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

Plant
SS = 0
10 units/day
$500/unit

warehouse
SS = 100
demand =
10 unit/day
value = 512/unit

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.

ussulally decision is for one year
الا اذا ذكر السؤال غير ذلك

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

	Rail	Truck
Capacity	400 unit/Railcar	100 unit/truck
Cost	$1,200/car Load	$700/truckLoad
Transit time	20 days	3 days

Inventory cost:

بتساعدني احسب ال opportunity cost (زي استثمار)

Value of items at plant: $500 / unit

Value of items at warehouse: $512 / unit

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

مثلاً 25% x cost

Plant space cost: $20 / unit*year

$q_{optimal} \times 20/\text{unit} \times \text{year}$

Warehouse space cost: $25 / unit*year

plant

SS = 0

مدري D و P = 10 unit/day

value of item = $500/unit

holding cost Rate = 25%/year

plant space cost = $20/unit year

warehouse

SS = 100

d = 10 unit/day

value of item = $ 512/unit

plant space cost = $25/unit year

holding cost Rate = 25 %/year

inventory cost analysis
→ we have 3 sepatate inventory costs
→ inventory held at the origin

$q_{optimal} \to q \neq q$
Rail truck

$q = 600 \to 400$, $\to 200$

ناقص
عن
عوزته
17/10

المطلوب
(cost per / year)

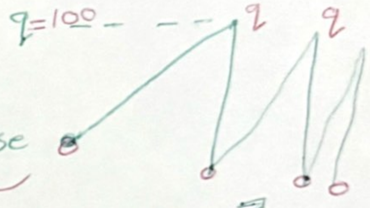
Goal :

- 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 $q \leq$ capacity

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?



تضاريس
زي الرسمة
Sawteeth

step ① (inventory space cost)
* the maximum accumulation of item at the plant is q
therefor space cost is given by inventory space cost $= \frac{\$20\ q}{\text{unit year}}$
$q = \$20\ q/\text{year}$

~~the maximum accumulation of~~

step ② (holding cost ($/year))
inventory holding cost = inventory holding cost rate x value of item stored at the plant

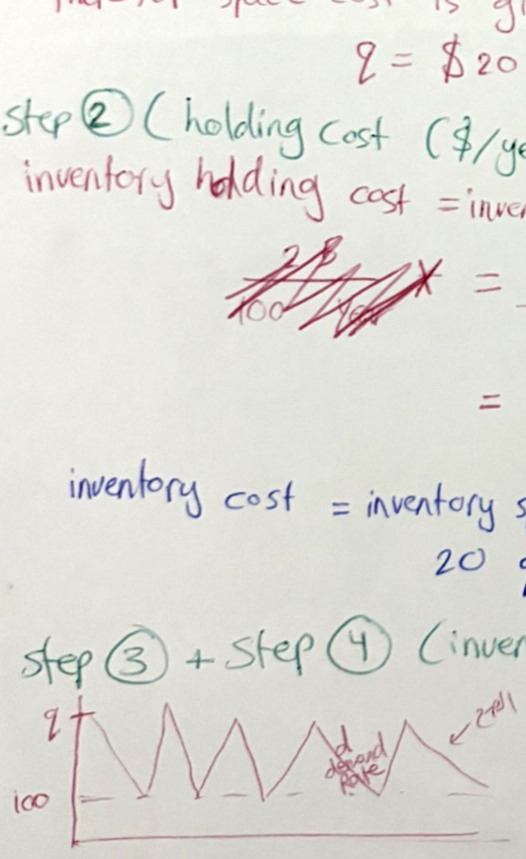
$$= \frac{25\%}{\text{year}} \times \left(\frac{\$500}{\text{unit}} \cdot \frac{q}{2} \text{ unit} \right)$$

$$= \$\ 62.5q/\text{year}$$

inventory cost = inventory space cost + inventory holding cost

$$20\ q + 62.5\ q = \$\ 82.5\ q/\text{year}$$

step ③ + step ④ (inventory cost at the Destination (warehouse) في السؤال



Space Rate cost $= \frac{\$25}{\text{unit year}} \cdot (q + 100)$
$\$25\ q + \$2,500$ /year

(demand Rate = Production Rate)

inventory holding cost $= \frac{25}{100\ \text{year}} \times \frac{\$512}{\text{unit}} \times \left(\frac{q}{2} + 100 \right) \text{unit} = 64\ q + 12800$

total inventory cost $= 25\ q + 2500 + 64\ q + 12\ 800 = 89\ q + 1530$

Dijkstra's algorithm

Greedy

- 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:
            • $v(j) \leftarrow v(p) + c_{pj}$
            • $pred(j) \leftarrow p$
```

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.

31/10/2024

Example # 1: Given the graph in the Figure shown below, find the shortest paths to all nodes using the vertex A as the starting source. Use Dijkstra's algorithm.

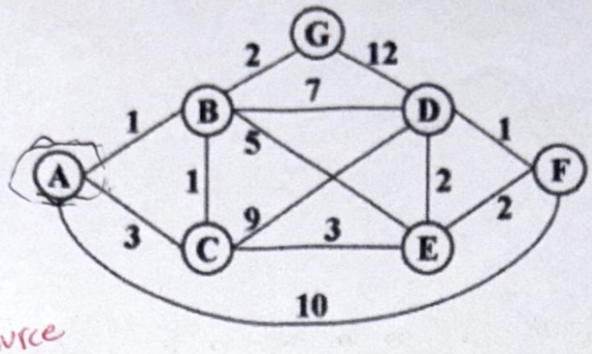
Advantange of Dijkstra

→ we Know number of iteration

(once visited we don't go bac

iterations → n + 1

(عدّ في iteration صفر)



source

current node

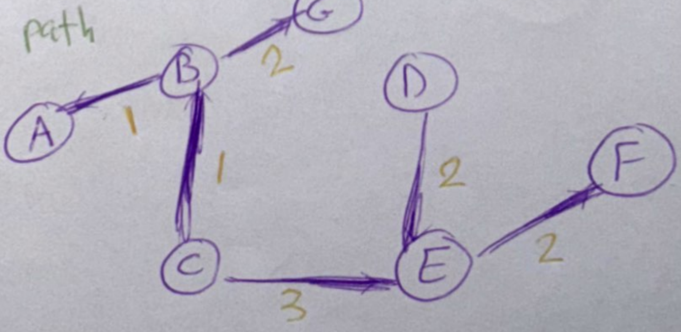
Iteration	A	B	C	D	E	F	G	P	PERM	$\overline{PERM}$
0	0(0)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	∞(-1)	—	—	—
1	0(0)*	1(A)	3(B)	∞(-1) ~~8(B)~~	∞(-1) ~~6(B)~~	10(A)	∞(-1 ~~3(B)~~	A	A	N/{A}
2	—	1(A)*	2(B)	8(B)	6B	10 A	3 B	B	A, B	N/{A, B}
3	—	—	2(B)*	8(B)	5C	10 A	3(B)	C	A, B, C	D, E, F, G
4	—	—	—	8(E)	5C*	10A	3(B)*	G	A, B, C, G	D, E, F
5	—	—	—	7E	5C*	7E	—	E	N/{D, F}	D, F
6	—	—	—	7E*	—	7E	—	D	N/{F}	F
7	—	—	—	—	—	7(E)*	—	F	N	∅

اذا كنت مهتمة فقط ب G بمجرد ما اوصلها بقدر اوقف

*what is the min cost path from A to F ?

P = {(A,B),(B,C) (C,E),(E,F)}
Path

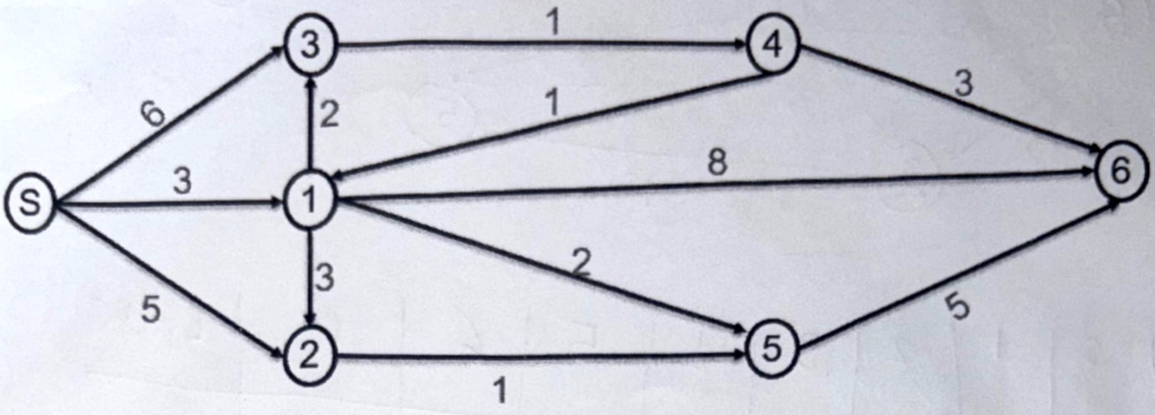
cost=7



ARCS = n-1
7-1=6

undirected so no Arrows

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										
1										
2										
3										
4										
5										
6										
7										

Bellman-Ford algorithm

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

Initialization:

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

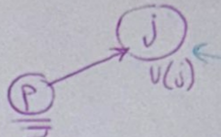
$$pred(j) \leftarrow \begin{cases} 0 & j = s \\ -1 & \text{otherwise} \end{cases}$$

$LIST = \{s\}$

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 $j \notin LIST$, $LIST \leftarrow LIST + \{j\}$
```

