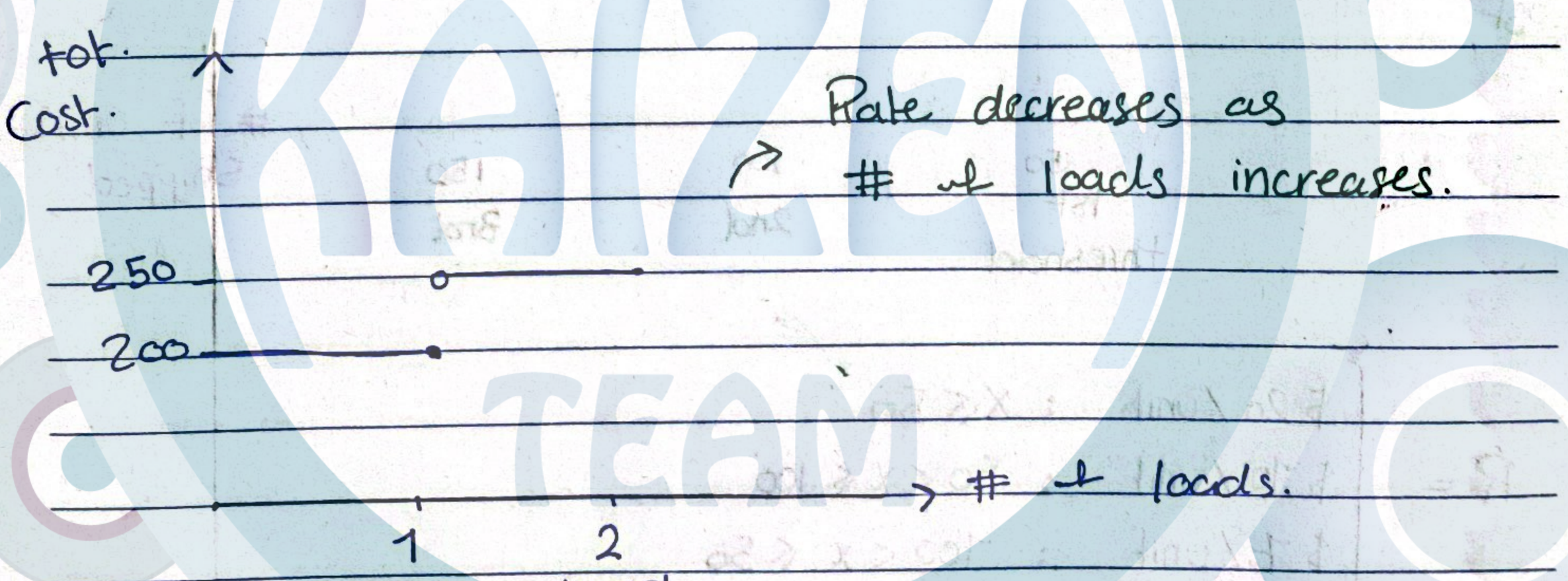
↓ to transfer 170 units *

Scanned with CamScanner

② FTL, FCL : Full truck load Case:-



③ Incentive Rate :-



Scanned with CamScanner

[2] Rate structure - Related to Distance.

① Uniform rate → rate independent of Distance.

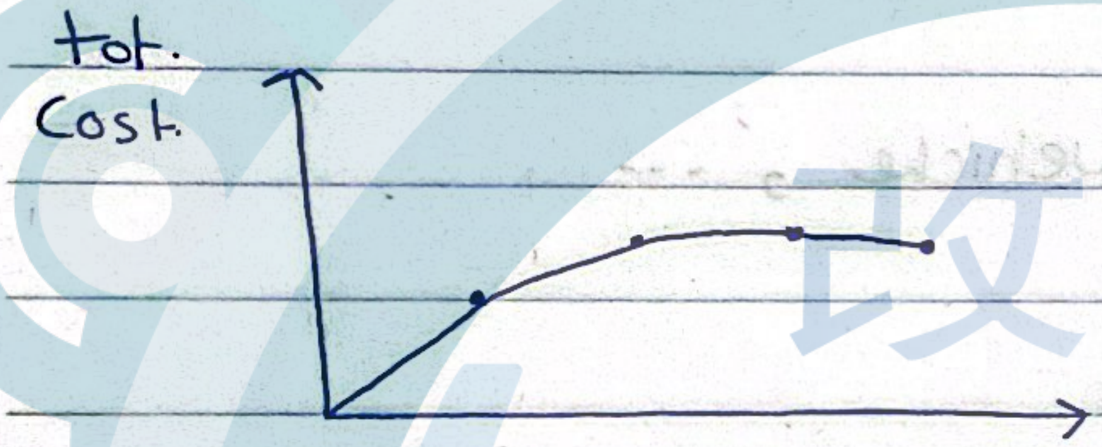
② Propotional Rate → Fixed Cost + variable Cost

Cost: ذي فتحة العداد Taxi

variable Cost: بحدها بتعتمد على كم المسافة.

③ Tapered Rate → cost increases with Distance But at a decreasing Rate.

ذي شركات الشحن



④ Blanket Rate → Constant rate for certain intervals of Distance.

بيعتمد على المنطقة و الماشين.

[3] Rate structure - Related to Demand.

→ Rate deperels on how the Customer is willing to afford.

[4] Rate structure - Related to Product Shipped.

Scanned with CamScanner

✱ Cost Characteristics by mode of trans. :-

① Fixed Costs → vehicles, equipments, ...

② Variable Costs → fuel, maintenance, drivers, ...

③ Treminal Costs. → loading & unloading ...

- Comparison (Rail / truck & water / Air) on topic 3 Slides.

✱ Cost Classification:

① Transportation Cost
→ Cost of movement via vehiche, ...

② Handling Costs
→ Packing, Un-Packing, moving into & out of storage.

③ Holding Costs
→ opportunity cost of capital for time waiting.

④ Facility rent Costs.
→ rent, storage infrastructure and mainteance ...

Scanned with CamScanner

⊛ Mode Choice Problem :-

Steps

① Determining the inventory Space cost at the origin (Production Plant).

② Determining the Holding cost at the Orgin

③ Determining the inventory Space cost at the Destination (warehouse)

④ = = Holding Cost at the Destination

⑤ = = Pipeline inventory Cost (...intransit inventory Cost)

⑥ = transportation Cost for each mode

⑦ = total Cost for each mode

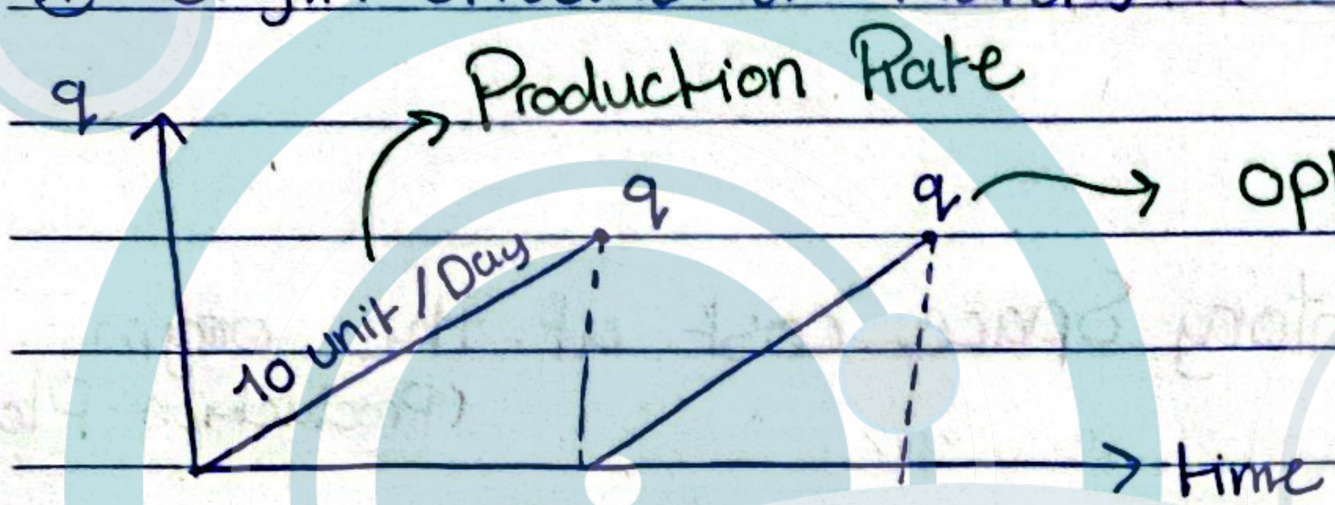
- ⊛ $\overline{TC}_{Rail} = \overline{TC}_{rail}(q)$ → optimal quantity.] tot. Cost as function of q.
- ⊛ $\overline{TC}_{truck} = \overline{TC}_{truck}(q)$

⑧ Determine optimal q for each mode at transportation

⑨ = the most economical (optimal) transportation mode. ✳

Scanned with CamScanner

* Orgin (Production Plant) :-



هي بلي رح اشحنها
فا بدي أحسب
قيمتها

↳ Considering Safety Stock at zero

$SS = 0$.

Scanned with CamScanner

In-class Example

Choosing a Transportation Mode

Hypothetical company has:

- 1 plant, producing 1 product
- 1 warehouse, storing the product from which customers supplied directly
- Plant keeps 0 safety stock inventory, warehouse maintains 100 units
- Average customer demand rate at warehouse: 10 units/day
- Plant produces 10 units/day

Suppose we would like to compare two modes of transportation for moving product from the plant to the warehouse. We want to make a mode choice decision that minimizes total cost.

Rail:

- Shipments in railcars containing at most 400 units
- Cost: $1200 / carload
- Transit time: 20 days

Truck:

- Truckloads of at most 100 units
- Cost: $700 / truckload
- Transit time: 3 days Inventory costs

Inventory cost:

Value of items at plant: $500 / unit

Value of items at warehouse: $512 / unit

Inventory holding cost rate: 25% of value per year held

Plant space cost: $20 / unit*year

Warehouse space cost: $25 / unit*year

Scanned with CamScanner

Goal :

- Our goal is to determine the optimal shipment size q for each mode, and the total cost of the optimally configured system.

Assumptions:

- Assume for now that we will never ship more than a single railcar or a single truckload at a time when minimizing cost

1. What are the annual inventory holding costs at the production plant? (Include the space and holding cost in this calculation.)
2. What are the annual inventory holding costs at the warehouse (Include the space and holding cost in this calculation.)
3. What are the pipeline inventory costs for each mode?
4. What are the transportation costs for each mode?
5. What are the total cost equations for each mode of transportation?

Scanned with CamScanner

* In-class Example one :-

Step 1 : inventory space cost at origin.

Space cost = $20/unit. year * q (unit)

= $20 q / year #

Step 2 : Holding cost at origin.

Inventory Holding cost = inventory holding cost Rate → given 25%.
* value of item x at origin → given 500
* Avg. # of units held in the inventory → $\frac{q}{2}$

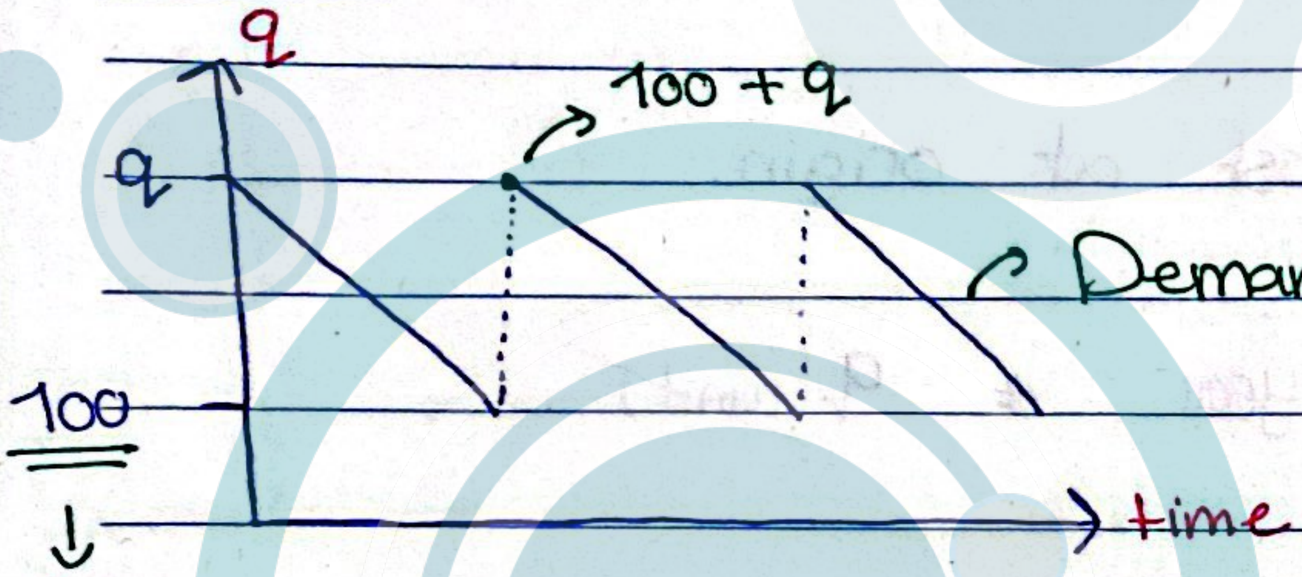
= 0.25 * 500 * (q/2) = 62.5 q #

now; inventory Cost = Space Cost + holding Cost
↙ = 20 q / year + 62.5 q
step 1 + step 2

= 82.5 q / year #

Scanned with CamScanner

Step ③ : inventory Space cost at Destination (WH)



So, inventory Space cost at Dest. =

= $25 / unit . year * (100 + q)

↳ max q
ممكن يكون موجودة.

= $(2500 + 25q) / year.

Step ④ : Holding Cost at Destination (WH)

holding Cost at Destination (warehouse)

$$= 0.25 * 512 * \left(100 + \frac{q}{2}\right)$$

$$= 64q + 12,800 \text{ / year.}$$

⑧ now tot. inventory Cost at WH.

= Step 3 + Step 4

= 2500 + 25q + 64q + 12,800

= 89q + 15,300 #

Scanned with CamScanner

Step ⑤

Pipeline Inventory → inventory cost in-transit.

* P.I = 25% * value of item intransit.

* value of item = $\frac{\text{its value at orgin} + \text{its value at Dest.}}{2}$ ← Avg.
in transit

$= \frac{500 + 512}{2} = 506\ \$ / year.$

* Holding Cost = 25% * 506 = $126.5 /unit.year
Pipeline intransit

↳ Doesn't depend on Shipment Size, it depends on the mode of transportation.

imp
* 10 unit / Day → 365 day / year. ⇒ 3650 units / year
# لي بدي أنقلهم كاملين، إما عالـ truck أو عالـ Rail.

* now given :-

- Rail : capacity = 400 units, Cost = 1200 / Car, time intransit = 20 Days.
- Truck : Capacity = 100 units. Cost = 700 / truck, time intransit = 3 Days.

Scanned with CamScanner

⊛ Pipeline Inventory Cost by Rail ($ / year)

$= 3650 \frac{\text{unit}}{\text{year}} * 20 \text{ days} * \frac{1 \text{ year}}{365 \text{ days}} * \frac{126.5 \$}{\text{unit . year}}$

$= \$\ 25300 / \text{year}$ #

⊛ Pipeline Inventory Cost by truck ($ / year)

$= 3650 \frac{\text{unit}}{\text{year}} * 3 \text{ days} * \frac{1 \text{ year}}{365 \text{ day}} * \frac{126.5 \$}{\text{unit year}}$

$= 3795 \$ / \text{year}$ #

↳ Pipeline inventory cost is constant (not as Punchion of q) ← ما رح يتأثر بـ q لأنه كل ما يصير عندي رح أوزعها

Step 6 : transportation cost for each mode.

- Transportation cost depends on:
  - # of Shipments (عدد الشحنات)
  - transportation mode

$\#\ \text{of Shipments} = \frac{3650}{q}$ ↳ train, truck $[q_{train} \neq q_{truck}]$

Scanned with CamScanner

⊛ Transportation Cost by Rail

$= \frac{1200\ \$}{\text{Railcar}} * \frac{3650\ \text{unit/year}}{q_{rail}} = \frac{4{,}380{,}000\ \$ / \text{year}}{q}$ #

⊛ Transportation Cost by Truck

$= \frac{700\ \$}{\text{Truckcar}} * \frac{3650\ \text{unit/year}}{q_{truck}} = \frac{2{,}555{,}000\ \$ / \text{year}}{q}$ #

Step 7 Total cost for each mode

⊛ total Cost for Rail / truck :-

= inventory cost at origin ⊕ inventory cost at Destination
(holding + Space) step 2 (holding + Space) step 4

⊕ Pipeline inventory cost step 5 ⊕ Transportation Cost step 6

① $\overline{TC}$ for Rail $= \overline{TC(q)} = 82.5q + 15300 + 89q + 25300 + \frac{4380000}{q_{rail}}$

② $\overline{TC}$ for truck $= \overline{TC(q)} = 82.5q + 15300 + 89q + 3795 + \frac{2{,}555{,}000}{q_{truck}}$

Step 8 Determine optimal q for each mode

⇒ Derive $\overline{TC}$ eq. ⇝ q optimal when $\frac{d\overline{TC}}{dq} = 0$ #
with respect to q

⊛ total Cost equation :

. $f(x) = \frac{A}{X} + BX + C$ ⇝ $X^* = \sqrt{\frac{A}{B}}$ → q

Scanned with CamScanner

* To find q_{Rail} (optimal) $\rightarrow \frac{d\,\overline{TC}_{rail}}{q} = 0$

$\Rightarrow 82.5 + 0 + 89 + 0 - \frac{4380000}{q^2} = 0$

$q_{rail} = 160$ units # < 400 capacity ✓

by sub. q_{rail} into $\overline{TC}_{rail}(q)$ Eq.

↳ $\overline{TC}_{rail} = 95,415$ \$ / year.

* To Find q_{truck} (optimal) $\rightarrow \frac{d\,\overline{TC}_{truck}}{q} = 0$

$\Rightarrow 82.5 + 0 + 89 + 0 - \frac{2,555,000}{q^2} = 0$

$q_{truck} = 122$ # > 100 capacity !

So 2 trucks needed.

by Sub q_{truck} into $\overline{TC}_{truck}(q)$ Eq.

↳ $\overline{TC}_{truck} = 61,795$ \$ / year ← lower cost by truck mode ✓

Step 9 :- Ans.

# of trucks needed $= \frac{3650}{100} = 36.5 = 37$ truck.

Scanned with CamScanner

Sunday
30/10

* we already discussed different modes of transportation
  - for mode selection → we break down the cost into different Categories in order to understand total Cost of Particular mode:

→ transportation Cost
→ Handling Cost
→ Holding Cost for the inventory
→ Pipline inventory holding cost.
→ physical facility Cost (space, rent)

→ choose the most economical optimal mode based on the tot. Cost.

* Uncertainty in Supply :-

Sol.
- Safety Stock → to mitigate the effect of Supply Uncertainty.

* Uncertainty in Demand :-

- when Demand > Avg. Demand $\xrightarrow{\text{Sol.}}$ Safety stock of finished Products. [extra units]

- when Demand < Avg. Demand $\xrightarrow{\text{Sol.}}$ Extra Space to hold undelivered units.

Scanned with CamScanner

⊛ Short - Haul transportation

↳ • imp notations and definitions :-

⊛ Network = graph :-

→ include a set of Nodes / vertices and a set of Arcs / edges.

N

A

$G = (N, A)$ → أي graph بعرفه على شكل arcs & nodes.

G ← graph

N ← nodes

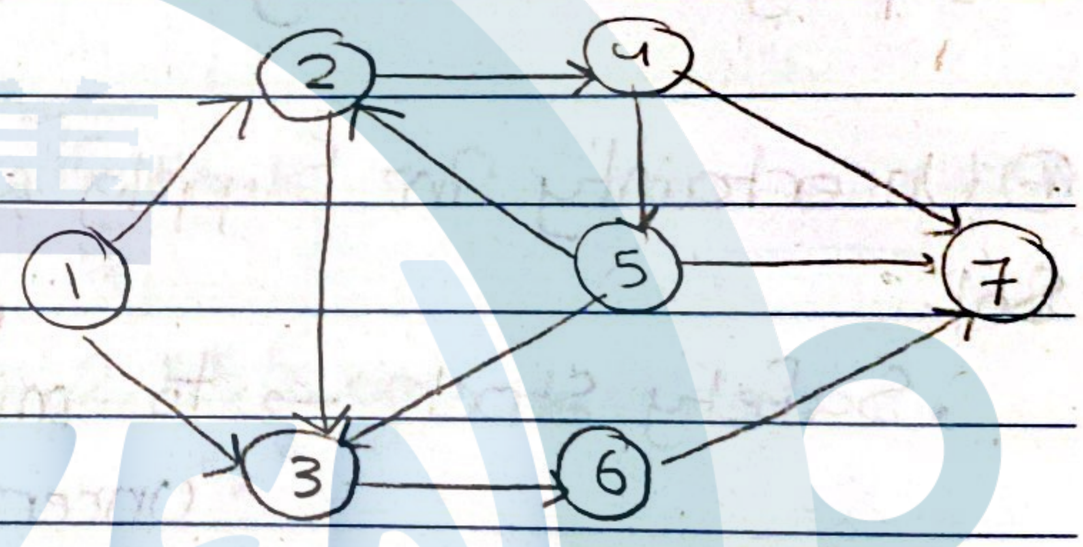
A ← arcs

• example

set of nodes.

⊛ $N = \{1, 2, 3, 4, 5, 6, 7\}$

↳ $|N| = 7$ # of nodes.



⊛ $A = [(1,2), (1,3), (2,4), (2,3), (3,5), (3,6), (2,5), (4,5), (4,7), (5,7), (6,7)]$ → Set of Arcs

$|A| = 11$ → # of Arcs.

Side Note
- node = vertex
- Arc = edge
- graph = Network
- N = set of nodes
- A = set of arcs

Scanned with CamScanner

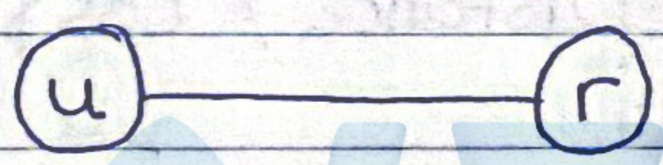
⊛ Direct & Indirect Graphs :-

- Direct graph :
  - → direct arcs →
  - → ordered pair of nodes



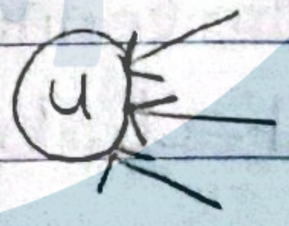
→ not Symmetrical

- Indirect graph :
  - → Indirect arcs —
  - → unordered pair of nodes.



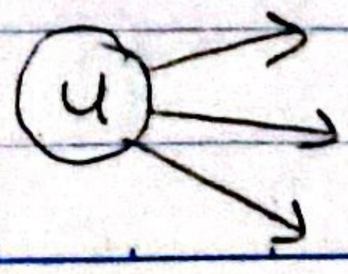
⊛ Degree of the node

- indegree of the node
  - → the number of incoming arcs of the node
  - → for node u : # of arcs which u is the tail or (terminal node.).



indegree = 3.

- out degree of the node
  - → the number of outcoming arcs of the node.
  - → for node u : # of arcs which u is the head (initial node)



outdegree = 3

Scanned with CamScanner

• degree of nodes = # of in-degree + # of outdegree

• indegree of all nodes = out degree of all nodes = # of arcs

⊛ Edges / Arcs may have number of characteristics

• Capacity : max flow on the arc



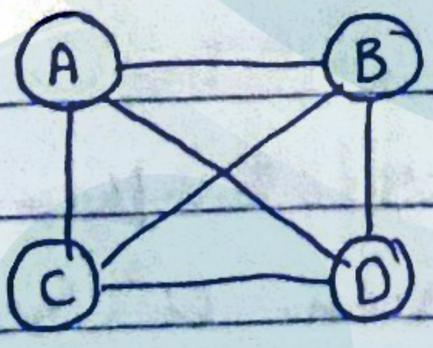
$U_{ij} \neq U_{ji}$

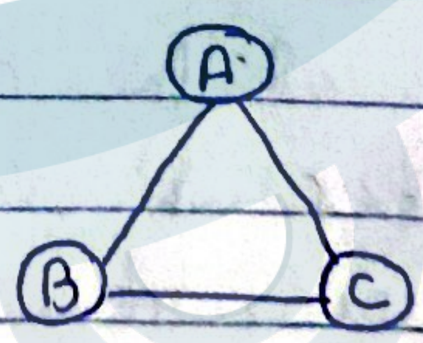
or • Cost : time / Distance / cos$ ($C_{ij} \neq C_{ji}$)

⊛ Complete Graph

→ is a simple undirected graph in which a distinct vertices (nodes) are connected by a unique arc / edge.

→ graph G is connected when each node / vertex is connected by every node.

[



] → there is a direct arc bet. every 2 nodes.

Scanned with CamScanner

→ number of arcs out from each node in a complete graph :

= # of nodes − 1 .

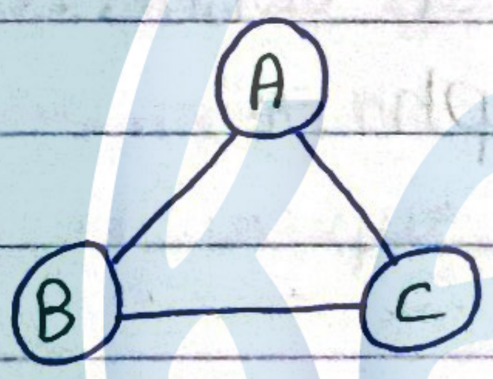
→ number of arcs / edges in a complete graph :

= $\frac{n(n-1)}{2}$, where n is number of nodes.

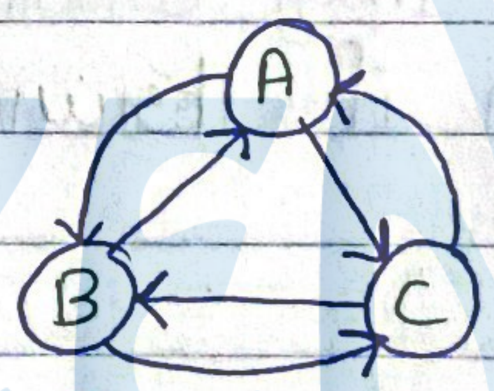
2 لأنه مقسوم على undirected

⊛ Complete di-graph :-

→ is a Complete direct graph



Complete graph



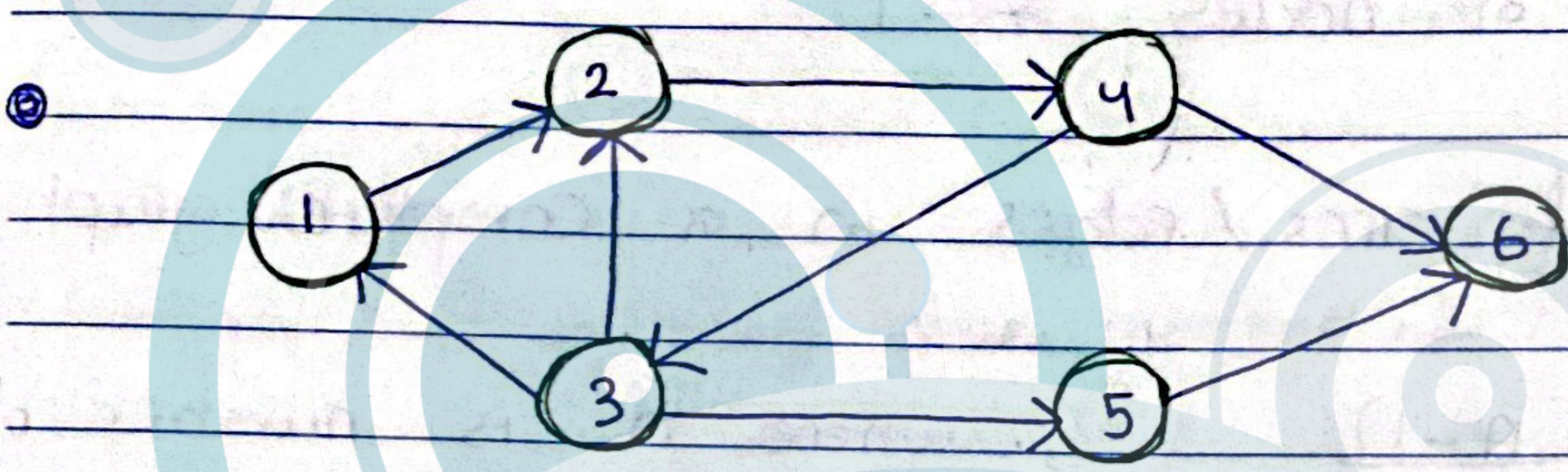
Complete di-graph

⊛ • Connechivity : we Say that nodes i and j are Connected , if the graph Contain at least one path from i to j

• A graph is Connected if every pair of its nodes is Connected , otherwise the graph is unconnected

Scanned with CamScanner

- Based on Connechivity Definition :-
  every Complete graph is Considered to be Connected . والعكس غير صحيح



Direct graph

↳ Un Connected graph , there is no path between node 5 and node 4 and ... etc.

