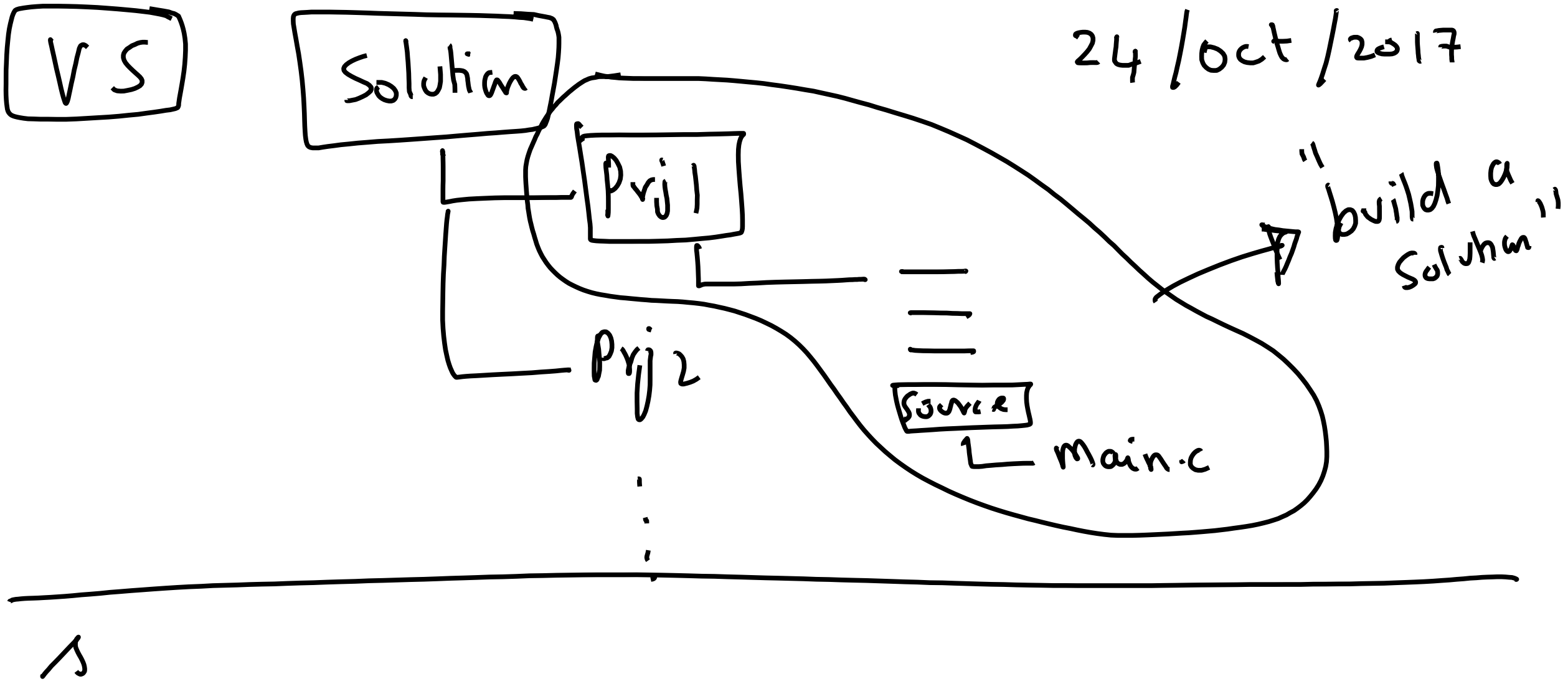
VS

24/oct/2017



Syntax

printf ( )

scanf ( ) vs scanf_s ( )

strcpy (a,b) ⟼ strcpy_s ( )

---

if - else if - else

switch

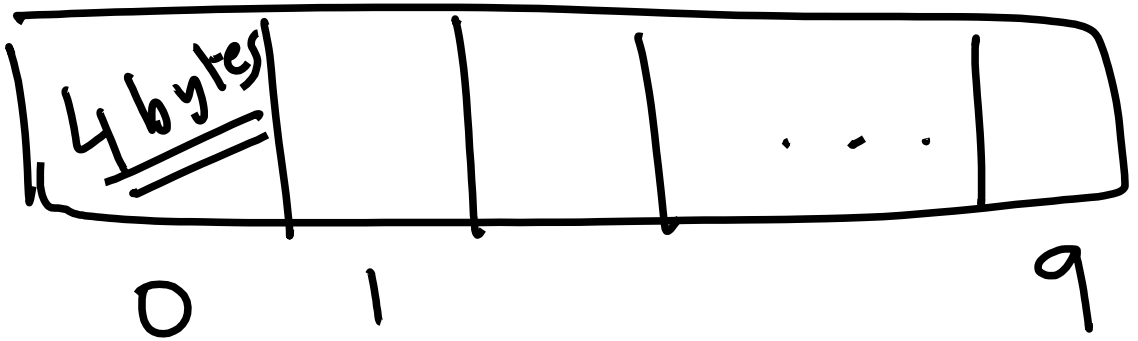
loops — while

do - while

for

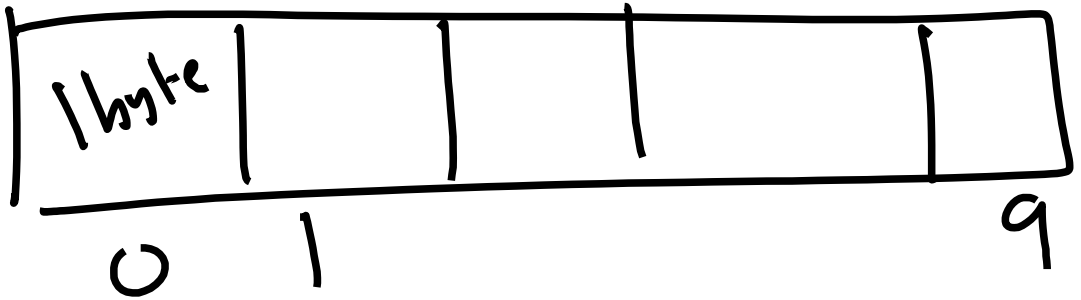
Arrays

```
int array[10];
```




```
float values[10];
```

