CSE

Final

Spanning tree

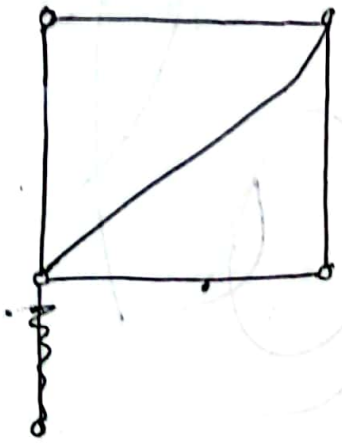
A spanning tree is a sub-graph of an undirected comp- connected graph which includes all the vertices of the graph with a maximum possible number of edges. If a vertix is missed then it is not a spanning tree.

Complete vertex n^{n-2}

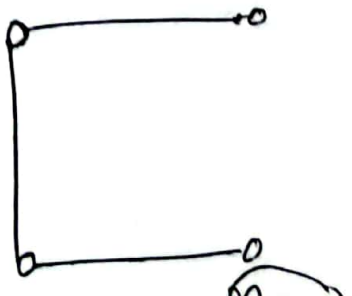
5^{5-2}

$= 5^3$

$= 125$



(I) Complete graph
(II) Sub graph হবে
(III) vertix travel
(IV) disconnected হওয়া যাবে না
(V) cycle থাকা যাবে না
(VI)



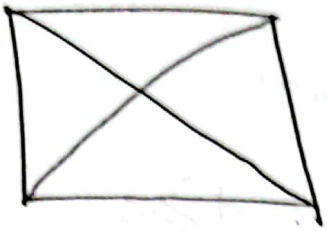
^nC_r

edge → E $C_{E'}$ — no of cycle

where, $E' = |V| - 1$

3A graph

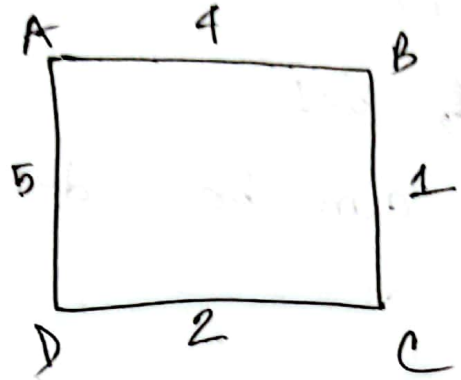
complete graph =



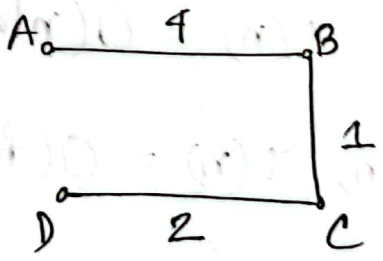
$\therefore V-2$

MST ⟶ minimum spanning tree

Weighted graph =



=



Master Theorem

if $T(n) = a\,T\left(\left[\frac{n}{b}\right]\right) + O(n^d)$

for constant $a > 0$, $b > 1$, $d \geq 0$

Or,

if $T(n) = a\,T\left(\left[\frac{n}{b}\right]\right) + f(n)$

for constant $a \geq 1$, $b > 1$

Order of growth can be determined using

1. if $f(n) < n^{\log_b a}$ then $T(n) = O(n^{\log_b a})$
2. if $f(n) = n^{\log_b a}$ ", $T(n) = O(n^{\log_b a} \log n)$
3. if $f(n) > n^{\log_b a}$ then, $T(n) = O(f(n))$

Example: $T(n) = 9T\left(\frac{n}{3}\right) + n$

Soln: $T(n) = aT\left(\left[\frac{n}{b}\right]\right) + f(n)$; $a \geq 1$, $b > 1$