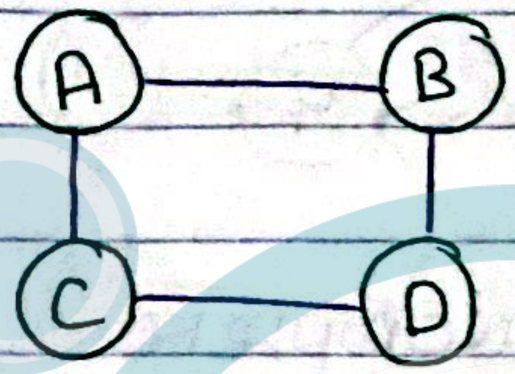
↳ we may Consider it as a Connected graph if it is indirect ...

✱ Path :-

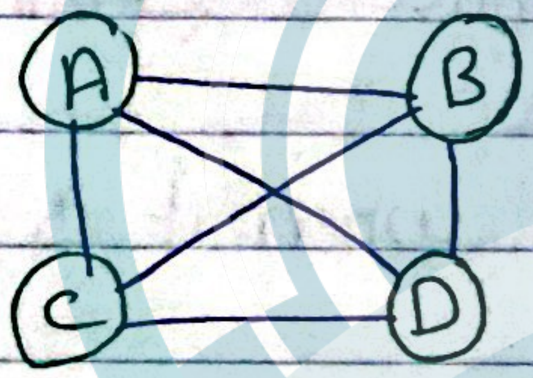
- Assume you have n vertices in V , m edges in A.
- Path exists from v_0 to v_n if there is a list of vertices : $\{v_0, v_1, \dots, v_n\}$ Such that $(v_i, v_{i+1}) \in A$, For all $0 \leq i \leq n$.

Scanned with CamScanner

Tuesday
1/11



→ Connected
→ incomplete
→ in direct.

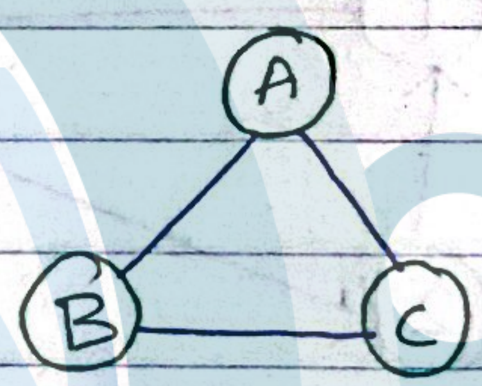


→ Connected
→ Complete
→ in direct.

Recall :-

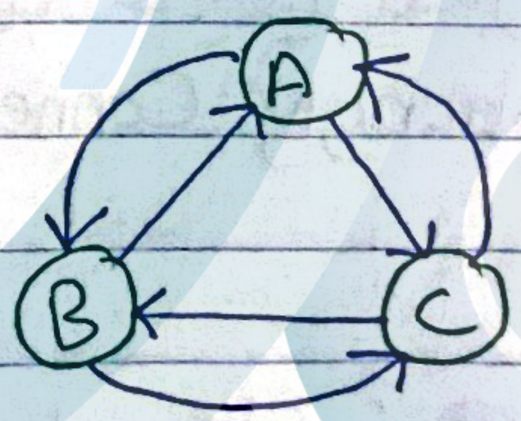
✱ Complete graph:

→ Undirected graph in which a distinct nodes are connected by a unique arc.

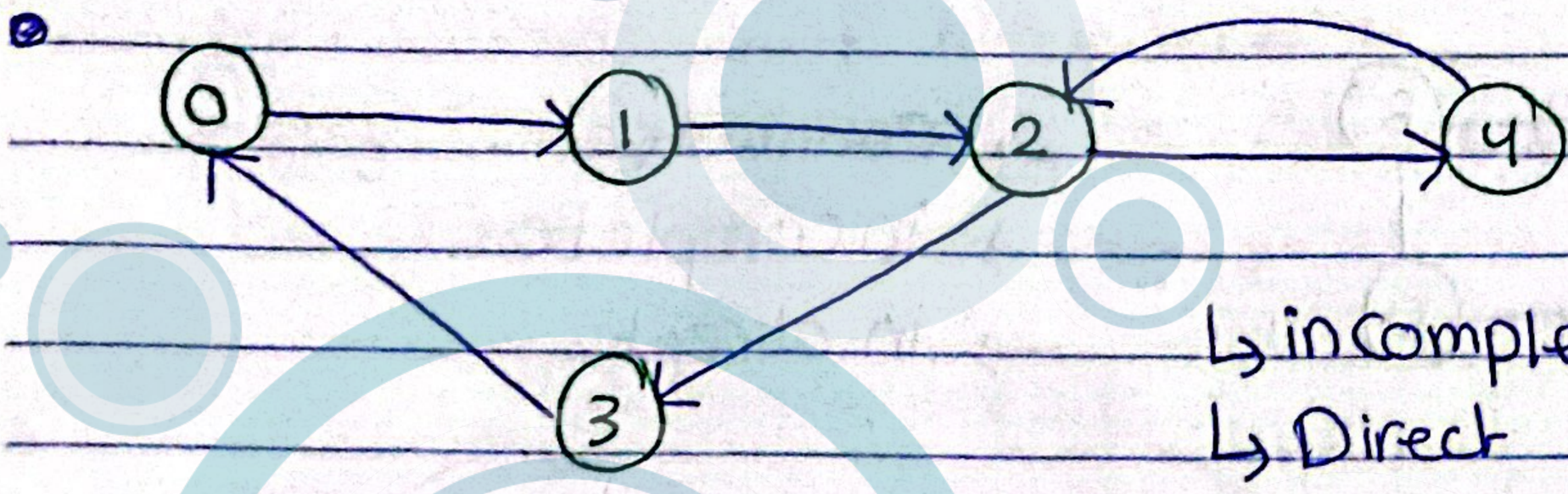


✱ Complete di-graph

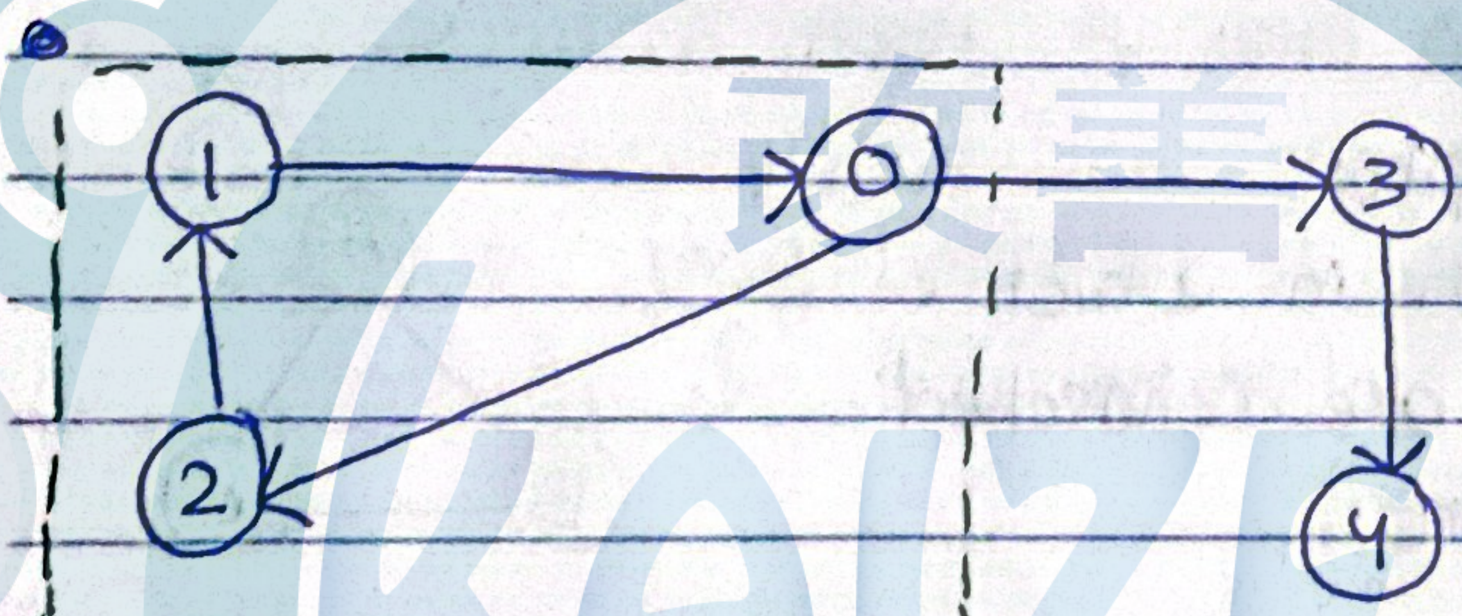
→ directed graph in each every pair of nodes is connected by a pair of unique edges (one in each direction)



Scanned with CamScanner



↳ incomplete
↳ Direct
↳ Connected ← Strongly Connected



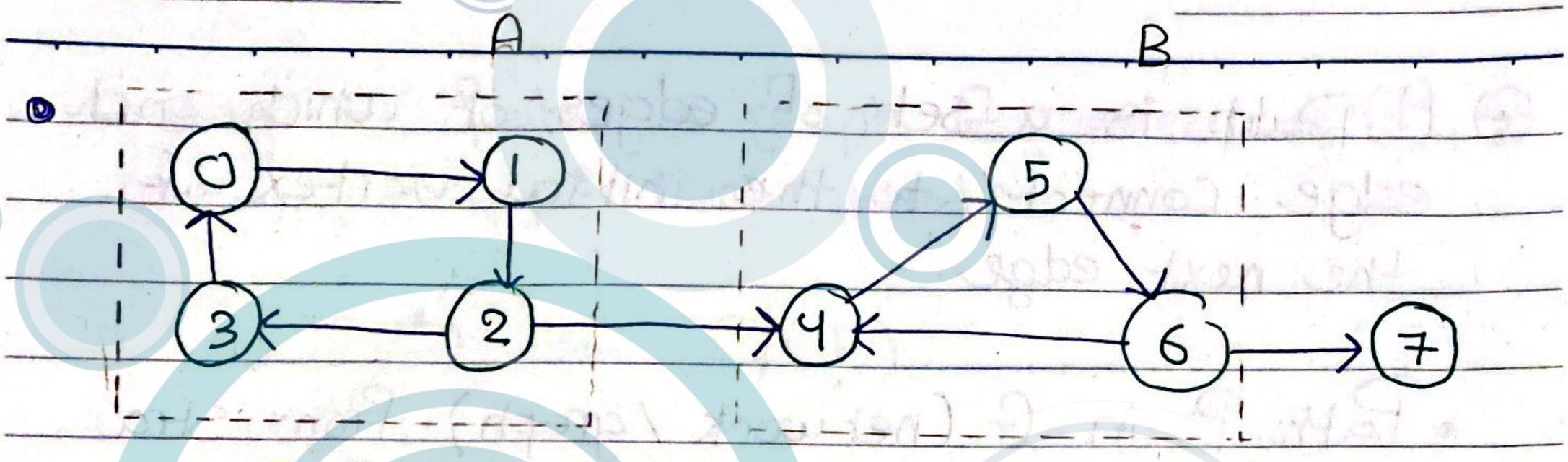
→ not strongly Connected.

↳ لو أخذنا هاي الجزئية لحالها فَهي تعتبر Strongly Connected.

* Strongly Connected" is associated with Direct graph, there is a route between each pair of nodes.

* Strongly Connected = Direct + Connected

Scanned with CamScanner

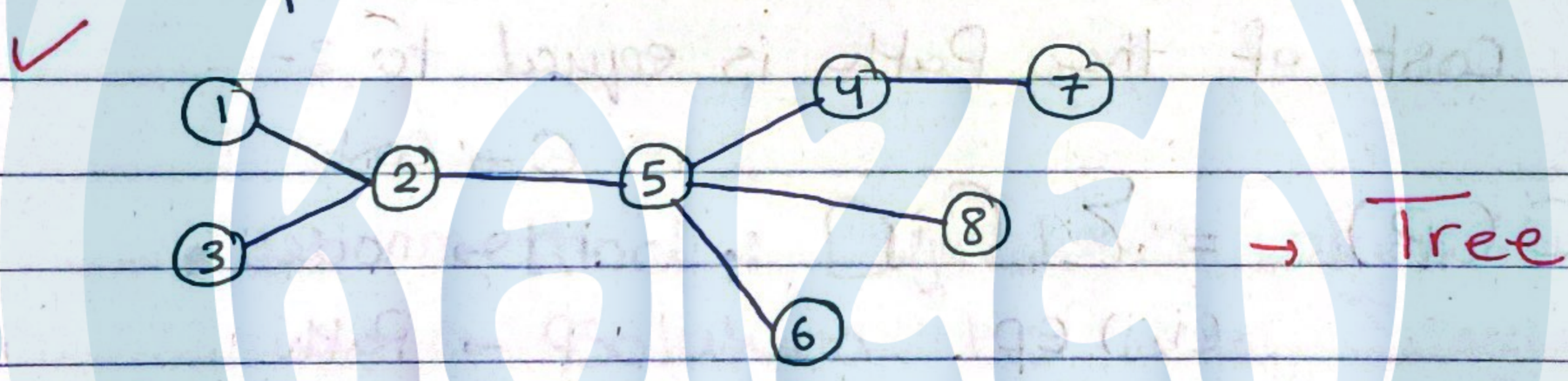


↳ Part A is Strongly Connected
↳ Part B is Strongly Connected
↳ the whole graph is NOT.

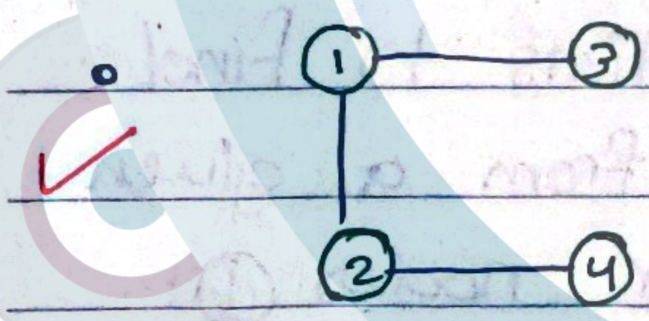
⊛ Tree :

→ a tree is a Connected graph that contains no Cycle.

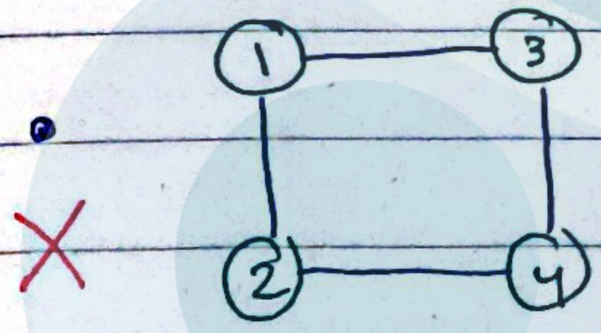
• example



→ Tree



→ Tree



→ Not a tree
Since it contains a Cycle #

Scanned with CamScanner

* A Path is a set of edges of which each edge connects to the initial vertex of the next edge.

- Path P in G (network / graph) from node (S) to Destination node (t) is: an ordered sequences of arcs that Connect S to t.

- $P = \{(S, i), (i, j), \dots, (k, t)\}$

Starting nodes S — destination node t

- the cost of the Path is equal to :-

$$\Rightarrow C(P) = \sum_{(i,j) \in P} C_{ij}$$

C → Cost
i, j → nodes
P → Path

- min cost Path questions : Aims to find the path with min cost from a given Source node (S) to all other nodes (j).

Scanned with CamScanner

- Among all Path in graph G connecting (s) to (t), choose p^* such that:

$$C(p^*) \leq C(p)$$

min Cost Path.

↳ Cost of any Path Connecting (s) to (t)

⊛ Single Source Shortest Path Problem :-

- aim to find shortest Path from a Single Source : node to all other nodes in the graph.

- 1st method : Dijkstra's algorithm
  → a solution to the Single Source - shortest path problem in graph theory.

⊕ example with steps

Scanned with CamScanner

Dijkstra's algorithm

- Requires that $c_{ij} \geq 0$

Initialization:

$$v(j) \leftarrow \begin{cases} 0 & j = s \\ +\infty & \text{otherwise} \end{cases}$$

$$pred(j) \leftarrow \begin{cases} 0 & j = s \\ -1 & \text{otherwise} \end{cases}$$

$PERM = ; \overline{PERM} = N$

Iterations:

while $\overline{PERM} \neq \emptyset$:

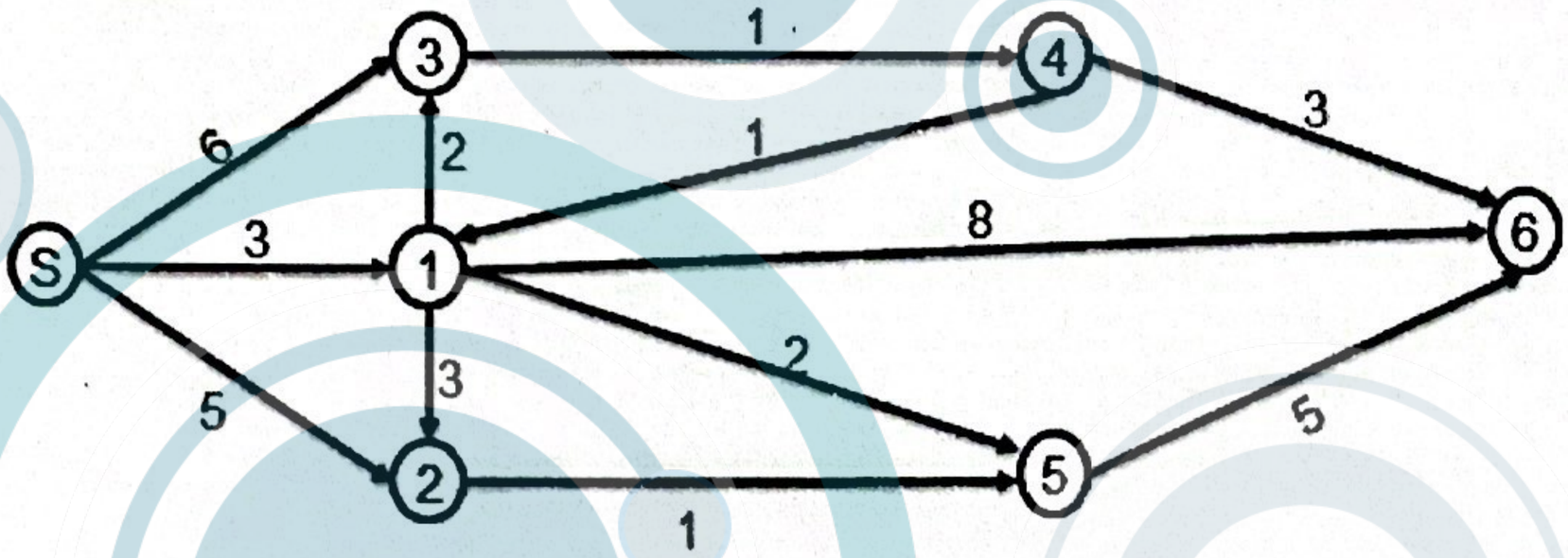
1. Let $p \in \overline{PERM}$ be node for which $v(p) = \min\{v(j), j \in \overline{PERM}\}$
2. $PERM \leftarrow PERM \cup \{p\}$
3. $\overline{PERM} \leftarrow \overline{PERM} \setminus \{p\}$
4. For every arc $(p, j) \in A$, $j \in \overline{PERM}$ leading from node p:
   (a) if $v(p) + c_{pj} < v(j)$, then:
   - $v(j) \leftarrow v(p) + c_{pj}$
   - $pred(j) \leftarrow p$

Dijkstra's Algorithm Steps :-

1. Assign to every node a distance value. Set it to zero for our initial node and to infinity for all other nodes.
2. Mark all nodes as unvisited. Set initial node as current.
3. For current node, consider all its unvisited neighbors and calculate their tentative distance (from the initial node). For example, if current node (A) has distance of 6, and an edge connecting it with another node (B) is 2, the distance to B through A will be 6+2=8. If this distance is less than the previously recorded distance (infinity in the beginning, zero for the initial node), overwrite the distance.
4. When we are done considering all neighbors of the current node, mark it as visited. A visited node will not be checked ever again; its distance recorded now is final and minimal.
5. If all nodes have been visited, finish. Otherwise, set the unvisited node with the smallest distance (from the initial node) as the next "current node" and continue from step 3.

Scanned with CamScanner

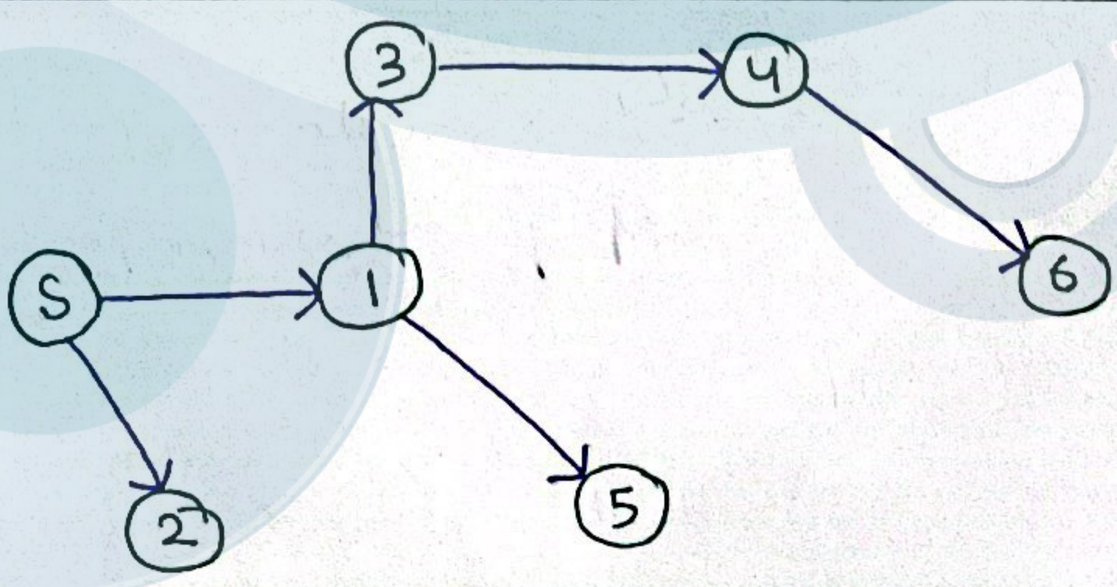
- **Example:**
  one



↗ initial (starting) node

Iteration	s	1	2	3	4	5	6	P	PERM	PERM
0	0(0)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	—	∅	N
1	0(0)*	3(s)	5(s)	6(s)	∞(-1)	∞(-1)	∞(-1)	S	S	N/{S}
2	—	3(s)*	5(s)	5(1)	∞(-1)	5(1)	11(1)	1	S,1	N/{S,1}
3	—	—	5(s)*	5(1)	∞(-1)	5(1)	11(1)	2	S,1,2	N/{S,1,2}
4	—	—	—	5(1)*	6(3)	5(1)	11(1)	3	S,1,2,3	N/{S,1,2,3}
5	—	—	—	—	6(3)	5(1)*	10(5)	5	S,1,2,3,5	N/{S,1,2,3,5}
6	—	—	—	—	6(3)*	—	9(4)	4	S,1,2,3,5,4	N/{S,1,2,3,5,4}
7	—	—	—	—	—	—	9(4)*	6	N	∅

#



min cost path tree from the
⇒ Single Source (5)
to all other nodes.

Scanned with CamScanner

Thursday
3/11

Condt. → min cost Path From Source node (S) to node (4) :-

P = [(S,1),(1,3),(3,4)] S-1-3-4 X

* more about Dijkstra's algorithm :-
  * Conditions
    1. Used only in Single Source shortest Path Problem

    2. works on both direct and indirect graphs.

    3. all edges must have non-negative Cost/weight.

  * Outputs
    1. $v^*(j) \quad \forall \quad j \in N$

min Cost ← From Source to node j ; ∀ For all ; j ∈ N → for all j in the set of nodes arcs (N)

    2. $Pred^*(j)$: the set of N Preceding j on the optimal Path From (S) to (j)

Predecessor ←

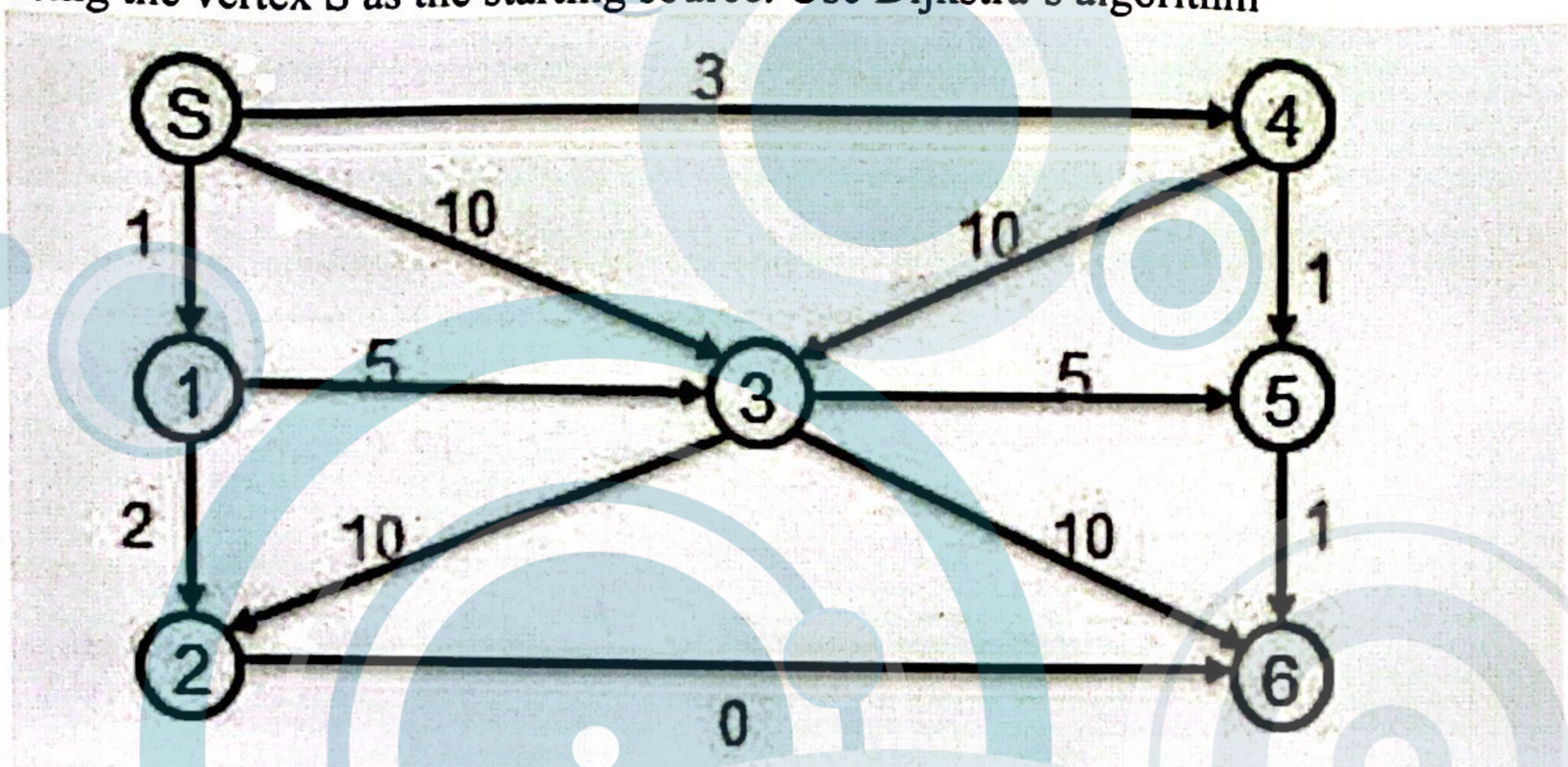
    3. Min Cost Path tree

* Dijkstra determine min. Cost Path tree given a Specific starting node

(+) example 2

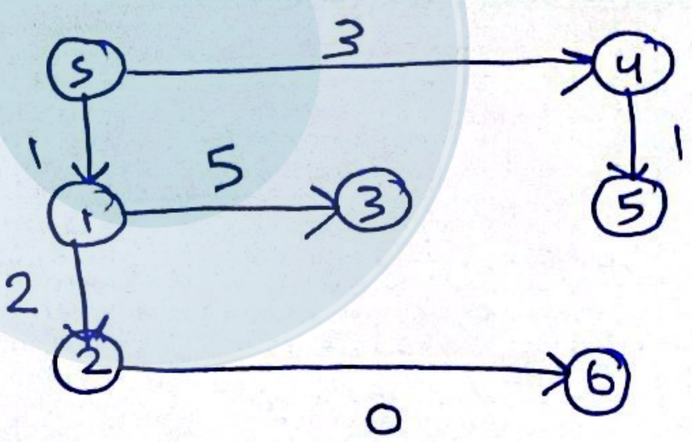
Scanned with CamScanner

Example # 2: Given the graph in the Figure shown below, find the shortest paths to all nodes using the vertex S as the starting source. Use Dijkstra's algorithm



Iteration	S	1	2	3	4	5	6	P	PERM	$\overline{PERM}$
0	0(0)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	—	∅	N
1	(0(0)*	1(S)	∞(-1)	10(S)	3(S)	∞(-1)	∞(-1)	S	S	N/{S}
2	—	1(S)*	3(1)	6(1)	3(S)	∞(-1)	∞(-1)	1	1,S	N/{S,1}
3	—	—	3(1)*	6(1)	3(S)	∞(-1)	3(2)	2	1,2,S	N/{S,1,2}
4	—	—	—	6(1)	3(S)*	4(4)	3(2)	4	1,2,4,S	N/{1,2,4,S}
5	—	—	—	6(1)	—	4(4)	3(2)*	6	1,2,4,6,S	3,5
6	—	—	—	6(1)	—	4(4)*	—	5	1,2,4,6,5,S	3
7	—	—	—	6(1)*	—	—	—	3	N	∅

(*) Min Cost Path tree



Scanned with CamScanner

Sunday
6 / 11

* Recall that min. Cost Path tree is one of Dijkstra's algorithm outputs

- this type of tree is called Rooted-out
  لـ يلي بنتج عن الـ Dijkstra
- because it gives the shortest path from one Single Source (S) to all other nodes

* Rooted-out tree [given the Single Source (S) as a starting node] :-

→ Is a tree on G such that there is a path exist from (S) to all other nodes.