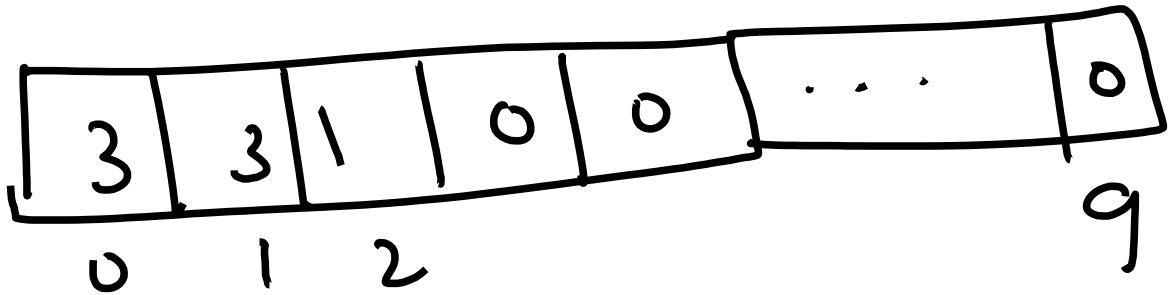
```
char name[10];
```



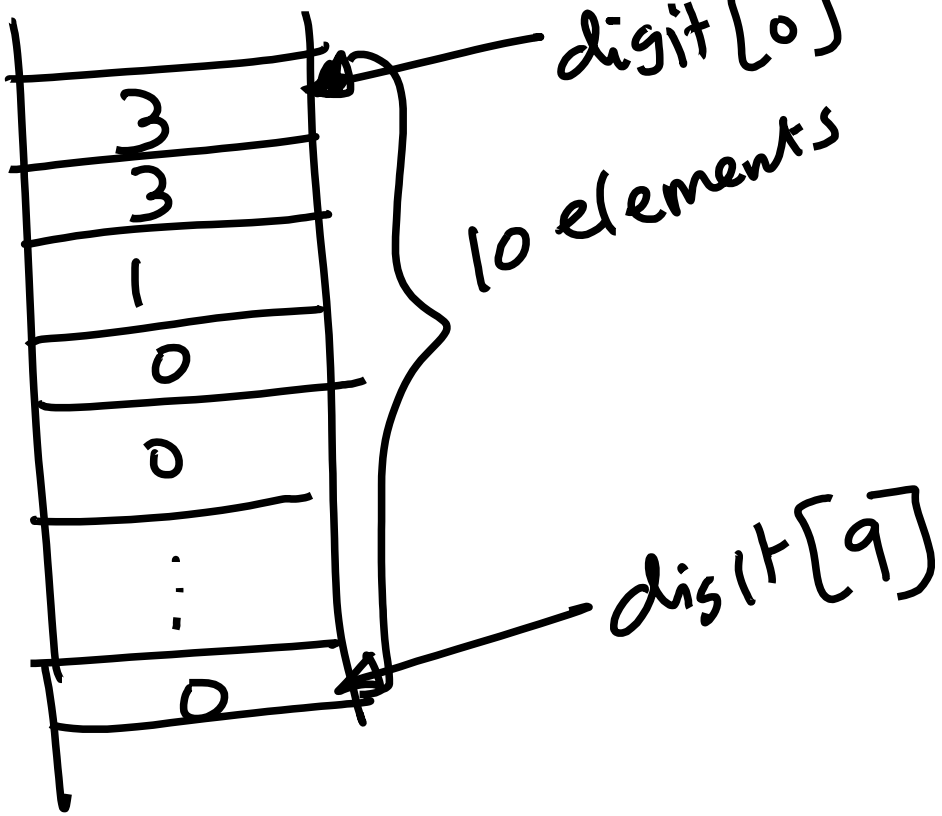

```
array[i] = array[i] + 10;
```

13-1 (a)

```
int digits[10] = {3, 3, 1};
```



RAM



T3.2

```
float data[100], d[100], avg;
.
.
for (i=0; i<100; i++) {
    avg = avg + data[i];
}
avg = avg/100; /* avg/strlen ? try this! */
for (i=0; i<100; i++) {
    d[i] = data[i] - avg;
```

float array[10][10];

row

col

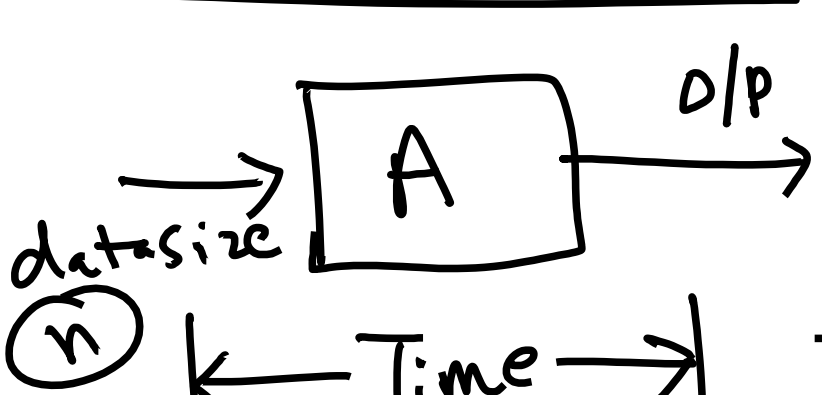
$$\begin{bmatrix} (0,0) & (0,1) & \cdots & (0,9) \\ (1,0) & & & (1,9) \\ \vdots & & & \\ (9,0), & (9,1), & \cdots & (9,9) \end{bmatrix}$$

array[3][4] = {1, 2, 3, 4, ..., 12};

array

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

Time Complexity



$T(n) = an^2 + bn + c$

n	$T(n)$
1	(a, b, c)
5	$(a5^2, b5)$
⋮	
500	$(500^2 \quad 500)$
⋮	
1 mil	$1mil^2, 1mil$

n^2 term dominates

$n^3 + n^2 + n + k$

Big O(.)

Order of complexity is $O(n^2)$.

for (i=0; i < 100 ; i++) {

for (j=0; j < 100 ; j++) {

a[i][j] = a[i][j] + 1;

