9/11/17

6. Graph Theoretic Algorithms And Computer Programs

→ Algorithm: step by step procedure

Features

- finiteness
- definitenes
- input
- output
- Efficient

Algorithm can be written in

→ English

→ Computer language.

→ Flowchart

→ Efficiency depends on:

- memory
- computation time

} size of input.

I/p data

Graph

There are 5 ways of representation of graph

1) Adjacency matrix

2) Incidence matrix.

3) Edge listing

4) two linear array .

5) Successor

Adjacency matrix X(G)

$n \times n$ matrix

binary matrix.

No: of elements $= n^2$

Pack of bits - word

w - word length - no: of bits in a word.

Each row can be → $[n/w]$

for that n rows → $\boxed{n[n/w]}$

- Storage requirements

Advantages

- Popular matrix
- upper/lower triangle of matrix is $\frac{n(n-1)}{2}$.

Disadvantages

- Complexity & toot computation time high
- parallel edge

Incidence matrix:

rows - vertices

columns - Edges.

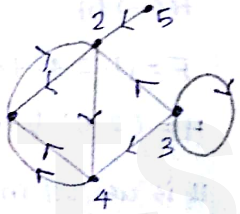
size - $n \times e$

storage requirement $n \times e > n^2$ Because $e > n$

Application - Switching network

disadvantages - high storage requirement

Edge Listing.



- Each edge representing using vertices.

(1,2) (2,1) (4,1) (2,4)

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

- Storage requirement

b-bits → Each vertex.

$e \to 2v$ bits

$e.2b = 2eb$

This is usefull when $2eb < n^2$

Sparse matrix - no: of zero's is many

So, it is easy to represent in Edge listing than Incident matrix.

Disadvantage

Retreiving, manipulation is difficult ∴ we may need to use search technique.

Two Linear Arrays

$F = \{ f_1 \ f_2 \ \dots \ f_e \}$

$H = \{ h_1 \ h_2 \ \dots \ h_e \}$

for every ith member there is a edge b/w $f_i \longleftrightarrow h_i$

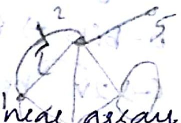
$F = (1, 2, 4, 4, 2, 3, 3, 3, 5)$

$H = (2, 1, 1, 1, 4, 2, 4, 3, 2)$

It is usefull in sorting requirements

Successor Listing

1, 2, 3, ... n



For each vertex K form a linear array another array with immediate successor.

1: 2

2: 1, 4

3: 2, 3, 4

4: 1, 1

5: 2

Storage requirement

$d_{av} \rightarrow$ average degree of vertices

$SR = n(1 + d_{av})$

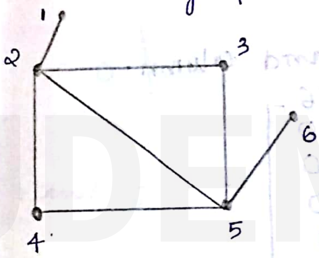
Used in DFS, BFS

14/11/17

Some Basic Algorithms

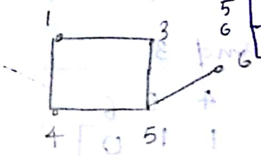
1) Connectedness & components

→ Efficient - fusing of vertices



$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & ● & 0 & 1 & 1 & 1 & 0 \\
3 & 0 & 1 & 0 & 0 & 1 & 0 \\
4 & 0 & 1 & 0 & 0 & 1 & 0 \\
5 & 0 & 1 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

Vertex	adjacency vertex
•	
1	fuse ← 2
1	fuse ← 3
	← 4
	← 5
1	← ⑥



→ To fuse Vi and Vj :-

- add ith row with jth row
- add ith column with jth coloumn

} logic OR

0+1 = 1
1+0 = 1
1+1 = 1
0+0 = D

- remove jth row & column

To fuse 1 and 2

Add 1st row and 2nd row.

add 1st ~~row~~ column and 2nd column.