Rules modification
if j ∉ List
↳ if j has never been in the List

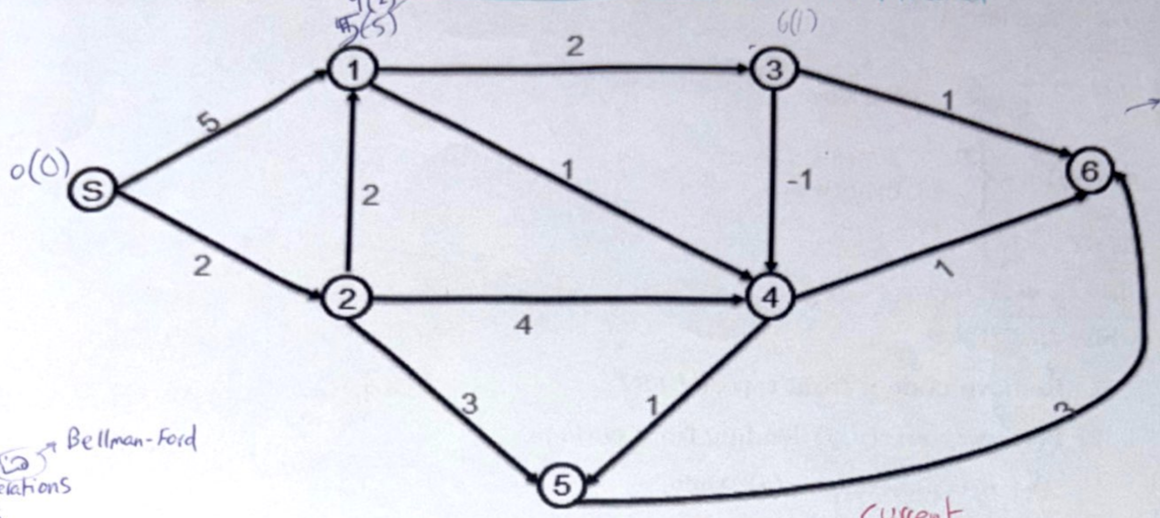


$v(P) + C_{Pj} < v(j)$

$v(j) \leftarrow v(P) + C_{Pj}$

pred (i) = P

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. → Just for Directed



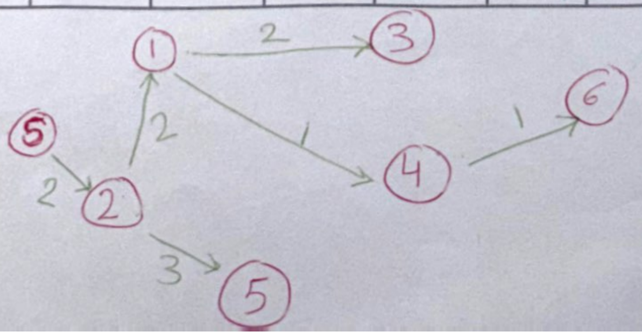
Bellman-Ford ← (هنا) * of iterations
غير معروف

Iteration	s	1	2	3	4	5	6	P (Current)	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 → in the List because they never been in the list
3	0(0)	4(2)	2(s)*	7(1)	6(1)	5(2)	∞(-1)	2	1, 3, 4, 5 (بالبداية) لأنها تغيرت
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	∅

we take it from the top of the list

No Neighbor

Minimum Cost Path Tree



*What is the Minimum cost path from S to 6?

↳ P = {(S,2), (2,1) (1,4) (4,6)} = 6

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:

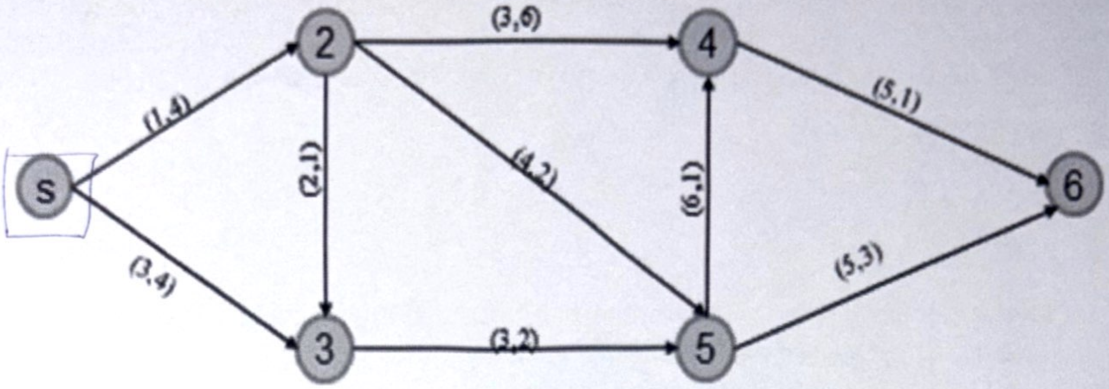
while $LIST \neq \emptyset$:

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 $(d_r, t_r) = (d_k + d_{pj}, t_k + t_{pj})$
   - (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.

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	$(0,0)^{s}_{1}$	$(1,4)^{2}_{2}$ Pred(2) = 1	$(3,4)^{3}_{3}$ Pred(3) = 1 $(3,5)^{3}_{4}$ X Pred(4) = 2	$(\overset{4}{\cancel{5}},\overset{10}{\cancel{6}})^{4}_{5}$ Pred(5) = 2 $(11,7)^{4}_{9}$ Pred(9) = 6	$(5,6)^{5}_{6}$ Pred(6) = 2 $(6,6)^{5}_{7}$ X Pred(7) = 3	$(9,11)^{6}_{8}$ Pred(8) = 5 $(16,10)^{6}_{10}$ Pred(10) = 9	~~S$(0,0)^{s}_{1}$~~ ~~$(1,4)^{2}_{2}$~~ ~~$(3,4)^{3}_{3}$~~ $(3,5)^{3}_{4}$ X ~~$(4,10)^{4}_{5}$~~ ~~$(5,6)^{5}_{6}$~~ $(6,6)^{5}_{7}$ X ~~$(9,11)^{6}_{8}$~~ $(11,7)^{4}_{9}$ $(16,10)^{6}_{10}$

Minimum Cost Paths

Question .

Suppose you have gone to the grocery store and stocked up with a week of groceries. You would like to make it home in a reasonable amount of time without traveling too far so that your ice cream does not melt. The grocery store will be your starting location, referred to as *s*, and your house will be the final destination, referred to as *t*. The table below shows all connections in the transportation network you will use to travel home (if a connection is not in the table, that arc does not exist in the network). The table also gives the distance and time between any pair of nodes for which a connection exist.

Connection	Distance	Time
(s,1)	4	1
(s,2)	3	1
(1,3)	4	2
(1,4)	1	3
(2,3)	2	4
(2,4)	3	2
(3,t)	2	2
(4,t)	2	2

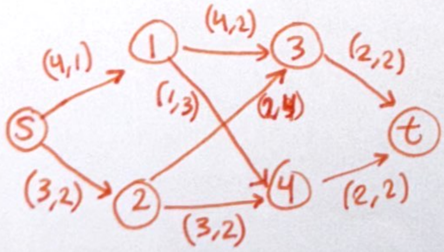
(a) Draw the network, labeling each node with the appropriate node number $\{s,t,1,2,3,4\}$ and labeling each arc with the cost (d_{ij},t_{ij}) where d_{ij} is the distance of the connection and t_{ij} is the time of the connection.

(b) Use the **multi-label algorithm** to find the minimum cost paths from the grocery store *s* to your home *t*, where the two types of costs being considered are described below. Show your work in the form of a table as we did in class.

i. Find the minimum distance path from *s* to *t*.