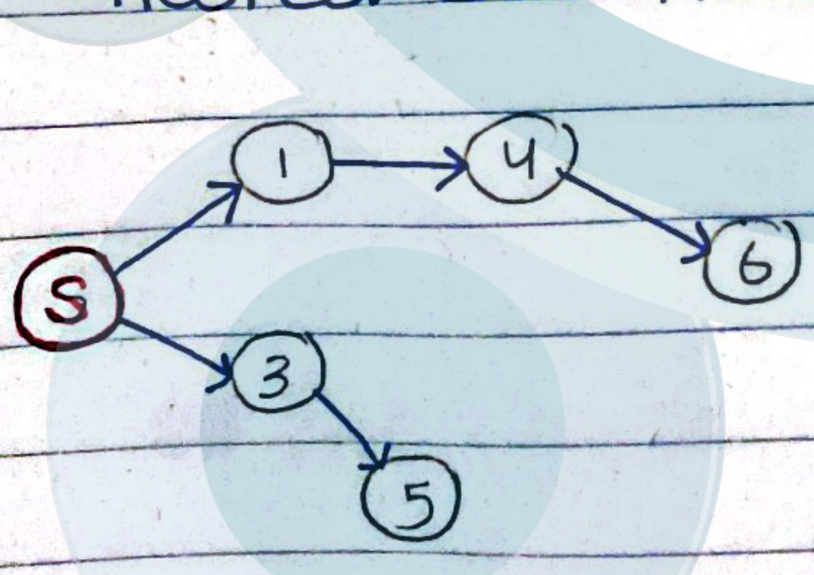
* Rooted-in tree [given one ending node t]:

→ is a tree on G such that there is a Path exist from any node to node (t).

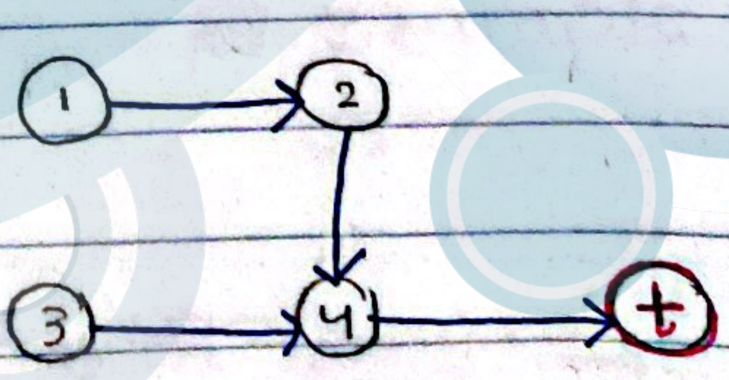
لـ يعني لازم من كل الـ nodes يكون في Distination الـ Path

- Examples :-

- Rooted-out tree



- Rooted-in tree



Scanned with CamScanner

⊛ Tree.

- Subnetwork of G ⇒ T

- $T = (N, A_T)$
  - N: all set of nodes
  - A_T: subset of arcs
    - $A_T \subseteq A$
      - ↳ subset.

- # of ars = # of nodes - 1
  - $A_T = N - 1$

- For undirect tree, a Path exists between all pairs of nodes in N
  - « as long as we ignore the direction »

- No cycle (or loop):

Kaizen TEAM

Scanned with CamScanner

⊛ Bellman - Ford algorithm «Slides»

- used in direct weighted arcs.
- no requirement for non-negative arc cost.

- PaPes modification

→ if $j \notin$ List (if j doesn't belong to the Current list)

↳ we have 2 cases :-

① If j has never been in the list

List ← List + [j]

القديمة الجديدة

(we add j to the end of the list)

② If j has been in the list

List ← [j] + List

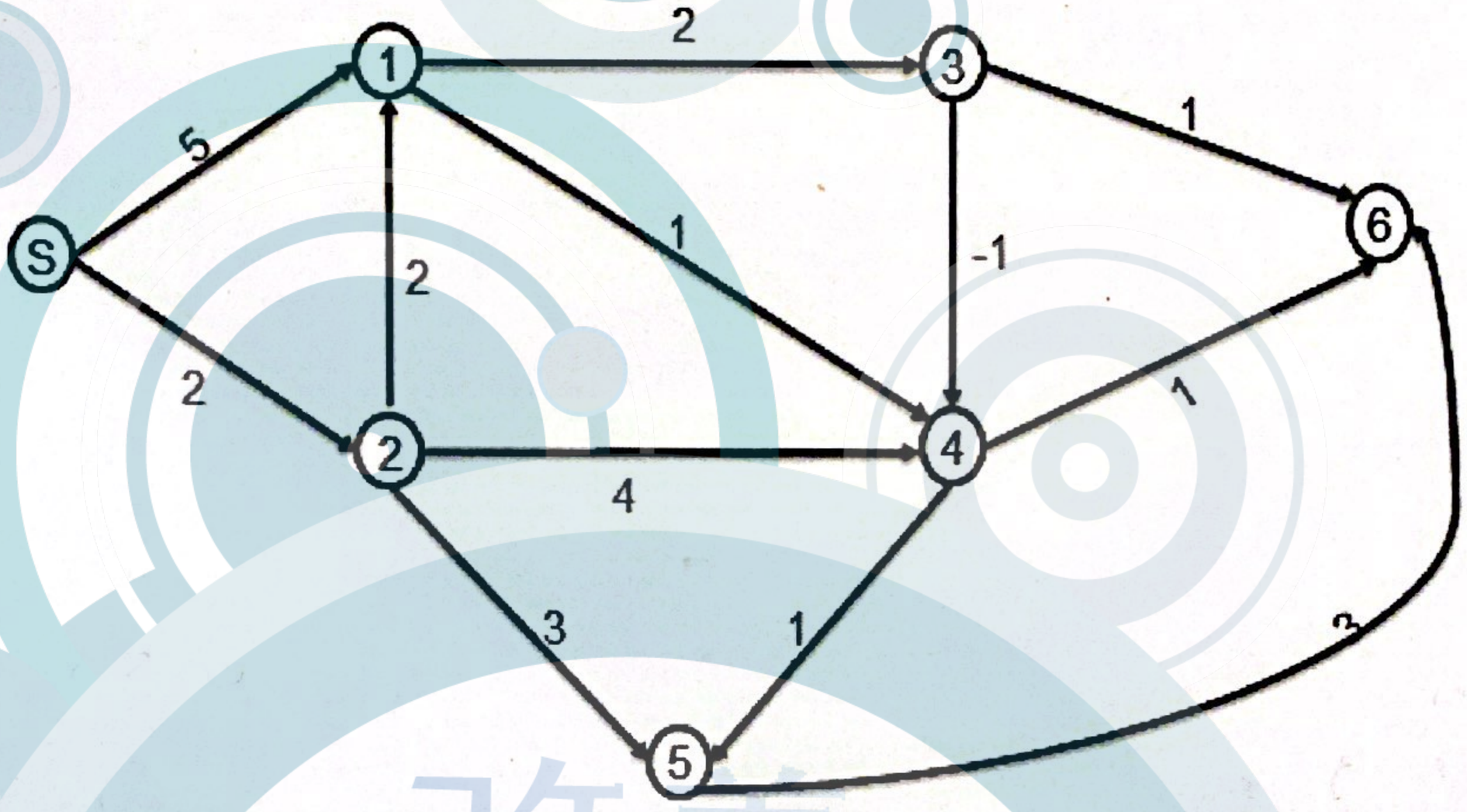
(we add j at the beginning of the list)

⊕ example

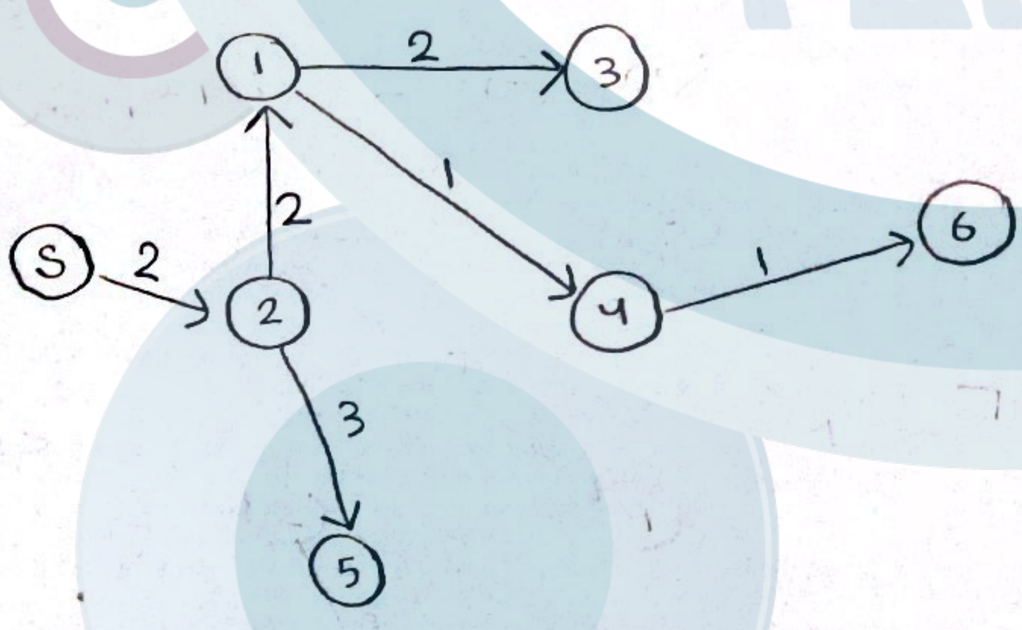
⊕ Bellman & Dijkstra's Pros & Cons →

Scanned with CamScanner

Example:
Given the graph in the Figure shown below, find the shortest paths to all nodes using the vertex s as the starting source. Use Bellman-Ford algorithm.



Iteration	s	1	2	3	4	5	6	P	LIST
0	0(0)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	—	S
1	0(0)*	5(S)	2(S)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	S	1, 2
2	0(0)	5(S)*	2(S)	7(1)	6(1)	∞(-1)	∞(-1)	1	2, 3, 4
3	0(0)	4(2)	2(S)*	7(1)	6(1)	5(2)	∞(-1)	2	1, 3, 4, 5
4	0(0)	4(2)*	2(S)	6(1)	5(1)	5(2)	∞(-1)	1	3, 4, 5
5	0(0)	4(2)	2(S)	6(1)*	5(1)	5(2)	7(3)	3	4, 5, 6.
6	0(0)	4(2)	2(S)	6(1)	5(1)*	5(2)	6(4)	4	5, 6
7	0(0)	4(2)	2(S)	6(1)	5(1)	5(2)*	6(4)	5	6
8	0(0)	4(2)	2(S)	6(1)	5(1)	5(2)	6(4)*	6	∅



• what is the Cost of the Path ⑤ to ⑥ ?

P : {(S,2), (2,1), (1,4), (4,6)}

C(P) = 6 #

Scanned with CamScanner

Dijkstra's algorithm

- **Pros** الـ Dijkstra أحسن من الـ Bell-man لأنه الـ node يلاقي بوصلها ما برجعلها

1. Achieves fastest running times (theoretically and practically) for dense networks. Typical geographic networks are not dense, though.
2. Can be truncated. Each iteration, one node p is given a permanent, optimal minimum cost label and predecessor. Thus, if you are interested only in paths to some subset of the nodes $T \subset N$, you can stop the algorithm once $PERM \supseteq T$.

↳ Subset or equal to

- **Cons** الـ bellman أحسن

1. Alternative methods are faster (practically) on sparse networks.
2. Efficient implementations require programming a priority queue (heap).

↳ in Bill-man, the Priority is for the 1st node in the list.
↳ where in Dijkstra there is no Specific Priority.

KAIZEN TEAM
Scanned with CamScanner

Bellman-Ford algorithm

Bellman-Ford algorithm

Sometimes, *label-correcting* algorithm. No requirement of non-negative arc costs. Often more efficient in practice for sparse transportation problems.

***Initialization*:**

$$v(j) \leftarrow \begin{cases} 0 & j = s \\ +\infty & \text{otherwise} \end{cases}$$

$$pred(j) \leftarrow \begin{cases} 0 & j = s \\ -1 & \text{otherwise} \end{cases}$$

$LIST = \{s\}$

***Iterations*:**

while $LIST \neq \emptyset$:

1. **Remove node p from top of $LIST$**
2. **For every arc (p, j) leading from node p:**
   (a) **if $v(p) + c_{pj} < v(j)$, then:**
   - $v(j) \leftarrow v(p) + c_{pj}$
   - $pred(j) \leftarrow p$
   - if $j \notin LIST$, $LIST \leftarrow LIST + \{j\}$

Bellman-Ford algorithm

- Pros
  1. Achieves fastest running times (practically) for sparse networks, especially when implemented with Pape's Modification.
  2. Very simple to implement.
- Cons
  1. Labels and predecessors not optimal until full completion of algorithm.
  2. Running time may be slow on dense networks due to frequent label updates.

Scanned with CamScanner

Tuesday
8/11

⊛ Multi-label algorithm.

- In some Problems we want to find path that balance multiple objectives.

- for example we might want to find the path with min. distance that doesn't exceed a time limit. → (multi-label)

- or we might want to find the path with min distance that doesn't exceed a time limit (time constraint)

- in multi-label we have 2 measures of cost: Distance & time.

- nodes are given labels with 2 components

$$(d_k , t_k)^{i \rightsquigarrow \text{node}}_{k \rightsquigarrow \text{label \#}}$$

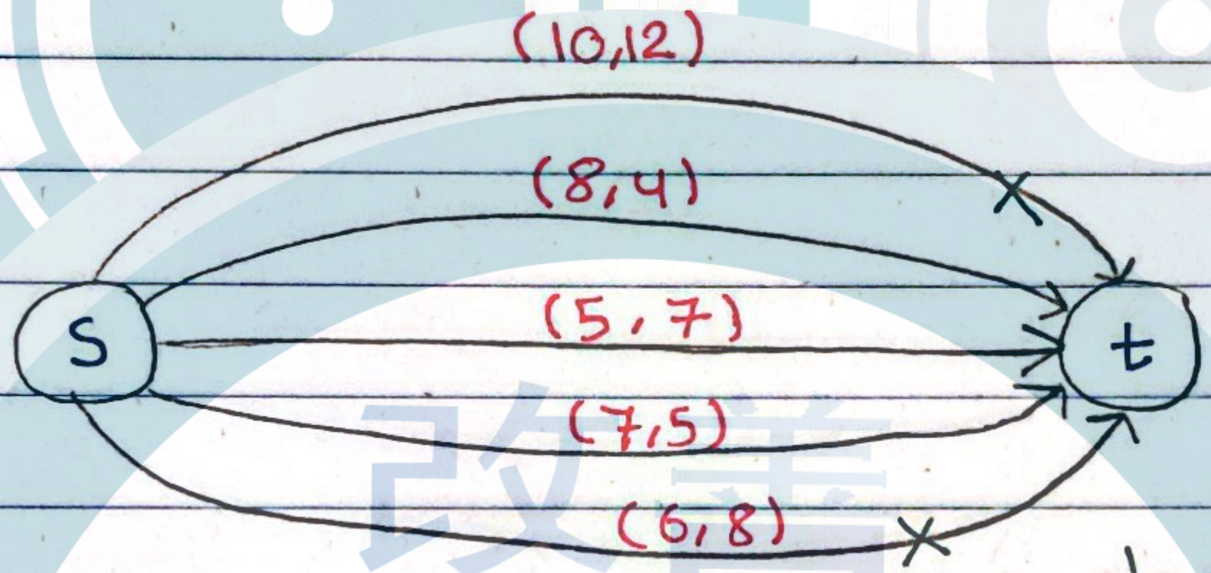
where : d_k : 1st cost component (tot. Distance) required to reach node (i) from the source node (s).

t_k : 2nd cost component (tot. time) required to reach node (i) along some path from node (s)

Scanned with CamScanner

- Example
  - node may have multiple label resulting from different paths.

  - one path to a node may have longer distance but shorter time than other paths.



↳ dominated by (5,7).

↳ we will never choose (10,12) or (6,8) label / path, because (5,7) dominate both of them

↳ however we can't easily choose between (8,4) and (5,7).

Scanned with CamScanner

Sunday
13/11

Multi-label algorithm

Initialization:

Initially, the origin node s is given label $(0,0)_1^s$. (The subscript indicates the reference number $k = 1$ for this label, and the superscript indicates that this label exists at node s) All other nodes receive no labels. The predecessor for this label, $pred(1)$ is 0.

$LIST = \{(0,0)_1^s\}$ $r = 2$

Iterations:

while $LIST \neq \emptyset$:

1. Remove label k from the beginning of the list. Suppose label k has components $(d_k, t_k)_k^p$, so that it exists at node p
2. For every arc (p, j) leading from node p:
   (a) Create a new label r at node j where $(d_r, t_r) = (d_k + d_{pj}, t_k + t_{pj})_r^j$
   (b) Define $pred(r) = k$
   (c) Add label r to the end of the $LIST$
   (d) Compare label r to each existing label (d_q, t_q) currently at node j:
      i. If $d_r \geq d_q$ -and- $t_r \geq t_q$, then label r is dominated by (d_q, t_q). Discard the new label, and remove it from $LIST$.
      ii. Else If $d_q \geq d_r$ -and- $t_q \geq t_r$, then label r dominates label q. Discard the label (d_q, t_q), and if necessary remove it from $LIST$.
      iii. Else retain both labels

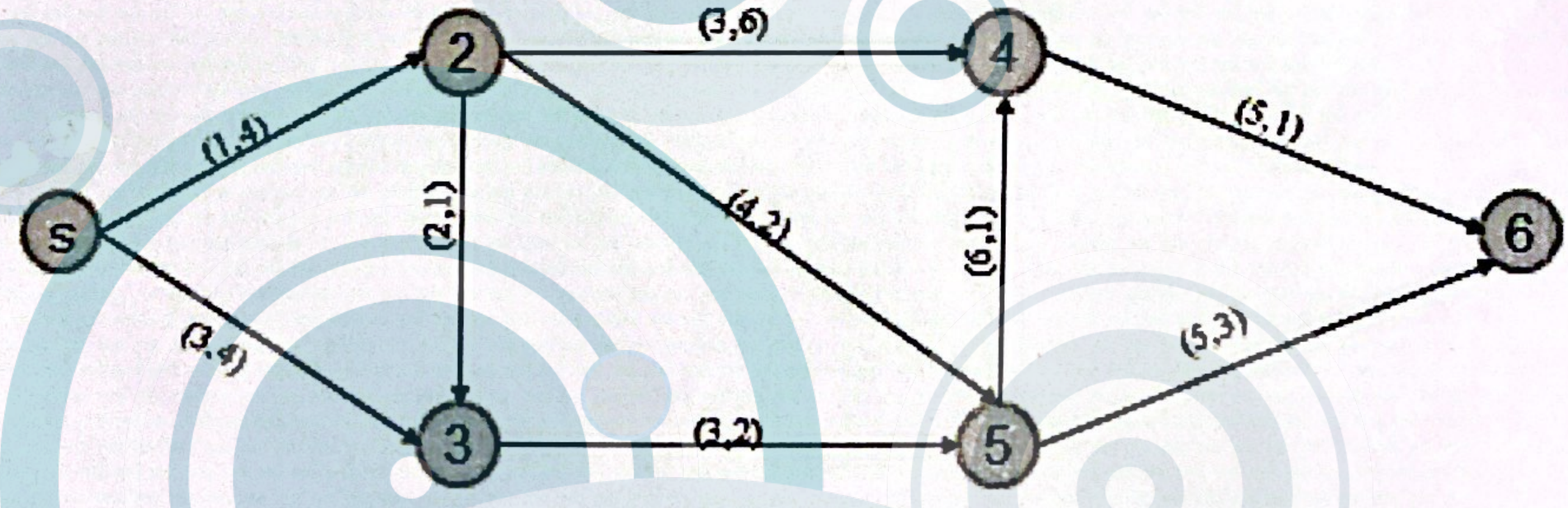
Stopping criterion:

When no labels remain in $LIST$, this indicates that no remaining label corrections can be found, and all of the paths have been found.

KAIZEN TEAM

Scanned with CamScanner

Example: Given the graph in the Figure shown below, find the shortest paths to all nodes using the vertex s as the starting source. Use **Multi-label algorithm**



	S	2	3	4	5	6	LIST
Labels	$(0,0)^S_1$ Pred(1)=0	$(1,4)^2_2$ Pred(2)=1	$(3,4)^3_3$ Pred(3)=1 $(3,5)^3_4$ X Pred(4)=2	$(4,10)^4_5$ Pred(5)=2 $(11,7)^4_{10}$ Pred(10)=6	$(5,6)^5_6$ Pred(6)=2 $(6,6)^5_7$ X Pred(7)=3	$(9,11)^6_8$ Pred(8)=5 $(10,9)^6_9$ Pred(9)=6 $(16,8)^6_{11}$ Pred(11)=10	$(0,0)^S_1$ X $(1,4)^2_2$ X $(3,4)^3_3$ X $(3,5)^3_4$ X $(4,10)^4_5$ X $(5,6)^5_6$ X $(6,6)^5_7$ X $(9,11)^6_8$ X $(10,9)^6_9$ X $(6,6)^5_7$ X $(11,7)^4_{10}$ X $(16,8)^6_{11}$ X $\emptyset$

KAIZEN TEAM

Scanned with CamScanner

Sunday
13/11

Multi-label algorithm

Initialization:

Initially, the origin node s is given label $(0,0)_1^s$. (The subscript indicates the reference number $k = 1$ for this label, and the superscript indicates that this label exists at node s) All other nodes receive no labels. The predecessor for this label, $pred(1)$ is 0.

$LIST = \{(0,0)_1^s\}$ $r = 2$

Iterations:

while $LIST \neq \emptyset$:

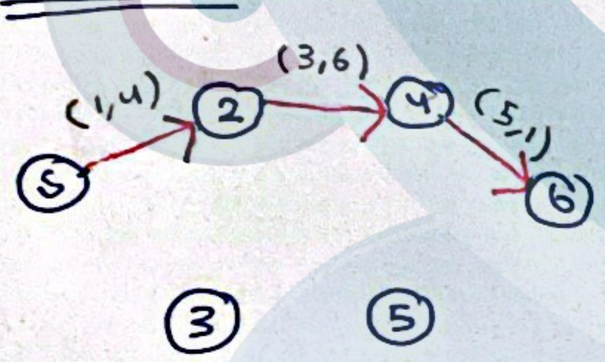
1. Remove label k from the beginning of the list. Suppose label k has components $(d_k, t_k)_k^p$, so that it exists at node p
2. For every arc (p, j) leading from node p:
   (a) Create a new label r at node j where $(d_r, t_r) = (d_k + d_{pj}, t_k + t_{pj})_r^j$
   (b) Define $pred(r) = k$
   (c) Add label r to the end of the $LIST$
   (d) Compare label r to each existing label (d_q, t_q) currently at node j:
      i. If $d_r \geq d_q$ -and- $t_r \geq t_q$, then label r is dominated by (d_q, t_q). Discard the new label, and remove it from $LIST$.
      ii. Else If $d_q \geq d_r$ -and- $t_q \geq t_r$, then label r dominates label q. Discard the label (d_q, t_q), and if necessary remove it from $LIST$.
      iii. Else retain both labels

Stopping criterion:

When no labels remain in $LIST$, this indicates that no remaining label corrections can be found, and all of the paths have been found.

Sol. from (S) → (6) ⟹ 3 Paths. (min. Cost Path from (S) → (6))

Path 1

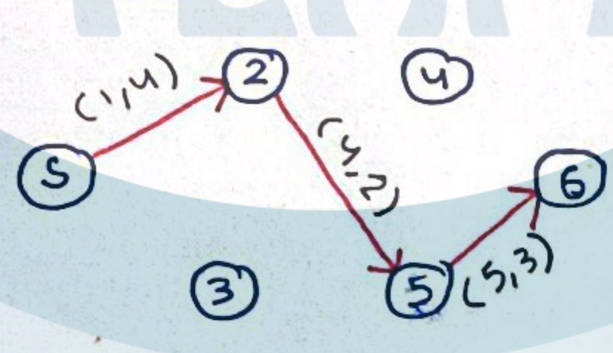


Same as Path $(9,11)_8^6$ So Correct

1 + 3 + 5 = 9

4 + 6 + 1 = 11

Path 2

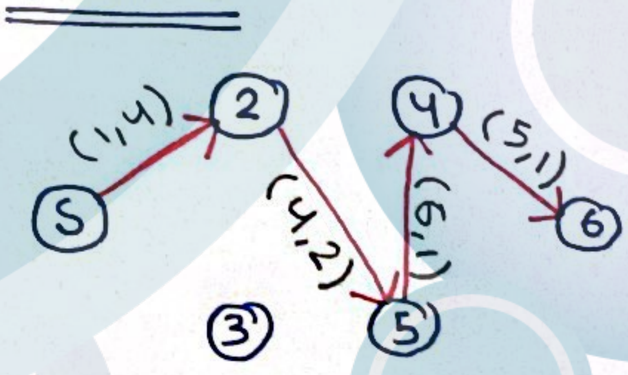


Same as Path $(10,9)_9^6$

1 + 4 + 5 = 10

3 + 2 + 4 = 9

Path 3



Same as Path $(16,8)_{11}^6$

5 + 6 + 4 + 1 = 16

1 + 1 + 2 + 4 = 8

Scanned with CamScanner

Tuesday
15/11

Recall :

- Tree : Subnetwork → $\overline{T} = (N, A_T)$

$|A_T| = |N| - 1$

⊛ Minimum Spanning Tree (MST) or (T^*):

- In a network where each arc (i,j) has C_{ij} : the MST is a tree such that the tot. cost of arcs in the tree is minimized.

- mathematically :-

$$T^* = \underset{T \in G}{\text{arg min}} \left[\sum_{(i,j) \in T} C_{ij} \right]$$

→ among all of the tree in our graph G, it's the one that minimizes the cost of arc tree (tot. cost).

⊛ There are 2 algorithms to Find the MST :-

① Kruskal's algorithm

② Prim's algorithm

" Optimization problem. "

Scanned with CamScanner

1 ⊛ Kruskal's algorithm :-

- initialization : Tree = ∅
- iterations :-

① Sort arcs in non-decreasing order of Cij into LIST.