Here, $a = 9$, $b = 3$; $f(n) = n$

Now, $n^{\log_b a} = n^{\log_3 9} = n^2$

Here, $f(n) < n^{\log_b a}$

$\Rightarrow n < n^2$

$\therefore T(n) = O(n^{\log_b a}) = O(n^2)$

1. $T(n) = T(n/3) + 1$

Here, $a = 1$, $b = 3$, $f(n) = 1$; $a \geq 1$, $b > 1$

Now, $n\log_b a = n\log_3 1$

$= n^0 = 1$

Here, $f(n) = n\log_b a$

then, $T(n) = O(n\log_b a \log n)$

$= O(1)$

$= O(1 . \log n)$

$= O(\log n)$

* $T(n) = 8T(n/2) + n^2$

$a = 8$, $b = 2$, $f(n) = n^2$; $a \geq 1$, $b > 1$

now, $n\log_b a = n\log_2 8 = n^3$

Here, $f(n) < n\log_b a$

then, $T(n) = O(n\log_b a)$

$= O(n^3)$

2. $T(n) = 3T\left(\frac{n}{4}\right) + n\log n$

Here, $a = 3 \quad b = 4 \quad f(n) = n\log n \; ; a \geq 1, b > 1$

now, $n\log_b a = n \log_4 3$

$= n^{0.792}$

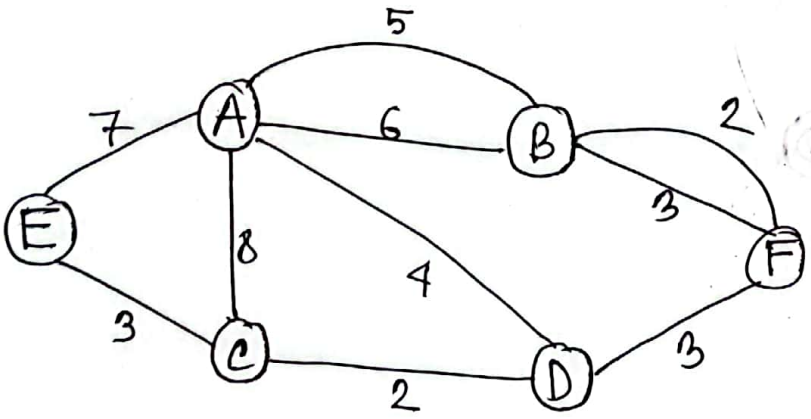
Here, $f(n) > n\log_b a$

So, $T(n) = O(f(n))$

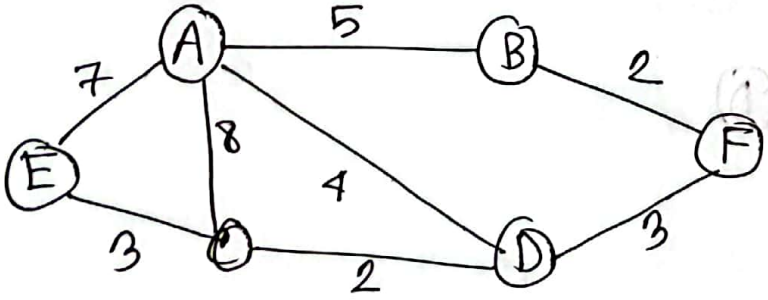
$= O(n\log n)$

Kruskal

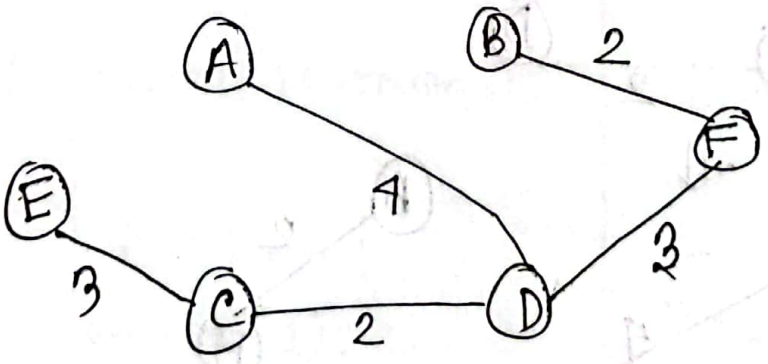
Subset of graph
No of cycle
No disconnected vertices
No parallel edge
Delete loop from graph.