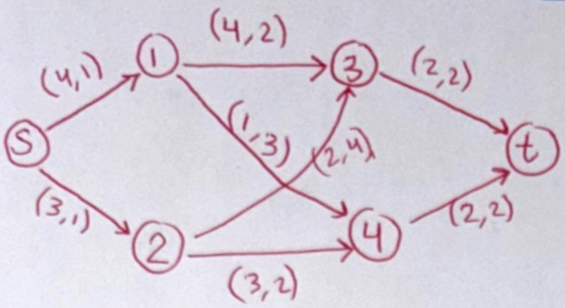
ii. Find the path from *s* to *t* that minimizes the total time.

if it is the same Question
and it asked for one objective only
we can use dijkstra
or bellman
اذا مو محدد طريقة



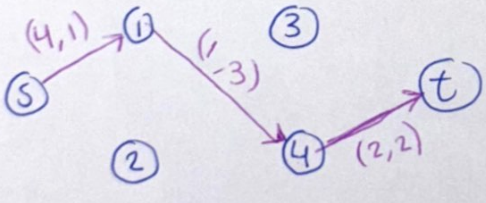
Example (2)

Multi-Label Algorithm



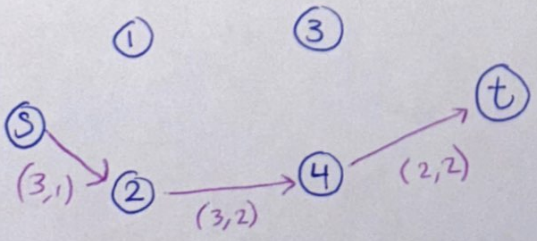
LABELS	S	1	2	3	4	t	List
	$(0,0)^s_1$ Pred(1)=0	$(4,1)^1_2$ Pred(2)=1	$(3,1)^2_3$ Pred(3)=1	$(8,3)^3_4$ Pred(4)= 2	$(5,4)^4_5$ Pred(5)=2	$(10,5)^t_8$ Pred(8)= 4	$(0,0)^s_1$ X $(4,1)^1_2$ X $(3,1)^2_3$ X
				$(5,5)^3_6$ Pred(6)= 3	$(6,3)^4_7$ Pred(7)=3	$(7,6)^t_9$ Pred(9) = 5	$(8,3)^3_4$ X $(5,4)^4_5$ X $(5,5)^3_6$ X
				No Domination	No Domination	~~$(7,7)^t_{10}$ Pred(10)=6~~ X	$(6,3)^4_7$ X $(10,5)^t_8$ X
						$(8,5)^t_{11}$ Pred(11)=7	$(7,6)^t_9$ X $(7,7)^t_{10}$ X $(8,5)^t_{11}$ X ϕ

* Draw the Minimum distance path from (s to t)



$(7,6)^t_9$
Pred (9) = 5

Draw the Minimum total time path from (s to t)



$(8,5)^t_{11}$
Pred (11) = 7

Lower Bound

A simple lower bound for the TSP

Often, we generate lower bounds using a decomposition strategy. For example, suppose a feasible solution to a problem can be decomposed into two components. If these two components are optimized separately, the resulting solution may not be feasible for the original problem, but the resulting cost is a lower bound on the optimal cost of the original problem.

Consider any feasible solution to the TSP, that is a tour visiting all nodes in G. Suppose $TOUR$ is a set containing all of the arcs used in the tour; clearly $|TOUR| = n$. If one arc (i, j) from the tour is removed, the remaining structure is a $TREE$ on G; all nodes are connected and the $TREE$ contains $n-1$ arcs. In set terminology, we could write this decomposition as:

$$TOUR = TREE + \{(i, j)\}$$

The cost could be written as follows:

$$C(TOUR) = C(TREE) + c_{ij}$$

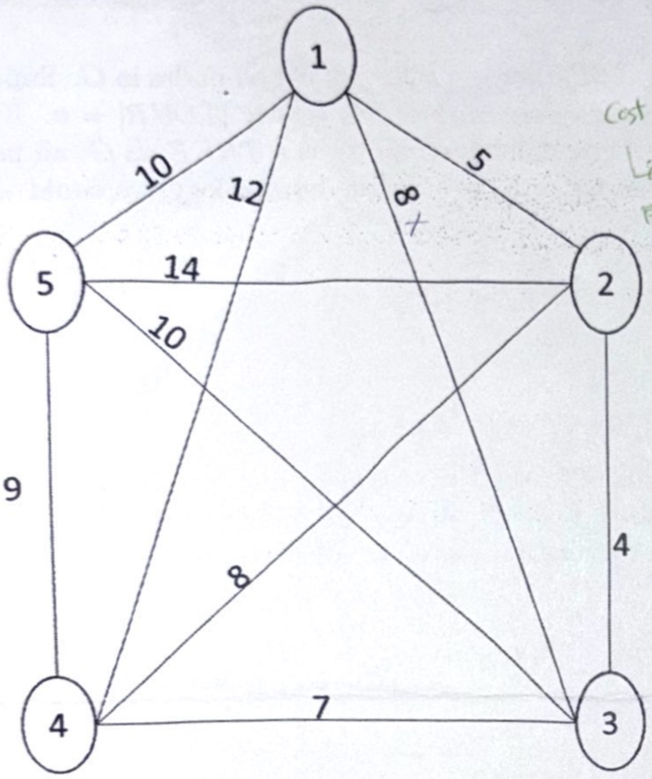
Now let's minimize the decomposition. Clearly, the minimum cost tree in a network is given by the MST. And, in general, (i, j) could be any arc in the network, but more specifically is some arc that is not in the TREE. Therefore, a lower bound on the optimal TSP cost is the following:

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

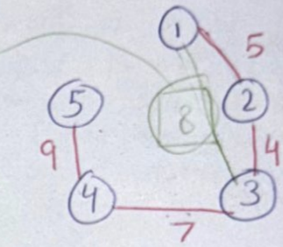
where c_{ij}^* is minimum cost arc not in MST.

يشبه سؤال سنوات للسكنز

Example 1: 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.
(not the optimal one)



MST by Prim
لأنها اسهل واسرع



ضفنا اقل cost
عشان نحولها
ذي ~~[illegible]~~
Lower
band

number of Acsc = n-1
= 5-1 = 4
اذا نوقف

C(MST) = 25

LB = C(MST) + C_{ij}

LB = 25 + 8 = 33

Complete the following table that shows the lower bounds using several techniques:

technique	Lower bound
MST+arc	33
$1-TREE_1$	33
$1-TREE_2$	33
$1-TREE_3$	33
$1-TREE_4$	34
$1-TREE_5$	35

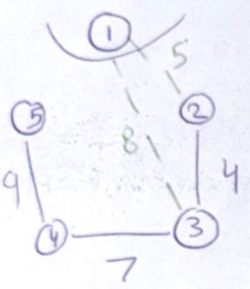
the optimal because it gave me a tour

the more usefull is the highest LB and (it gave me a tour)
يعني احسن واحسن

in a Minimization Problem → higher lower bound is a closer optimal (more useful)

Extra page:

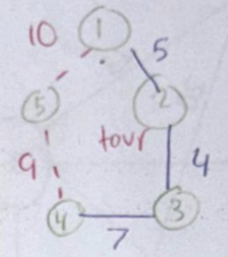
عدد ال N = عدد ال LB



$C(1\text{-Tree}_1) = 20$

$LB = C(1\text{-Tree}) + (i,j)^* + (i,k)^*$

$= 20 + C_{12} + C_{13}$



$C(1\text{-Tree}_5) = 16$

$LB = C(1\text{-Tree}_5) + C_{54} + C_{51}$

$= 16 + 9 + 10$

$= 16 + 19$

$= 35$

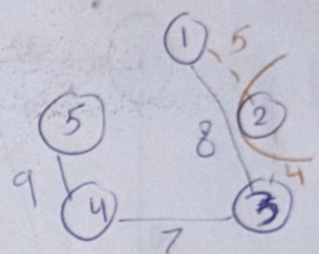
optimal $\geq LB$

$C(T^*) \geq 33$

≥ 33

≥ 35

1 - Tree 2



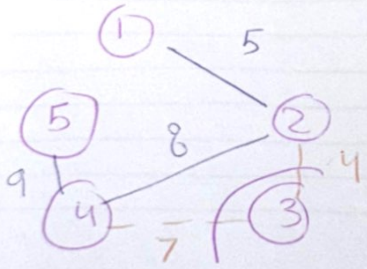
$C(1\text{-Tree}_2) = 24$

$LB = C(1\text{-Tree}_2)$

$= 24 + (4+5)$

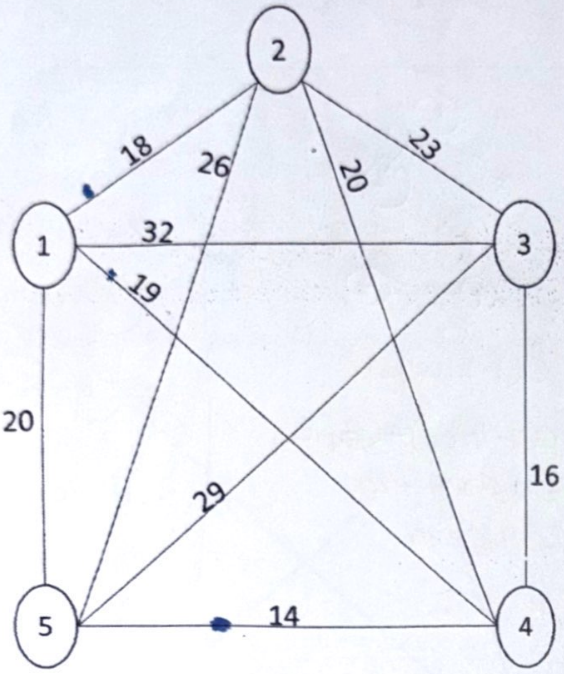
$= 24 + 9 = 33$

1 - Tree 3

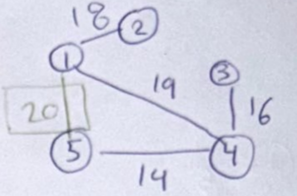


$C(1\text{-Tree}_3) = 22 + 11 = 33$

Example 2: 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.