② while |TREE| < n-1 :-

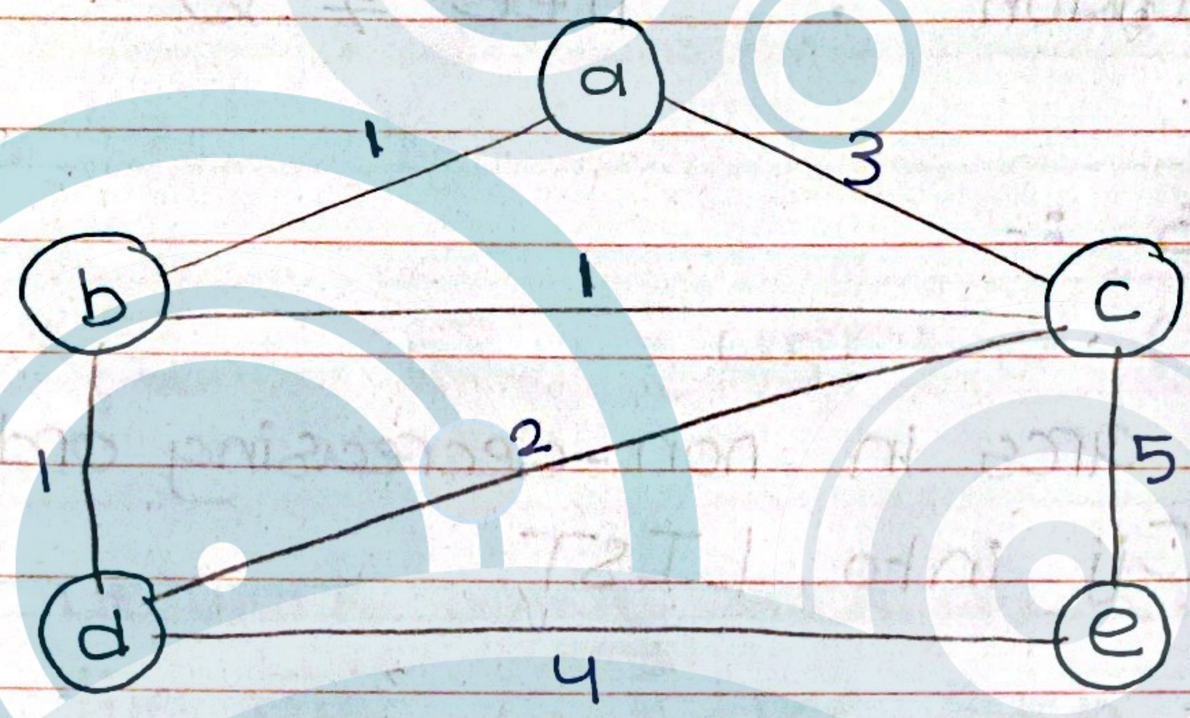
a) Remove arc (i,j) from the top of the LIST.

b) add arc (i,j) to the tree, if doing so does not create a cycle

③ T* ← Tree #

Scanned with CamScanner

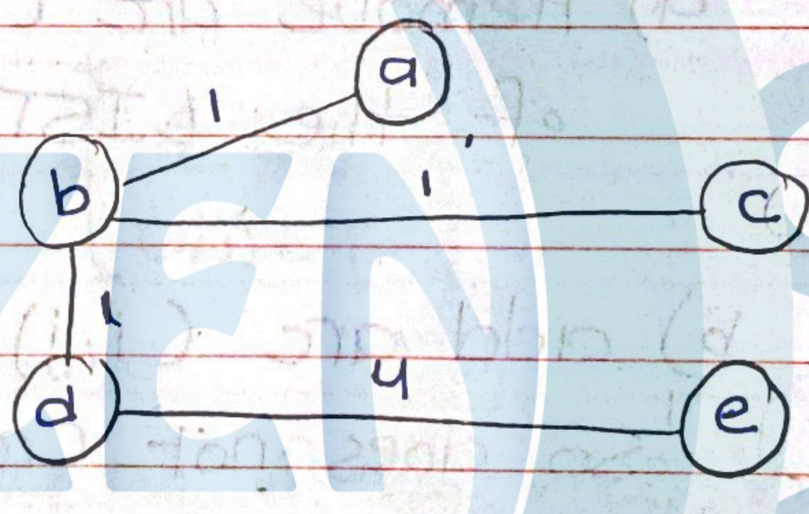
⊛ Example



LIST :

(a,b) 1
(b,d) 1
(b,c) 1
(d,c) 2
(a,c) 3
(d,e) 4
(c,e) 5 → stop.

⊛ MST :



tot. Cost = 7 #

لما iteration بوقف ال ← ممكن يكون في أكتر من MST ←
# of arcs = n-1
بس بالنهاية كلهم بيطلعو
نفس ال tot. Cost

Scanned with CamScanner

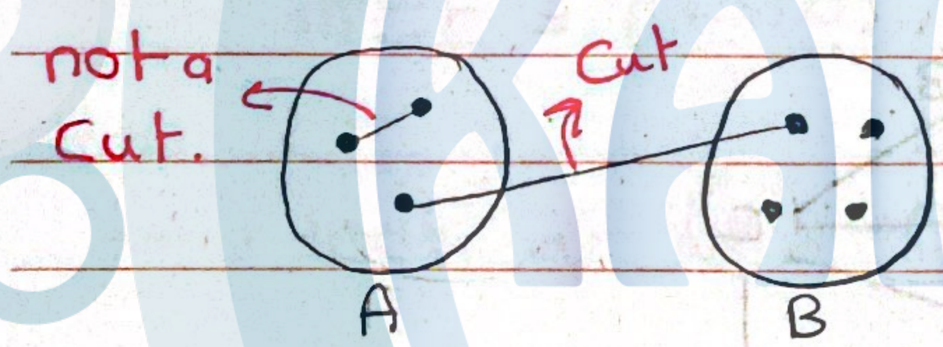
2 ⊛ Prim's algorithm

- Cuts in the network « مبدأها »

- Suppose the Nodes in N are Partitioned in two independent Subsets, A & B Such that :

$$A \cup B = N$$
$$A \cap B = \emptyset$$

- then the Cutset (A,B) is given by all arcs (i,j) such that $i \in A$ and $j \in B$ or $i \in B$, $j \in A$.



- Prim's algorithm

: initialization : $\overline{TREE} = \emptyset$

$A = \{1\}$, $B = N / \{1\}$

Scanned with CamScanner

• iteration : while | TREE | < n-1

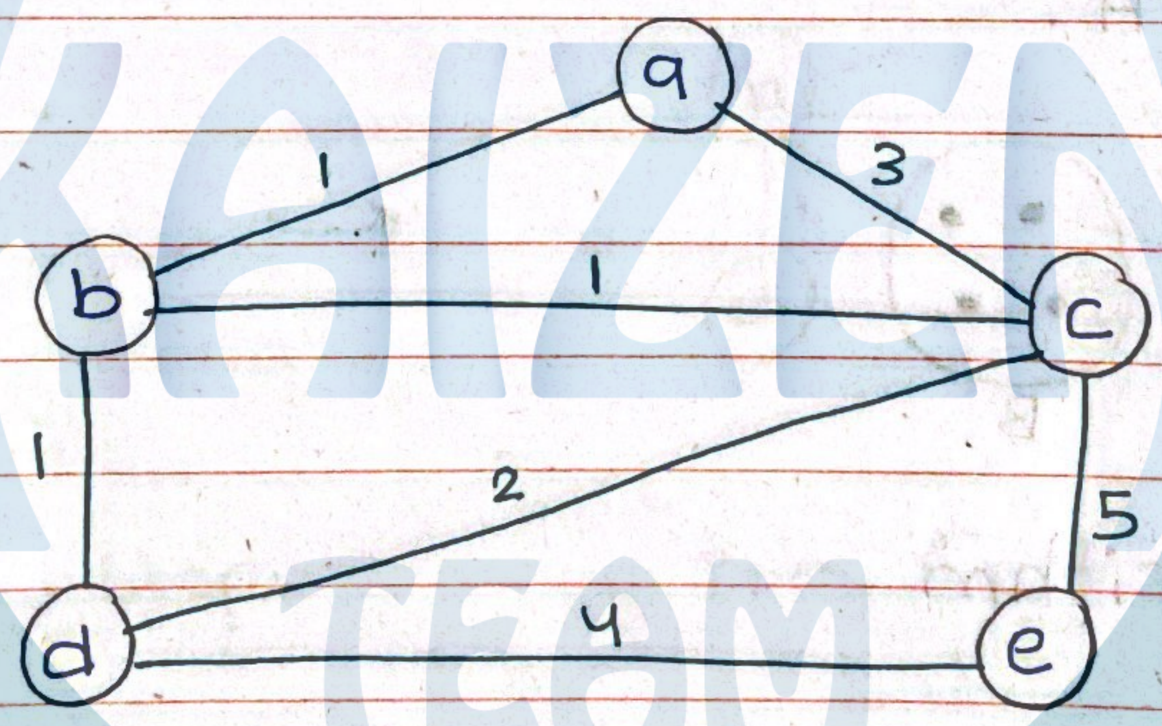
① Find arc (i,j) in (A,B) with min C_{ij} ← cost.

② Add this arc to the TREE

③ Remove j from B and add it to A.

at the end , $T^* \leftarrow$ TREE #

@example :-



عادي ممكن نختار أي node تانية

→ A = [c]

B = [a, b, d, e]

TREE = ∅

Scanned with CamScanner

* A = {c, b}
  B = {a, d, e}

TREE = {(c, b)}

→ now look at arcs ac, ab, ce, bd, cd and choose one.
                   3   1   5   1   2
A و B لـ يلي جزء منهم بـ
التاني بـ B

→ add arc ab ⇝ لأنّه أقل واحد

So * A = {a, b, c}
     B = {d, e}

TREE = {(c, b), (a, b)}

* ما بضيف ال arc يلي
* Cycle بعمل

→ now look at arcs ce, cd, bd
                   5   2   1

* A = {a, b, c, d}
  B = {e}

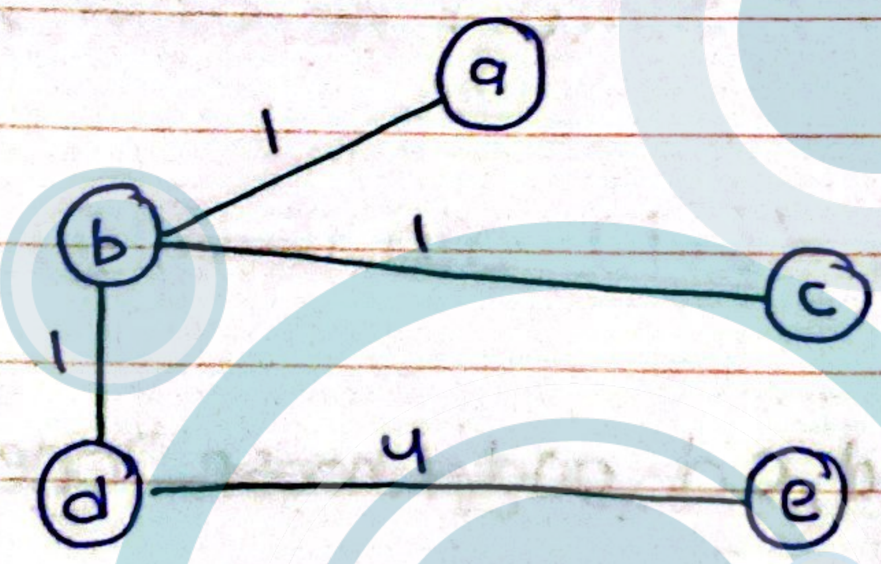
TREE = {(c, b), (a, b), (b, d)}

* A = {a, b, c, d, e}
  B = ∅ → وقف بون

TREE = {(c, b), (a, b), (b, d), (d, e)} cost = 7 #

Scanned with CamScanner

MST :-



tot. Cost = 7 #

مادة المس #

KAIZEN TEAM

Scanned with CamScanner

Thursday
17/11

⊛ Short-haul transportation : Single vechile routing.

- Setting : we want to visit number of locations, starting from and returning to a depot with minimum tot. travel cost (time, distance, cost).

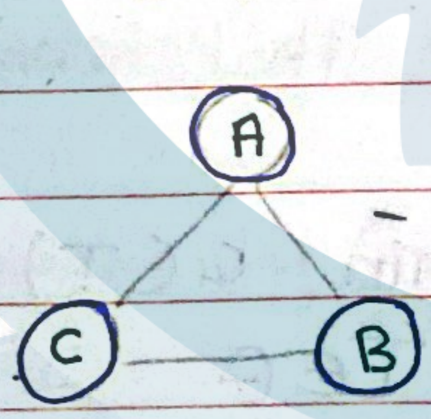
- Question: in what order should the locations be visited ?

- Answer : Sequence of locations
  ↳ this sequence is called "tour".

- Assumption :
  - we start with a complete, undirected network

$$G = (N, A) \rightarrow \#\text{ of arcs} = \frac{n * (n-1)}{2}$$

- The graph satisfies triangle inequality

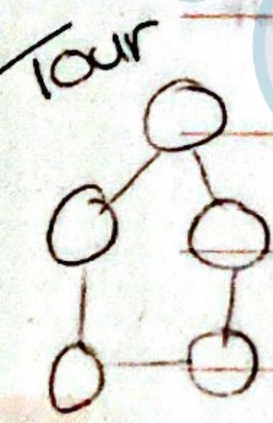


$AC + CB \geq AB$

Scanned with CamScanner

- TSP → Travel Saleman Problem

- TSP tour is a visitation sequence of all Nodes in N.



$T = [V_1, V_2, V_3, \dots, V_n, V_1]$, $V_i \in N$

لـ ع ارجع لنقطة البداية

- Subtour : is a sequence of nodes with Common start and end, such that not all nodes in N are visited.



$= \{V_1, V_2, \dots, V_q, V_1\}$, $1 < q < N$

ك برضو ع أرجع لنقطة لبداية بس ما بكون زرت كل ال nodes

- Total Cost :

$$C(T) = \sum_{i=1}^{n-1} C_{vi, vi+1} + C_{vn, V_1}$$

↳ Simply : Sum. For the costs of arcs that belong to the tour.

- Optimal tour = TSP^* = Argmin $C(T)$

TEG

Scanned with CamScanner

- TSP → Optimization Problem → tour with min. cost.

  ↳ IP formulation (integer Problem formulation)

[1] Decision variable :

$$X_{ij} = \begin{cases} 1 & : \text{if arc } (i,j) \text{ belong to the tour} \\ 0 & : \text{otherwise} \end{cases}$$

arc bet (i) & (j)

[2] objective : Minimize $\sum C_{ij} \cdot X_{ij}$

[3] Constraint (sub. to) : → node لكل

- $\sum_{i \in N/[j]} X_{ij} = 1 \quad \forall j \in N$ → unique arc entering node j

- $\sum_{j \in N/[i]} X_{ij} \quad \forall \in N$ → unique arc leaving node i

Simply ← لازم كل node يدخل عليها arc و يطلع منها arc

← يعني دخ أوصل ال location و أغادره ، مارح أزوره مرتين.

- Connectivity Constraints / Subtour elimination Constraints.

  ↳ عشان يكون ال tour كامل و مو Subtour

- $\sum_{i \in S} \sum_{j \notin S} X_{ij} \geq 1 \quad , \forall S \in N \quad |S| \geq 2$

  ↳ S → Size of the Subset (# of nodes)

  ↳ ∀ = for all

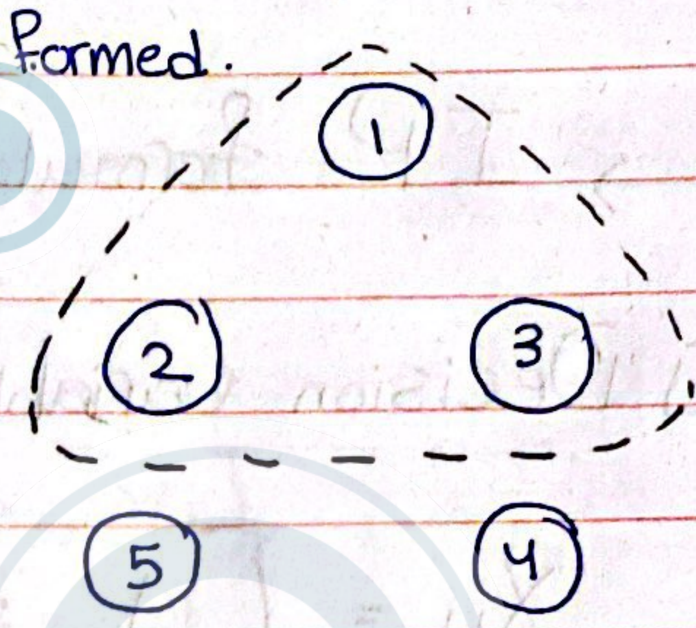
  ↳ N → Set of all nodes

OR

- $\sum_{i \in S} \sum_{j \in S} X_{ij} \leq |S| - 1 \quad , \forall S \in N$

Scanned with CamScanner

- we cannot have more than 2 arcs in the subset, else a subtour will be formed. $S = [1,2,3]$, $|S| = 3$



- $X_{12} + X_{13} + X_{23} \leq 2$
  ↳ 3-1
- $X_{ij} \in \{0,1\} \; \forall (i,j) \in A$
  ↳ set of arcs

↳ Because of subtour elemination Constraints we end-up with exponential # of Constraints.

↳ Conclusion : TPs is hard to solve for medium-large questions.

↳ we need to find smart heuristic to solve the problem in reasonable amount of time

KAIZEN TEAM

Scanned with CamScanner

Sunday
20/11

⊛ Travel Saleman Problem :

- Goal ① : identify efficient heuristic that give good (but not necessary optimal Solution)

- Goal ② : find heuristic where worse Case quality of a Solution can be guaranteed to be within Some Constant α of the true optimal Solution.

C^* : true optimal Case (unknown)

αC^* : worst heuristic Solution Cost

$$\rightarrow C^* \leqslant C_{heuristic} \leqslant \underbrace{\alpha C^*}_{\hookrightarrow \text{worst Case}}$$

⊛ Lower Bound for hard optimization Problem

- the LB is used to evaluate the quality of the heuristic Solution.

- Assume minimization Problem :

Suppose T^H : is a feasable tour identified by heuristic H.

→

Scanned with CamScanner

→ $C(T^H) \geq C(TSP)^*$ Optimal

lower bound

→ $\underline{\subseteq}(TSP^*)^* \leq C(TSP)^*$

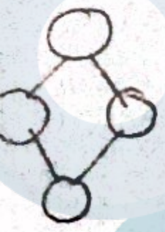
lower bound

a number ↵

→ we can't make a Comparison between heuristic Solution and optimal Solution, but if we know $\subseteq(TSP^*)$ then we know sth. about how to choose our heuristic Solution to the optimal.

⊛ Simple lower bound for TSP:

- Observe one given feasible Solution tour.



tour

$|TOUR| = n$, $n = |N|$ → # of nodes

↳ # of arcs



tree

↳ If one arc is removed from the tour, then the remaining form a tree #

↳ $TOUR = TREE + [(i,j)]$

$C(Tour) = C(Tree) + Cij$

$C(TSP^*) = MST + Cij^*$) minimization for both sides #

→ where Cij is min. Cost arc not in the MST.

Scanned with CamScanner

- $C(TSP^*) \geq MST + C_{ij}^*$

- $\underline{C}(TSP^*) \leq C(\overline{TSP^*}) \leq C(TSP)^H$

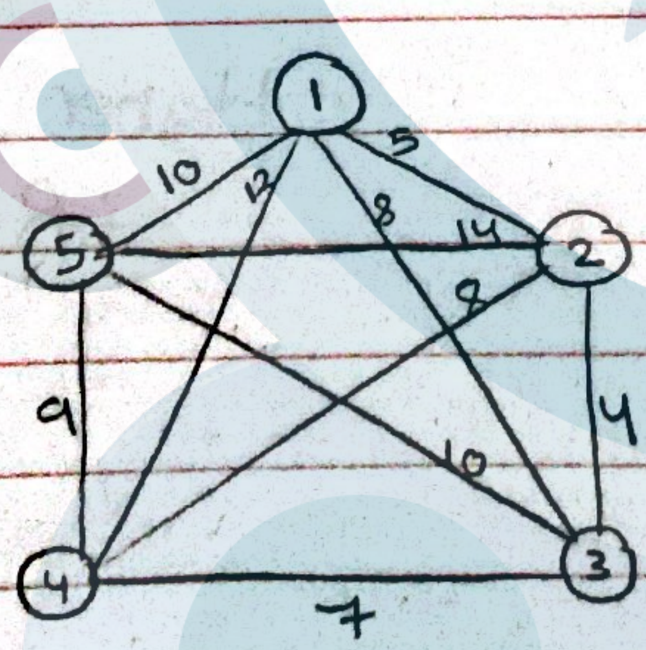
$$LB = C(MST) + C^*(i,j)$$

- Steps of finding Simple LB for TSP :

① Decompose feasible solution into 2 Pieces.
② optimize each piece
③ Combine the piece that we have broken out

- Note :
This may not be feasible tour solution to the original TSP, but the result gives LB ≤ opt. ولو أعطتني feasible مباشرة هو ال

* Example :



→ Find the LB for the TSP.

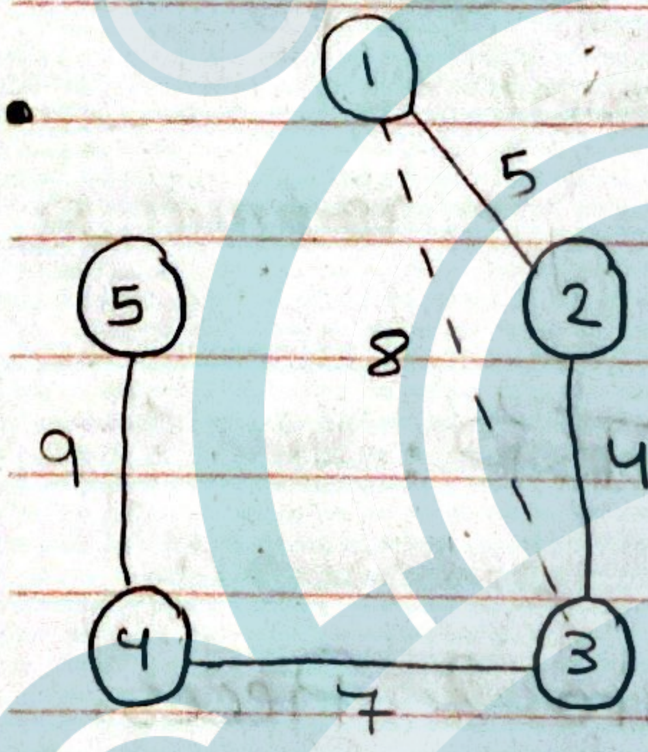
→ LB = MST + C(i,j)*

Scanned with CamScanner

Sol.

Draw the MST then add min. Cost arc that is not in MST

حتى لو عمل cycle عادي بنضيفه



↳ not a fecusible tour
(there is subtour)

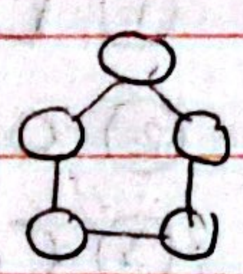
↳ $LB = C(MST) + C_{ij}^*$

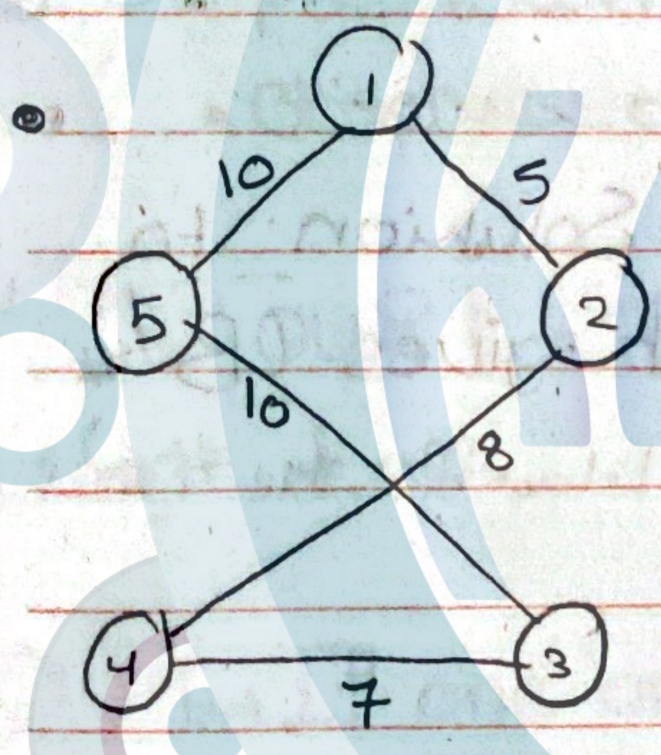
$= 25 + 8 = 33$ #

$C(TSP^*) \geq C(TSP)$

$C(TSP^*) = 35$

$C(TSP^*) \geq 33$





↳ $LB < opt. < Huerisric$

$33 < 35 < 40$

↳ $\frac{40-33}{33} * 100\% = 20\%$

Higher than LB

(*) if $C^H < LB$ → wrong Sol.

(=) if $C^H = LB$ → optimal.

(≥) if $C^H \geq LB$ → correct but not opt.

Scanned with CamScanner

Tuesday
22/11

* 2nd method for determining the lower bound for TSP

↳ 1-TREE LB for TSP.

- Definition : 1-TREE is a tree on n-1 nodes
  ↳ # of arcs in 1-TREE = (n-1) - 1
  = (n-2)

⇒ now for finding MST for this graph → node يلي حصولها أسهل اك
stop solving MST before adding the last node #

- TOUR = 1-TREE$_i$ + [(i,j), (i,k)]
  ↳ commited node (ignored)
  = TREE + 2 arcs from node i

- LB = MST$_i$ + (i,k) (i,j)

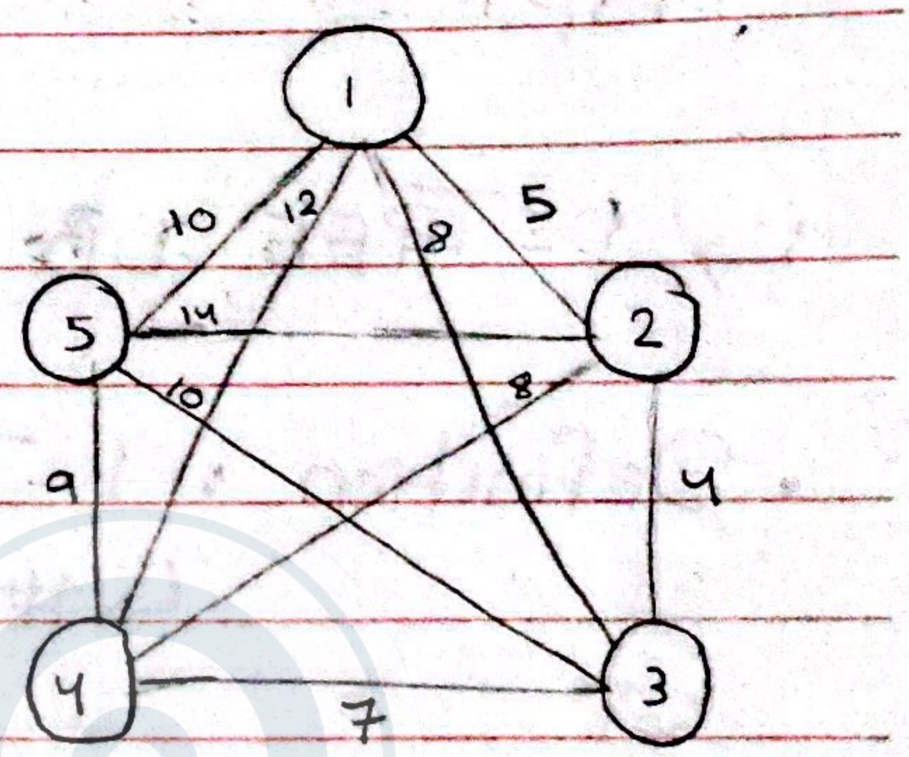
- C(TOUR) = C(1-TREE^i) + C_{ij} + C_{ik}

- C(TSP)* ≥ C(1-TREE) + C_{ij}^* + C_{ik}^*
  ↳ 2 smallest arc cost from (i)

Scanned with CamScanner

⊛ Example

↳ Complete Undirect graph that Satisfies triangle inequality.