⇒

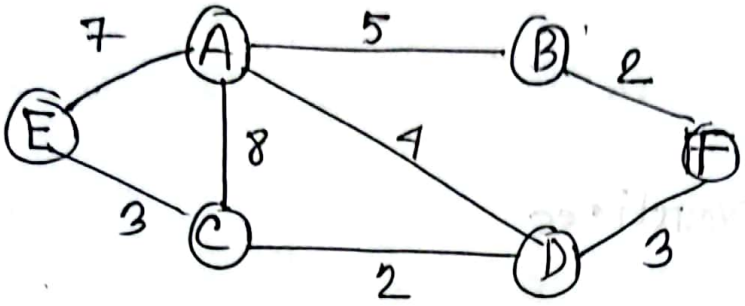


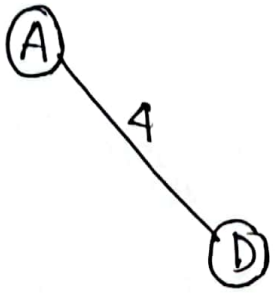
CD → 2
BF → 2
DF → 3
EC → 3
AD → 4
AB → 5
AE → 7
AC → 8

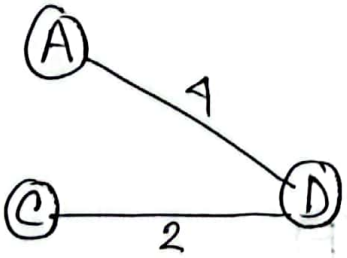


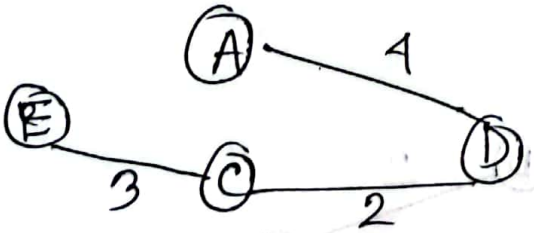
maximum weight = 14

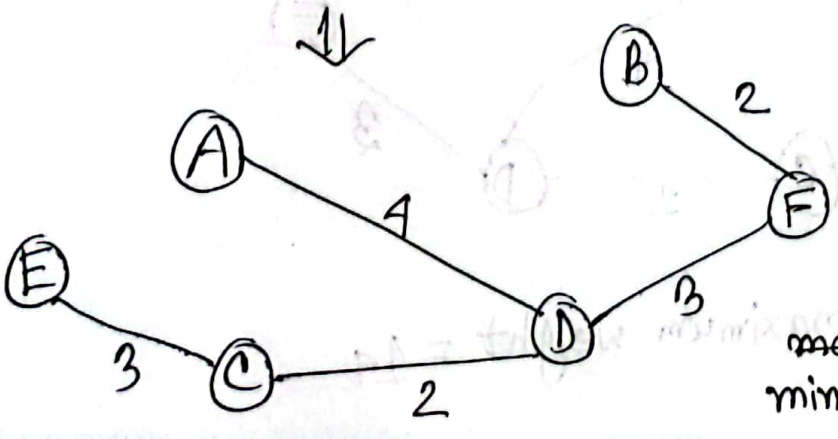
Prims











maxi: minimum spanning tree
minimum weight = 14

Recurrence Relation

Many Algorithm particularly divide and conqure algorithms have time complexities which are naturally modeled by recurrence relations.

A recurrence relation is an equation which is defined in terms of itself.

So, when an algorithm contain a recursive call to itself, it's running time can be defined by recurrence equation.

Example: Recurrence relation of merge sort.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + n, & \text{if } n > 1 \end{cases}$$

Ex-1

```
void func(int n)
{
  if (n<=0) return 0;
  else {
        print(n);
        func(n-1);
       }
}
```

T(n)
1
1
n-1

Soln: $T(n) = T(n-1) + c$

$$T(n) = \begin{cases} 1 & , \text{if } n \leq 0 \\ T(n-1) + c, & \text{if } n > 0 \end{cases}$$

Ex-2

```
void fun (int n)                    | T(n)
{                                   |
  if (n>0)                          | 1
for (i=1; i<=n; i++)                | n+1
    Print (n);                      | n*1
    func (n-1);                     | T(n-1)
}                                   |
```

Soln: $T(n) = T(n-1) + n$

$$T(n) = \begin{cases} 1, & \text{if } n \leq 0 \\ T(n-1) + n & \text{if } n > 0 \end{cases}$$

Ex-03

```
int fib(int n)                          | T(n)
{ if (n <= 0) return 0;                 |  1
  if (n == 1) return 1;                 |  1
  else
    { return fib(n-1) + fib(n-2);       | T(n-1) + T(n-2) + 1
    }
}
```