}

}

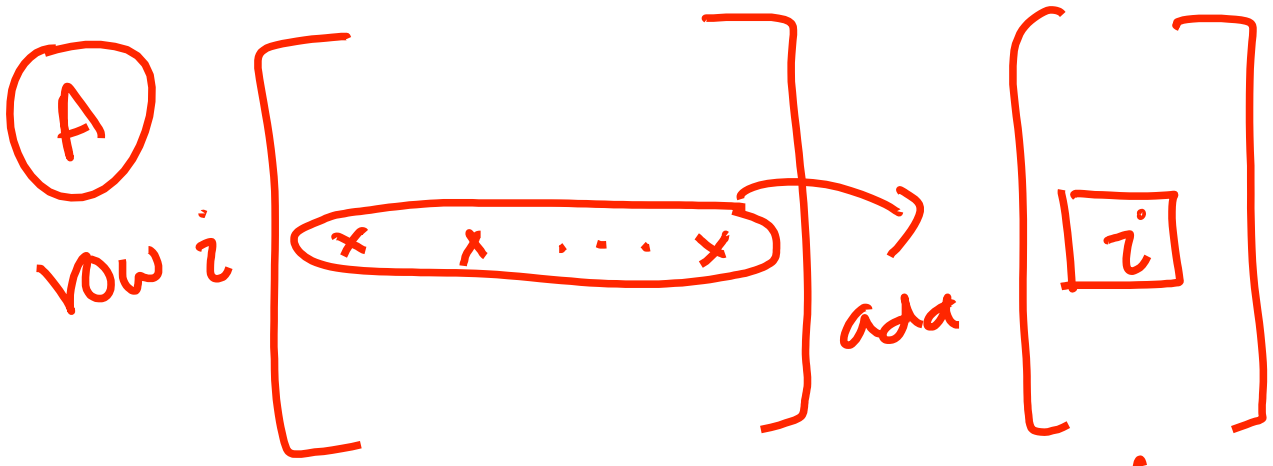
$O(n^2)$.

i — 100

for a fixed i j runs 100x

& that j runs 100x

100 x 100 = 10000



$$b[i] = \sum_{j=0}^{N-1} a[j]$$

N×N

```
for (i=0; i<N; i++) {
    sum=0;
    for(j=0; j<N; j++)
        sum = a[i][j]+sum;
    b[i]=sum;
}
```

row

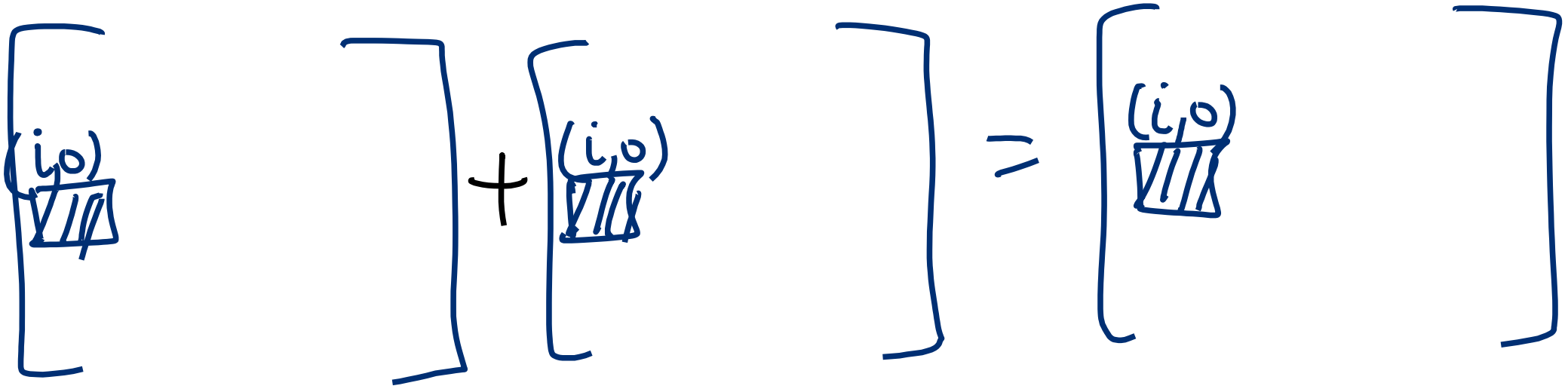
col

```
b[i]=b[i]+
a[i][j];
```

Time Complexity ?

$O(N^2)$

B



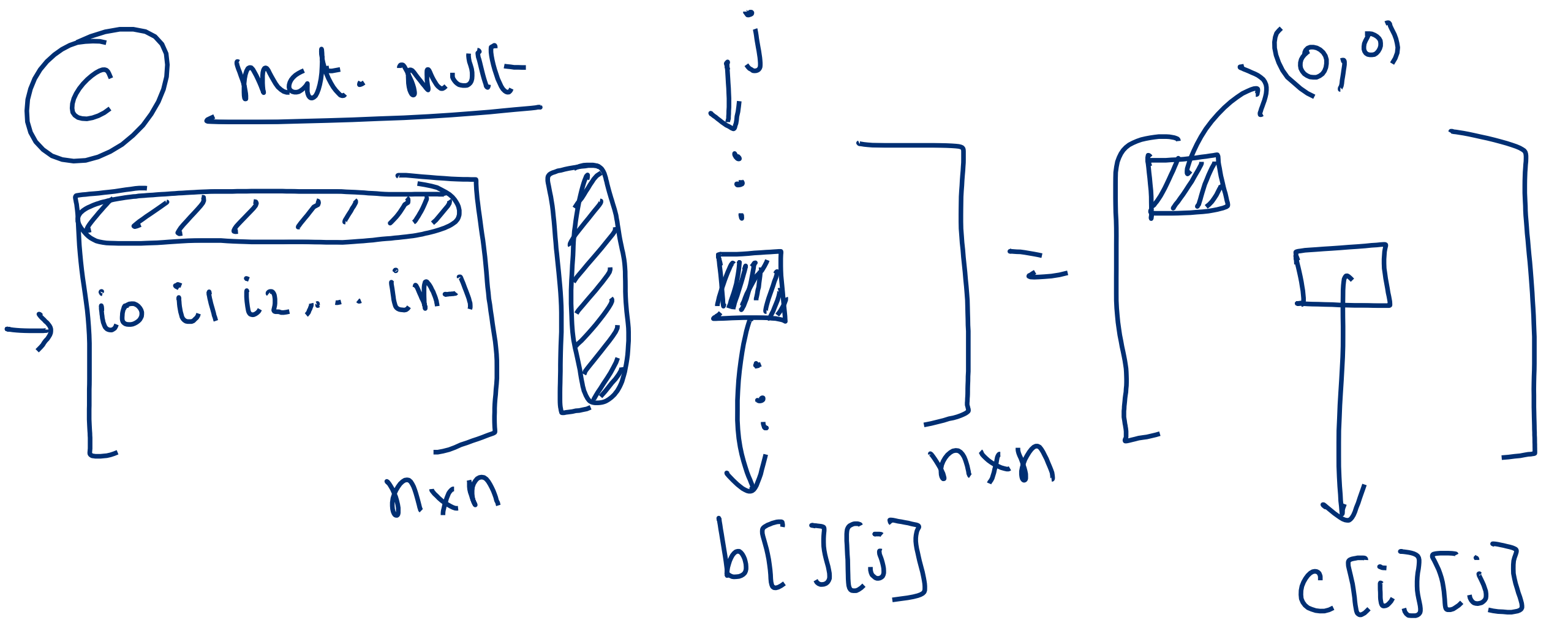
a[i][o] + b[i][o] ⇒ c[i][o]

```
for (i=0; i<N; i++){ /*row*/
    for(j=0; j<N; j++){ /*col*/
        c[i][j]= a[i][j]+ b[i][j];
    }
}
```

Time complexity ?