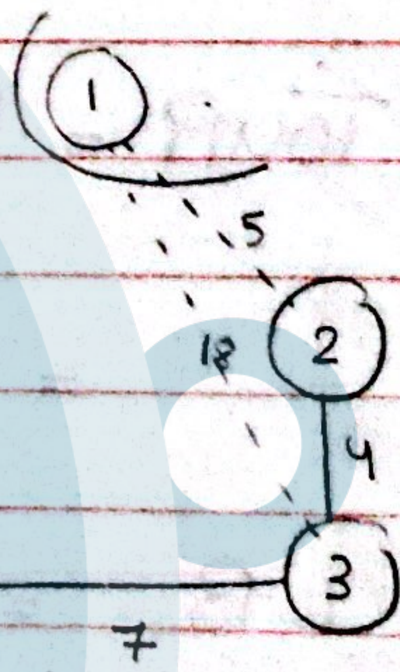


↳ UB = MST + $C(i,j)^*$ = 33

⊛ 1 - TREE,

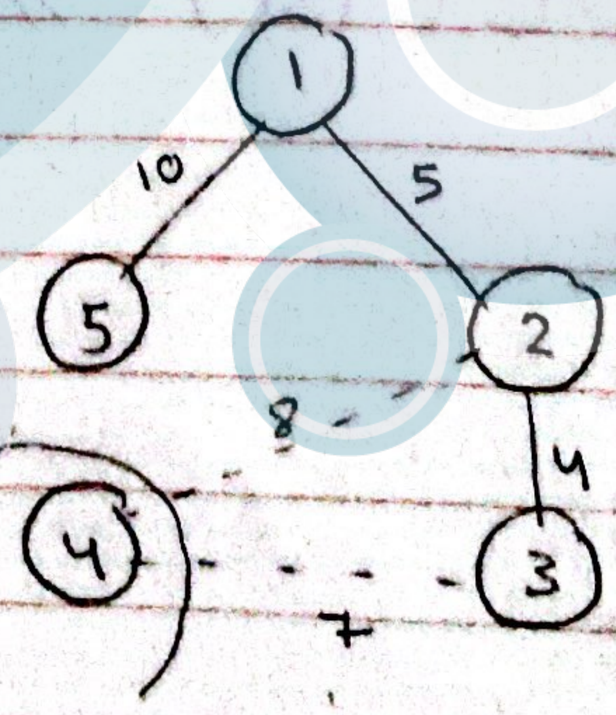
1) neglect node ①

2) take 2 arcs from the neglected node.



1 - TREE, = 33 غشوائيه node لَ ignore بعدك .

نفس الرقم ← $C(TSP^*) \geq 33$.



⊛ 1 - $TREE_4$ = 34

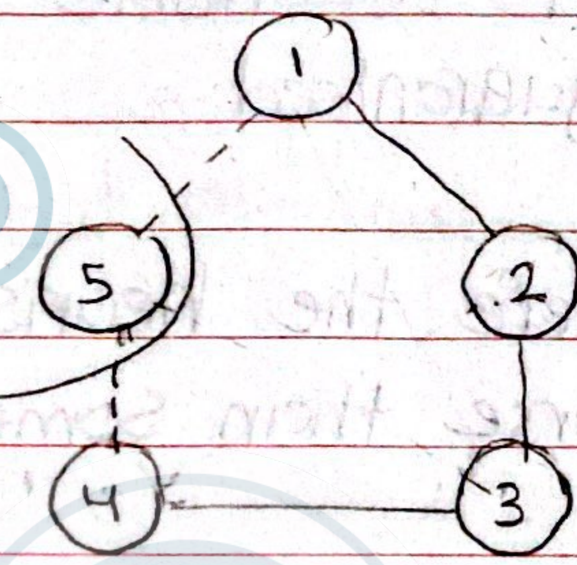
$C(TSP^*) \geq 34$

Scanned with CamScanner

*edited

⊛ $1 - \overline{TREE}_5 = 35$

$C(TSP^*) \geq 35$



- كل ما يزيد الرقم عما هو عليه يقرّب على ال optimal.

- يختار الأعلى من ال LB لأنه عليك يكون أقرب من ال optimal ← يعني هون ال 35 هي الأحسن، كلهم valid بس ال $C(TSP^*) \geq 35$ هي ال useful

KAIZEN TEAM

Scanned with CamScanner

* TSP heuristic based on MST with Performance guaranteed → يعني احنا عارفين انه ال worst case يكون ضعف ال opt.

↳ here the heuristic will never give a solution worse than some particular bound

- α optimal heuristic:
  ↳ also known as α approximation heuristic.
  ↳ You will never return a solution whose cost is more than α * true optimal solution

a) 2 MST - based heuristic

1. $\alpha = 2$ → twice around MST heuristic.
2. $\alpha = 1.5$ → Christofied heuristic

→ the heuristic sol. will never be > twice opt. sol.
→ " " " " " " > 1.5 * opt. sol.

1 ⇒ twice around MST heuristic :-

1. find MST of nodes N
2. let LIST contain 2 copies of each arc in MST
3. Create a walk of nodes in N by using arc in LIST once

Some nodes will be re-visited ← walk = $\{v_1, v_2, v_3, \dots, v_{2(n-1)}, v_1\}$

↳ →

Scanned with CamScanner

4. Create heuristic tour « T^H » from the walking by taking Shortcuts.

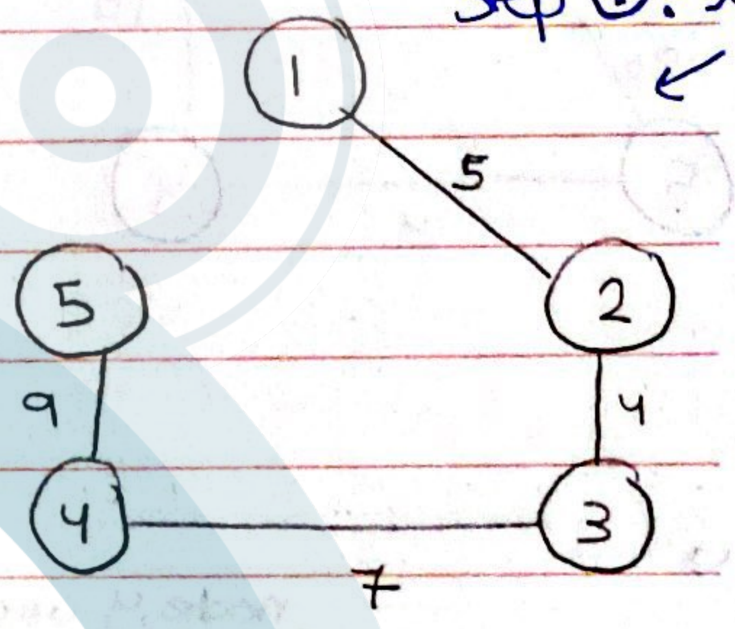
↳ (Follow same order as walk, but eliminate repeated visits to nodes).

* Example و نفس لي قبل صفحتين

step 2 LIST

(1,2)
(1,2)
(2,3)
(2,3)
(3,4)
(3,4)
(4,5)
(4,5)

step ①: MST ←



بدأت من أي node عادي

step 3 → walk = { 1, 2, 3, 4, 5, 4, 3, 2, 1 }

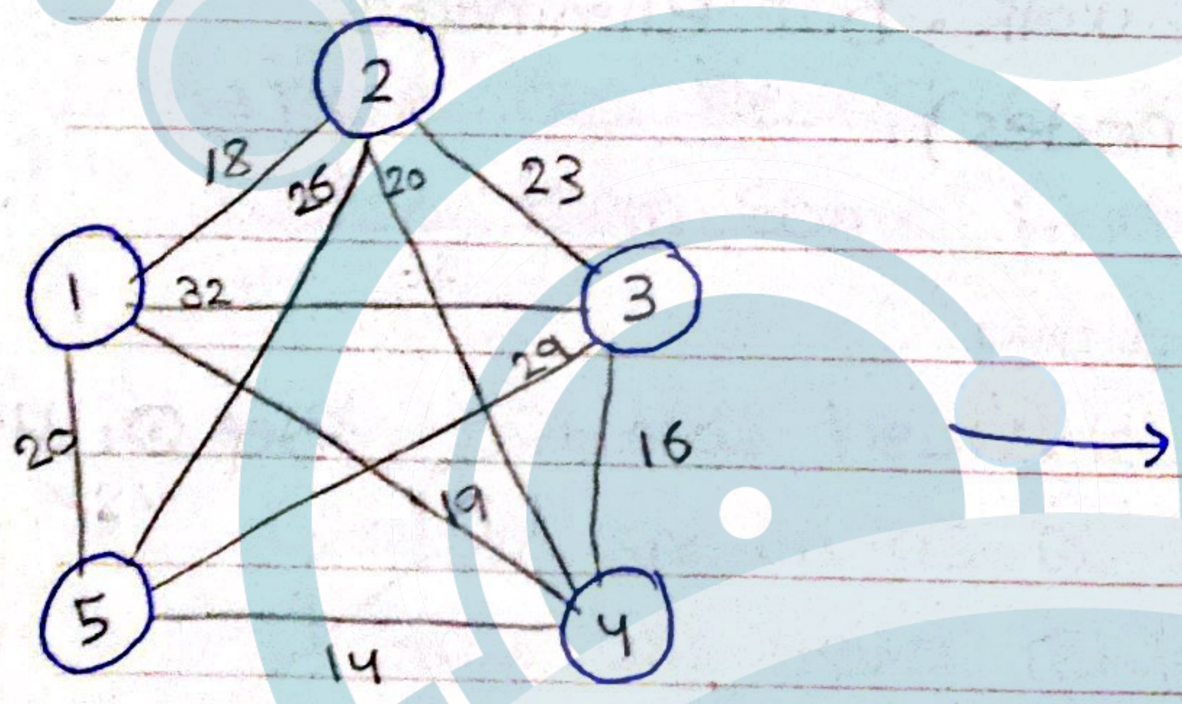
step 4 → TOUR = [1, 2, 3, 4, 5, 1] → بنشطب ال repeat لحد نرجع لنقطة البداية



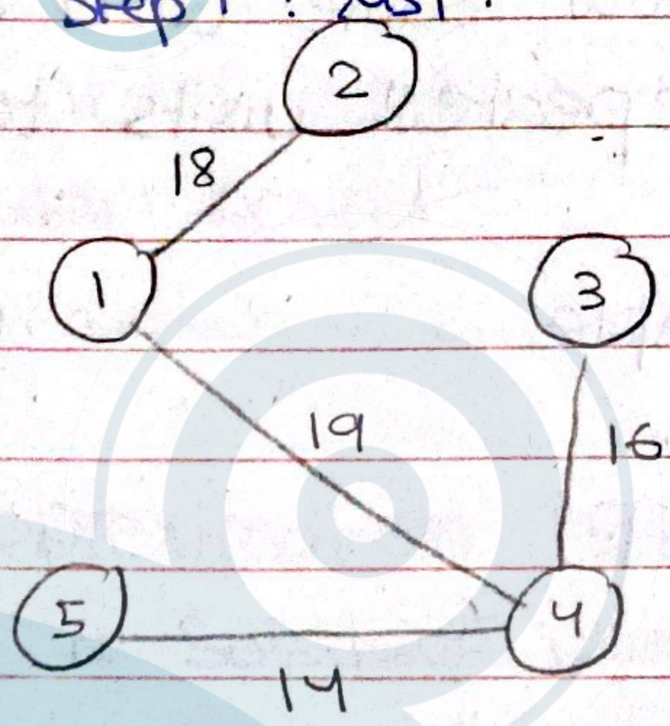
→ $C^H = 35$.#

Scanned with CamScanner

✱ example 2 :



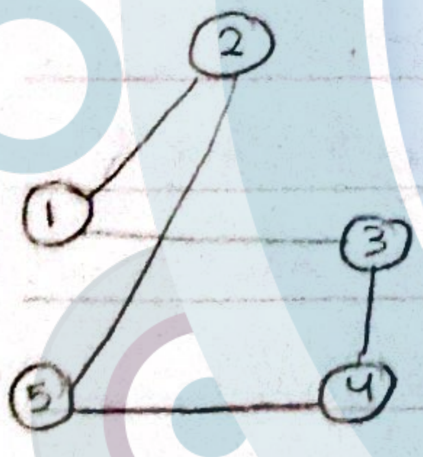
step 1 : MST :



↳ بدأنا الحل من node 4

step 3 ↳ walk = { 4, 3, 4, 1, 2, 1, 4, 5, 4 }

step 4 ↳ tour = [4, 3, 1, 2, 5, 4]



$C^H = 106 \neq$

↳ 14 + 16 + 32 + 26 + 18

step 2

LIST

(1, 2)
(1, 2)
(1, 4)
(1, 4)
(3, 4)
(3, 4)
(4, 5)
(4, 5)

- لو بدأنا من node تانية ممكن يطلع هذا Cost مختلف
- بس دايمًا يكون أقل من ضعف ال opt.

Scanned with CamScanner

Tuesday
29/11

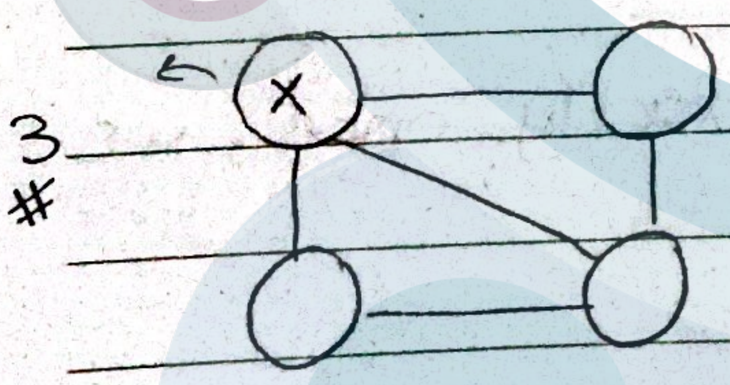
2 ⇒ Christofides heuristic
↳ the best Performance guarantee heuristic for TSP.

- TSP → OPtimal Solution ممكن ما أعرفه
  - hard to solve for med-large size Problems

→ Heuristic
- we use it to solve med-large size Prob. (instead of opt. method)

→ lower bound (LB)
- we use it to judge the heuristic evaluation
- Can be easily calculated, but usually not a Feasible tour
- but if the result UB tour is Feasible, then it's the optimal #

↳ node degree :
# of arcs adjasent to a node
↳ regadless of in or out arc.



Scanned with CamScanner

S → Set of nodes

↳ Perfect node Matching : Given a set of nodes (S), such that |S| is even, a Perfect node matching (w) is a set of |S|/2 node Pairings, such that each node in (S) is exactly in one Pairing

The cost of match $C(w) = \sum_{(i,j) \in w} c_{ij}$

* Christofides steps :-

[1] Find MST on the original graph

[2] let S be nodes of G with odd degree in MST.

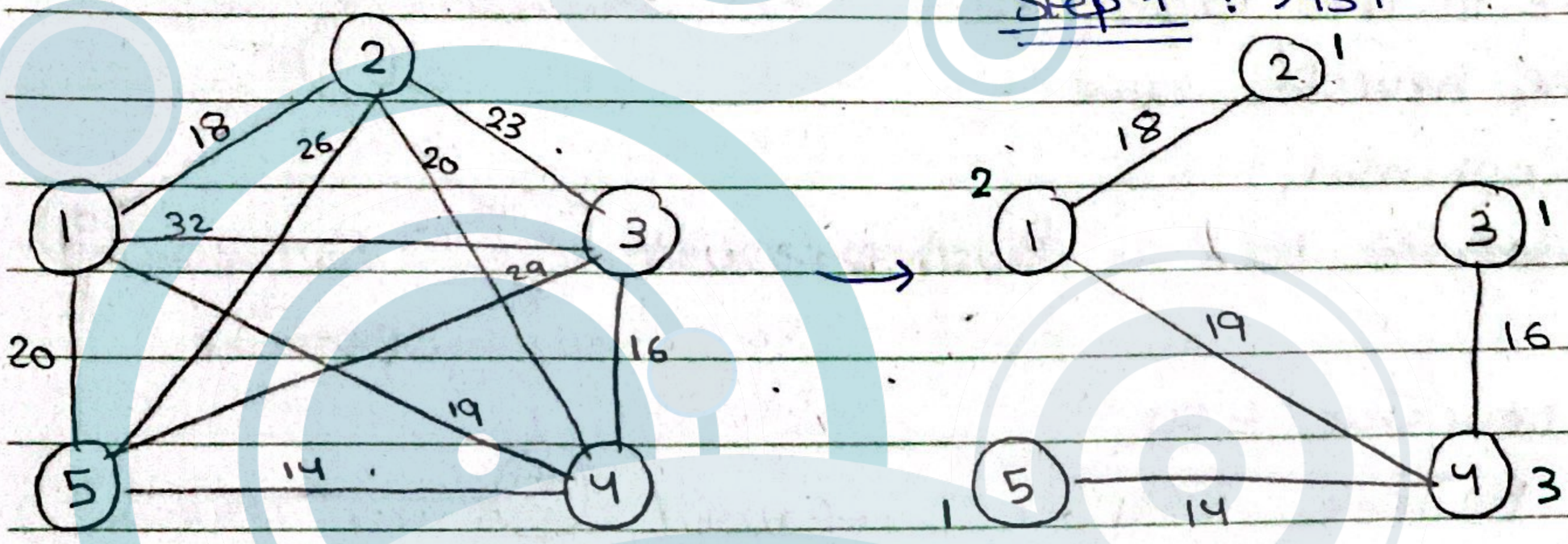
[3] Find the Perfect node matching (w^*) of nodes in (S) with min. Cost.

[4] Using arcs for MST and w^* exactly once, we generate a walk.

[5] Generate tour T^H by shortcutting the node (remove repeated nodes).

Scanned with CamScanner

⊛ example :-



Step 2

$S = \{2, 3, 4, 5\}$ → sets of nodes with odd degree.

Step 3

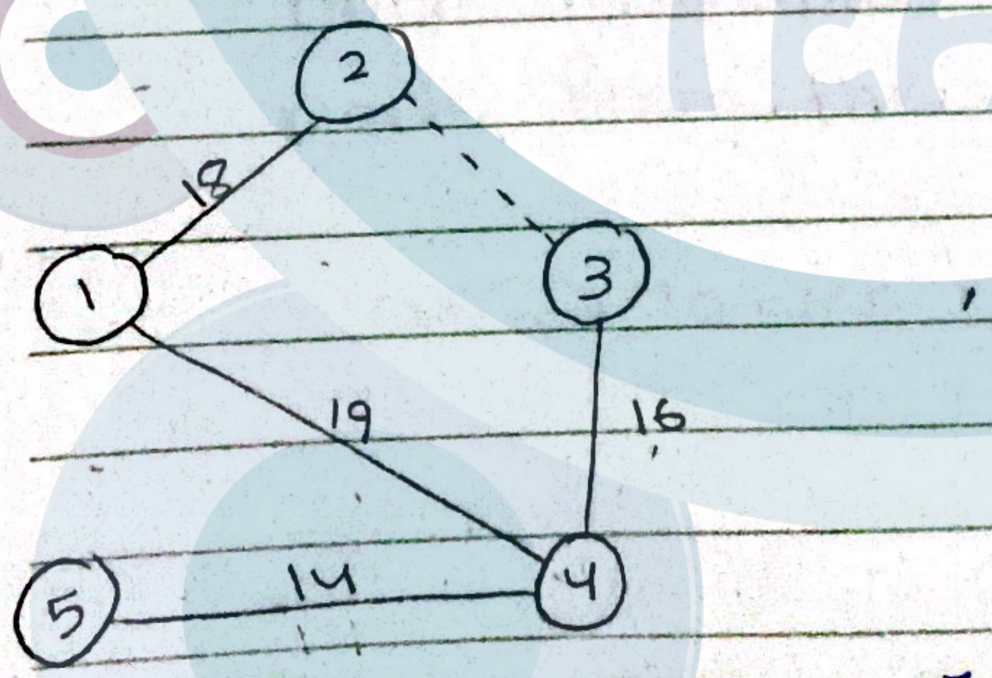
Matching: least cost.

$2-3$ & $4-5$ → $C_{23} + C_{45} = 23 + 14 = 37$ *5

$2-4$ & $3-5$ → $C_{24} + C_{35} = 20 + 29 = 49$

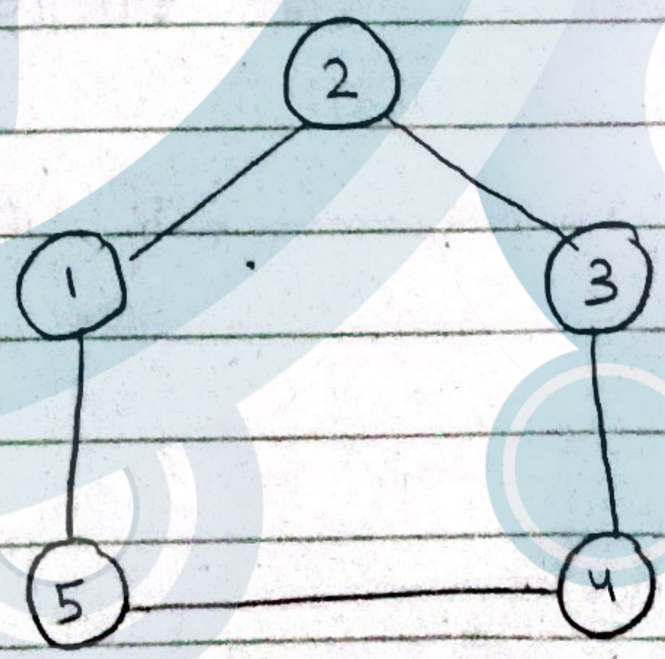
$2-5$ & $3-4$ → $C_{25} + C_{34} = 26 + 16 = 42$

Step 4



$W = \{1, 2, 3, 4, 5, 4, 1\}$

Step 5



Tour = { 1, 2, 3, 4, 5, 1 }

start, end.

Scanned with CamScanner

Tour :

→ heuristic = 91

1 - TREE = 91

→ if optimal = 100 :-

Twice around ≤ 200

Christofied ≤ 150

↳ 100 * 1.5



改善
Kaizen
TEAM

Scanned with CamScanner

Sunday
4/12

⊛ α-optimal heuristic, the performance is guaranteed, (we know that the worst case for C^A is to be twice or 1.5 the opt. Cost, "Practical heuristic" means that there is no perfor. guarate

⊛ Practical TSP heuristic :.

[1] Construction Approach
nearcast neighbor, nearest insertion cost, furthest insertion Cost.

[2] improvement approach

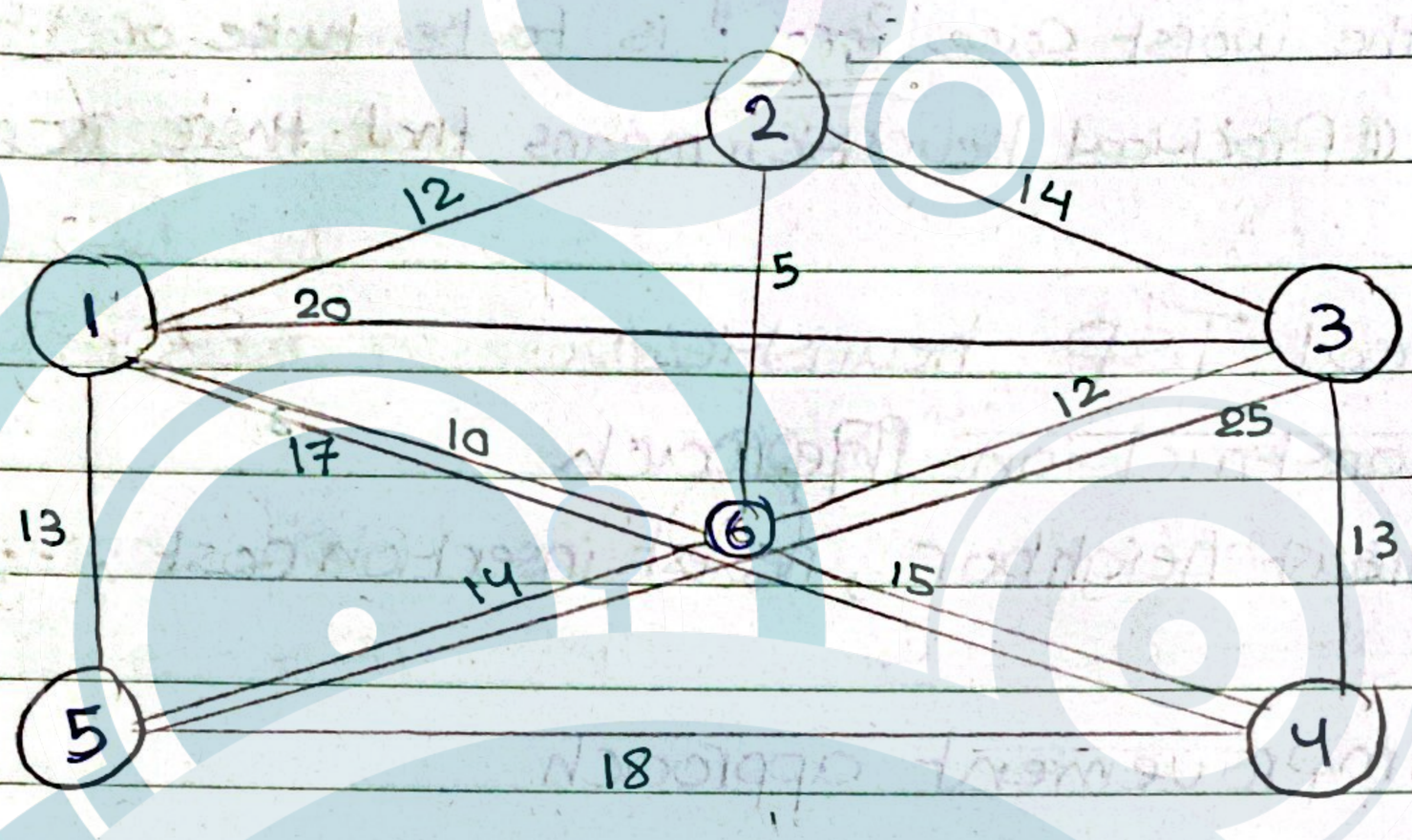
⊛ Nearest neighbor heuristic
↳ doesn't look any further than immediate move
↳ Considred to be a greedy heuristic.

• Steps :-
1. we start with the lowest cost arc, or with arbitrary node.

2. at each node, we extend Path to nearest node reachable from either start or ending node from the Path.

Scanned with CamScanner

* example



Sol.

* Starting with the lowes

Cost arc : (2-6)

Start with arc 2,6. → end (2,6)

add arc 6,1 → end (1,2)

~~add arc 1,2~~ → Cycle X

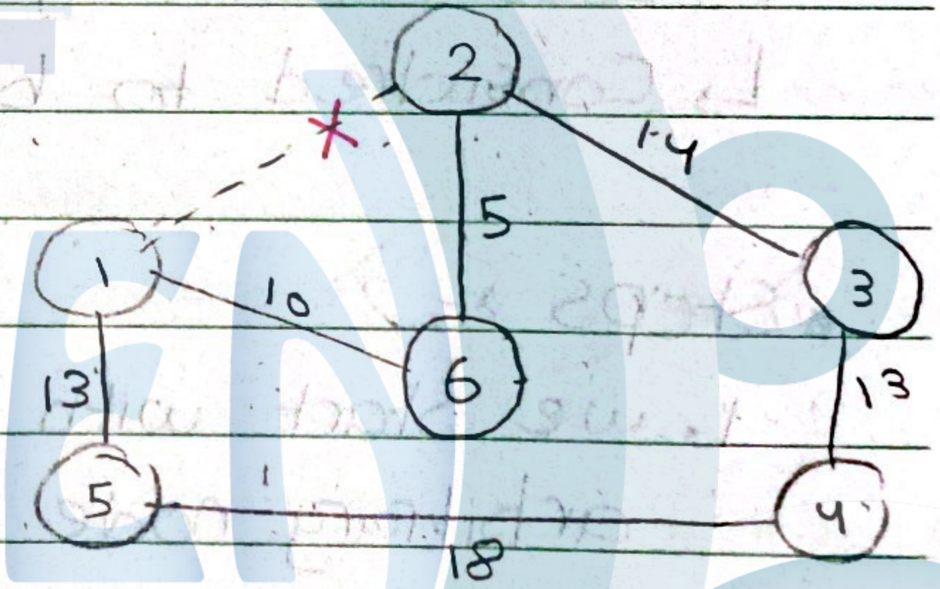
↳ " = 1,5 → " = (2,5).

" = 2,3 → " = (3,5)

" = 3,4 → " = (4,5)

" = 4,5 → end. *

Tour :-



← بشوف ال arcs يلي مرتبطين بال ending & start. nodes

ويختار ال lowest cost arc

ويضيفه على ال tour و

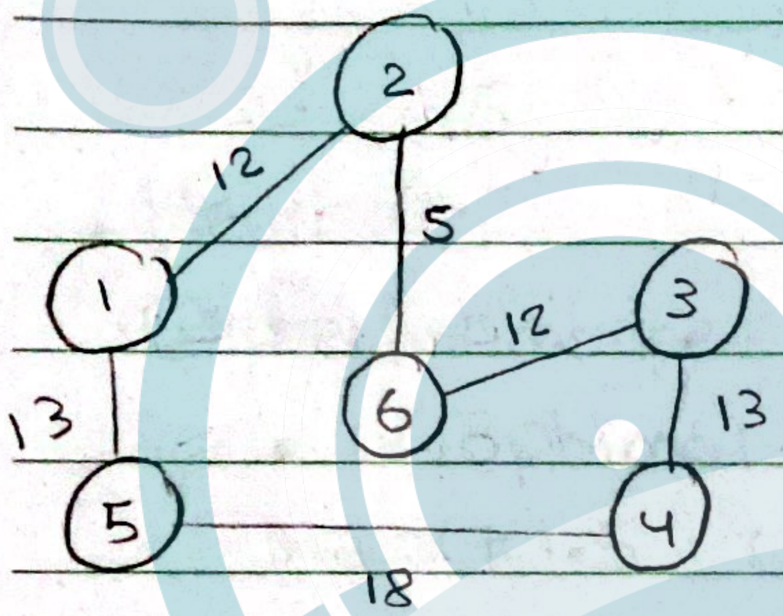
هكذا ...