Remove 2nd row and column.

$$\begin{array}{c|ccccc} & 1 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & 1 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 & 0 \\ 4 & 1 & 0 & 0 & 1 & 0 \\ 5 & 1 & 1 & 1 & 0 & 1 \\ 6 & 0 & 0 & 0 & 1 & 0 \end{array}$$

Fuse 1 and 3

$$\begin{array}{c|cccc} & 1 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & 1 & 0 \\ 4 & 1 & 0 & 1 & 0 \\ 5 & 1 & 1 & 0 & 1 \\ 6 & 0 & 0 & 1 & 0 \end{array}$$

Fuse 1 and 4:

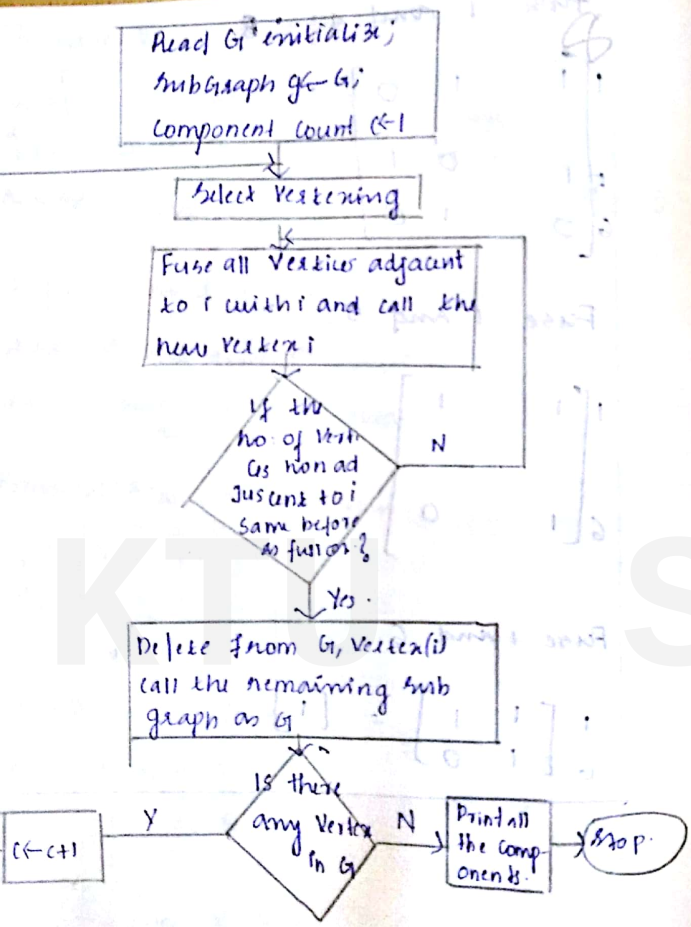
$$\begin{array}{c|ccc} 1 & 1 & 1 & 0 \\ 5 & 1 & 0 & 1 \\ 6 & 0 & 1 & 0 \end{array}$$

Fuse 1 and 5:

$$\begin{array}{c|cc} 1 & 1 & 1 \\ 6 & 1 & 0 \end{array}$$

Fuse 1 and 6

$$\begin{array}{c|cc} 1 & 1 & 1 \\ 6 & 1 & 0 \end{array} = \begin{bmatrix} 1 \end{bmatrix}$$


Read G initialize;
subGraph gf←G;
component count C←1
Select Vertexing
Fuse all vertices adjacent to i with i and call the new vertex i
If the no of vertices non adjacent to i same before as fusion?
N
Yes.
Delete from G, vertex(i) call the remaining sub graph as G
Is there any vertex in G
Y
N
C←C+1
Print all the components.
Stop.

15/11/17

Spanning Tree Algorithm:

- Connected graph → Spanning Tree
  disconnected graph → Spanning forest

- min weight/distance (Salemsman problem)

- Fundamental circuit.

At the K^{th} stage $1 \leq K \leq e$ taking each edge (f_K, h_K) lies in any of the following cases.

case-1: if neither vertex f_K or h_K lies in any of the tree then K^{th} edge is named as a new tree and its end vertices f_K and h_K are given component no c after incrementing c by one.

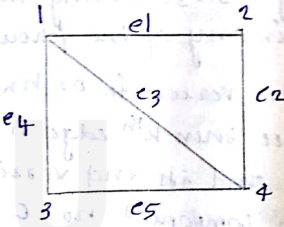
case-2: if vertex f_K is in some tree $T_i = \{1, 2, \dots c)$ and h_K is in some tree $T_j = \{1, 2, \dots c\}$ then K^{th} edge is used to join these two trees and every vertex in T_j is given component no: of T_i and c is decremented by 1.

case-3 if both vertices are in the same tree the edge (f_K, h_K) forms a fundamental circuit and is not considered further.