O(N2).

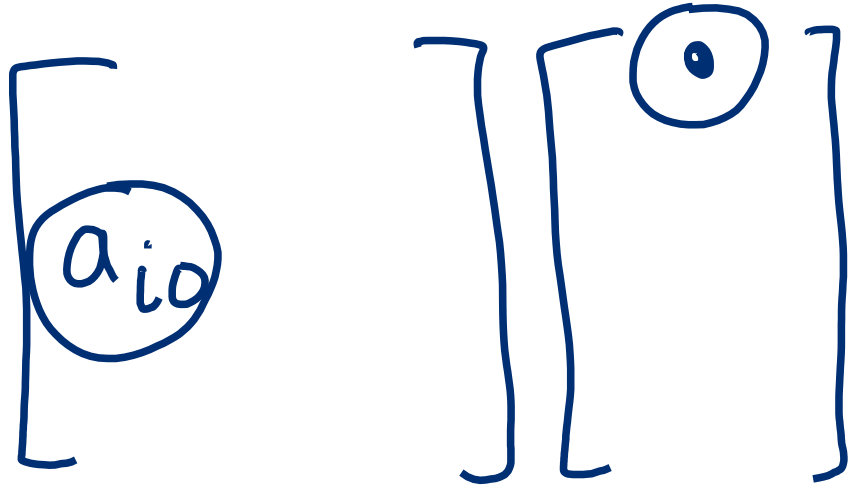
```
int a[20][20], b[20][20],
    c[20][20];
```


C
mat. mult-
i0 i1 i2 ... in-1
n x n
j
b[ ][j]
n x n
(0,0)
c[i][j]

$$a_{i0}\, b_{0j} + a_{i1}\, b_{1j}$$

$$+ a_{i2} \cdot b_{2j} + \cdots$$

$$+ a_{ik} \cdot b_{kj} + \cdots a_{in-1} \cdot b_{n-1\,j} = c_{ij}$$

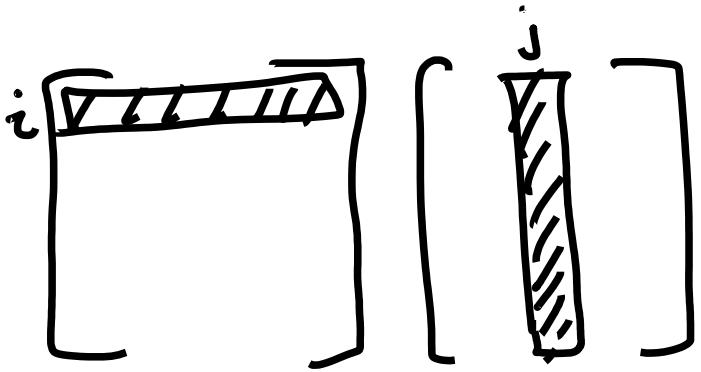


$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} \cdot b_{kj}$$

$i = 0, 1, \ldots, n-1$

$j = 0, 1, \ldots, n-1.$

```
for(i=0; i<N; i++){   /* row */
    for(j=0; j<N; j++){   /* col */
```




```
        for(k=0; k<N; k++){
            c[i][j] = c[i][j] +
                a[i][k] * b[k][j];
        }
    }
}
```

Time complexity = $O(N^3)$

Space Complexity

integer size

(A) $4n^2 + 4n = 4(n^2+n) \rightarrow O(n^2)$.

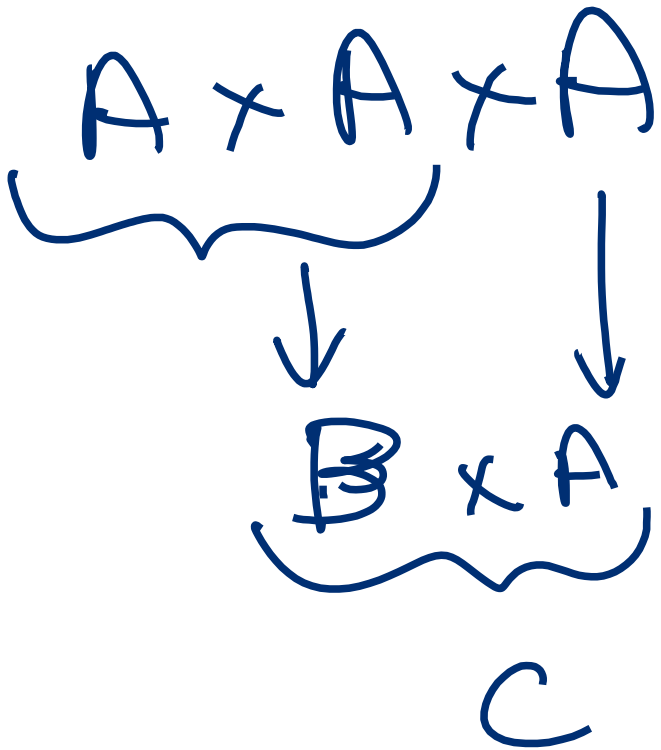
(B) $n^2 + n^2 + n^2 = 3n^2 \rightarrow 4 \times 3n^2 = 12n^2$

(C) $n^2 + n^2 + n^2 = 3n^2 \rightarrow 12n^2$

(d) $A \times A = A^2 = B$

$N \times N \quad N \times N$

A3 ?



Time Complexity = ?.