Soln: $T(n) = T(n-1) + 1 + T(n-2) + 1 + 1$

$= T(n-1) + T(n-2) + 3$

$= T(n-1) + T(n-2)$

$$T(n) = \begin{cases} 1, & \text{if } n <= 1 \\ T(n-1) + T(n-2) & \text{if } n > 1 \end{cases}$$

Ex-4

$$T(n) = \begin{cases} 1 & \text{if } n = 0 \\ T(n-1)+1 & \text{if } n > 0 \end{cases}$$

Substitution

$T(n) = T(n-1) + 1$

$= [T(n-2)+1] + 1$

$= T(n-2) + 2$

$= [T(n-3)+1] + 2$

$= T(n-3) + 3$

$\vdots$

$= T(n-k) + k$

$= T(n-n) + n$

$= T(0) + n$

$= 1 + n = n$

$\therefore T(n) = O(n)$

$T(n) = T(n-1) + 1$

$T(n-1) = T(n-1-1) + 1$

$= T(n-2) + 1$

$T(n-2)$

Let,

$n - k = 0$

$n = k$

$\therefore k = n$

Ex-5

$$T(n) = \begin{cases} 1 & , \text{if } n=0 \\ T(n-1)+n & , \text{if } n>0 \end{cases}$$

$T(n-1) = T(n-1-1)+n$
$= T(n-2)+n$

$T(n) = T(n-1)+n$

$= [T(n-2)+n]+n$

$= T(n-2)+2n$

$= [T(n-3)+n]+2n$

$= T(n-3)+3n$

.
.
.

$= T(n-k)+kn$

Let,
$n-k=0$
$\Rightarrow n=k$

$= T(n-n)+n.n$

$= T(0)+n^2$

$T(n) = O(n^2)$

H.W

$$T(n) = \begin{cases} 1 & \text{, if } n=1 \\ T(n/2)+c & \text{, if } n>1 \end{cases}$$

~~$T(n/2)+c$~~

$T(n) = T(n/2) + c$

$= [T(n/4)+c] + c$

$= T(n/2^2) + 2c$

$= [T(n/2^3)+c] + 2c$

$= T(n/2^3) + 3c$

$\vdots$

$= T(n/2^k) + kc$

$= T(n/n) + kc$

$= T(1) + kc$

$= 1 + n\log n$

$= O(n\log n)$

let,

$n/2^k = 0$

$\Rightarrow n = 2^k$

$\log n = \log 2^k$

$\log n = k \log 2$

$k = \log n$

Substitution Method

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ n * T(n-1) & \text{if } n > 1 \end{cases}$$

$T(n) = n * T(n-1)$ ——— (I)

$T(n-1) = (n-1) * T(n-1-1)$

$= (n-1) * T(n-2)$ ——— (II)

$T(n-2) = (n-2) * T(n-2-1)$

$= n-2 * T(n-3)$ ——— (III)

$T(n) = n * T(n-1)$

$= n * (n-1) * T(n-2)$

$= n * (n-1) * (n-2) * T(n-3)$

⋮

$= n * (n-1) * (n-2) * T\{n-(n-1)\}$

$= n * (n-1) * (n-2) * T(n-n+1)$

$= n * (n-1) * (n-2) * T(1)$

$= n * (n-1) * (n-2) * 1$

$= O(n^n)$

Sieve of Eratosthenes

~~1~~ 2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~
~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~
17 ~~18~~ 19 ~~20~~.

Code

```
Loop 2 : N {
if (visited[i] == true) continue)
else {
    for( j = i*2 ; j*j <= N ; j += i)
    {   visited = true;
    }
}
}
```

69

$i = 2$ $\quad j = 9$ $\quad j += 2 = 9+2 = 11$

$j = 6 * 2$

$j = 6$ $\quad j += 6+3$

$i = 3$ $\quad j = 6$

$j = 9$

$j = 12$

$j = 15$

Recursion time method

Drawing a picture of the back substitution process given a idea of what is going on.