Scanned with CamScanner

↱ Same example

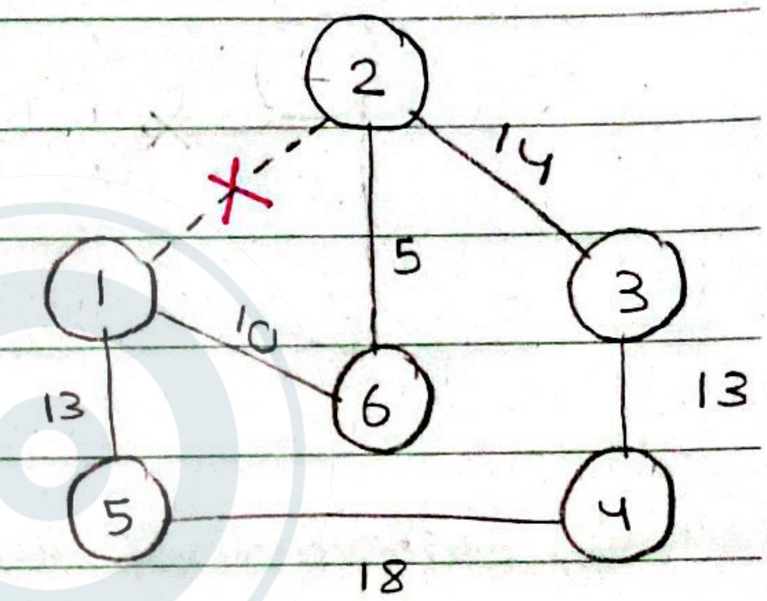
⊛ Starting with arbitrary node:

• node ④



↳ Cost = 73

• node ①



↳ Cost = 73

↳ ممكن بعاي الطريقة أحصّل

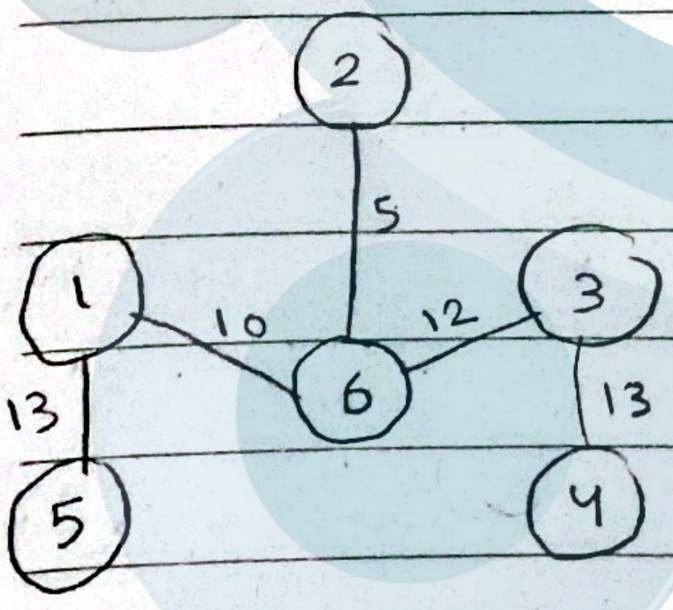
Diff. arcs & Diff tour.

→ there is no one unique Sol.

⊛ Same example

• LB = MST + (i,j)*

• MST:



→ MST Cost = 53

→ (i,j)* Cost = 12

→ LB = 53 + 12 = 65

→ opt. Sol. ≥ 65

Scanned with CamScanner

Tuesday
6/12

Nearest insertion heuristic

Initialization:

→ Given an undirected network $G = (N, A)$, with costs c_{ij} associated with each $(i, j) \in A$,
→ where c_{ij} satisfies the Δ-inequality.
Initial. ← Find the arc (i, j) with minimum cost in A. Let $T = \{i, j, i\}$

Steps:

while $|T| < n + 1$:

1. Find the node $j \notin T$ that is *closest* to a node currently in T. That is, find $j \notin T, i \in T$ such that c_{ij} is minimized.

2. Find the *insertion location* for node j into the existing tour. To do so, find the arc (ℓ, k) in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert j between ℓ and k in T.

Farthest insertion heuristic

Initialization:

Given an undirected network $G = (N, A)$, with costs c_{ij} associated with each $(i, j) \in A$, where c_{ij} satisfies the Δ-inequality.

Find the arc (i, j) with maximum cost in A. Let $T = \{i, j, i\}$

Steps:

while $|T| < n + 1$:

1. Let $C_j(T)$ be $\min_{i \in T} c_{ij}$ for all nodes $j \notin T$. The node j to be inserted into the tour is one with maximum $C_j(T)$.

2. Find the *insertion location* for node j into the existing tour. To do so, find the arc (ℓ, k) in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert j between ℓ and k in T.

end;

In English, Step 1 finds the node j not on the tour for which the minimum distance to a vertex on the tour is maximum, and adds that node to the current tour.

. Both nearest and farthest are ① considered to be better than nearest neighbor.

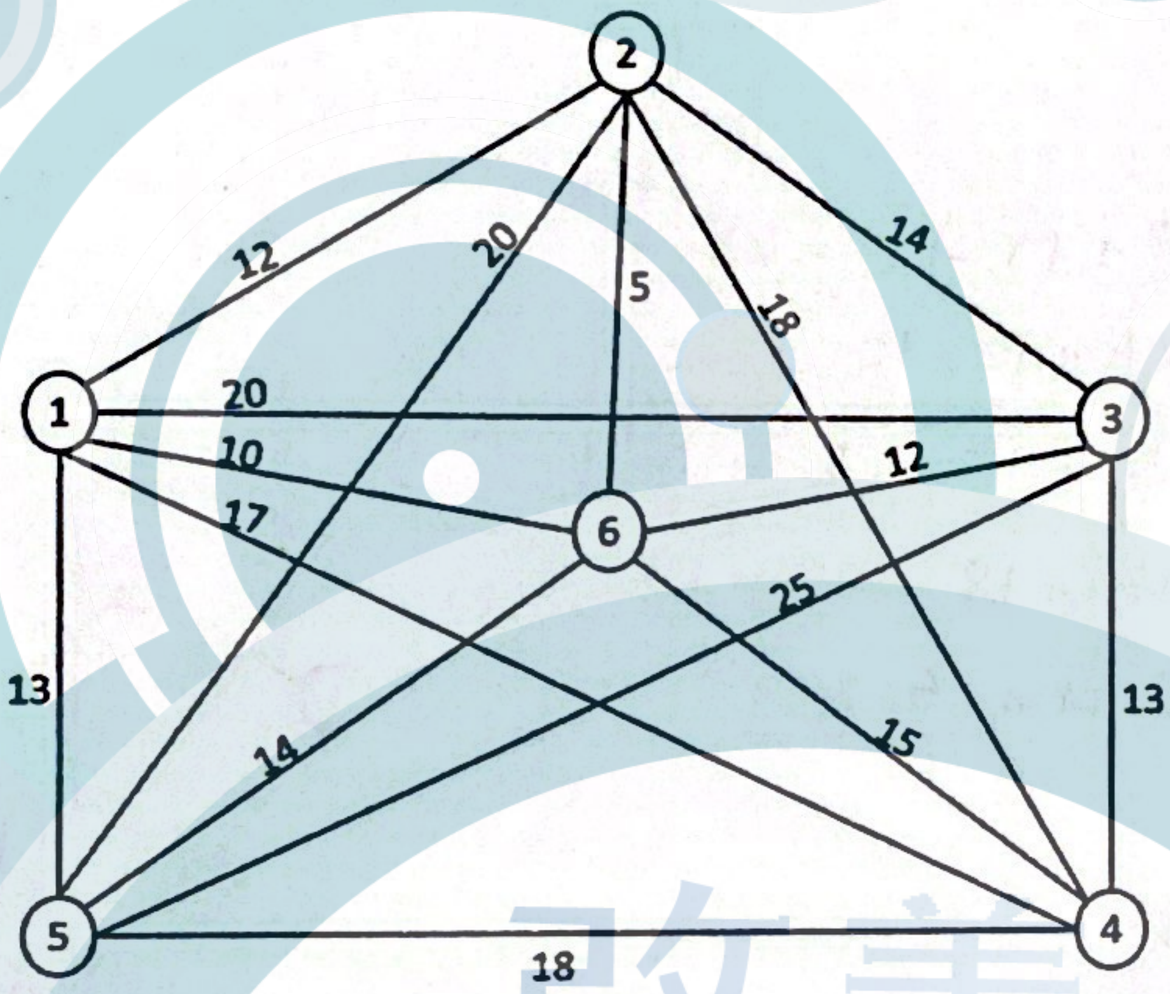
② Practical Heuristic (not guaranteed Performance)

Scanned with CamScanner

Example:

Consider the following symmetric, undirected network G where the numbers next to each arc are the arc costs. Suppose that your goal is to find a good traveling salesman tour that visits all of the nodes of G.

نزور كل ال nodes ب



- Find the trave saleman tour using nearest insertion heurishic.

- **initialization:**

↳ we start with min cost arc : (2,6)

$T = [2, 6, 2]$, $C(T) = 10$ →

(T: tour, st.: 2, end.: 2)

iteration ①:

- **Step 1:** [to determine which node to insert]

  - iteration ①:

j ∉ tour	i ∈ tour	C_{ij}
1	6	10 ←
3	6	12
4	6	15
5	6	14

↳ we choose to insert node 1 to the tour

- **Step 2:** [to determine where to add the node]

Node 1 will be inserted

btw node 2 & 6 $T = [2, 1, 6, 2]$

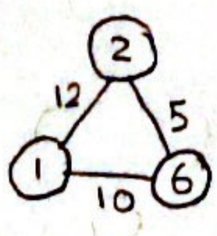
insertion cost $= C_{21} + C_{16} - C_{26}$

$= 12 + 10 - 5 = 17$

$T = [2, 1, 6, 2]$,

$C(T) = C(T)_{\text{Previous}} + \text{insertion Cost}$

$= 10 + 17 = 27$ #



Iteration ②:

Step 1

j ∉ T	i ∈ T	C_{ij}
3	6	12 ←
4	6	15
5	1	13

↳ we choose node ③ to insert.

Step 2

insert node 3 either:

btw 2,1 $= C_{23} + C_{31} - C_{21} = 14 + 20 - 12 = 22$

≡ 1,6 $= C_{13} + C_{36} - C_{16} = 20 + 12 - 10 = 22$

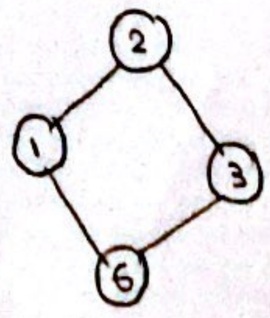
≡ 6,2 $= C_{63} + C_{32} - C_{62} = 12 + 14 - 5 = \boxed{21}$

↳ we will insert it btw 6 & 2.

$T = [2, 1, 6, 3, 2]$.

$C(T) = C(T)_{\text{Prev.}} + \text{insertion Cost}$

$= 27 + 21 = 48$ #



Cont'd. →

Scanned with CamScanner

Iteration (3):

Step 1

j∈T	i∈T	C_{ij}
4	3	13
5	1	13

↳ we choose node 4 to insert

Step 2

Insert node 4 either

btw 2,1 = $C_{24} + C_{41} - C_{21}$ = 18 + 17 - 12 = 23

btw 1,6 = $C_{14} + C_{46} - C_{16}$ = 17 + 15 - 10 = 22

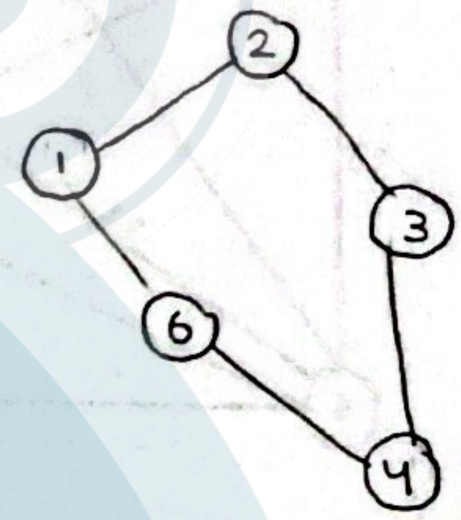
btw 6,3 = $C_{64} + C_{43} - C_{63}$ = 15 + 13 - 12 = **16**

btw 3,2 = $C_{34} + C_{42} - C_{32}$ = 13 + 18 - 14 = 17

↳ we choose to insert node 4 btw 6 & 3

$T = \{2, 1, 6, 4, 3, 2\}$

$C(T) = 48 + 16 = 64$ #



Iteration (4):

Step 1

node 5

$T = [2, 1, 6, 4, 3, 2]$

Step 2

inserting node 5 either:

btw 2,1 = $C_{25} + C_{51} - C_{21}$ = 21

= 1,6 = $C_{15} + C_{56} - C_{16}$ = 17 ✓

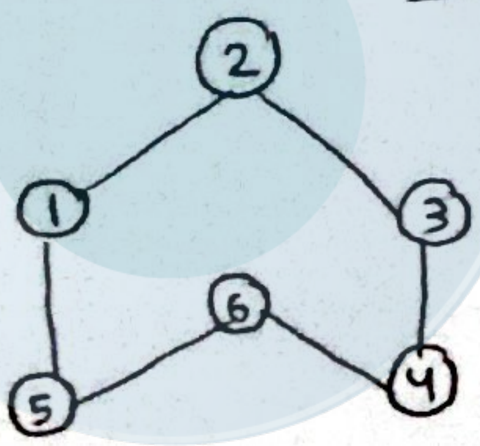
= 6,4 = $C_{65} + C_{54} - C_{64}$ = 17 ✓

= 4,3 = $C_{45} + C_{53} - C_{43}$ = 30

= 3,2 = $C_{35} + C_{52} - C_{32}$ = 31

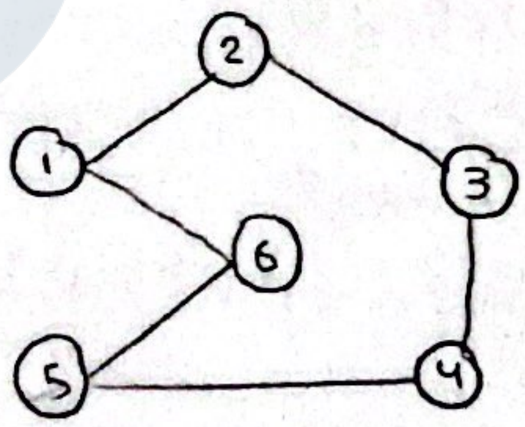
(*) btw 1 & 6 → $T = \{2, 1, 5, 6, 4, 3, 2\}$

$C(T) = 64 + 17 = 81$



(*) btw 6 & 4 → $T = \{2, 1, 6, 5, 4, 3, 2\}$

$C(T) = 64 + 17 = 81$

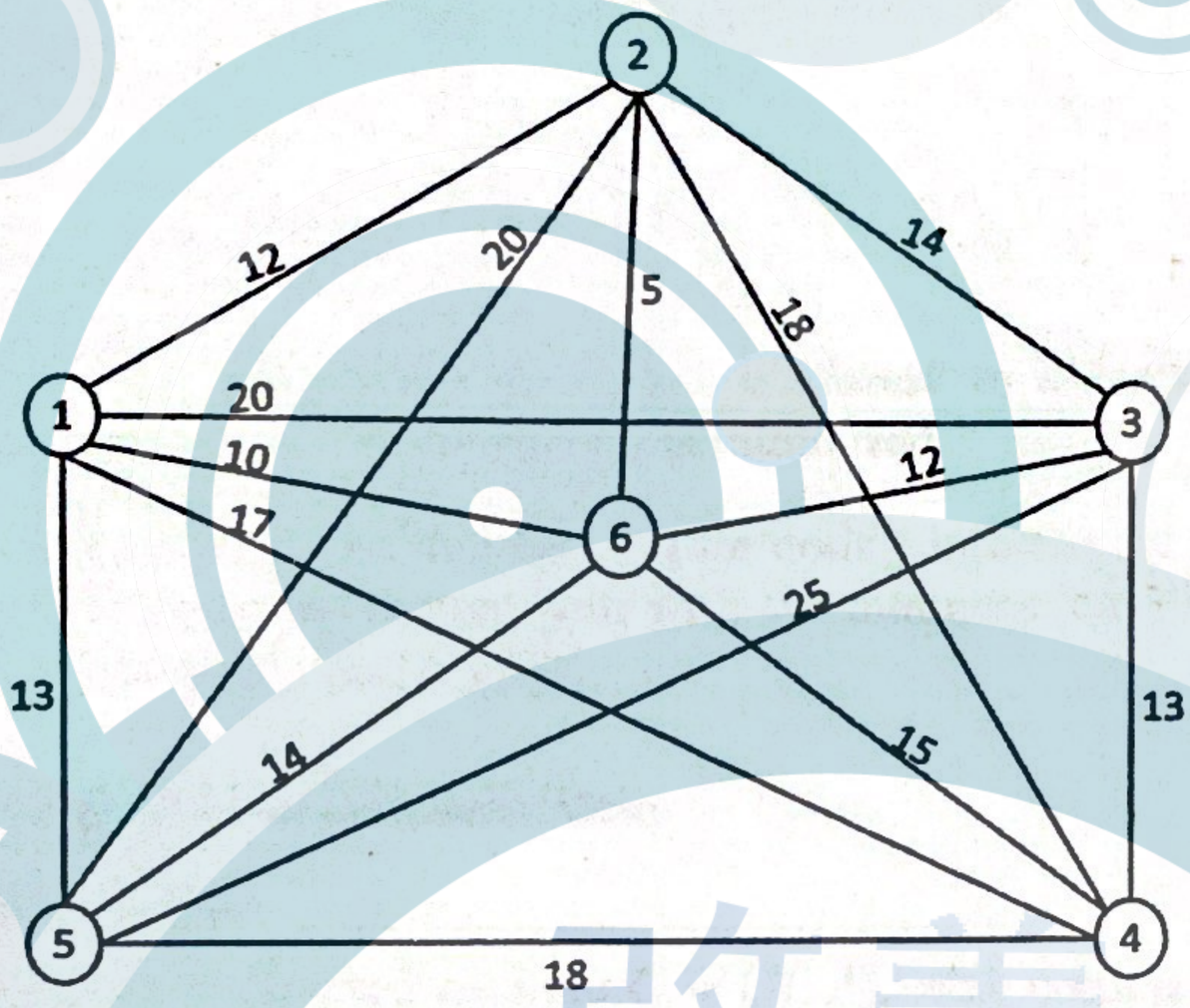


Scanned with CamScanner

Thursday
8/12.

Example:

Consider the following symmetric, undirected network G where the numbers next to each arc are the arc costs. Suppose that your goal is to find a good traveling salesman tour that visits all of the nodes of G.



ال . min هـون
باخدها بس عند ال
insertion
← الباقي كلّه max .

Initialization : (3,5)
↳ max cost arc.

$T = \{3, 5, 3\}$, $C(T) = 50$

Iteration 1 : $T = \{3, 5, 3\}$

step1

j∉T	i∈T	C_{ij}
1	5	13
→ 2	3	14
4	3	13
6	3	12

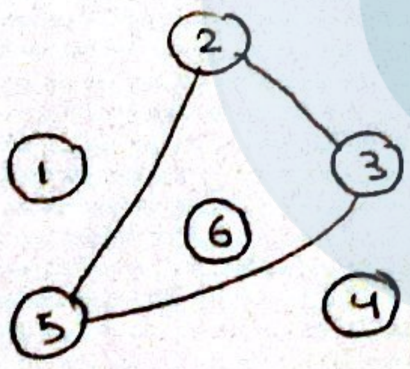
we choose to insert node 2 → max. cost

step 2

insertion cost = $C_{32} + C_{25} - C_{35}$

$= 14 + 20 - 25 = 9$

$C(T) = 50 + 9 = 59$ #



Iteration 2:

step1

j∉T	i∈T	C_{ij}
1	2	12
4	3	13
6	2	5

↳ we choose to insert node (4)

step 2 → $T = \{3, 2, 5, 3\}$

btw 3,2 = $C_{34} + C_{42} - C_{32} = 17$
= 2,5 = $C_{24} + C_{45} - C_{25} = 16$
= 5,3 = $C_{54} + C_{43} - C_{53} = 6$ ←

$T = \{3, 2, 5, 4, 3\}$

$C(T) = C(T)_{Prev.} + insertion$

$= 59 + 6 = 65$ #

→

Scanned with CamScanner

Iteration (3) :-

Step 1

J ∉ T	i ∈ T	C_{ij}
1	2	12 ←
6	2	5

↳ we choose node ① to insert

Step 2 $T = \{3, 2, 5, 4, 3\}$

btw 3,2 $= C_{31} + C_{12} - C_{32} = 18$

= 2,5 $= C_{21} + C_{15} - C_{25} = 5$ ←

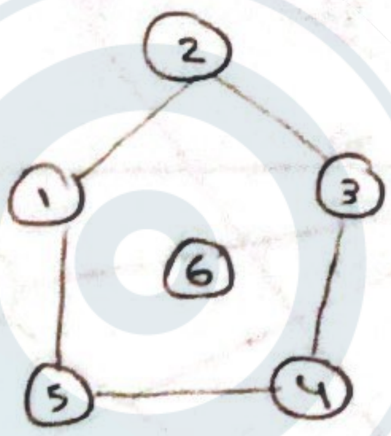
= 5,4 $= C_{51} + 5_{14} - C_{54} = 12$

= 4,3 $= C_{41} + C_{13} - C_{43} = 24$

↳ we will insert it btw 2 & 5

$T = \{3, 2, 1, 5, 4, 3\}$

$C(T) = 65 + 5 = 70$ #



Iteration (4) :-

Step 1

node ⑥ will enter the tour.

Step 2

btw 3,2 $= C_{36} + C_{62} - C_{32} = 3$

2,1 $= C_{26} + C_{61} - C_{21} = 3$

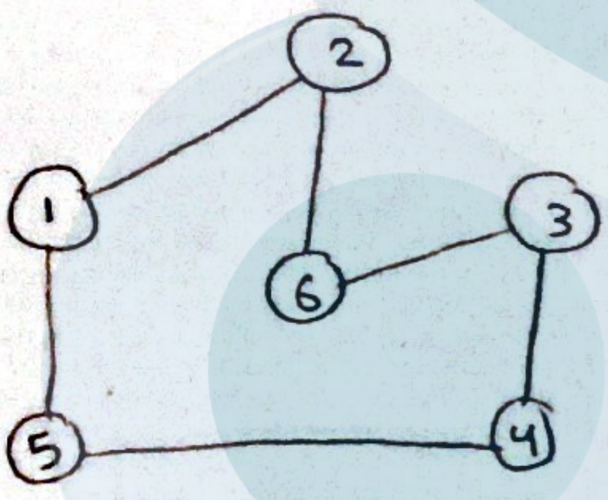
1,5 $= C_{16} + C_{65} - C_{15} = 11$

5,4 $= C_{56} + C_{64} - C_{54} = 11$

4,3 $= C_{46} + C_{63} - C_{43} = 14$

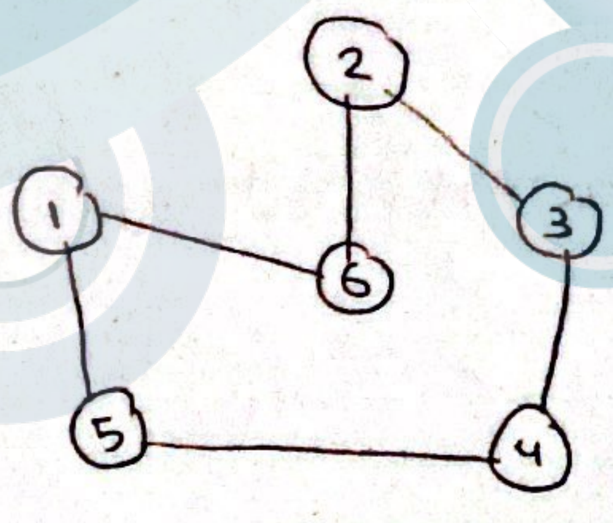
btw 3,2 → $T = \{3, 6, 2, 1, 5, 4, 3\}$

$C(T) = 70 + 3 = 73$



btw 2,1 → $T = \{3, 2, 6, 1, 5, 4, 3\}$

$C(T) = 70 + 3 = 73$ #



Scanned with CamScanner

Sunday
11/12

⊛ Revision

- Recall TSP → Optimal Sol.
  → Heuristic
  → LB

- Heuristic

① Performance guaranteed.:

- α optimal heuristic

1 → $\alpha = 1.5$ (Christofides) → best Performance guarantee

2 → $\alpha = 2$ (twice around).

② Performance not guaranteed: (Practical heuristic)

[1] Construction approach

- nearest neighbor heuristic
- nearest insertion heuristic
- farthest insertion heuristic

[2] improvement approach

Scanned with CamScanner

⊛ improvement heuristic

- idea : given an existing tour, Consider small, local changes to decrease tot. Cost.

Current Sol. لـ لأنها تعتمد على ال

- the existing tour could by one of the Previous heurishic Solutions mentioned.

- Common approach: λ- exchange

↳ Choose Some λ (integer #) arcs to remove, and then choose λ arcs to add to create Feasible Solution.

لـ يعني tour مو Subtour أو ...

- λ optimal

a tour is λ-optimal if there is noway to replace λ arcs Currently in the tour with λ arcs not in the tour to Create a feasible Solution.

← لازم كل tour يكونه ك improvement

← كل exchange يطلعي tour جديد

Scanned with CamScanner

- given a complete undirected network $G_1 = (N, A)$ and tour T_i, $A(T_i)$ be set of arcs in T_i

such that $\Rightarrow$ $A(T_i) = \{(v_1, v_2), (v_2, v_3), \dots, (v_{n-1}, v_1)\}$

- then T_i is $\exists$ optimal if there do not exist two set of arcs

Set of arcs removed ← $\therefore X \in A(T_i)$

Set of arcs added ← $\therefore Y \in A / A(T_i)$

↳ except

λ opt. ② Conditions

1. $|X| = |Y| = \lambda$

2. $\sum_{(i,j) \in X} C_{ij} - \sum_{(i,j) \in Y} C_{ij} > 0$ → يعني لازم أكون عندك Saving أو Cost أقل من min.

3. the set $A(T_i) + Y \setminus X$ allows a new feasible tour. T_{i+1}

Scanned with CamScanner

⊛ Steps : ٢ optimal heuristic (exchange heu.)

• initialization : given some initial feasibal tour (T_0) → K=0

لـ عـدّاد الـ iteration

• Iteration :

[1] identify a feasible λ-exchange with positive saving, That is, find two sets :-

$X \in A(T_i)$

$Y \in A / A(T_i)$

↳ such that Conditions 1-3 mentioned before are met.

. if no sets can be found, stop.

because → T_K is λ-optimal #

[2] let T_{K+1} be new tour generated by removing (X) and adding (Y)

[3] K++, Repeat from 1

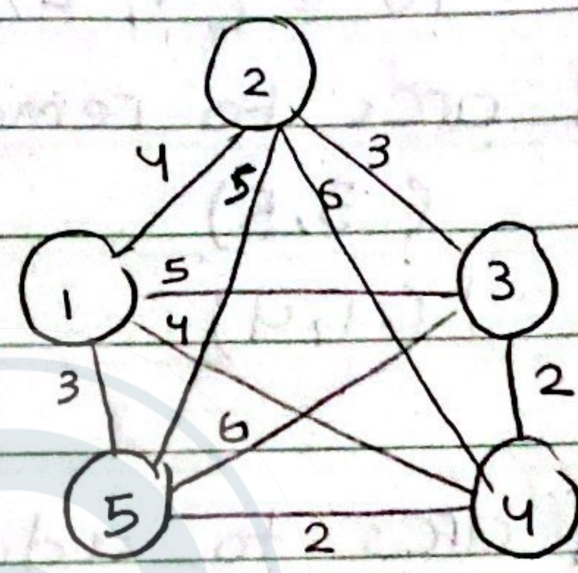
قيمة الـ λ الكتورة بتحددها باكمتان "٢"

لأنّه ما بتعدّ بشوف كل الـ Possibilities

⊛ example

Suppose we have Complete Undirect graph G

G



$T_o = [1, 2, 4, 5, 3, 1]$

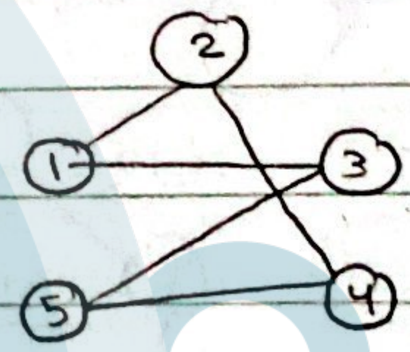
→ Perform 2 exchange heuristic on this tour T_o, to try to improve cost.