$A \times A \longrightarrow B$ N^3

$B \times A \longrightarrow C$ N^3

$2N^3$

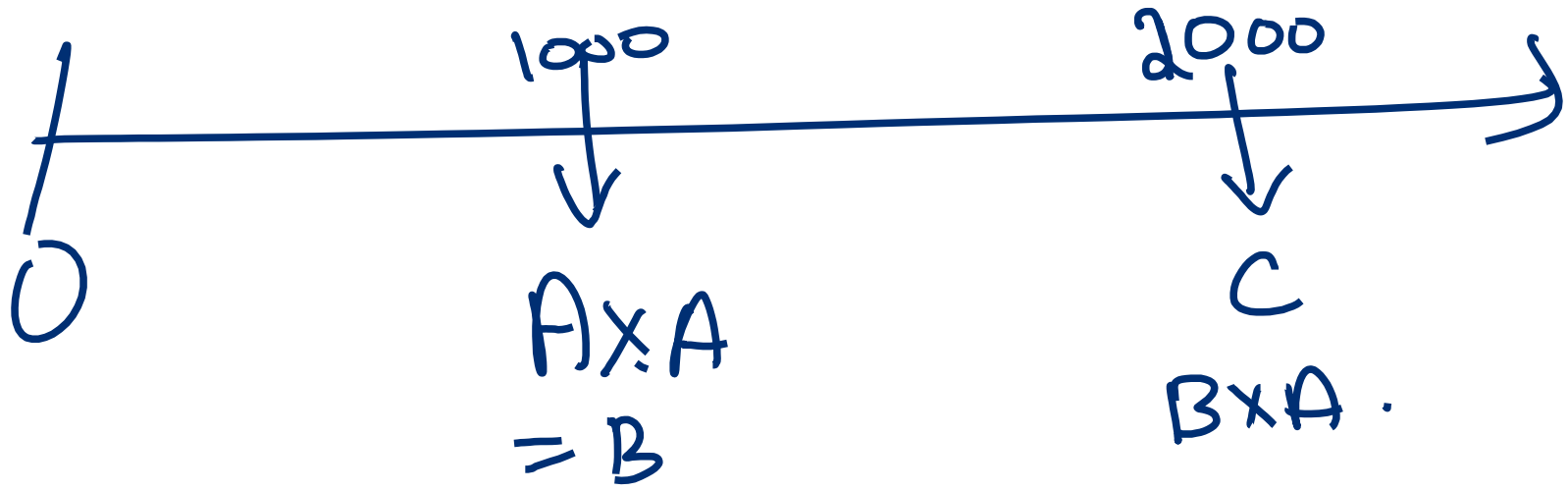
N = 10 t = 1msec

$A^2 = O(N3)$ $10 \times 10 \times 10 = 1000 \times 1 = 1000$ msec

↓

B

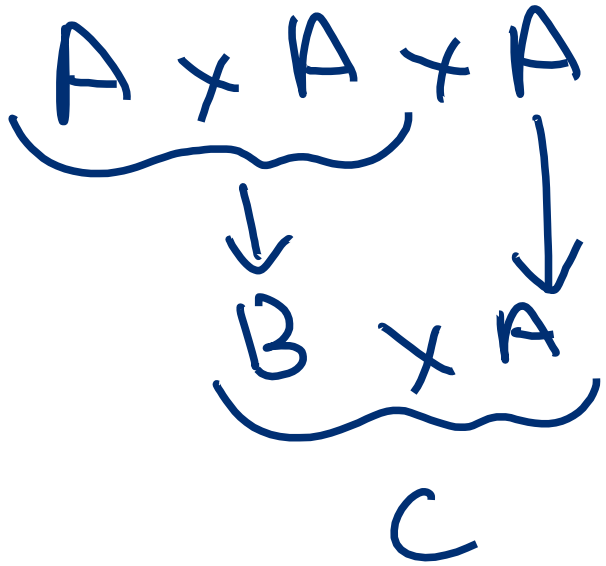
BXA = O(N3) = 1000msecs



Space complexity

A, B, C

n^2 n^2 n^2 $= 3n^2$



int → $4 \times 3n^2 = 12n^2$ bytes

When the algo terminates mem. used $= 2n^2$

$= 8n^2$ bytes

26/oct/2017

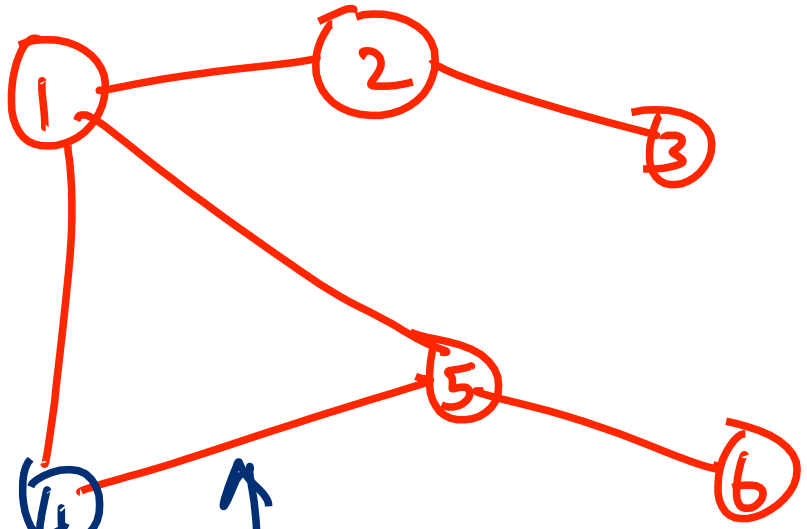
```
for (i=0 to N) {
    for (j=0 to i) {
        x = i+j;
    }
}
```