Case 4 If vertex fk is in a tree Ti and hk is no a tree The edge (fk, hk) is added to Ti by assigning component number of Ti to hk also.

Case 5: If vertex fk is in no tree and hk is in a tree Tj the edge (fk, hk) is added to Tj by assigning the component no: of Tj - Fk

Consider a graph.



$e_1 = (1,2)$

$e_5 - (3,4$

$e_2 - (4,2)$

$e_3 - (1,4)$

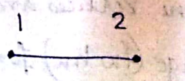
$e_4 - (1,3)$

$F = \{\overset{\textcircled{1}}{1}, \overset{\textcircled{1}}{\overset{\textcircled{2}}{3}}, 4, 1, 1\}$

$H = \{\overset{\textcircled{1}}{2}, \overset{\textcircled{2}}{4}, 2, 4, 3\}$.

Algorithm

1: $e_1 - (1,2)$



ase-1 C=1.

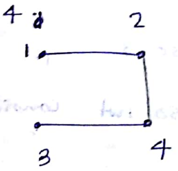
② $e_5 - (3,4)$

case-1 C=2

③ $e_2 - (4,2)$

already in different case-2

Join 2 and 4 :



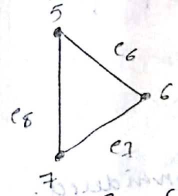
C=1 (decrement)

4 $e_3 - (1,4)$ (case - 3)

5. $e_4 - (1,3)$ (case - 3)

Now consider another graph with above graph

Disconnected:



$F = \{\overset{\textcircled{1}}{1}, \overset{\textcircled{1}}{3}, \overset{\textcircled{1}}{4}, \overset{\textcircled{1}}{1}, \overset{\textcircled{1}}{1}, \overset{\textcircled{2}}{5}, 6, 5\}$

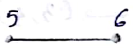
$H = \{2, 4, 2, 4, 3, \overset{\textcircled{2}}{6}, 7, 7\}$

$e_6 - (5,6)$

$e_7 - (6,7)$

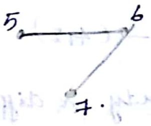
$e_8 - (5,7)$

(6) $e_6 - (5,6)$ case-1



$C = 1+1 = 2.$

(7) $e_7 - (6,7)$ case-4



(8) $e_8 - (5,7)$ case-3

forms circuit so not considered.

21/11/17 Shortest path algorithm.

- Shortest path b/w two specified vertices
- Shortest path b/w one vertex and all other vertex

Shortest path b/w every pair of vertices.

1) Shortest Path b/w two Specified Vertices.

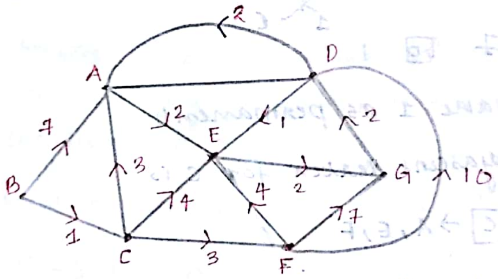
Dijkstra's Algorithm.

- weighted directed graph is considered.
- Generate a matrix D_{ij}

d_{ij} = length of distance from i to j

$d_{ij} = 0$

$d_{ij} = \infty$ no edge from i to j



* column indicates & vertices

* rows are specified by step.

i − permanent
j − perma.

P · S ⟷ t

	A	B	C	D	E	F	G
1.	∞	$\boxed{0}$	∞	∞	∞	∞	∞
2.	7	$\boxed{0}$	1	∞	∞	∞	∞
3.	7	$\boxed{0}$	$\boxed{1}$	∞	∞	∞	∞
4.	4	$\boxed{0}$	$\boxed{1}$	∞	5	4	∞
5	4.	$\boxed{0}$	$\boxed{1}$	∞	5	$\boxed{4}$ ✓	∞
6.	i=1 4	i=2 $\boxed{0}$	$\boxed{1}$	14	5	$\boxed{4}$	i=n 11
7	$\boxed{4}$ ✓	$\boxed{0}$	$\boxed{1}$	14	5	$\boxed{4}$	11
8.							
9.							