For Lower bound



C(MST) = 67

LB = (CMST) + Cij

= 67 + 20 = 87

Complete the following table that shows the lower bounds using several techniques:

technique	Lower bound
MST+arc	87
$1-TREE_1$	87
$1-TREE_2$	87
$1-TREE_3$	90
$1-TREE_4$	91 ←
$1-TREE_5$	87

Twice around MST Heuristic

The MST 2-approximation heuristic will generate a TSP tour with a cost no greater than double the optimal cost for networks satisfying the triangle inequality. This is not good, but at least it's a guarantee!

An α-approximation heuristic: MST 2-approximation

Steps:

1. Generate a minimum spanning tree MST^* of the nodes N.
2. Let $LIST$ contain two copies of each arc in the minimum spanning tree, MST^*.
3. Generate a $WALK$ of nodes of G using each arc in $LIST$ exactly once; $\{v_1, v_2, ..., v_{2(n-1)}, v_1\}$. Note that v_i are not unique.
4. Create a heuristic tour T^H by following the nodes in the order specified by $WALK$, but skipping repeat visits to any node (the "taking shortcuts" process)

In this method, recognize that $WALK$ does not fit with our definition of a tour, since it may contain nodes that are visited more than once. So, we simply eliminate repeat visits in our WALK to identify T^H, sometimes referred to as an embedded tour.

Christofides' Heuristic

Christofides' heuristic has the best known worst-case performance bound for the traveling salesman problem on complete networks satisfying the triangle-inequality. Similar to the Twice-Around MST heuristic, Christofides' heuristic utilizes the concept of minimum spanning trees. In addition, the heuristic relies on the idea of minimum cost perfect matchings (MCPM). Consider the following definitions and facts:

(Christofides Worst-case Performance Bound) Christofides heuristic is 1.5-*optimal for every complete network G satisfying the Δ-inequality, that is:*

$$C(T^C) \leq \frac{3}{2} C(TSP^*)$$

where TSP^* is an optimal traveling salesman tour on G.

Christofides' Heuristic

Steps:

1. Given an undirected, complete network G satisfying the Δ-inequality, find a minimum spanning tree MST^* on G.

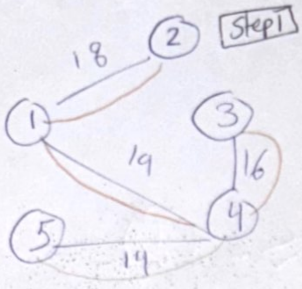
2. Let S be the nodes of G with odd degree with respect to MST^*.

3. Find a perfect node matching W^* of the nodes in S with minimum total cost. (This problem can be solved in $O(n^3)$ time by the method of Lawler).

4. Generate a tour of the nodes T^C using arcs in the set $MST^* \cup W^*$ exactly once. (T^C can be improved by introducing shortcuts to avoid revisiting nodes).

Consider the following 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.

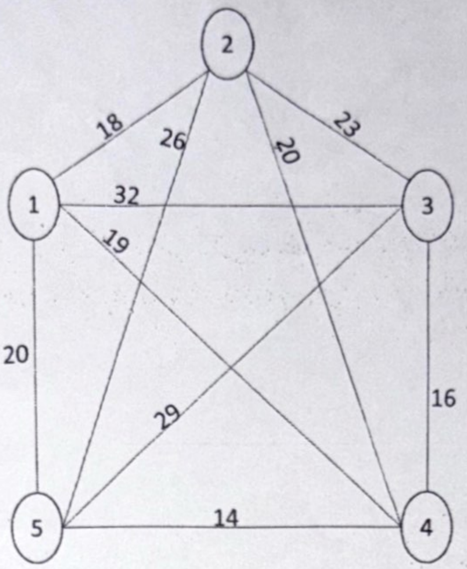
Step1



Step 2

List

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



a) Determine the TSP heuristic tour yielded by applying the twice around MST heuristic.

b) Determine the TSP heuristic tour yielded by applying Christofides heuristic.

Step 3

we chose node $\boxed{4}$ to walk

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

→ لازم نرجع
إلى نقطة البداية

Step 4

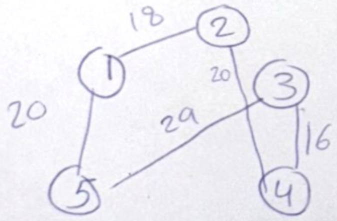
T_{tour} = {4,3,5,1,2,4} فكرة ال Tour انه بمشي عالترتيب = Tour

بال walk

وما بكرر ال Nodes

Step 5

Draw the Tour

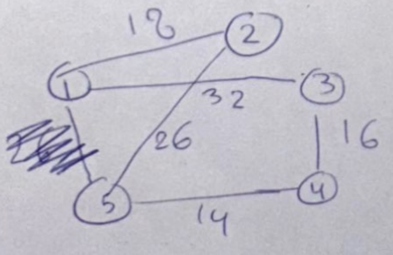


C(T) = 103

same Node but we took a different path

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

T = {4,3,1,2,5,9}



C(T) = 106

|TREE| <n-1

* Kruskal's algorithm

Initialization: $TREE = \emptyset$

Iterations:

[illegible] tie لو لن تصير increasing

1 و هكذا

1
1
1
2
3

```
1. Sort the arcs in order of non-decreasing $c_{ij}$ into the $LIST$ →
2. while $|TREE| < n - 1$:
    (a) Remove $(i, j)$ from the top of the $LIST$
    (b) Add $(i, j)$ to $TREE$ if so doing does not create a cycle
```

List
~~(a,b) → 1~~
~~(b,d) → 1~~
~~(b,c) → 1~~
~~(d,c) → 2~~
~~(a,c) → 3~~
~~(d,e) → 4~~
~~(e,c) → 5~~

Non decreasing

we Finish

ما يجوز هذا يكون عندي
cycle نهائياً

At the end, $T^* \leftarrow TREE$.

* Cuts

To explain the next algorithm, it necessary to understand the concept of a *cut* in a network.

Suppose the nodes in N are partitioned into two independent subsets, S and T. That is, $S \cup T = N$ and $S \cap T = \emptyset$. The cut set (S, T) is given by all arcs (i, j) such that $i \in S$ and $j \in T$ or $i \in T$ and $j \in S$.

*Suppose the Nodes N are portitioneed into 2 independent subset A and B, where

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

* The cut set (A,B) is given by all ~~arcs~~ arcs (i,j), such that $i \in A$ $j \in B$ or $j \in A$ $i \in B$

Prim's algorithm

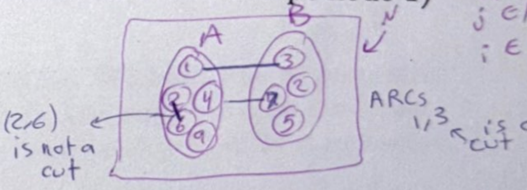
→ cut in Network

Initialization:

$TREE = \emptyset$

$S = \{1\}, T = N \setminus S$

(In English, the set S contains node 1, while the set T contains all nodes except node 1)



Iterations:

cut set $\boxed{A,B}$

```
while $|TREE| < n - 1$:
    1. Find the arc $(i, j)$ in $(\cancel{S}, \cancel{T})$ with minimum cost $c_{ij}$
    2. Add $(i, j)$ to $TREE$
    3. Remove $j$ from $\overset{B}{\cancel{T}}$ and add $j$ to $\overset{A}{\cancel{S}}$
```

contiue until $B = \emptyset$

At the end, $T^* \leftarrow TREE$.

* Prim's Algorithm

Inatialization Tree $= \emptyset$

$A = \{i\}$ → any arbitrary node , $B = N / \{A\}$

iteration while |Tree| <n-1

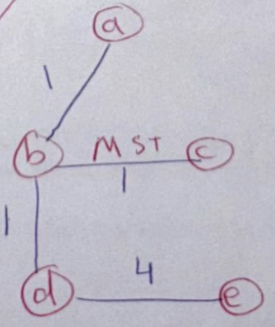
initialization $A = \{c\}$, $B = \{a, b, d, e\}$

أختار Tree $= \emptyset$.

iteration 1
- arc (c,b) has min cost
- add arc (c,b) to the tree
- $A = \{c, b\}$, $B = \{a, d, e\}$

iteration 2
- arc (b,d) has min cost
- add (b,d) to tree $A = \{c, b, d\}$
$B = \{a, e\}$

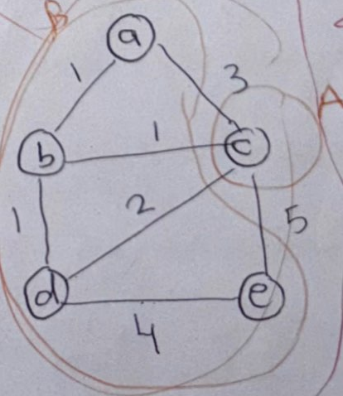
For



5 nodes so
I want 4 Arcs

(MST) = 7 cost?