Sol.

1] identify arcs to remove:

(3,1)
(2,4)] → يلي بتقاطعوا

T_o ↘

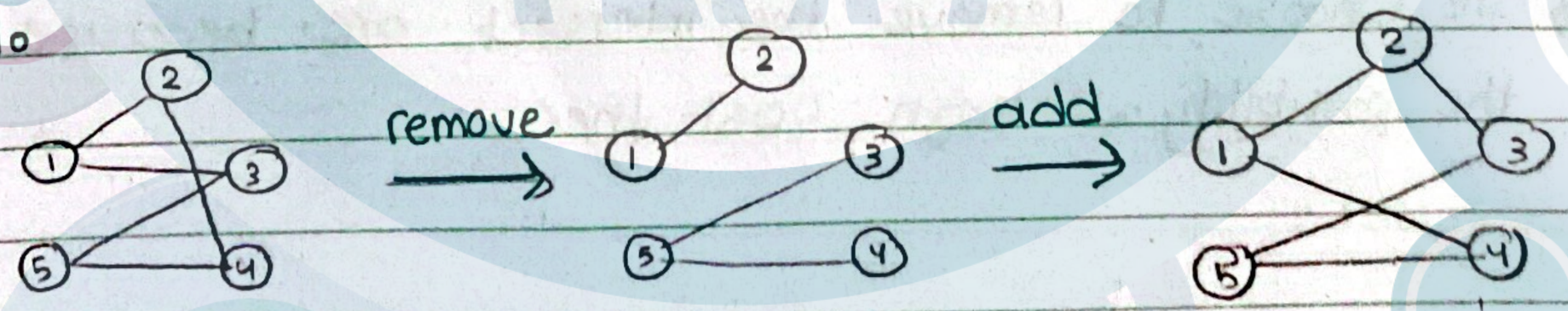


2] identify arcs to add

(2,3)
(1,4)

3] Check for improvement.

T_o



- $(C_{31} + C_{24}) - (C_{23} + C_{41}) \overset{!?}{>} 0$

$11 - 7 > 0$ ✓

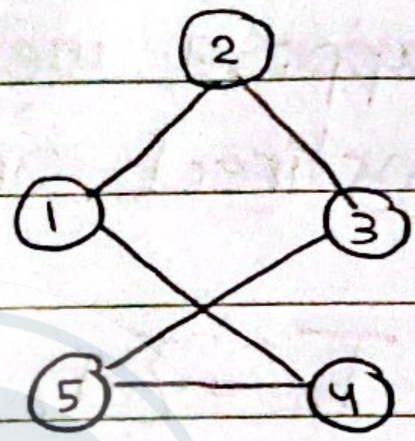
↳ Since improvement is checked, this will become T_o for next iteration

Scanned with CamScanner

Iteration [2]

$T_o = \{1, 2, 3, 5, 4, 1\}$

T_o ↴



1] arcs to remove :

(3,5)

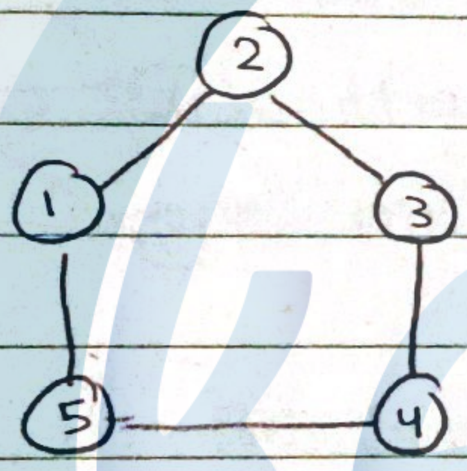
(1,4)

2] arcs to add :

(1,5)

(3,4)

3] Check for improvement :-



$(C_{14} + C_{35}) - (C_{15} + C_{34}) > 0$

$4 + 6 - 3 + 2 > 0$

$5 > 0$ → exchange accepted. #

↳ tour after removing & adding arcs.

$C(T) = 14$ #

⊕ we choose to remove the intersect arcs because of the possibility of high cost there.

مادة الكويز #

Scanned with CamScanner

Tuesday
13/12

⊛Bin Packing Problem : « BPP »

- Description of the Problem :

↳ we have some # of identical bins (or truck, container, ...) each with Capacity Q

↳ we have some # of items that need to be Packed into bins, item i has size q_i

- $q_i \in [0, Q]$ ← q_i قيمة ال

↳ also we cannot split items (order) between bins.

↳ let's assume we are given items & sizes in a list :-

$$L = [q_1, q_2, \dots, q_n]$$

- Problem : assign each item to a bin such that :

[1] Capacity is observed, total weight / volume of all items in a bin $\leq Q$

[2] Smallest # of bins is used

- Example :

11 groups of People of Sizes : 2,2,3,3,3,4,4,4,5,5,5 need to fit to mini buses with Capacity of 10 but must stay together in their groups,

→ Find Smallest # of buses need to transport them.

Scanned with CamScanner

Sol.

↳ bin 1 = [5, 5]

↳ bin 2 = [3, 3, 4]

↳ bin 3 = [5, 3, 2]

↳ bin 4 = [4, 4, 2]

⊛ IP Formulation

- Decision variables :

$$y_j = \begin{cases} 1 \rightarrow & \text{if bin is Utilized} \\ 0 \rightarrow & \text{otherwise} \end{cases}$$

$$X_{ij} = \begin{cases} 1 \rightarrow & \text{if item is assigned to bin (j)} \\ 0 \rightarrow & \text{otherwise} \end{cases}$$

- Objective :

$$\text{Min} \sum_{j=1}^{m} y_j$$

- Such that (Constraints) :

Capacity Constraint [1] $\sum_{i=1}^{n} q_i X_{ij} \leq Q \rightarrow \forall_j$ ↳ For all j

each order should be served in 1 bin [2] $\sum_{j=1} X_{ij} = 1 \rightarrow \forall_i$

Connects btw X_{ij} & y_j [3] $X_{ij} \leq y_j \rightarrow \forall_j, \forall_i$

Scanned with CamScanner

- example
  - if $X_{13} = 1$ $\xrightarrow{\text{then must}}$ $Y_3 = 1$
  - $=$ $X_{13} = 0$
    - ↳ $Y_3 = 1$ or $Y_3 = 0$

- BPP, in theory : easy to understand
  + hard to solve] just like TSP

⊛ Approaches to Solve BPP :

[1] Online methods :

→ Data are received overtime

→ Irrecoverable decisions are made based on known data with some time limit

→ Short local visibility

[2] Offline methods :

→ Data about all items are available before any decision need to be finalized.

Scanned with CamScanner

Sunday
18/12

Heuristic for BPP :

* Online heuristic

1. Next fit heuristic (NF) → Performance $\alpha = 2$

- NF :

→ only one open bin at each step of the heuristic
→ Given list L of items each with size q_i
→ each bin has a Capacity of Q.

- Steps :

① Take item from the top of the list.

② If it fits in the current open bin, Place it & remove item from the list (then go back to step 1)

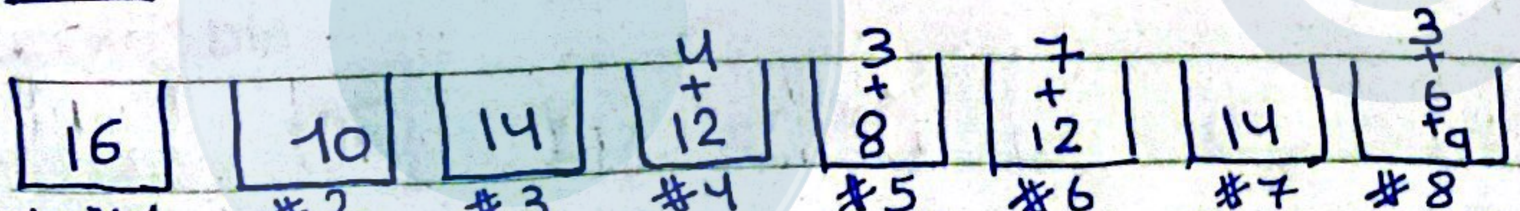
③ If it doesn't fit, Close the Current bin, open a new bin and Place item there, then remove item from the list, and go back to step 1.

↳ Continue until there is no item in our LIST (all Placed in bins #)

- Example :

L = [16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3], Q = 20

Sol.



↳ Capacity Left

⇒ So # of bins needed using NF heuristic is 8 bins #

Scanned with CamScanner

2 First-Fit heuristic (FF) $1.7 \leq \alpha \leq 1.75$

- idea : Keep all bins open until Completely Full or entire list has been processed.

- Steps :

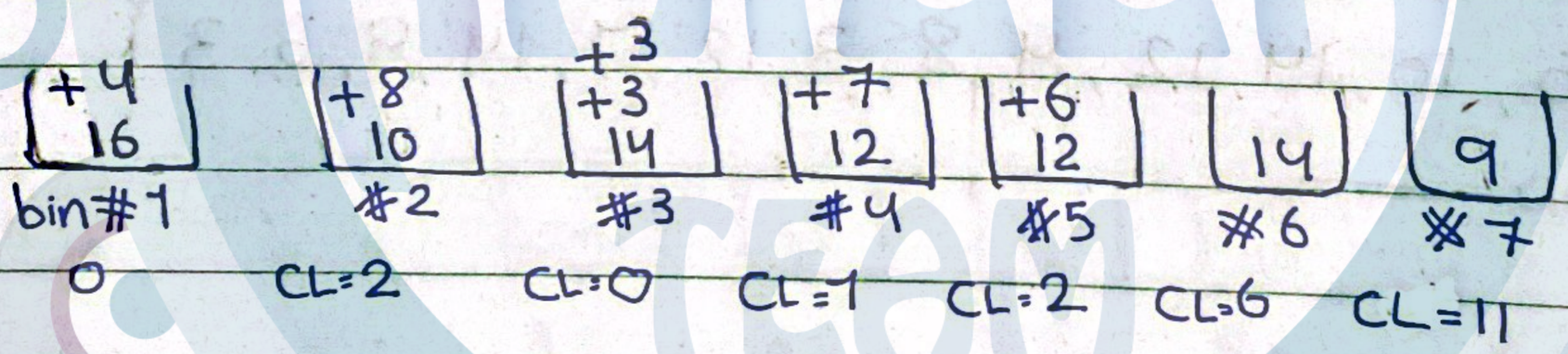
① Place next unplaced item in the lowest numbered bin, in which it fit, if no such bin exists, open new bin bin & place it there

② move to the next item & repeat.

- Example

$L = \{16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3\}$ $Q = 20$

Sol.



⇒ So # of bins needed using FF heuristic is 7 bins #

Scanned with CamScanner

[3] Best Fit heuristic $1.7 \leq \alpha \leq 1.75$

- idea: Keep all bins open until Completely full or entire list has been processed.

- Steps

① Place next unplaced item in bin such that:
   a) item fit
   b) residual Capacity at Choosen bin is minimized.
   c) if ties, Choose lower numbered bin.

② if no such bin exists, open new bin & place it there.

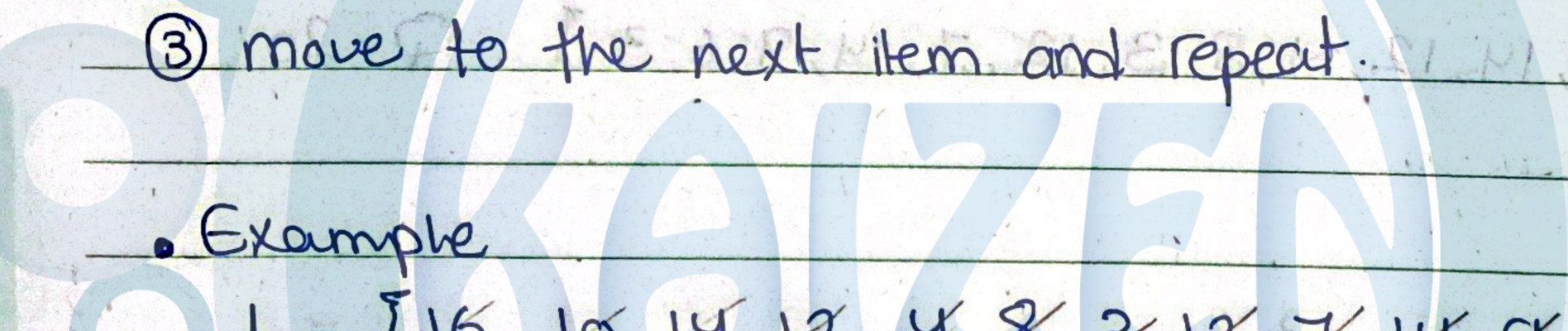
③ move to the next item and repeat.

- Example

$L = \{16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3\}$

$Q = 20$

Sol.



- $LB = \left\lceil \frac{\sum q_i}{Q} \right\rceil$ → Not feasible
  ↳* order might be Split.
  (in BPP we assumed that orders (items) Cannot be Split)

أكبر عدد صحيح.

Scanned with CamScanner

Tuesday
20/12

* BPP offline heuristic
↳ assume the list of items is completely known in advance:
1. Sort L (LIST) in non-increasing order of item size
↳ Size نفسها increasing بس انه مسموع بكون عندي ٢ نفس ال

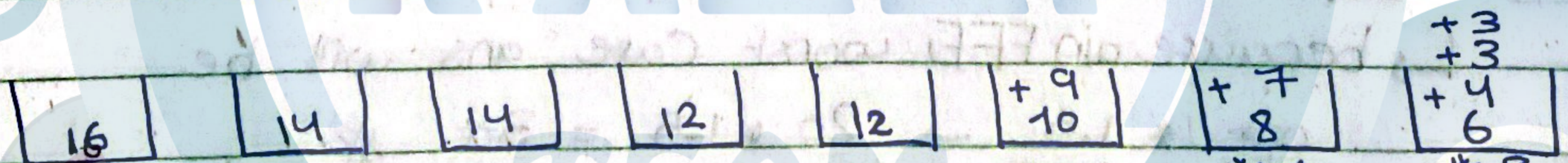
2. Run NF, BF, FF heuristic:

↳ NF Sorted data → Next Fit - decreasing
↳ BF Sorted data → Best Fit - decreasing
↳ FF Sorted data → First Fit - decreasing
offline heu. (data is already available)
$\alpha = 1.5$ (worst Case)

Example:

a) apply NF - decreasing on the following items
$L = [16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3]$, $Q = 20$.

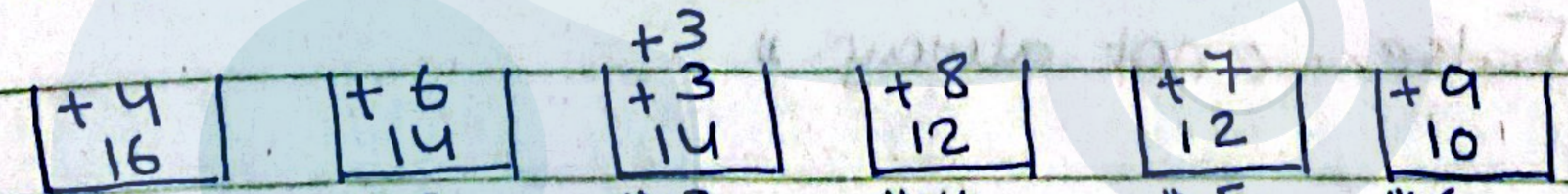
Sol. 1- $L = [16, 14, 14, 12, 12, 10, 9, 8, 7, 6, 4, 3, 3]$ ← Sorted
2- NF:



bin #1 #2 #3 #4 #5 #6 #7 #8
Capacity Left = 4 C.L=6 C.L=6 C.L=8 C.L=8 C.L=1 C.L=5 C.L=4
↳ 8 bins needed

b) apply FF - decreasing on the same list.

Sol.



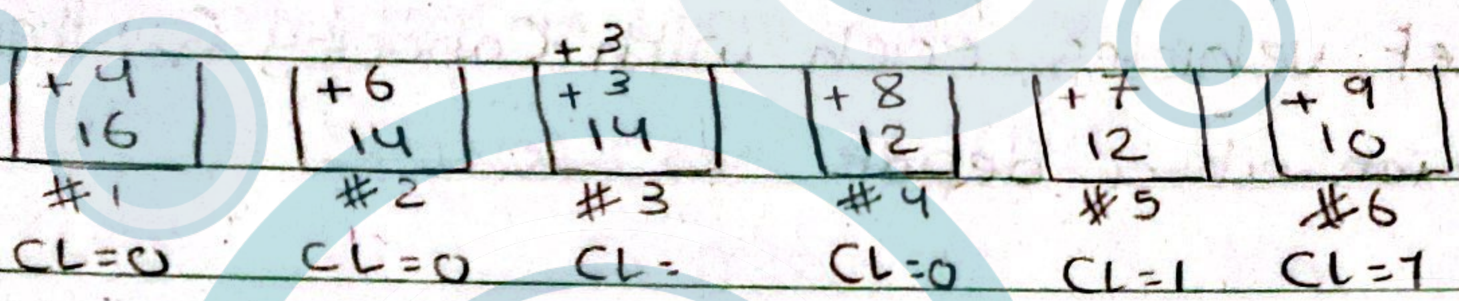
bin #1 #2 #3 #4 #5 #6
C.L=0 C.L=0 C.L=0 C.L=0 C.L=1 C.L=1
↳ 6 bins needed.

c)

Scanned with CamScanner

c) apply BFD → (best fit decreasing)

Sol.



改善
KAIZEN
TEAM

Scanned with CamScanner

note

- Performance for BF & FF decreasing is $\alpha = 1.5$
  ↳ this means that there is an adv. if having data already

- offline heu. in it's worst case will be equal to the online hey, not always better, but never worse.

- Bonus question :- → Bin Packing Problem
⇒ assume you solve BPP for optimality and your solution was 20 bins required, your friend applied FF-decreasing and solution was 35 bins required then :-

a) Your friend is wrong
b) Your friend is right
c) You don't have enough info. to judge.

Ans. [a] → Your friend is wrong.
↳ because at FFD worst case ans will be
opt * 1.5 → 20 * 1.5 = 30 #
لـ يعني مستحيل الـ FFD تعطيني رقم أكبر من 30 .

- True or False ?!
⇒ BF-heuristic always yeilds a sol. better than FF.
Ans. False. «not always»

Remember :
- BF → Best Fit → online heuristic
- BFD → Best Fit-decreasing → offline heuristic
⇒ and so for NF, FF.

Scanned with CamScanner

⊛ Vehicle Routing Problem « VRP »

- Description :
  - we have fleet vehicles, each with Capacity Q, these vehicles station at a depot.

  - Customers (n), where customer i has demand q_i

⊛ demand
- Delivery demand : order توصل ال
- Pickup demand : order تاخد ال

⊛ VRP « Basic » :

→ all orders are either delivered orders or Pickup orders.

→ All vehicles have homogenous Capacity Q. (Same Capacity)

- Goal of VRP :

To Find set of routes to serve all Customers with minimum distance or cost or time without violating vehicle Capacity

- Basic VRP is a combo between TSP & BPP.
  « Travel Saleman Problem & Bin Packing Problem »

- Every TSP is a vehicle routing Problem #
- TSP → 1 vehicle
- VRP → fleet of vehicles.

Scanned with CamScanner

Thursday
22/12

* • TSP is Special Case of VRP:

↝ when Capacity is ∞ or large enough to have all orders order in one vehicle → 1 tour → 1 tour = TSP #

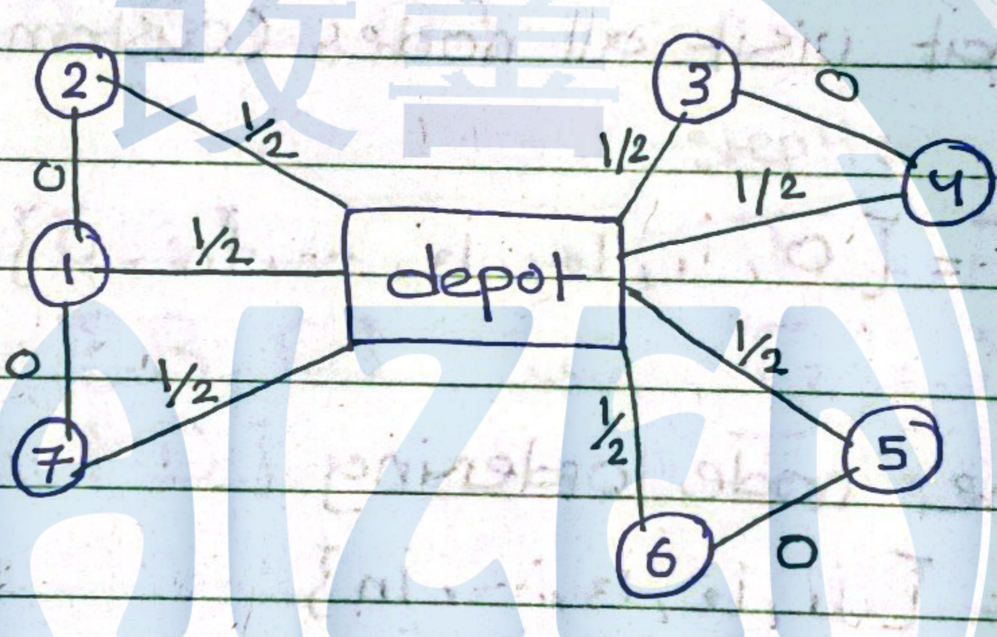
* • BPP is Special Case of VRP:

↝ BPP = → min. # of bins utilized.
VRP = → min distance / cost / time / ...

↳ From VRP to BPP:

↳ distances btw Customers = 0
↳ distance btw Customers & the depot = 1/2

هيك لو حليت VRP ال زي كأني حليت BPP



• VRP [just like TSP & BPP] is a hard-to-solve Problem which means it cannot be solved by optimality So we will use heuristic algorithms to solve it.

Scanned with CamScanner

⊛ VRP heuristics

⇒ 3 - heuristic algorithms.

[1] TSP Partitioning heu. (route 1st cluster 2nd)

steps

1) let „T" be an optimal or near-optimal TSP tour that visit all nodes (customers) & the depot.

↳ $T = \{0, i_1, i_2, i_3, \cdots, i_n, 0\}$

2) Define the node ordering

$T_0 = \{i_1, i_2, i_3, \cdots, i_n\}$ → Depot بس شلنا ال

3) Create Customer Clusters by running the NF bin Packing heuristic with Customer order (demand) q_i and vehicle Capacity Q on the Customer order defined by T_0.

4) Suppose the generated Clusters are

$\{i_1, i_2, \cdots, i_K\}$, $\{i_{K+1}, \cdots, i_L\}$, $\{i_{L+1}, \cdots, i_n\}$

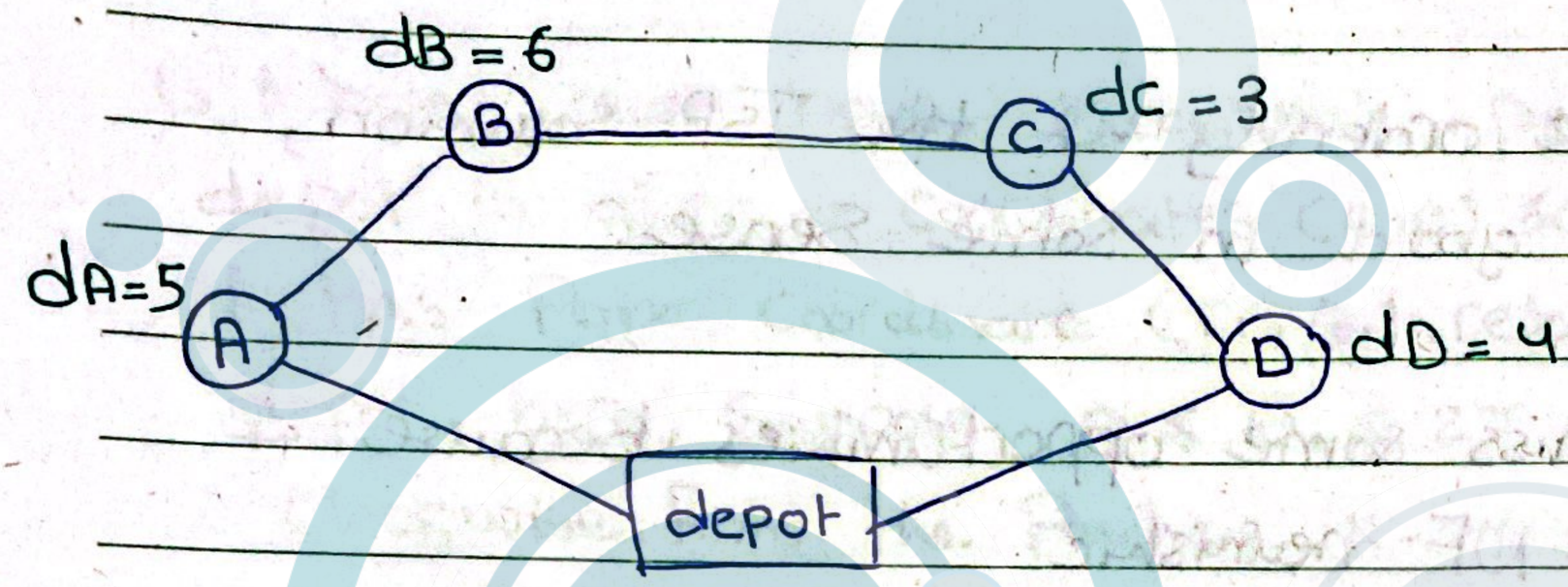
So the final VRP tours are given as :

$\{0, i_1, i_2, \cdots, i_K, 0\}$, $\{0, i_{K+1}, \cdots, i_L, 0\}$, etc...

↳ the order of Customers in the VRP tours is identical to the order in the original TSP tour #.

Scanned with CamScanner

example



, Q = 10 units

Sol.

step 1 : $\overline{T} = \{ depot, A, B, C, D, depot \}$

step 2 : $\overline{T_0} = \{ A, B, C, D \}$

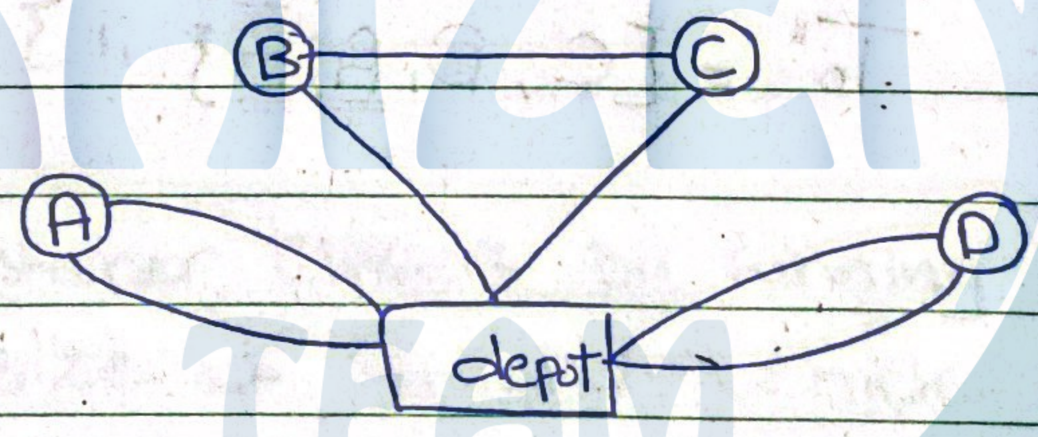
step 3 : NF

step 4:

Cluster 1 = {A} ↳ $T_1 = \{ 0, A, 0 \}$

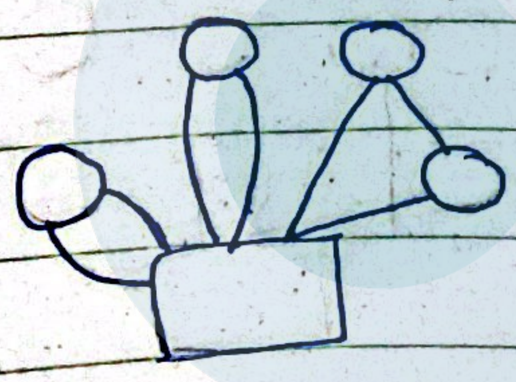
Cluster 2 = {B, C} ↳ $T_2 = \{ 0, B, C, 0 \}$

Cluster 3 = {D} ↳ $T_3 = \{ 0, D, 0 \}$

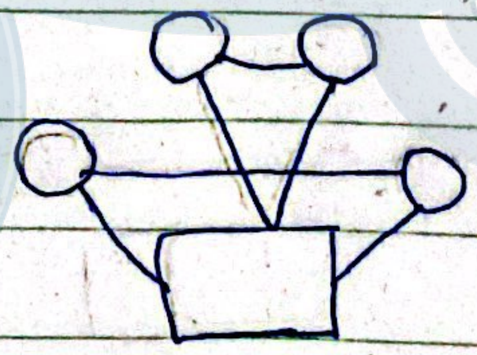


↳ 3 vehicles

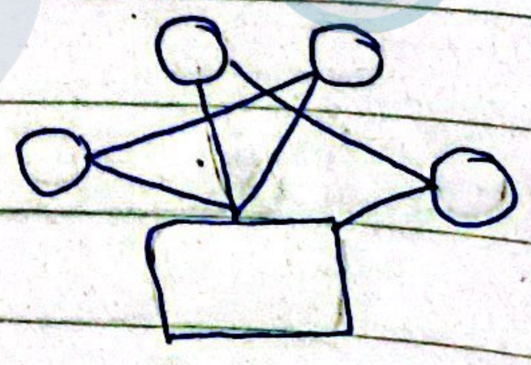
↳ other Possibilities if we didn't apply NF:



3 vehicles



2 vehicles



2 vehicles.