$\boxed{7}$ ✓

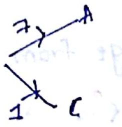
1<6

Source (S)	destination (t)
B	G

* Each label can be temproricy/ permantly

* permanent ⇒ □ V

directed from B is



Step 2: 7 $\boxed{C}$ 1

Step 3: make 1 as permanent.

Step 4: adjascent vertex for C is

$\overset{i}{\boxed{C}} \rightarrow \overset{j}{A}, E, F$

we can write directly

In each iterat^{n} a vertex get a permanent label according to follow

i. Every vertex j ie not a permanantly label gets a new temprory label whose value is given by minimum of old label of j, old label of i plus dij

min[old[label of j[old label i +dij)]

ii. Smallest value among all temprory values is found and that becomes the permanant label of the corresponding vertex.

(−) A E F

A → 4 min(7, 1+3)

E → 5 min (∞, 1+4)

F → 4 min(∞, 1+3)

Smallest value = 4 (tie comes) Select any one randomly and make it as permanant.

we choosen F

Now consider F

(i)

F → E, G, D

E → min(5, 4+4) = $\underline{\underline{5}}$

G → min (∞, 4 + 7) = 11

D → min (∞, 4 + 10) = 14

Here the smallest value is $\overset{4}{B}$. Then makes it as permanant.

```
Read D, n, S, t
LABEL ← ∞
LABEL(S) ← 0
VECT ≤ 0
i ← S, VECT(S) ← 1

j ← 1
m ← ∞

IS
VECT(j)=1          Z=dij + LABEL(i)   IS
?

IS
j=n?

Vect(p) ← 1
i ← p.

Is
i=t ?

Print
L
```

22/11/17

Shortest path between all pair of Vertices

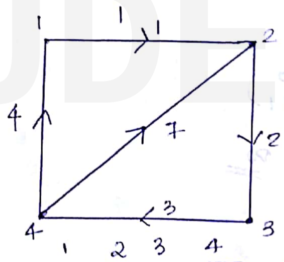
Floyd-warshall algorith:

$D_1 \quad D_2 \quad D_3 \quad \ldots \quad D_n$

$$d_{ij}^{K} = \begin{cases} w_{ij} & k=0 \\ \min(d_{ij}^{K-1}, d_{iK}^{K-1} + d_{Kj}^{K-1}) & \text{for } K \geq 1 \end{cases}$$

$$d_{ij}^{0} = \begin{cases} w_{ij} & i \neq j \\ 0 & i=j \\ \infty & \text{no edge.} \end{cases}$$

Consider a graph with 4 ~~edg~~ vertices.



K= iteration value.

$$d^{0} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & \infty & \infty \\ 2 & \infty & 0 & 2 & \infty \\ 3 & \infty & \infty & 0 & 3 \\ 4 & \cancel{\infty}\,4 & 7 & \infty & 0 \end{array}$$

$$d^{1} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & \infty & \infty \\ & \infty & 0 & \underline{2} & \underline{\infty} \\ 3 & \infty & \underline{\infty} & 0 & \underline{3} \\ 4 & 4 & \underline{5} & \underline{\infty} & 0 \end{array}$$

In d^1 ~~for~~ 1: diagonal elements are zero.

2: First row and column same as d^0

3: ~~d~~ $\min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$

to fill (2,3)

$i=2 \quad K=1 \quad j=3.$

$\min(d_{23}^{0}, d_{21}^{0} + d_{13}^{0}) = \min(2, \infty) = \underline{\underline{2}}$

(2,4) $i=2$ & $j=4$

$\min(d_{24}^{0}, d_{21}^{0} + d_{14}^{0})$

$\min(\alpha, \alpha) = \underline{\underline{\infty}}$

(3,1) $i=3$ & $j=1$

$\min(d_{31}^{0}, d_{31}^{0} + d_{11}^{0})$

$\min(\alpha, \alpha + 0) = \underline{\underline{\infty}}$

(3,4) $i=3$ & $j=4$

$\min(d_{34}^{0}, d_{11}^{0} + d_{14}^{0})$

$\min(\overset{3}{\cancel{\alpha}}, 0 + \infty) = \underline{\underline{3}}$

(4,2) $i=4$ & $j=2$

$\min(d_{42}^{0}, d_{41}^{0} + d_{12}^{0}) = (7, 4+\cancel{2}1)$

$= \underline{\underline{5}}$

Now calculate d^2:

Fill diagonal elements, 2^{nd} row and 2^{nd} column

$$d^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 3 & \infty \\ 2 & \infty & 0 & 2 & \infty \\ 3 & \infty & \infty & 0 & \underline{3} \\ 4 & 4 & 5 & 7 & 0 \end{array}$$

$\overset{i}{3},\overset{j}{1}$ ∞, ∞

ij
14 $(\alpha, 1+\infty)$

$(\overset{i}{3}, \overset{j}{1})$ $K=2$

$\infty, \infty +$

$(\underset{i}{3}, \underset{j}{4})$ $(3,$

$K=2$

$(\underset{1}{i}, \underset{2}{j})$ $K=3$ (1,3)

$$d^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 3 & 6 \\ 2 & \alpha & 0 & 2 & 5 \\ 3 & \alpha & \alpha & 0 & 3 \\ 4 & 4 & 5 & 7 & 0 \end{array}$$

$K=3$

$$d^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \underline{1} & \underline{3} & 6 \\ 2 & \underline{9} & 0 & \underline{2} & 5 \\ 3 & \underline{7} & \underline{8} & 0 & 3 \\ 4 & 4 & 5 & 7 & 0 \end{array}$$

$K=4$

i	j	
1	2	(1, 6) = 1
1	3	(3, 6) = 3
2	1	(α, 5+4) = 9
2	3	(2, 5+7) = 2
3	1	(α, 3+4) = 7
3	2	(α, 3+5) = 8

Now we need shortest Path for that we need S^0 S^1 S^2 S^3 S^4

To create S^0,

$$3 \quad z_{ij}^0 = \begin{cases} j & i \neq j \\ 0 & i = j \end{cases}$$

$$\begin{array}{c|cccc} 1 & 0 & 2 & 3 & 4 \\ 2 & 1 & 0 & 3 & 4 \\ 3 & 1 & 2 & 0 & 4 \\ 4 & 1 & 2 & 3 & 0 \end{array}$$

D S^1 can be filled from the S^0 if the $D^0 = D^1$ if $\neq$ then ~~copy~~ write k value.

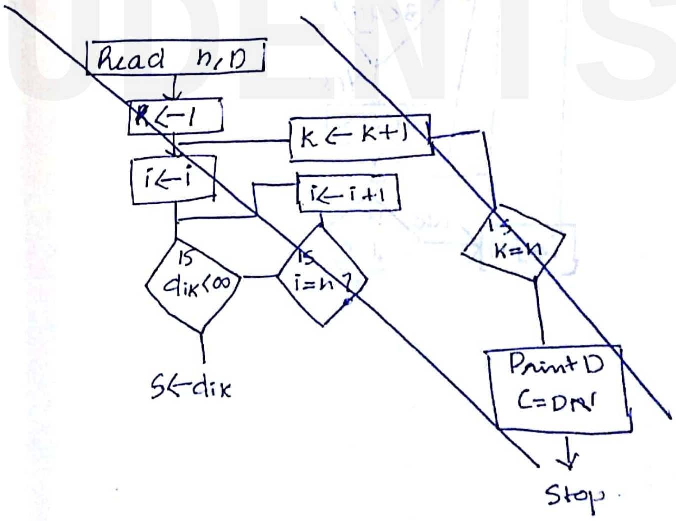
$$S^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 2 & 3 & 4 \\ 2 & 1 & 0 & 3 & 4 \\ 3 & 1 & 2 & 0 & 4 \\ 4 & 1 & 1 & 3 & 0 \end{array}$$