السؤال مثلاً :



for Kruskal algorithm

Route-first Cluster-second Heuristics

Heuristics in this class begin first by creating an ordering of the nodes through some procedure, and then create feasible customer clusters to be served by a vehicle using the fixed ordering. After clusters are created, most of these heuristics simply serve customers following the initial order. However, others (like the Sweep heuristic) perform yet another step in which optimal or near-optimal TSP tours are created for each customer cluster and the depot. Technically, the Sweep Heuristic might be called an Order-first Cluster-second Route-third heuristic.

Generic TSP Partitioning Heuristic RIC2

Steps:

1. Let T be an optimal or near-optimal TSP tour that visits all of the customers and the depot. Suppose wlog $T = \{0, i_1, i_2, ..., i_n, 0\}$.
2. Define the node ordering $T_O = \{i_1, i_2, ..., i_n\}$.
3. Create customer clusters by running the next-fit bin packing heuristic with lot sizes q_i and vehicle capacity Q on the customer order defined by T_O.
4. Suppose the generated clusters are $\{i_1, i_2, ..., i_k\}$, $\{i_{k+1}, i_{k+2}, ..., i_\ell\}$, $\{i_{\ell+1}, i_{\ell+2}, ...,\}$. The final VRP tours are then given as $\{0, i_1, i_2, ..., i_k, 0\}$ etc. where the order of the customers in the VRP tours is identical to the order in the original TSP tour T.

Cluster-first Route-second Heuristics CIR2

Often, in practical vehicle routing problems, the subproblem of determining which customers to group together in a cluster served by a single vehicle is relatively more important than routing considerations. Since effective clustering can lead to reduced fleet sizes through better packing, problems in which dispatch or fixed vehicle costs are high may include packing as a primary concern. However, it can be shown that by focusing on clustering before routing even in the traditional vehicle routing problem focused on distance-based costs can result in good heuristic solution methods.

Sweep Heuristic

In the sweep heuristic, nodes must have geographic positions. Without loss of generality, suppose that the depot is given cartesian coordinates $(0, 0)$, and suppose each customer i has polar coordinates (r_i, θ_i) relative to the depot. Here, suppose $0 \leq \theta_i < 2\pi$ is measured clockwise from the positive x-axis.

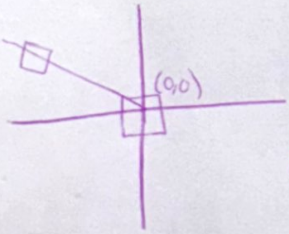
ترتيبهم حسب الزاوية

شعاع

1. Choose a start ray **r**, with a polar angle θ.
2. For each customer, calculate Δ_i as the relative angle between **r** and the customer location, measured clockwise.

   increasing
3. Sort the customers in order of non-decreasing Δ_i. Suppose the order created is $T_O = \{i_1, i_2, ..., i_n\}$.

   NF
4. Create customer clusters by running the next-fit bin packing heuristic with lot sizes q_i and vehicle capacity Q on the customer order defined by T_O.
5. Create final VRP tours by generating a separate TSP tour for each customer cluster and the depot.



(r_i, θ_i)

$0 \leq \theta_i < 2\pi$

360°

Δi

ray $= \theta_r$

$\Delta i = \theta_1 - \theta_r$

$\Delta i \Rightarrow$ increasing order

$T_O = \{ i_1, i_2, \dots i_n \}$

NF generate cluster

#of possible tours = $\frac{(n-1)!}{2}$

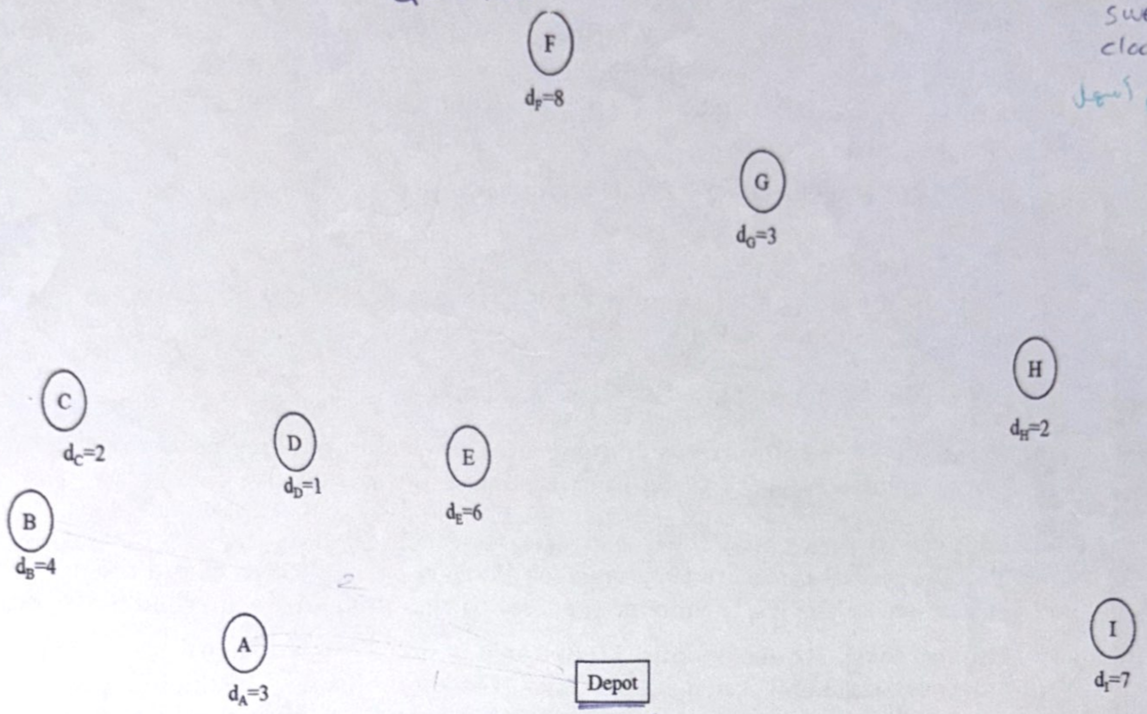
Example: Suppose you are the dispatcher for an LTL trucking firm, and you need to dispatch small delivery trucks for the local delivery of freight. Suppose you have many trucks available, each with the capacity to haul 10 tons of freight.

Q = 10 tons

لأنهم مرسومين دكذا ماني داعي نحسب الزوايا

sweep clock wise

بالقلم أسهل



start from here

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

CW

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

Cluster 1 = {A, B, C, D} → #of possible tours with the depot → $\frac{(5-1)!}{2} = \frac{4\times3\times2}{2} = 12$

Cluster 2 = { E }

Cluster 3 = { F }

Cluster 4 = { G, H }

Cluster 5 = { I }

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

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

Cluster 1 = {B, C, D}

Cluster 2 = {E}

Cluster 3 = { F }

Cluster 4 = {G, H}

Cluster 5 = {I, A}

*السؤال بس طالب clusters

(tour مو طالب)

السؤال هنا لا يحتوي رسمة ← إذاً لازم نحسب Δ ←(مهتمين بالزاوية)

Example 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}.

a) Using the Sweep Heuristic, generate clusters of customers each to be served by a single vehicle. Begin with a ray through customer 2 and rotate your ray clockwise. دائماً

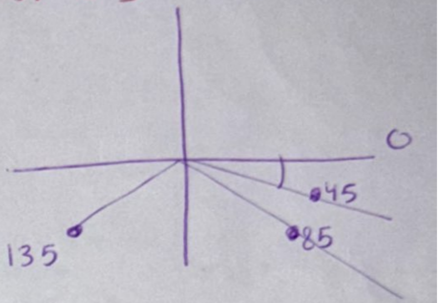
مو ضروري نحسبها

ترتيب حسب ال Δ

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

Customer	Demand	θ_i	Δ_i
C_1	7	45	45-105=-60
C_2	3	105*	0
C_3	4	220	220-105=115
C_4	5	135	135-105=
C_5	2	60	60-105=
C_6	4	85	
C_7	6	300	

السؤال حكى نبلش من هنا ← C_2



$\text{cluster}_1 = \{C_2, C_4\}$

$\text{cluster } 2 = \{C_3, C_7\}$

$\text{Cluster } 3 = \{C_1, C_5\}$

$\text{cluster } 4 = \{C_6\}$

اذا السؤال طلب tour بس نحط بالبداية depot و بالاخر depot

Example 3:

Suppose that we want to solve the vehicle routing problem with the following distance matrix and demands (units) at each customer location from 1 to 6. If each vehicle has a capacity of 15 units.