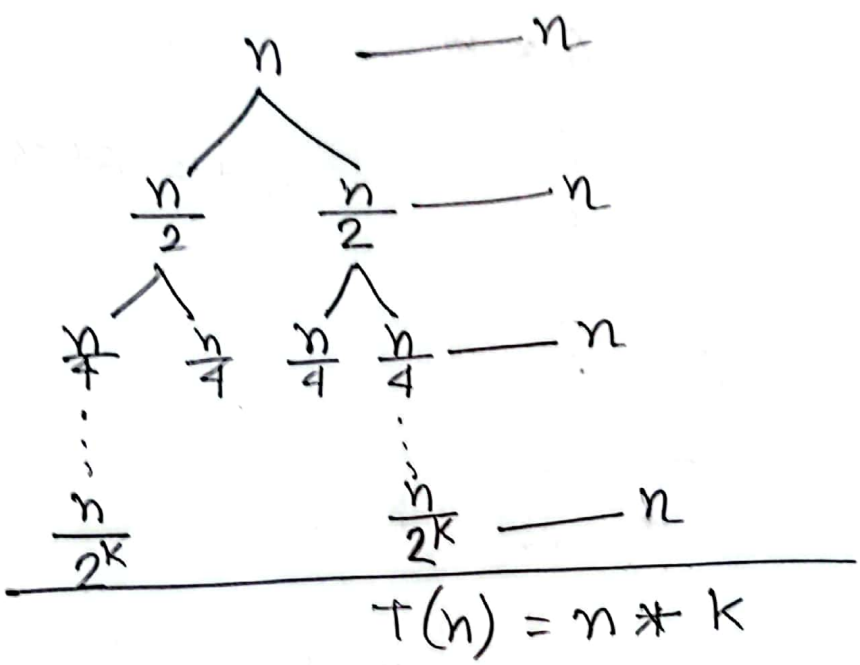
We must keep track of two things:

1). the size of remaining argument to the recurrence.

2) the additive stuff to be accumulated during this call.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n/2) + (n) & \text{if } n > 1 \end{cases}$$

↓ root node



$T(n) = n * k$

Let, $\frac{n}{2^k} = 1$

$\Rightarrow n = 2^k$

$\Rightarrow \log_2 n = k$

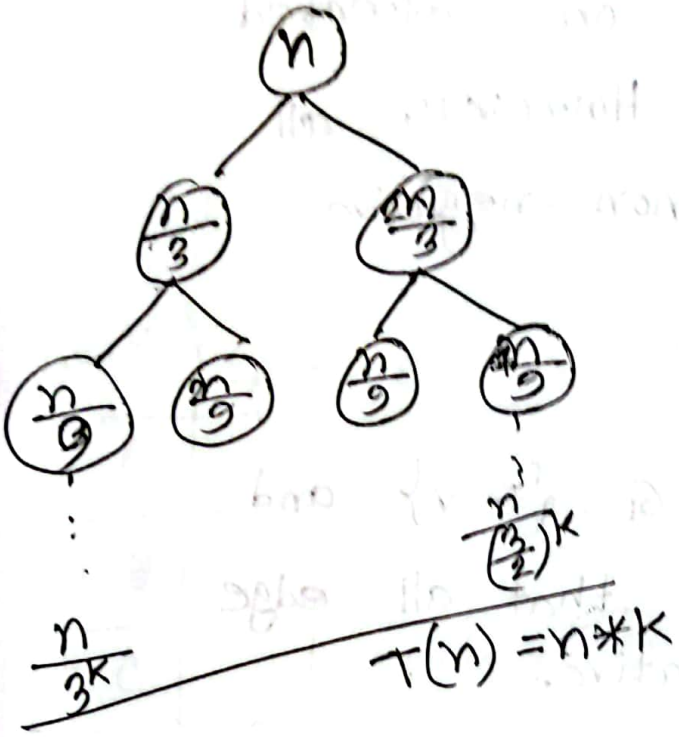
Here,

$T(n) = O(n*k)$

$= O(n \times \log_2 n)$

$= O(n \log_2 n)$

$$* T(n) = \begin{cases} 1 , & \text{if } n = 1 \\ T(n/3) + T(2n/3) + n & \text{if } n > 1 \end{cases}$$