$1+2+3+\cdots+N-1+N = \underbrace{\frac{N(N+1)}{2}}_{O(N^2)}$

Time complexity

i	j (# of times j runs)
0	1
1	2
2	3
⋮	⋮
5	6
⋮	⋮
10	11

$T = (1+2+3+\cdots+11)$

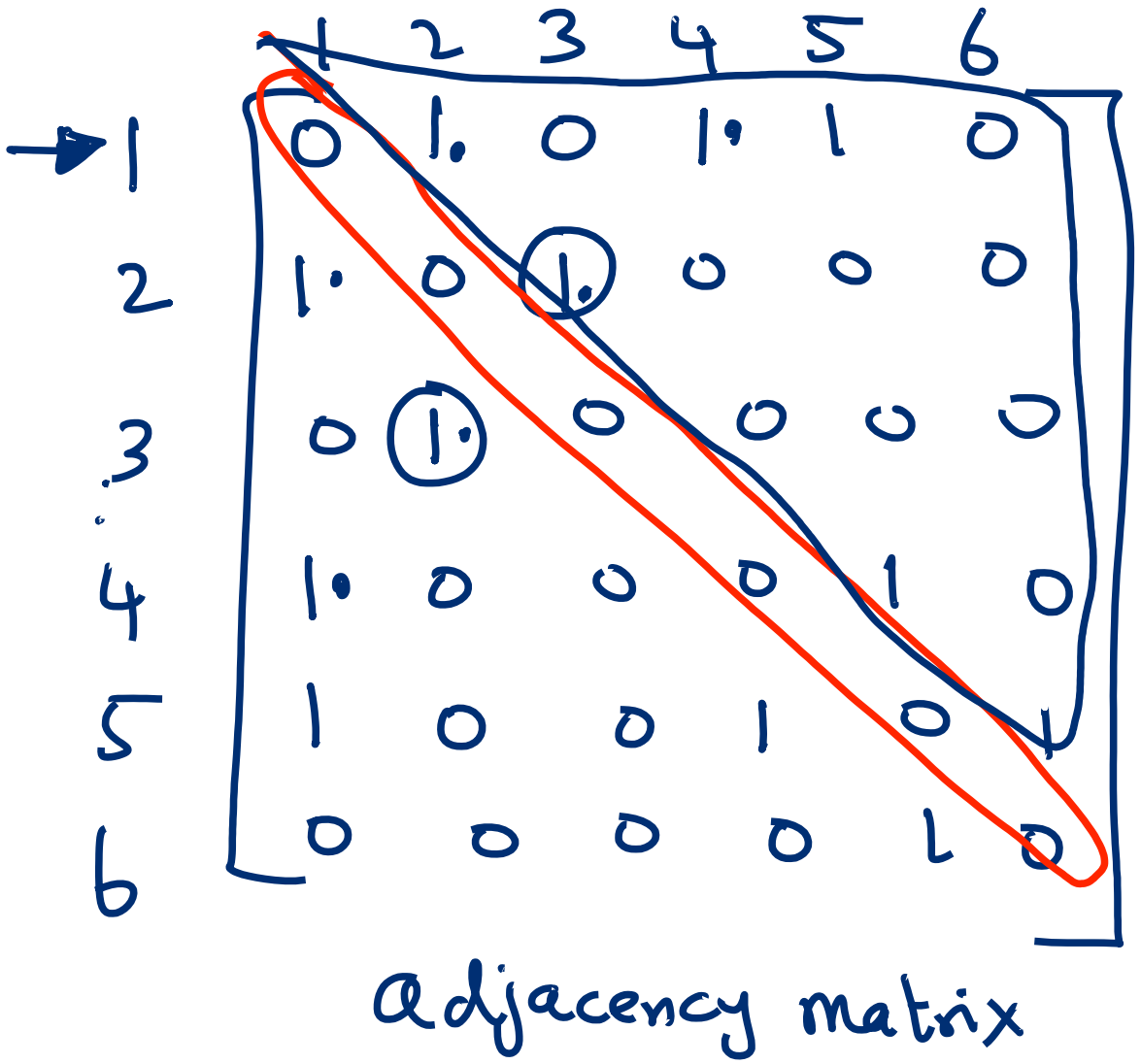


node (vertex)