Scanned with CamScanner

* Properties of (route 1st cluster 2nd) heu. :-

① It Preserve ordering of the TSP solution, which is good in some sense

② It may miss some opportunities because of using the NF heuristic

③ It can be good as starting point for improvement heuristic

④ It Can be easily modified
↳ by changing the order of To

ex. ↳ $T_0 = [A, B, C, D]$
$T_0' = [B, C, D, A]$
$T_0'' = [C, D, A, B]$

Scanned with CamScanner

Tuesday
3/1

[2] Sweep heuristic : Cluster 1st, Route 2nd

↳ Nodes must have geographic positions., Suppose depot is given a Coordinates (0,0) & each Customer i has Polar Coordinate (r_i, θ_i) relative to the depot. here, Suppose $0 \leq \theta_i \leq 2\pi$ is measured clockwise from the Positive x-axis.

Steps

[1] Choose a start ray, with Polar angle θ

[2] For each Customer, Calculate Δ_i as a relative angle between ray and Customer location, measured C.W.

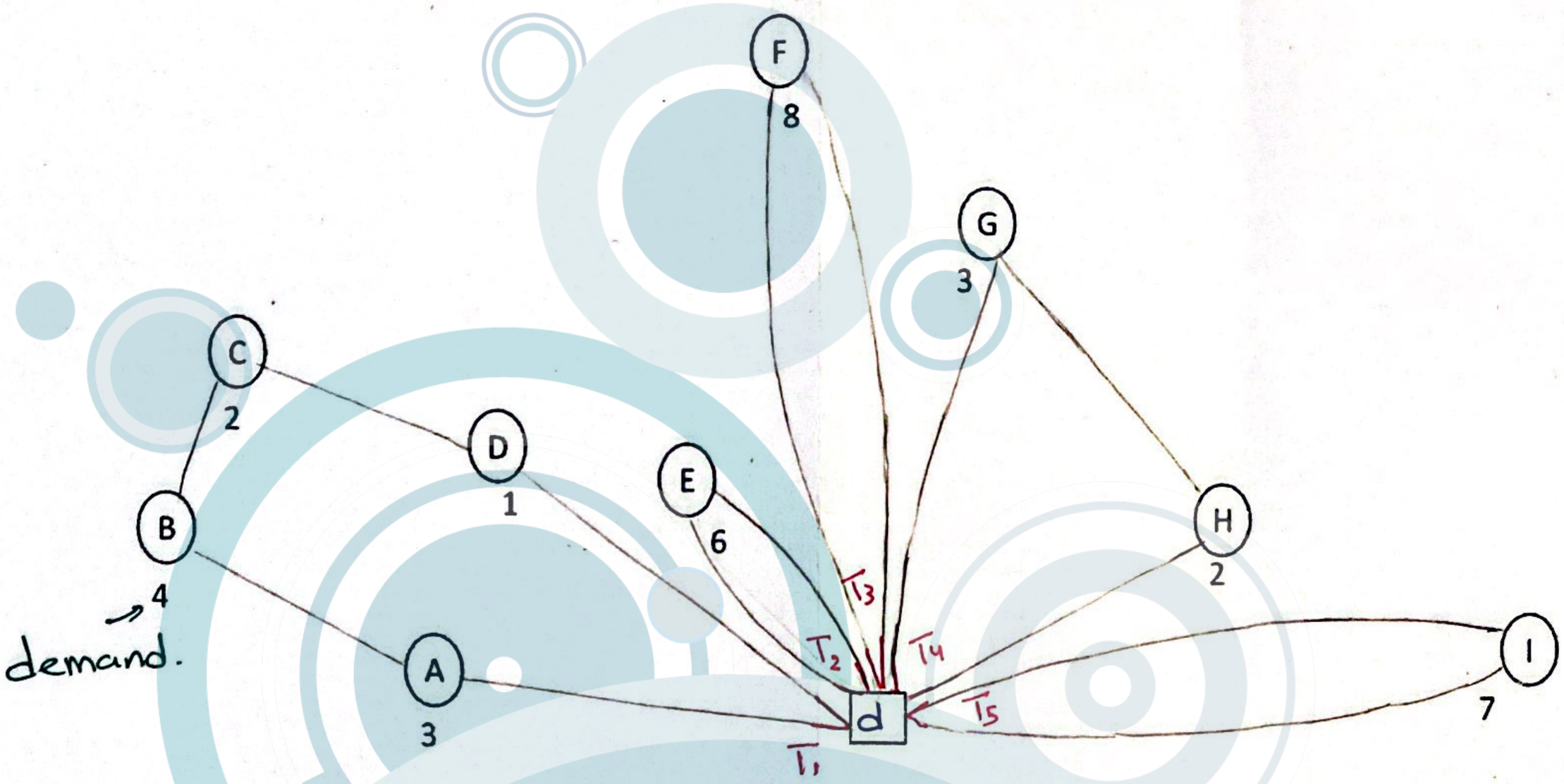
[3] Sort the Customers in order of non-decreasing Δ_i, Suppose the order created is
$T_o = \{i_1, i_2, i_3, \dots, i_n\}$.

[4] Create Customer Clusters by running NF heuristic with lot size of q_i and vehicle Capacity of Q on the customer order defined by T_o.

[5] Create Final VRP by generating a Separate TSP tour For each Cluster and the depot.

⊕ 2 examples

Scanned with CamScanner



- **Prob.:** Using the Sweep heu., generate Clusters of Customers each to be served by a Single vehicle. Begin with ray through Customer A, and rotate your ray clockwise. ($Q = 10$).

- **Sol.**

$T_0 = \{A, B, C, D, E, F, G, H, I\}$

NF (Next Fit heu. application)

Cluster 1 = {A, B, C, D} → $T_1 = \{0, A, B, C, D, 0\}$

= 2 = {E} → $T_2 = \{0, E, 0\}$

= 3 = {F} → $T_3 = \{0, F, 0\}$

= 4 = {G, H} → $T_4 = \{0, G, H, 0\}$

= 5 = {I} → $T_5 = \{0, I, 0\}$ #

- لو أعطانا السؤال Distances بنحل TSP.

Scanned with CamScanner

2

Suppose you are the dispatcher for an LTL trucking firm, and you need to dispatch small delivery trucks for the local delivery of freight. Suppose you have many trucks available, each with the capacity to haul 10 tons of freight. Suppose that the polar angle (relative to the x-axis, and measured clockwise in degrees) of the seven customers are {45, 105, 220, 135, 60, 85, 300}. Suppose that the demands of the seven customers in tons are {7, 3, 4, 5, 2, 4, 6}.

Using the Sweep Heuristic, generate clusters of customers each to be served by a single vehicle. Begin with a ray through customer 1, and rotate your ray clockwise

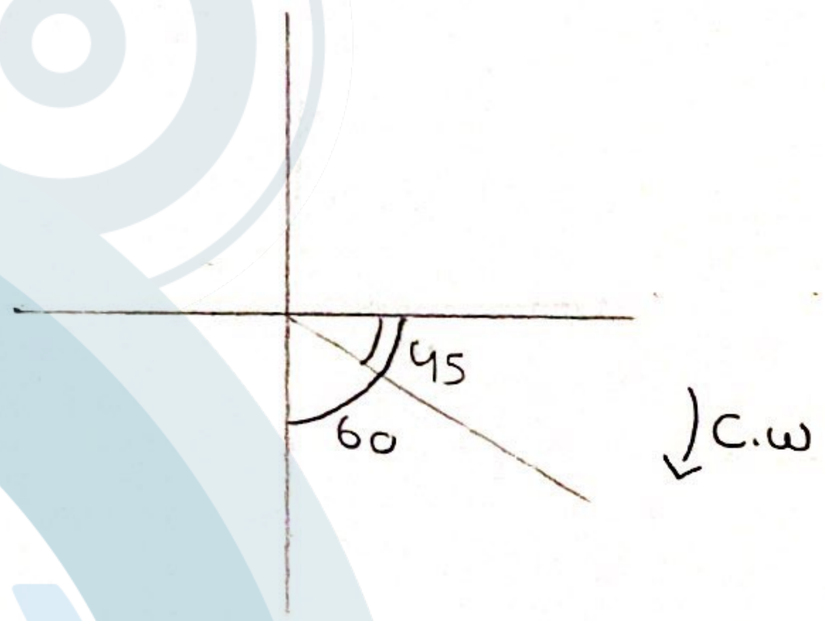
↳ given:

- Polar angles of the 7 customers

  {45 , 105 , 220 , 135 , 60 , 85 , 300}
  C_1 C_2 C_3 C_4 C_5 C_6 C_7

- Customers demands

  {7 , 3 , 4 , 5 , 2 , 4 , 6}
  C_1 C_2 C_3 C_4 C_5 C_6 C_7



Sol.

$T_0 = \{C_1, C_5, C_6, C_2, C_4, C_3, C_7\}$

NF Cluster 1 = $\{C_1, C_5\}$ → $T_1 = [0, C_1, C_5, 0]$

= 2 = $[C_6, C_2]$ → $T_2 = [0, C_6, C_2, 0]$

= 3 = $[C_4, C_3]$ → $T_3 = [0, C_4, C_3, 0]$

= 4 = $[C_7]$ → $T_4 = [0, C_7, 0]$.

#

Scanned with CamScanner

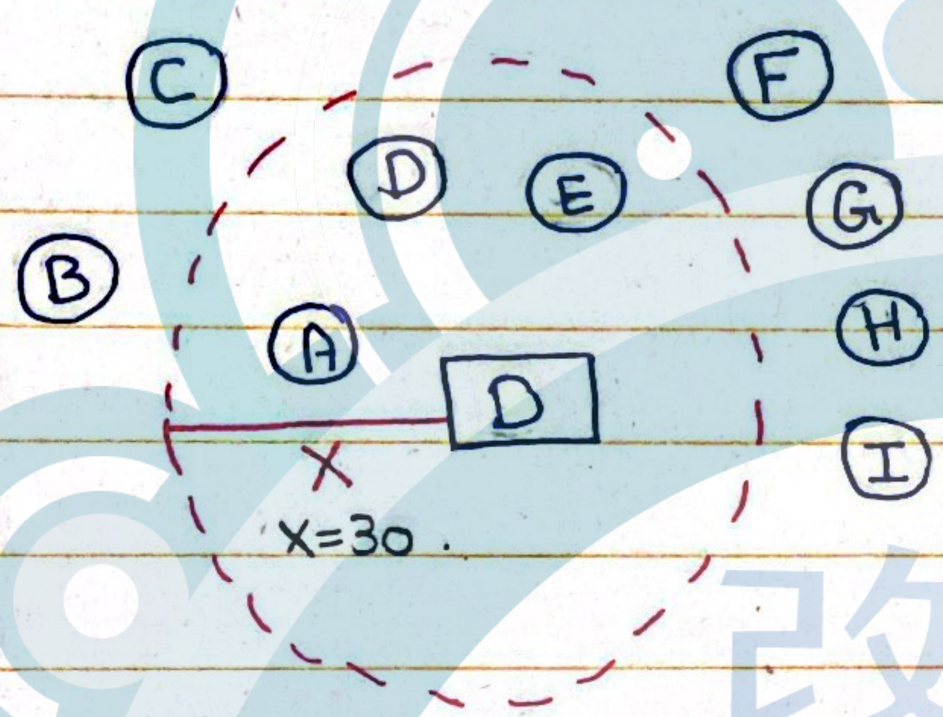
⊛ Modification layered sweep.

- idea : Setup regions based on radius from depot to Customer.

then ↳ Run sweep in each region.

↳ radius « X »

لـ بنرسم دائرة من عند الـ depot بـ radius given وبحدد جواها الـ Customers اللي جزء الـ region ومين برّاها.



↳ {A, D, E} → Close to the depot → Cluster 1

↳ {B, C, F, G, H, I} → far from the depot.

↳ region كل Clustering لـ بعملها لحالها

NF Cluster 1 = {A, D, E}

" 2 = {B, C}

" 3 = {F}

" 4 = {G, H}

" 5 = {I}

- # of Clusters = # of vehicles = # of tours.

Scanned with CamScanner

Sunday
8/1

↗ 3rd VRP heu.

[3] Clarke-Wright Savings Heuristic

idea : The Clarke-Wright method [begins with a poor initial solution in which each vehicle visits a single customer.] Tours are then combined as long as total cost savings are produced. Suppose the depot is node 0.

steps :

1. For each node $i \in N$, construct a tour $T_i = \{i\}$ beginning at the depot, traveling to node i, and returning to the depot. Let S be the set of tours.

2. Calculate the savings $S_{ij} = c_{0i} + c_{j0} - c_{ij}$ for all pairs of nodes i, j.

3. Sort the savings in nonincreasing order.

4. Walk down the savings combination list, and find the next feasible combination pair (i, j) such that:

   (a) i and j are on different tours, and i is the last node on its tour T_k and j is the first node on its tour T_ℓ.

   ← Capacity (b) $\sum_{k \in T_i \cup T_j} q_k \leq Q$.

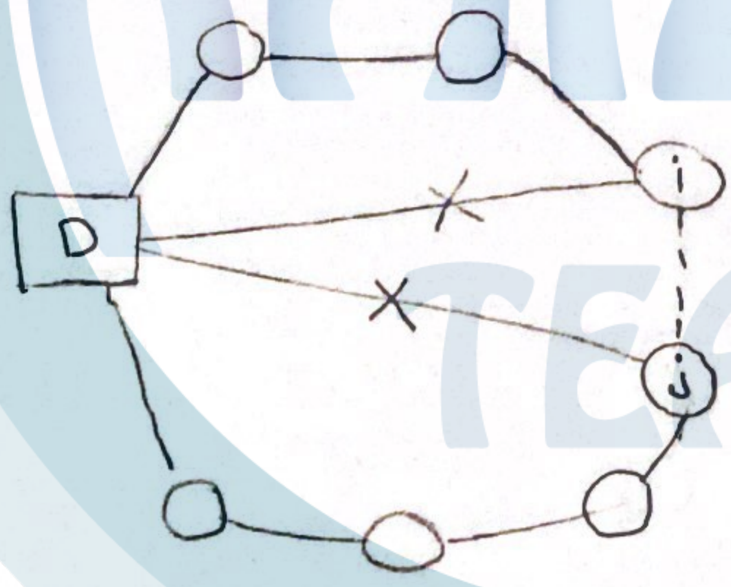
   ← Saving, LP (c) $S_{ij} \geq 0$

⇒ Add the combined tour $T_i + T_j$ to S, and remove T_i and T_j. Repeat this step as long as feasible combinations remain in the savings list.

arcs removed arc added

- Savings → $S_{ij} = C_{0i} + C_{j0} - C_{ij}$

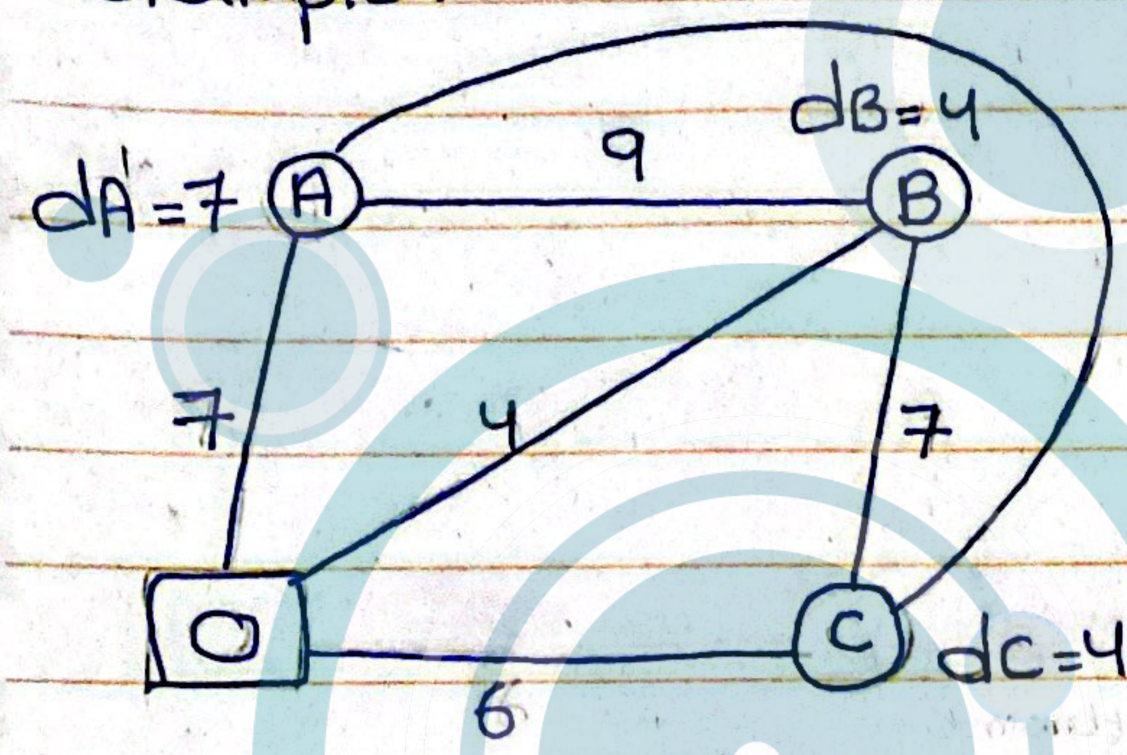
⇒



+ example

Scanned with CamScanner

example 2-



Q = 70 (truck Capacity)
d = demand
node = Customer.

↳ apply clarke - wright heuristic

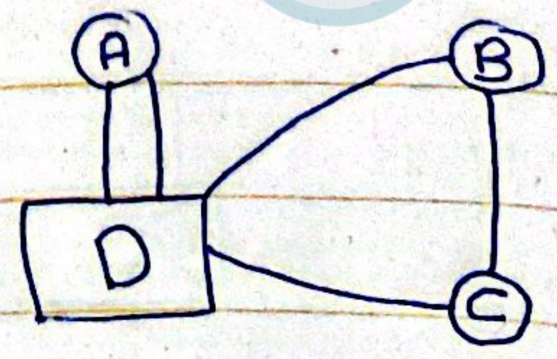
Sol :-

- $T_A = \{0, A, 0\}$
  $T_B = \{0, B, 0\}$
  $T_C = \{0, C, 0\}$] ↳ our initial Poor Solution.
  ↳ باي هو انه بكل Tour بنطلع من ال depot لعند Customer واحد بس ونرجع.

- $S_{AB} = C_{OA} + C_{OB} - C_{AB} = 2$
  $S_{AC} = C_{OA} + C_{OC} - C_{AC} = 3$
  $S_{BC} = C_{OB} + C_{OC} - C_{BC} = 3$] Savings for all Combinations.

[we will choose either BC or AC, depending on Capacity,
we can't Combine A&B because $d_A + d_B > 70$
= Can only Combine B&C → $d_B + d_C < 70$]

- $T_A = \{0, A, 0\}$
  $T_{BC} = \{0, B, C, 0\}$ →
  ↳ Cost = 31 #



Scanned with CamScanner

⊛ other improvements approaches for VRP

- Any improvement approach that we apply for Tsp Solution is also applicable for VRP Solution.

- Improvement heuristics for VRP :-

[I] Insert (i, j)

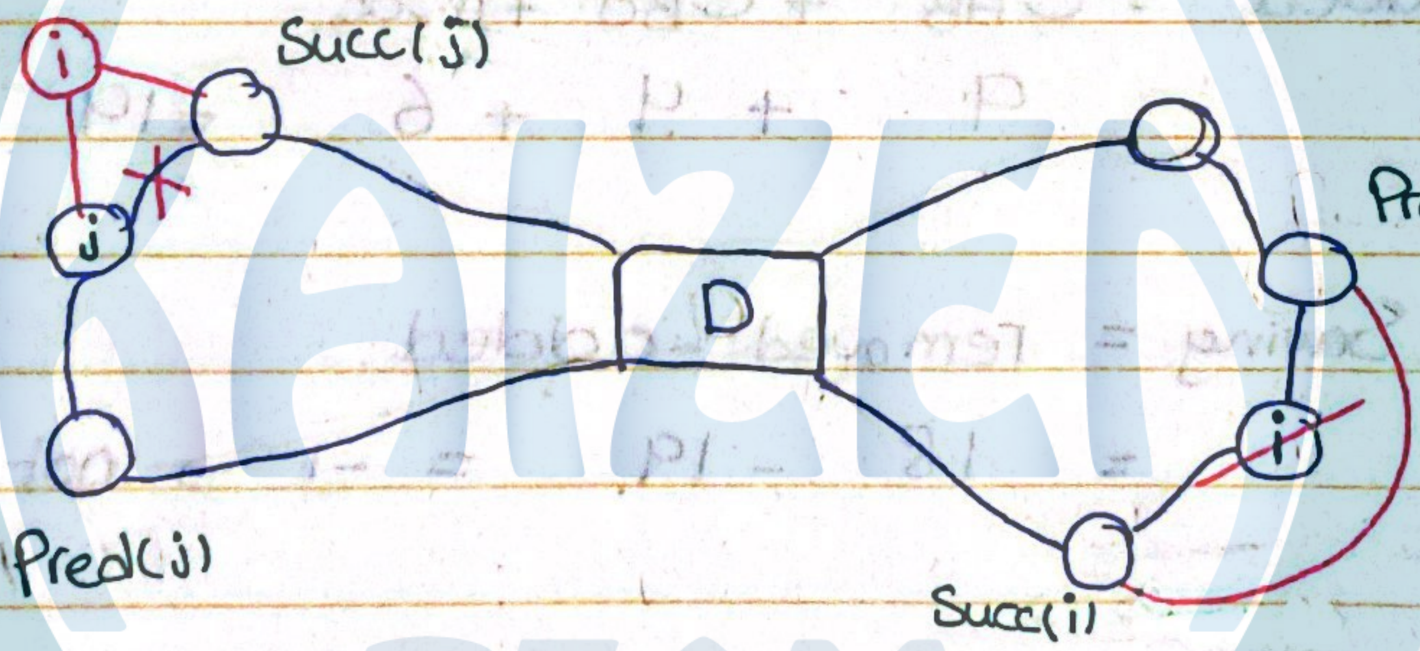
Steps 1) Remove customer (i) from the current tour.

2) Insert customer (i) between j and Succ(j)

↳ Successor

↳ only applicable if satisfies Capacity Constraint.

& Saving



$$\text{Cost Removed} = C_{pred(i),i} + C_{i,succ(i)} + C_{_,succ(j)}$$

$$\text{Cost Added} = C_{pred(i),succ(i)} + C_{ji} + C_{i,succ(j)}$$

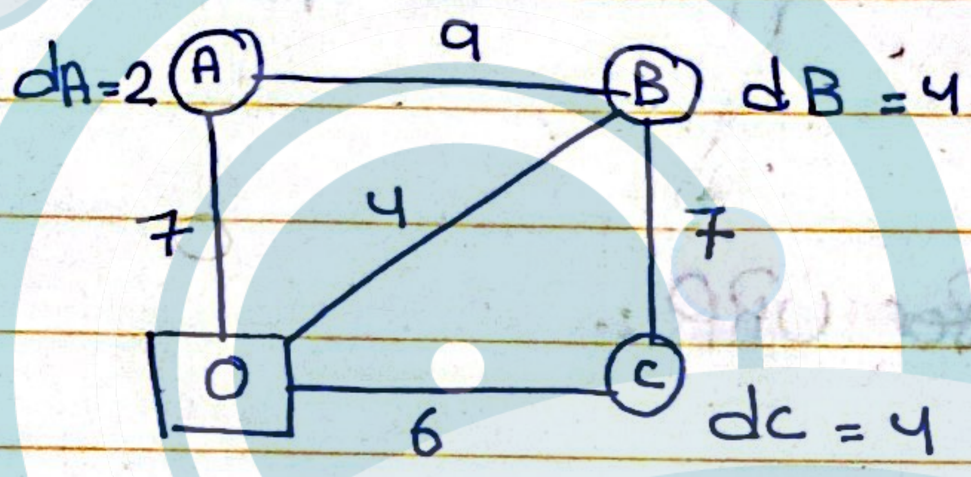
$$\text{Saving} = \text{Cost removed} - \text{Cost added.}$$

- Pred = Predecessor
  Succ = Successor

Scanned with CamScanner

example:

use insert heuristic to calculate Possible Saving for inserting Customer B to tour A, between A & Depot.



Sol.

Cost removed = $C_{OB} + C_{BC} + C_{OA}$

$4 + 7 + 7 = 18$

Cost added = $C_{AB} + C_{BO} + C_{OC}$

$9 + 4 + 6 = 19$

Saving = removed - added

$= 18 - 19 = -1$ → not possible #

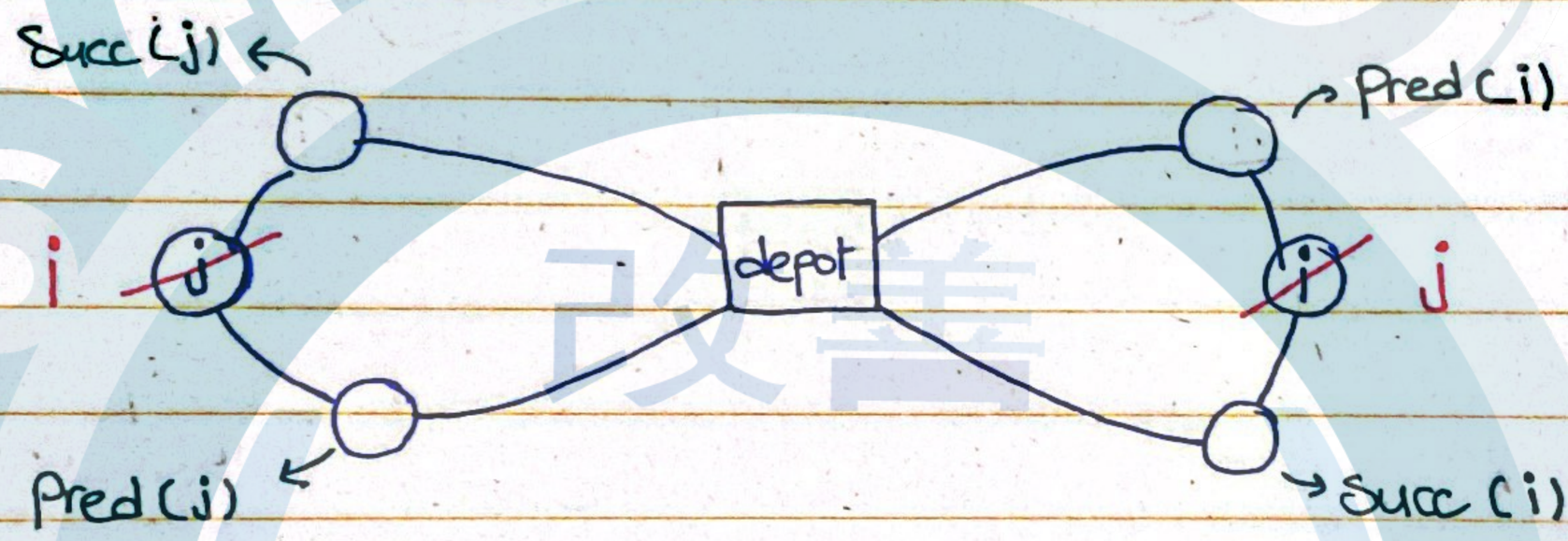
Scanned with CamScanner

[2] Swap (i, j)

↳ Move customer i to customer j's position, and j to customer i's position

↳ must satisfy Capacity Constraint

↳ Saving :-



$$\text{Cost Removed} = C_{pred(i),i} + C_{i,succ(i)} + C_{pred(j),j} + C_{j,succ(j)}$$

$$\text{Cost Added} = C_{pred(i),j} + C_{j,succ(i)} + C_{pred(j),i} + C_{i,succ(j)}$$

$$\text{Saving} = \text{Cost Removed} - \text{Cost Added}$$

#

Done.

Scanned with CamScanner