$Q = 15$

	C_1	C_2	C_3	C_4	C_5	C_6
Demand	4	6	3	5	3	6

the depot

	C_0	C_1	C_2	C_3	C_4	C_5	C_6
C_0	-----	20	18	14	16	12	19
C_1		-----	22	18	30	26	28
C_2			-----	32	20	22	21
C_3				-----	20	22	21
C_4					-----	30	32
C_5						-----	26
C_6							-----

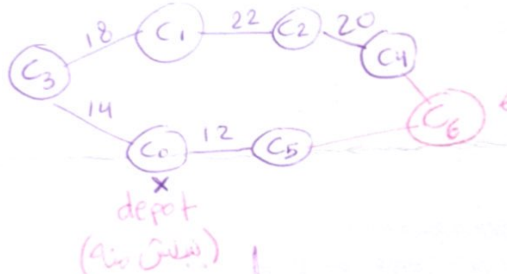
symmetric matrix

a) (Use the TSP tour partitioning heuristic (a standard route first, cluster-second approach)) to **generate a set of good vehicle routes** for this problem. Begin by using the Nearest Neighbor heuristic to generate a TSP tour. Determine all vehicles routes.

بنطلق بال Depot وبترجع ال

Build TSP optimal or near optimal NN - Heuristic ↓ NF Heuristic to generate clusters

بما انو ما حددنا ال node فإذاً هي ال Lower cost



حل node واحدة

Step 1 $T = \{C_0, C_3, C_1, C_2, C_4, C_6, C_5, C_0\}$

Step 2 $T_0 = \{C_3, C_1, C_2, C_4, C_6, C_5\}$

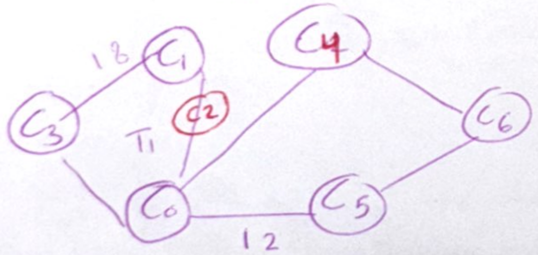
demand → 3 4 6 5 6 3

Step 3 cluster 1 $= \{C_3, C_1, C_2\}$ → $T_1 = \{0, C_3, C_1, C_2$

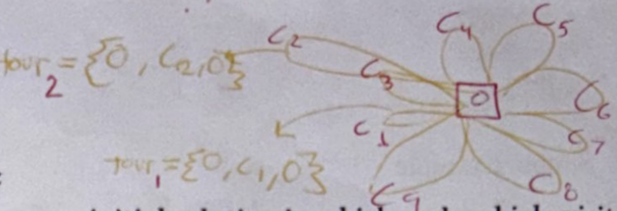
cluster 2 $= \{C_4, C_6, C_5\}$ → $T_2 = \{0, C_4, C_6, C_5$

Vehicle Route

تنطلق بال Depot وترجع ال



#of possible combinations or savings $= \left(\frac{n.(n-1)}{2} \right)$



Clarke-Wright Savings Heuristic

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.

Savings c$_1$-c$_2$ — 10 →
c$_1$-c$_3$ — 9
c$_1$-c$_4$ — 7

من الأعلى
↓
إلى الأقل

Generic Clarke-Wright

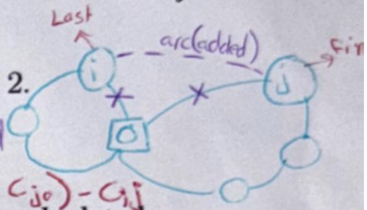
1. For each node $i \in N$, construct a tour $T_i = \{i\}$ beginning at the depot, traveling to node i, and returning to the depot. Let S be the set of tours.

2. For each pair of tours T_i and T_j in S such that the total customer demands of the two tours combined does not exceed Q (*i.e.*, $\sum_{k \in T_i \cup T_j} q_k \leq Q$), compute the savings: $S_{ij} = c_{end(i)0} + c_{0begin(j)} - c_{end(i)begin(j)}$.

3. Choose the pair of tours i and j that maximize S_{ij}. If $S_{ij} \geq 0$, then add $T_i + T_j$ to S and remove T_i and T_j from S. Else, end the heuristic.

4. If no tours were joined, end the heuristic. Else, repeat from Step 2.

Arc deleted - Arc Added



Implemented Clarke-Wright Heuristic Sort for Saving $S_{ij} = (c_{i0} + c_{j0}) - c_{ij}$

You may note in the preceding presentation that savings need not be calculated more than once for each pair of nodes. Consider the following version of the heuristic:

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.

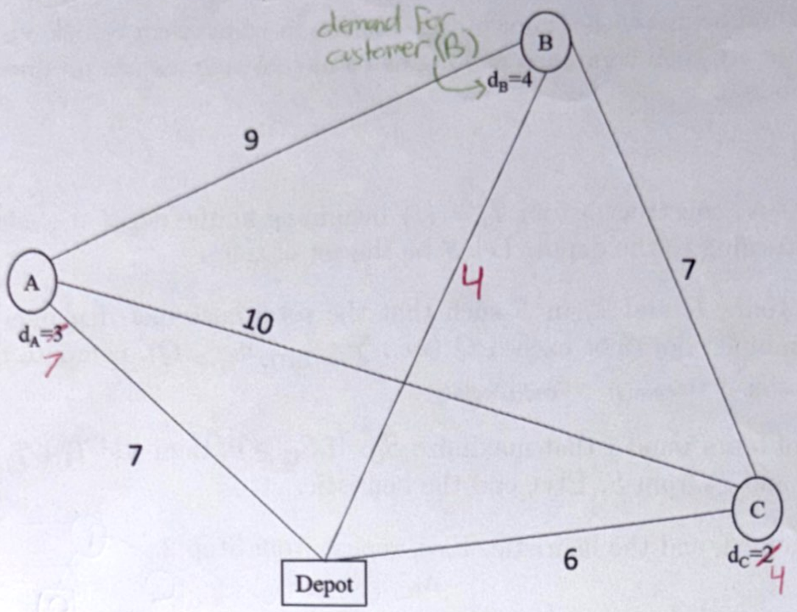
   (b) $\sum_{k \in T_i \cup T_j} q_k \leq Q$.

   (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.

Assume $\boxed{Q=10}$

Simple Example



it doesn't give a unique Solution

*لأنه في savings متساويين فممكن يختلف الحل

زي هاد السؤال بيجي بالامتحان ↓ اذا اجى بكون عليه ٨ علامات

{ Distance matrix
demand
capacity

Example 2:

Suppose that we want to solve the vehicle routing problem with the following distance matrix and demands (units) at each customer location from 1 to 6. If each vehicle has a capacity of 15 units. $\boxed{Q=15}$

	C_1	C_2	C_3	C_4	C_5	C_6
Demand	4	6	3	5	3	6

depot → 0

	C_0	C_1	C_2	C_3	C_4	C_5	C_6
C_0	-----	20	18	14	16	12	19
C_1		-----	22	18	30	26	28
C_2			-----	32	20	22	21
C_3				-----	20	22	21
C_4					-----	30	32
C_5						-----	26
C_6							-----

Use Clark Wright's heuristic to generate a set of good vehicle routes for this problem.

Step ① (construct a tour)

S List of tours

tours =

~~$T_1 = \{0, C_1, 0\} = 40$~~

~~$T_2 = \{0, C_2, 0\} = 36$~~

~~$T_3 = \{0, C_3, 0\} = 28$~~

~~$T_4 = \{0, C_4, 0\} = 32$~~

~~$T_5 = \{0, C_5, 0\} = 24$~~

~~$T_6 = \{0, C_6, 0\} = 38$~~

$T_{1+2} = \{0, C_1, C_2, 0\}$

$T_{1+2+3} = \{0, C_3, C_1, C_2, 0\}$

$T_{5+6} = \{0, C_5, C_6, 0\}$

$T_{4+5,6} = \{0, C_4, C_6, C_5, 0\}$

Step ② (calculate the savings)

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

# of possible savings $= \frac{6 \times 5}{2} = 15$ ← لأنه ما بهمنا الdirection

combination	Saving
$C_1 - C_2$	$C_{01} + C_{20} - C_{12} = 20 + 18 - 22 = 16$
$C_1 - C_3$	$C_{01} + C_{30} - C_{13} = \quad = 16$
$C_1 - C_4$	$C_{01} + C_{40} - C_{14} = \quad = 6$
$C_1 - C_5$	$C_{01} + C_{50} - C_{15} = 20 + 12 - 26 = 6$
$C_1 - C_6$	$C_{01} + C_{60} - C_{16} = \quad = 11$
$C_2 - C_3$	$C_{20} + C_{30} - C_{23} = 18 + 14 - 32 = 0$ ← يعني لا يوجد saving
$C_2 - C_4$	$C_{20} + C_{40} - C_{24} = \quad = 14$
$C_2 - C_5$	$C_{20} + C_{50} - C_{25} = \quad = 8$
$C_2 - C_6$	$C_{20} + C_{60} - C_{26} = \quad = 16$
$C_3 - C_4$	$C_{30} + C_{40} - C_{34} = \quad = 10$
$C_3 - C_5$	$C_{30} + C_{50} - C_{35} = \quad = 4$
$C_3 - C_6$	$C_{30} + C_{60} - C_{36} = \quad = 12$
$C_4 - C_5$	$C_{04} + C_{50} - C_{45} = \quad = -2$ ← عادي يكون بالسالب ومعناها انه بصير اسوء
$C_4 - C_6$	$C_{04} + C_{60} - C_{46} = \quad = 3$
$C_5 - C_6$	$C_{05} + C_{06} - C_{56} = \quad = 5$