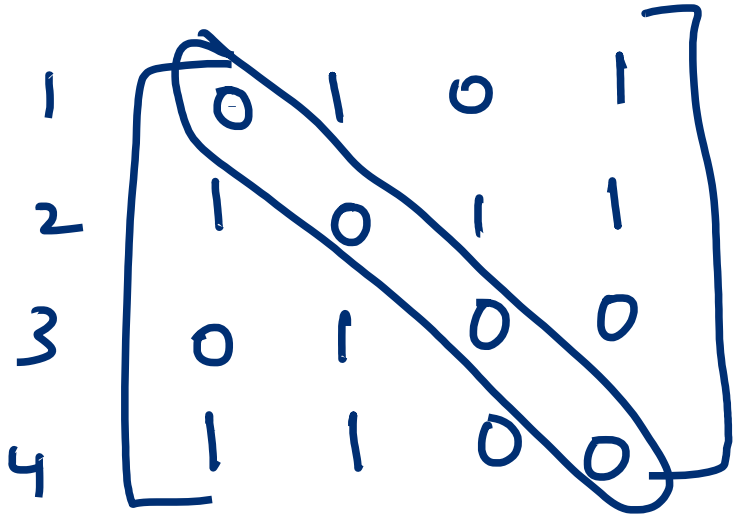
edge

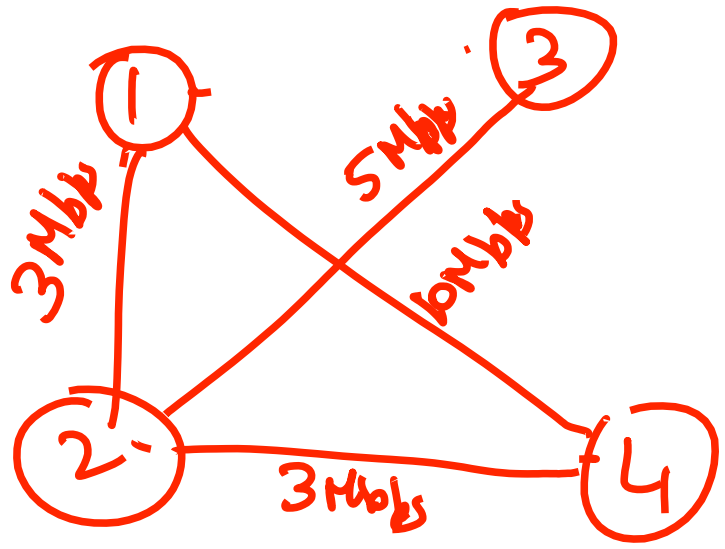
Graph

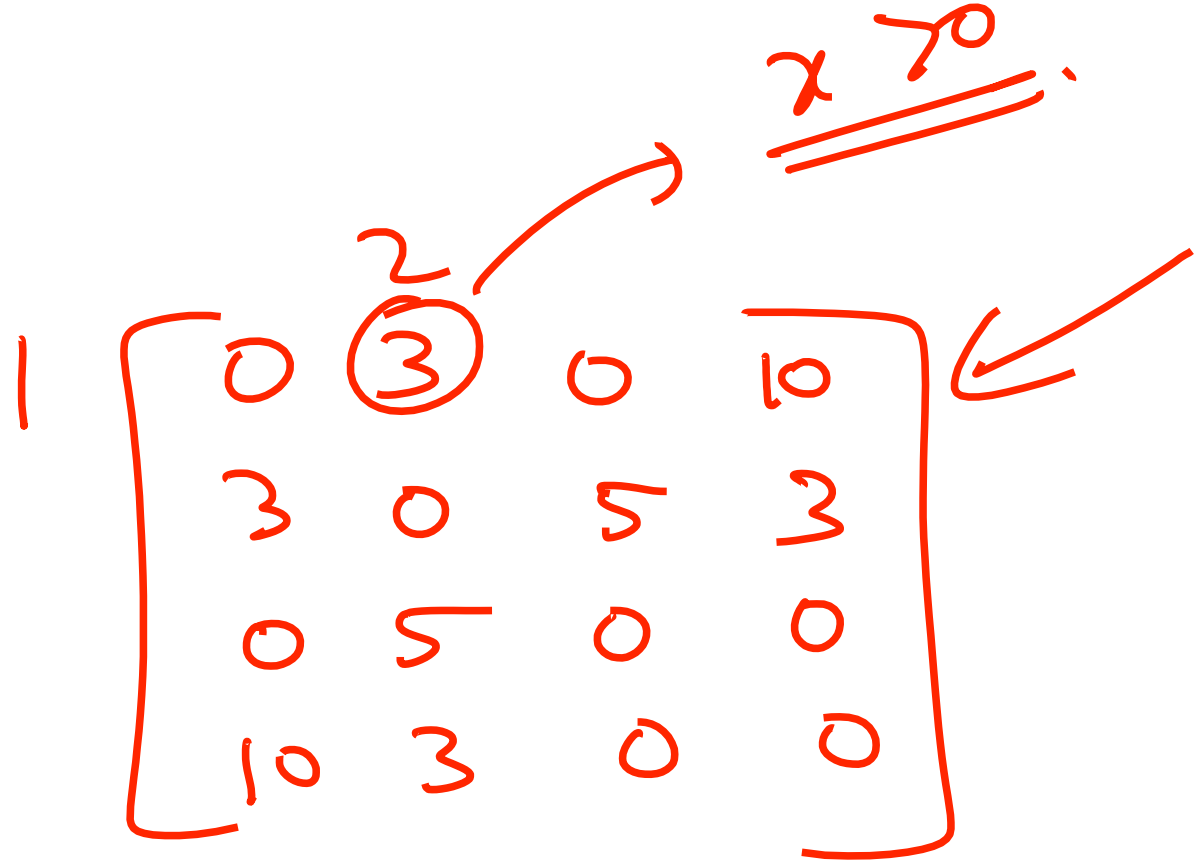
$G = (V, E)$



Adjacency matrix

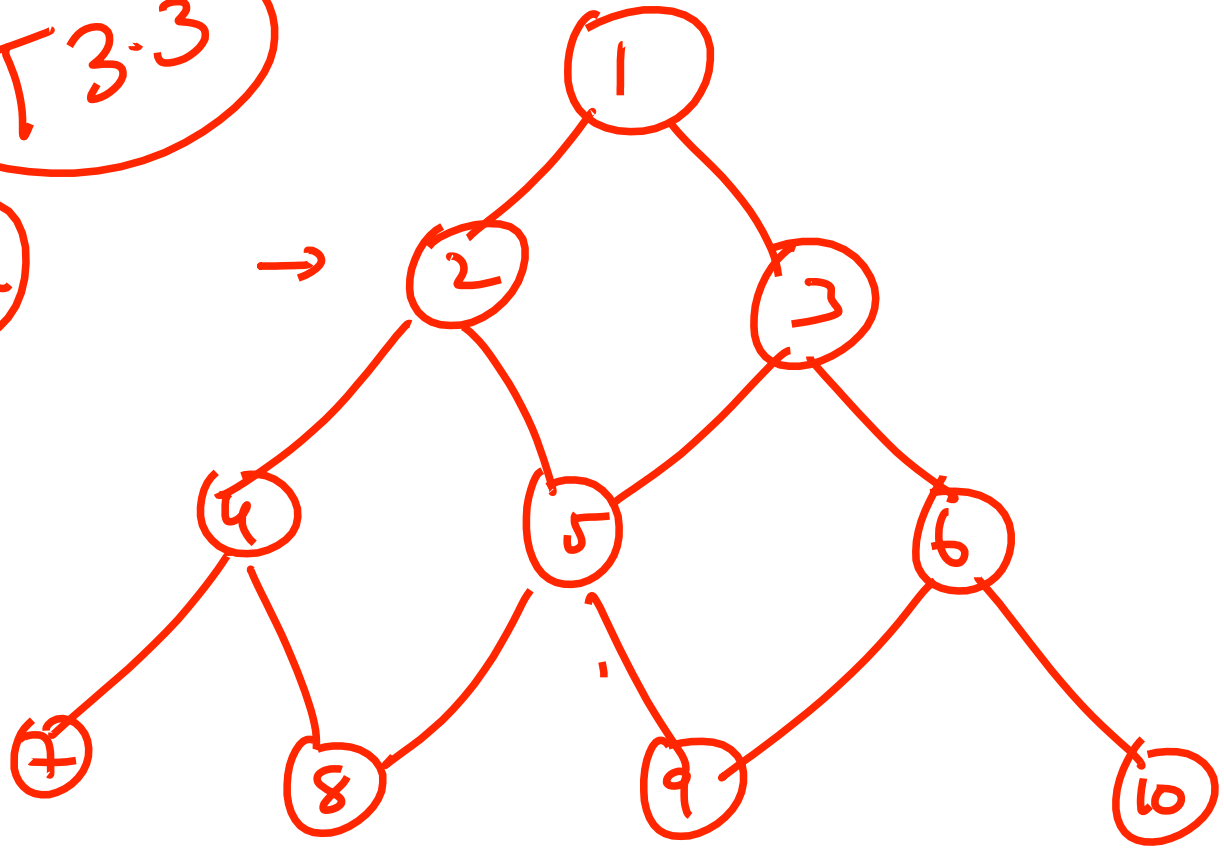

1 0 1 0 1
2 1 0 1 1
3 0 1 0 0
4 1 1 0 0


1
2
3
4
3Mbps
5Mbps
10Mbps
3Mbps

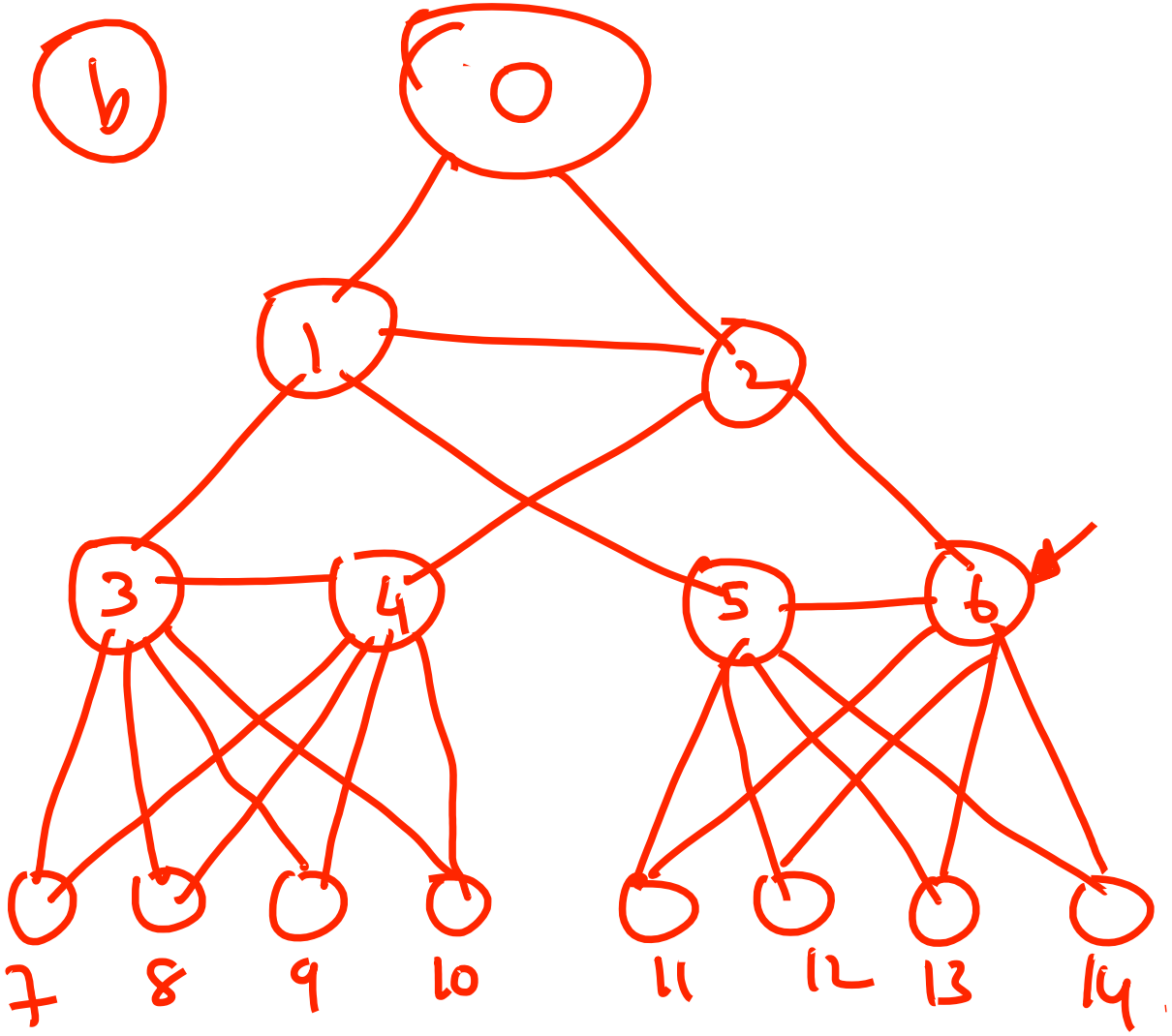

x 10
2
1
0 3 0 10
3 0 5 3
0 5 0 0
10 3 0 0

T3.3

(a)



	1	2	3	4	5	.	.	
1	0	1	1	0	.	.	.	0
2	1	0	0	1	1	0	.	0
3	1	0	0	0	1	1	0	. 0
4	0	1	0	0				1 1 . 0
.								