$T(n) = T(n/3) + T(2n/3) + n$

$T(n/3) = T(n/9) + T(2n/9) + n/3$

$= T(n/3^2) + T(2n/3^2) + n/3$

$T(2n/3) = T(2n/9) + T(4n/9) + 2n/3$

$= T(2n/3^2) + T(2^2 n/3^2) + 2n/3$

$\therefore T(n) = T(n/3) + T(2n/3) + n$

$= T(n/3^2) + T(2n/3^2) + n/3 + T(2n/3^2) + T(2^2 n/3^2) + 2n/3$

let, $\frac{n}{(3/2)^k} = 1$

$\Rightarrow n = (3/2)^k$

$\Rightarrow \log_{(3/2)} n = k$

$\Rightarrow k = \log_{(3/2)} n$

Hence,

$T(n) = k * n$

$= \log_{(3/2)} n * n$

So, $T(n) = O(n \log_{(3/2)} n)$

$T(n) = \begin{cases} 1 , & \text{if } n = 1 \\ 2T(n/2) + n , & \text{if } n > 1 \end{cases}$

Dijktra

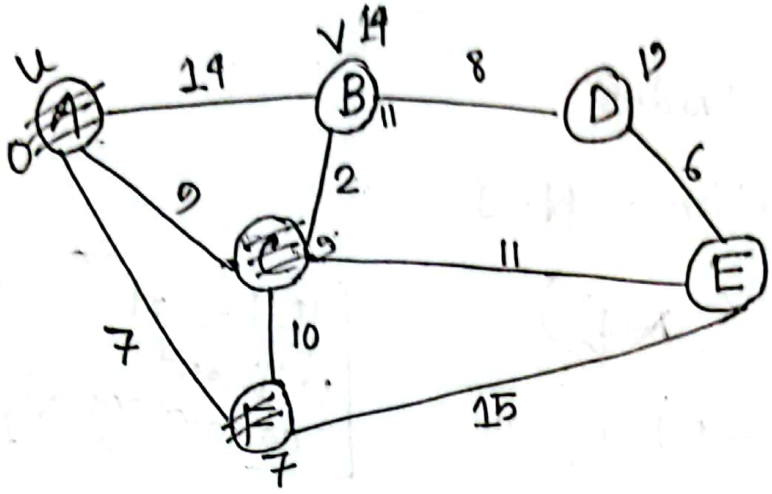
is a solution to the single source ~~sort~~ shortest path. Works on directed and undirected graph. However, all edges must have non-negative weights.

Greedy Approach

Input : Weighed graph ⊛ $G = \{E, V\}$ and source vertex such that all edge weights are non-negative.

Output: Length of the shortest path from a given source vertex to other vertices.

if $(d(u) + c(u,v) < d(v))$ then
update : $d(u) = d(u) + c(u,v)$ ~~then~~ } relaxation



$9+2$

$11 < 14$

$9 = 9+$

$11+8 < \infty$

$19 < \infty$

$25 < 20$

Visited

	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
F		14	9	∞	∞	[7]
C		14	[9]	∞	22	
B		[11]		∞	20	
D				[19]	20	
E					[20]	