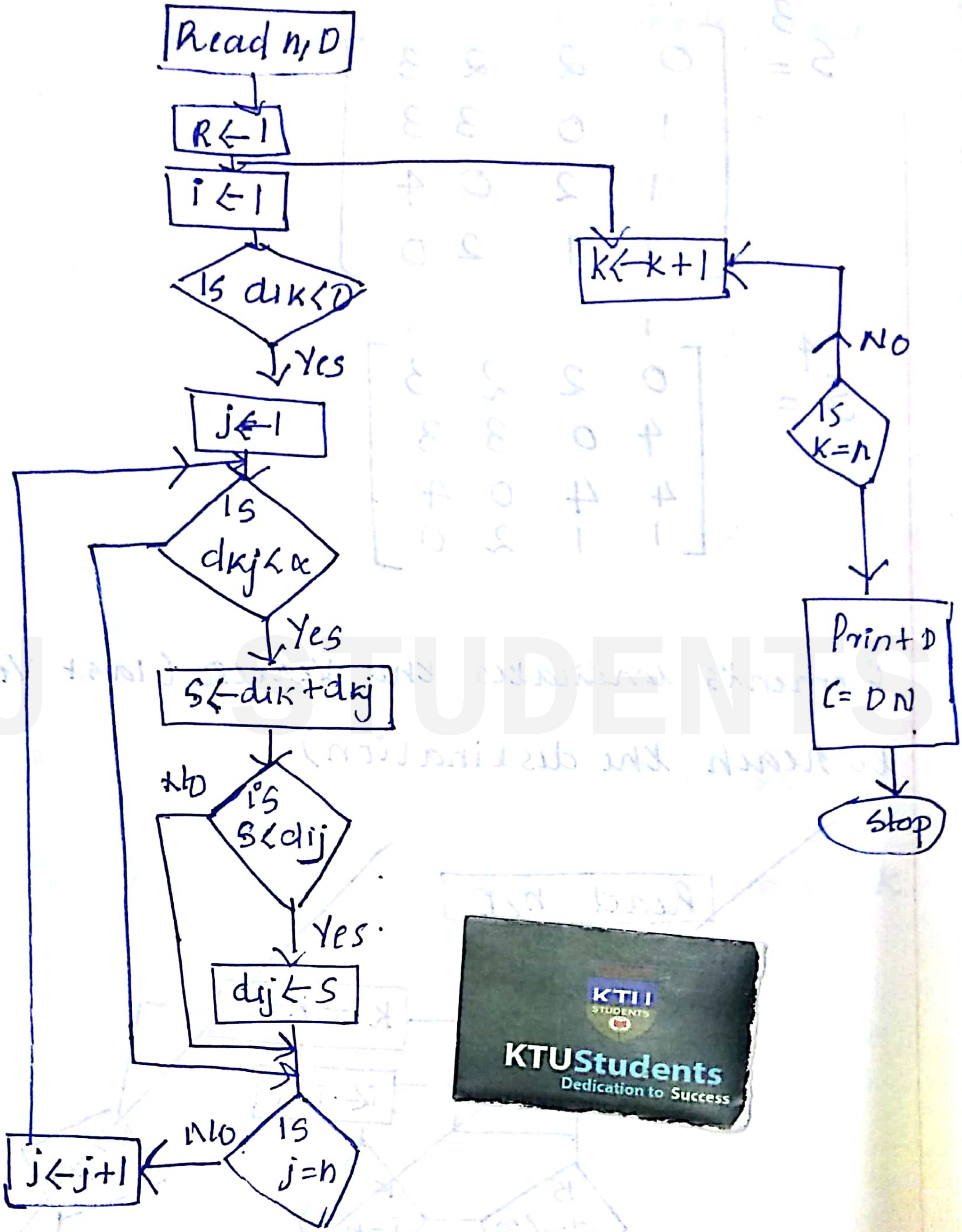
$$S^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 2 & 2 & 4 \\ 2 & 1 & 0 & 3 & 4 \\ 3 & 1 & 2 & 0 & 4 \\ 4 & 1 & 1 & 2 & 0 \end{array}$$

$$S^3 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 2 & 2 & 3 \\ 2 & 1 & 0 & 3 & 3 \\ 3 & 1 & 2 & 0 & 4 \\ 4 & 1 & 1 & 2 & 0 \end{array}$$

$$S^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 2 & 2 & 3 \\ 2 & 4 & 0 & 3 & 3 \\ 3 & 4 & 4 & 0 & 4 \\ 4 & 1 & 1 & 2 & 0 \end{array}$$

Elements indicates the vertex (last vertex to reach the destination)




Read n, D
R←1
i←1
Is dik<D
Yes
j←1
Is dkj<∞
Yes
S←dik+dkj
No
is S<dij
Yes.
dij←S
Is j=n
No
j←j+1
k←k+1
NO
Is k=n
Print D
C=DN.
Stop