.								
.								
								0


b
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

3.4

	1	2	3	4	5
1	0	3	0	2	0
2	3	0	0	2	2
3	0	0	0	2	0
4	2	2	2	0	0
5	0	2	0	0	0

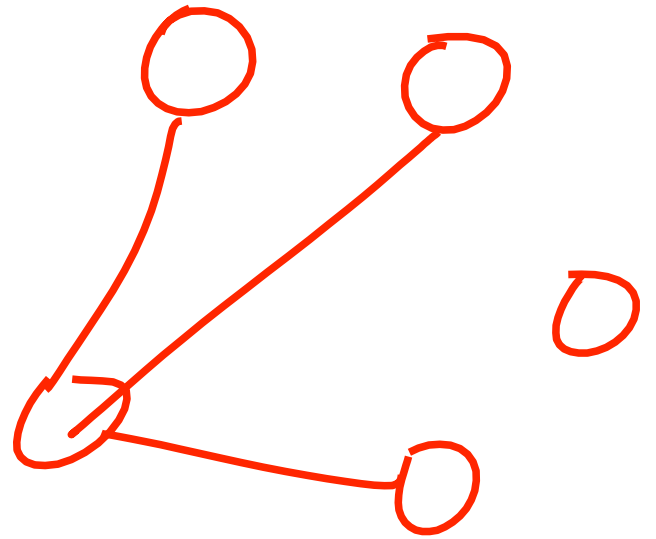
(b) (1) → 2, 4

(2) → 1, 4, 5

.

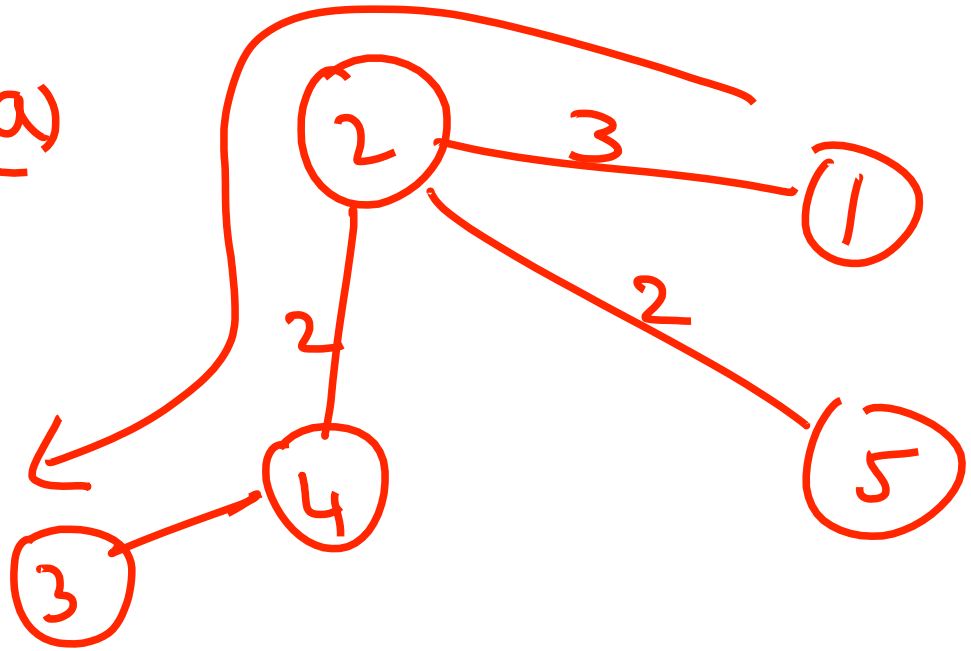
.

.



(c)

(a)



(d) degree of a node

(2) — 3 links incident on node 2

deg of node :

1	3
2	5 ←
3	6 ←
4	1
6	5
7	3

diameter of a graph

Maximum delay in that n/w.

Worst case delay in the network.

UROP