Step ③ (sort the Savings)

sorted list of savings

~~$C_1 - C_2 = 16$~~ ✓

~~$C_1 - C_3 = 16$~~ ✓

$C_2 - C_6 = 16$ ✗ → we can't because of capacity

$C_2 - C_4 = 14$ ✗ شرط ال capacity غير متحقق

$C_3 - C_6 = 12$ ✗

$C_1 - C_6 = 11$ ✗

$C_3 - C_4 = 10$ ✗

$C_2 - C_5 = 8$ ✗

$C_1 - C_4 = 6$ ✗

$C_1 - C_5 = 6$ ✗

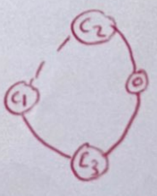
both in different tours { ~~$C_5 - C_6 = 5$~~ ✓

$C_3 - C_5 = 4$ ✗

~~$C_4 - C_6 = 3$~~ ✓

~~$C_2 - C_3 = 0$~~

~~$C_4 - C_5 = -2$~~

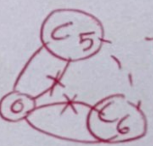


$T_{1+2} = \{0, C_1, C_2, 0\}$, combination ↑ $T_3 = \{0, C_3, 0\}$

$T_{1+2+3} = \{0, C_3, C_1, C_2, 0\} \doteq 13$

$\approx \{0, C_2, C_1, C_3, 0\}$ is equivalent to

→ $T_{5+6} = \{0, C_5, C_6, 0\}$



$T_{5+6} + T_4 = \{0, C_4, C_6, C_5, 0\}$ combination

$\approx \{0, C_5, C_6, C_4, 0\}$

★ Total cost $= 198 - (16 + 16 + 5 + 3) = 158$

TSP heuristics

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.

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
       $(\ell, k)$ in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert $j$
       between $\ell$ and $k$ in $T$.

end;
```

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 $T = \{i, j, i\}$

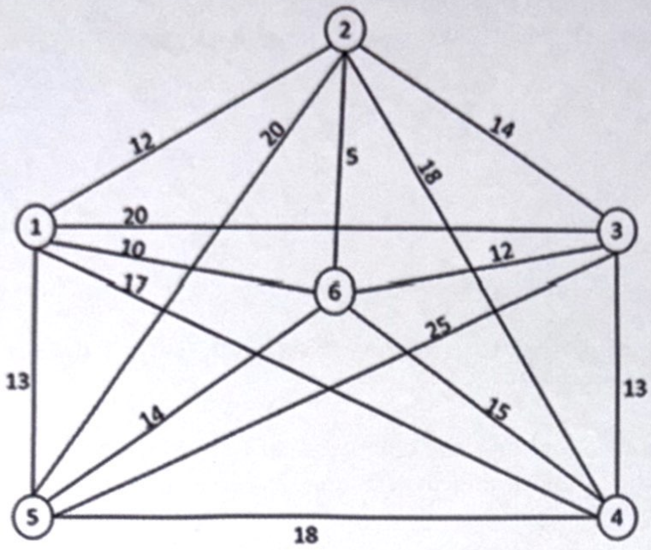
Steps:

```
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
       $(\ell, k)$ in the current tour with the minimum insertion cost: $c_{\ell j} + c_{jk} - c_{\ell k}$. Insert $j$
       between $\ell$ 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.

Extra page

Example: Consider the following 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.



a) Complete the following table that shows the lower bounds using several techniques:

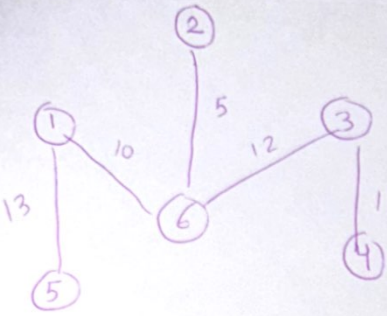
technique	Lower bound
MST+arc	
$1 - TREE_1$	
$1 - TREE_2$	
$1 - TREE_3$	
$1 - TREE_4$	
$1 - TREE_5$	
$1 - TREE_6$	

b) Determine the TSP heuristic tour yielded by applying the Nearest insertion heuristic.

c) Determine the TSP heuristic tour yielded by applying the Farthest insertion heuristic.

ج٩ $\boxed{b}$

Extra page



LB (MST) = 53 + 12
= 65

→ $\boxed{LB = 65}$

step ① (2,6)

(i,j)
T = {i, j, i} ← يعكس عكاه

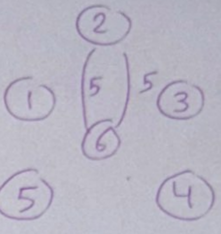
number of iterations

n − 2

T = {2,6,2} → C(T) = 10

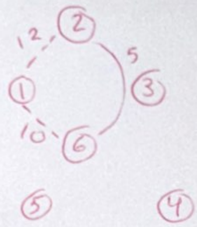
iteration 1 → 2 steps

$j \in T$	$i \in T$ (2 6)	C_{ij}
1	6	$C_{16} = 10$
3	6	$C_{36} = 12$
4	6	$C_{46} = 15$
5	6	$C_{56} = 4$



Now we have to determine which node to choose to enter the current Tou

step ②



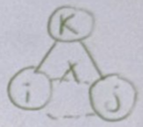
T = {2,6,2}

insert node 1 between 2,6

insertion cost = $C_{21} + C_{16} - C_{26}$
= 12 + 10 − 5
= 17

T = {2, 1, 6, 2}

current cost = Tour cost + insertion cost



insertion cost = $(C_{iK} + C_{Kj}) - (C_{ij})$

cost of Arc added − cost of Arc dele

مستحيل يكون negative

بسبب ال Triangle enquality

iteration 2 {2, 1, 6, 2} → C(T) = 27

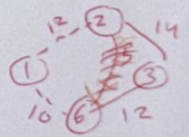
step 1

$j \in T$	$i \in T$ (1 2 6)	C_{ij}
3	6	$C_{36} = 12$
4	6	$C_{46} = 15$
5	1	$C_{15} = 13$

chose node (3) to be inserted in the current tour

step 2 insertion location

T = {2, 1, 6, 2}



options	insertion cost
btween 2,1	$= C_{23} + C_{31} - C_{21} = 14 + 20 - 12 = 22$
btw 1,6	$= C_{13} + C_{36} - C_{16} = 20 + 12 - 10 = 22$
tw 6,2	$= C_{63} + C_{32} - C_{26} = 12 + 14 - 5 = 21$

T = {2, 1, 6, 3, 2} = C(T) = 27 + 21 = 48

iteration 3 T = {2, 1, 6, 3, 2} → C(T) = 48
current

step 1

$j \in T$	$i \in T$ (1 2 3 6)	C_{ij}
4	3	13
5	1	13

node 4 will be inserted

step 2 insertion location T = {2, 1, 6, 3, 2}

options btw	
2,1	$= C_{24} + C_{41} - C_{21} = 23$
1,6	$= C_{14} + C_{46} - C_{16} = 22$
6,3	$= C_{64} + C_{43} - C_{63} = 16$
3,2	$= C_{34} + C_{42} - C_{32} = 17$

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

C(T) = 48 + 16 = 69

iteration 4

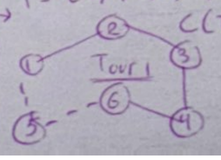
step 1 No need to do it in the Last iteration (Node 5 will be inserted to current tour
T = {2, 1, 6, 4, 3, 2}

step 2 insertion location

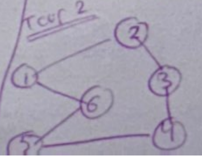
options	cost
2,1	$= C_{25} + C_{51} - C_{21} = 21$
1,6	$= C_{15} + C_{56} - C_{16} = 17$
6,4	$= C_{65} + C_{54} - C_{69} = 17$
4,3	$= C_{45} + C_{53} - C_{43} = 30$
3,2	$= C_{35} \quad - C_{32} = 31$

in the exam we say at the end we have 2 solutions

T = {2, 1, 5, 6, 4, 3, 2}
C(T) = 81



→ T = {2, 1, 6, 5, 4, 3, 2} → C(T) = 69 + 17
= 81



كلها مشينا بال iterations
بنجرب ال options to insert the nod

Bin Packing Problem (BPP)

Bin Packing

The bin-packing problem is another important problem in operations research with applications in logistics and transportation system operations.

Problem definition:

Suppose we have a supply of bins, and each bin has a capacity to hold Q units. Furthermore, suppose we have a number of items that need to be packed into bins. Each item i has a size $q_i \in (0, Q]$ which represents the portion of a bin's capacity that this item requires. Items cannot be "split" between multiple bins.