E, C, A

↳ A, C, E

$9 + 11$

$= (20)$

Bellman ford

N — Nodes

Edges Relaxation = N-1

$$O(|E|(V-1))$$
$$= O(n(n-1))$$
$$= O(n^2)$$

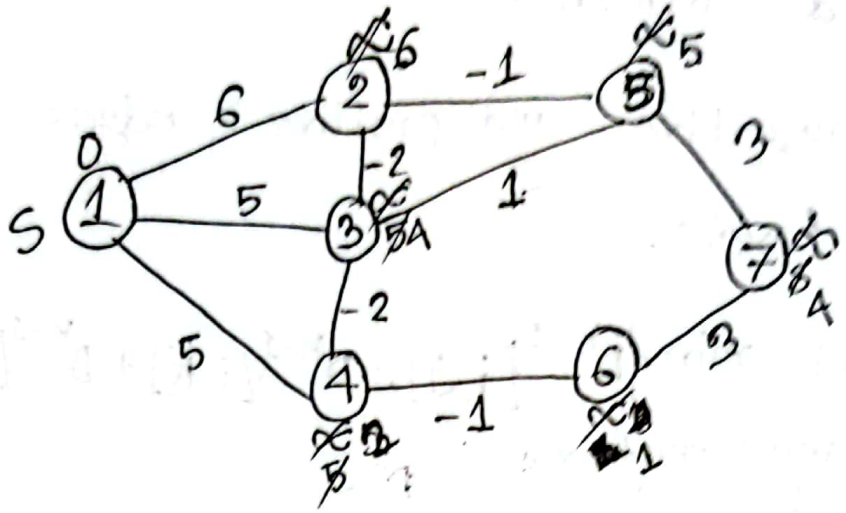
$|E| = n$ without complete graph

complete graph ⟶ $|E| = \frac{V(V-1)}{2} = \frac{n(n-1)}{2}$

$$O(|E|(V-1))$$
$$= O\left(\left\{\frac{n(n-1)}{2}(n-1)\right\}\right)$$
$$= O\left(\left\{\frac{n^2-n}{2}(n-1)\right\}\right)$$
$$= O(n^3)$$

if $(d(u) + c(u,v)) < d(v)$

update: $d(v) = d(u) + c(u+v)$

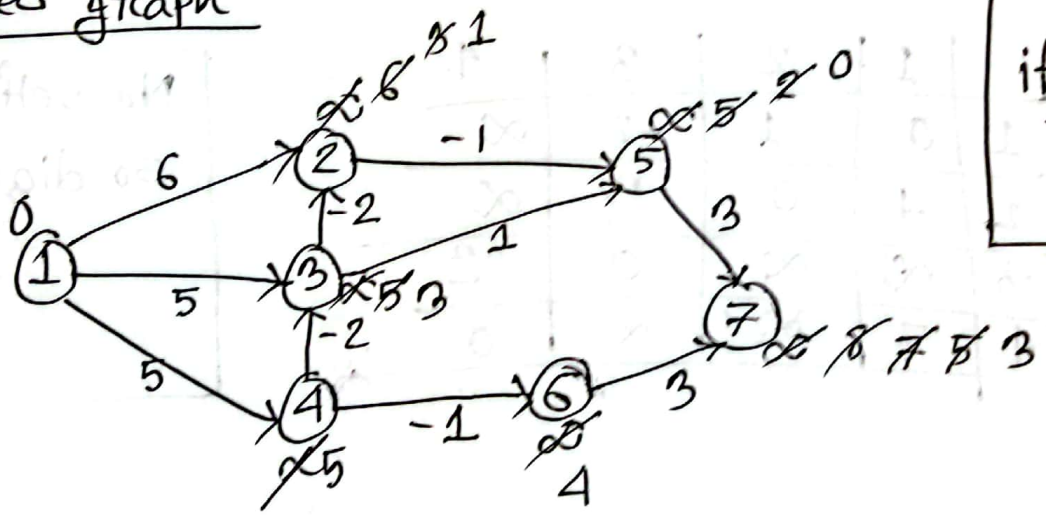


n-1/nth
↳ Update
2(n
Negative cycle
on(5

Edges: {1,2}, {1,3}, {1,4}, {2,5}, {3,2}, {3,5}, {4,3}, {4,6}, {5,7}, {6,7}.

1st
2nd
3rd

Directed graph



if it has 4th time update then negative cycle

Edges: {1,2} {1,3} {1,4} {2,5} {3,2} {3,5} {4,3} {4,6} {5,7} {6,7}

1st
2nd
3rd
4th
5th

Floyd Warshel

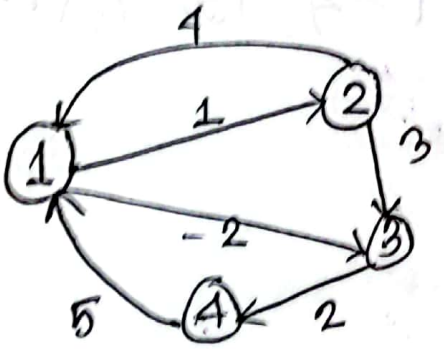
→ Works with negative and positive edges

→ No negative cycle.