Suppose for explanatory purposes the item sizes have been arranged into a list, $L = \{q_1, q_2, q_3, ..., q_n\}$.

Problem BPP: Assign each item to a bin such that:

1. The total weight of all items in a bin is no greater than Q (no bin violates capacity)
2. The smallest number of bins is used

Capacity need to be observed (total weight/volume of all items in bin $\leq Q$)

Integer Programming Formulation

Consider now the following mathematical programming formulation for the bin-packing problem. Here, we will assume that n bins are available for packing, since in the worst-case, each item would require its own bin.

Decision variables

we have 2 decision variable

# of bins

# of items

$$y_j = \begin{cases} 1, & \text{if bin } j \text{ is used} \\ 0, & \text{otherwise} \end{cases} \qquad \forall\, j = 1...m$$

$$x_{ij} = \begin{cases} 1, & \text{if item } i \text{ is assigned to bin } j \\ 0, & \text{otherwise} \end{cases} \qquad \forall\, i = 1...n\,,\ j = 1...m$$

$n \rightarrow$ # of items

$n \neq m$

number of items $\neq$ number of bins

(BPP)

our objective

m → # of bins avaibable

$j \rightarrow m$

$$\min \sum_{j=1}^{m} y_j \qquad \text{(OF)}$$

minimizing # of bins utilized

subject to:

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall\, i = 1...n \qquad \text{(C1)}$$

each item needs to be assigned to exactly one bin

items=100 , bin=5

each item needs to be Assigned exactly one bin

$$\sum_{i=1}^{n} q_i x_{ij} \leq Q \qquad \forall\, j = 1...m \qquad \text{(C2)}$$

# of items

total capacity of weight on the bin $\leq Q$ (bin capacity)

$$x_{ij} \leq y_j \qquad \forall\, i = 1...n,\ j = 1...n \qquad \text{(C3)}$$

توضيح

$x_{13}=1 \quad y_3=1$

$x_{13}=0 \quad y_3=0/1$

يعني ممكن

يكون ثاني items

على ال bin ← 3

لذا y صفر أو واحد

item$_1$ $\quad x_{11} + x_{12} + \overset{\text{شنو}}{x_{13}} + x_{14} + x_{15}$

$\quad\quad\quad\ \ 0 \quad\ 0 \quad\ \ 1 \quad\ \ 0 \quad\ 0$

$$0 \leq x_{ij} \leq 1 \qquad \forall\, (i,j) \qquad \text{(C4)}$$

$$x_{ij} \text{ integer} \qquad \forall\, (i,j) \qquad \text{(C5)}$$

يكون binary

لما صفر أو واحد

* Description of the problem

- we have some #

The minimization nature of this formulation attempts to set $y_j = 1$ for as few bins as possible, and the objective function counts the number of bins opened. Constraints (C1) ensure that each item is assigned to some bin, and (C2) require that no bin exceeds its capacity Q. Constraints (C3) require that closed bins $y_j = 0$ can have no items assigned to them.

Like the traveling salesman problem, no efficient algorithm has been developed for the bin-packing problem; it is in the problem class NP-hard. If one wishes to solve the problem optimally, it is necessary to enumerate all possible ways of assigning items to bins, and this process is inefficient.

Applications in Logistics

The primary application of bin-packing in logistics is in the loading of freight. Suppose that you have n pieces of freight to load with various sizes, and you wish to use the fewest containers as possible to transport the freight. This becomes a bin-packing problem. This situation also arises when designing vehicle routing tours. One potential problem in vehicle routing is to determine the minimum number of transportation vehicles required to serve a set of customer demands, under the assumption that customer demands cannot be split between vehicles.

For example, suppose that you operate a fleet of vehicles each with capacity of 20 tons. Further, suppose that you need to serve 13 customers on a given day, and the following is their demands in tons: $L = \{16, 10, 14, 12, 4, 8, 3, 12, 7, 14, 9, 6, 3\}$. How many trucks are required? Which trucks should serve which demands?

Online heuristics for BPP

-> data overtime

First, a general definition. An online heuristic is one in which input data are received over time, and irrevocable decisions are made based on known data within some time limit. Online heuristics have no foresight; they cannot anticipate future data that has not yet "arrived." Often, real-world problems require online heuristics. For example, if you are operating a truck fleet and planning which trucks will serve which demands, the demands from customers may arrive dynamically and you may not have anticipatory knowledge of what the next customer demand will be.

In the following online heuristics for BPP, let's assume that customer lot sizes become known one at a time. Upon the arrival of a lot, an irrevocable assignment of the lot to a bin is made. Such heuristics have practical significance in the loading of transport containers, since changing a decision of where to stow a particular piece of cargo would require unloading it (and potentially other lots) and reloading it into a different container.

Suppose that customer lots arrive one at a time, in the order specified by L. Let's develop some heuristics.

NF ⚹ *Next-Fit Heuristic* $\alpha=2$

- only 1 bin is open at each step of the heuristic
- Given List L of items each with size q_i
- each bin has capacity Q

Description: This heuristic places the next item in the currently open bin. If it does not fit, the bin is closed and a new bin is opened.

<u>steps</u>: 1) take item from the top of the List
2) if it fits in the bin (current open bin), place it there and remove item from the list then go back to step 1
3) if it doesn't fit, close current bin, open new bin and place item there, remove it from the list, go back to step 1

Initialization:
Given a list of item weights $L = \{w_1, w_2, ..., w_n\}$.
Place item 1 in bin 1 and remove from L. Let $i = 1$, $j = 2$.

Iterations:

1. If item j fits in bin i, place j in i. If not, open a new bin $i + 1$ and place j in bin $i + 1$. Let $i = i + 1$.
2. Remove item j from L. Let $j = j + 1$.
3. While items remain in L, repeat from Step 1.

NF

The next-fit heuristic has a special advantage over other online heuristics in this case, since it allows immediate dispatch. In the heuristic, whenever a container is closed it will not receive any future items and therefore can be immediately dispatched. This might mean a truck can leave a depot, or a container can be loaded onto a ship or train, or something similar.

FF ✱ *First-Fit Heuristic*

Description: This heuristic keeps all unfull bins open. It places the next item in the lowest-numbered bin in which the item fits. If it does not fit in any bin, a new bin is opened.

Initialization:
Given a list of item weights $L = \{w_1, w_2, ..., w_n\}$.
Place item 1 in bin 1 and remove from L. Let $j = 2$, $m = 1$.

Iterations:

1. Find the lowest numbered bin i in which item j fits, and place j in i. If j does not fit in any bin, open a new bin and number it $m + 1$ and let $m = m + 1$.
2. Remove item j from L. Let $j = j + 1$.
3. While items remain in L, repeat from Step 1.

BF ✱ *Best-Fit Heuristic*

Description: This heuristic attempts to create the fullest bin possible each time an item is assigned. Again, all unfull bins are kept open. It places the next item j in the bin whose current contents are largest, but do not exceed $1 - w_j$ (so the item fits). If it does not fit in any bin, a new bin is opened.

Initialization:

Given a list of item weights $L = \{w_1, w_2, ..., w_n\}$.
Place item 1 in bin 1 and remove from L. Let $j = 2$, $m = 1$.

Iterations:

1. Find the bin i whose contents are maximum but not greater than $1 - w_j$ (if S_i are the items in bin i, $\sum_{k \in S_i} w_k$ is the contents of bin i), and place j in i. If j does not fit in any bin, open a new bin and number it $m + 1$ and let $m = m + 1$.
2. Remove item j from L. Let $j = j + 1$.
3. While items remain in L, repeat from Step 1.

Performance guarantees
The next-fit heuristic is 2-optimal. It has been shown that the first-fit and best-fit heuristics are 1.7-optimal; that is, for any input item sizes L the number of bins that they will generate is no worse than 1.7 times the optimal number of bins.

Practical issues with first-fit and best-fit
Both of these online heuristics lack the dispatch advantage of the next-fit heuristic. Of course, if a container is completely filled at some stage of the processing it can be immediately dispatched. However, in the pure form of these heuristics, bins that are open but not yet full cannot be dispatched. Thus, to use these methods in practical settings it will be necessary to augment the methods with dispatch rules. For example, a container that has reached some minimum fill percentage ρ and/or has been open for some minimum time γ might be dispatched.

Offline heuristics for BPP

→ data in advance

Offline heuristics assume that you have access to all the item sizes when you start the heuristics. There are three important offline heuristics for bin-packing that all work by first sorting the items by non-increasing size. They are called respectively *next-fit decreasing, first-fit decreasing,* and *best-fit decreasing.*

A good example string of numbers is 6,6,5,5,5,4,4,4,4,2,2,2,2,3,3,7,7,5,5,8,8,4,4,5.

Sorted becomes 8,8,7,7,6,6,5,5,5,5,5,5,4,4,4,4,4,4,3,3,2,2,2,2.

Performance guarantees
It has been shown that the first-fit and best-fit decreasing heuristics perform better (usually) than the online versions. They are both 1.5-optimal.