Formula: $D^k[i,j] = \min\{D^{k-1}[i,j], D^{k-1}[i,k] + D^{k-1}[k,j]\}$

↳ Distance matrix

$k = 0, 1, 2$



$D^k \neq D^4$
vertex

$[4, 1] = 5$

Distance Matrix : D^0

	1	2	3	4
1	0	1	-2	∞
2	4	0	3	∞
3	∞	∞	0	2
4	5	∞	∞	0

No self Loop
so diagonals
0

$\infty, 5+1$

D^1 :

	1	2	3	4
1	0	1	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	5	6	3	0

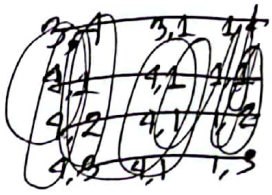
$D^1[2,3] = \min\{D^{1-1}[2,3],\ D^{1-1}[2,1] + D^{1-1}[1,3]\}$

$= \min\{3,\ 4 + (-2)\} = 2$

$D^1[3,1] = \min\{D^0[3,1],\ D^0[3,1] + D^0[1,1]\}$

$= \min\{\infty,\ \infty + 0\}$

$= \infty$



D^2 :

	1	2	3	4
1	0	1	-2	∞
2	4	0	2	∞
3	∞	∞	0	2
4	5	6	3	0

$D^2[4,3] = \min(D^1[4,3],\ D^1[4,2] + D^1[2,3])$

$= (\infty,\ 1+\infty) = \infty \quad (\infty, 0+\infty) = \infty$

$5, \infty+$

$3, \infty+2$

$\infty, \infty+\infty, \infty+0 \quad 2, \infty+$

Comput

D^3:

	1	2	3	4
1	0	1	-2	0
2	4	0	2	4
3	∞	∞	0	2
4	5	6	3	0

D^3: $\min\{ D^2[i,j], D^2[i,k] + D^2[k,j] \}$

4,3 4,3 3,3

3 ; 3+6

(1,2)

1,4 1,3 3,4

∞ , -1+2

2,4 ; 2,3 3,4

0 2 , 2+0

∞ , 2+2

3,4 4,1 , 3,3+ 3,2

2,0+2

∞ 0 ∞

4,2 4,

3,1 3,4 4,1

∞ , 2+5

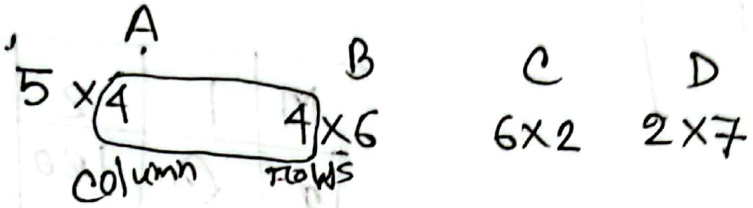
D^4:

	1	2	3	4
1	0	1	-2	0
2	-4	0	2	4
3	7	8	0	2
4	5	6	3	0

Matrix chain Multiplication

- is an optimization problem
- most efficient way to multiply a given sequence of matrices.

Suppose we have four matrices.



Step 1

M[1,1] M[2,2] M[3,3] M[4,4] = 0

A B C D

Step 2

M[1,2] M[2,3] M[3,4] = 2 matrix

A B B C C D

5x4 4x6 4x6 6x2 6x2 2x7

5 x 4 x 6 = 120

4 x 6 x 2 = 48

6 x 2 x 7 = 84

Step 3

M[1,3] = A → C

A . (B . C) (A . B) . C

5x4 (4x6, 6x2)

M[1,1] + M[2,3] + 5x4x2

Step-4 M[2,4] (B-D)

B.(C.D) | (B.C).D

4X6(6X2 . 2X7) | (4X6 .6X2) (2X7)

M(2,2) + M(3,4) + 4X6X7 | M[2,3] + M[4,4] + 4X2X7

= 0 + 84 + 168 = 252 | = 48 + 0 + 56 = 104 Min

M	1	2	3	4
1	0	120	88	158
2		0	48	104
3	no need		0	84
4	to fill			0

Step5 M[1,4]

A.(B.C.D) (A.B).(C.D) (A.B.C).D

A.(B.C.D)

↓

M[1,1] + M[2,4] + 5x4x7

= 0 + 104 + 140

= 244

(A.B)(C.D)

M[1,2] + M[3,4] + 5X6X7

= 120 + 84 + 210 = 414

(A.B.C).D

M[1,3] + M[4,4] + 5X2X7

= 88 + 0 + 70 = 158 Min

Formula: $M[i,j] = \text{Min}\{M[i,j] + M(k+1,j) + d_{i-1} * d_k * d_j$