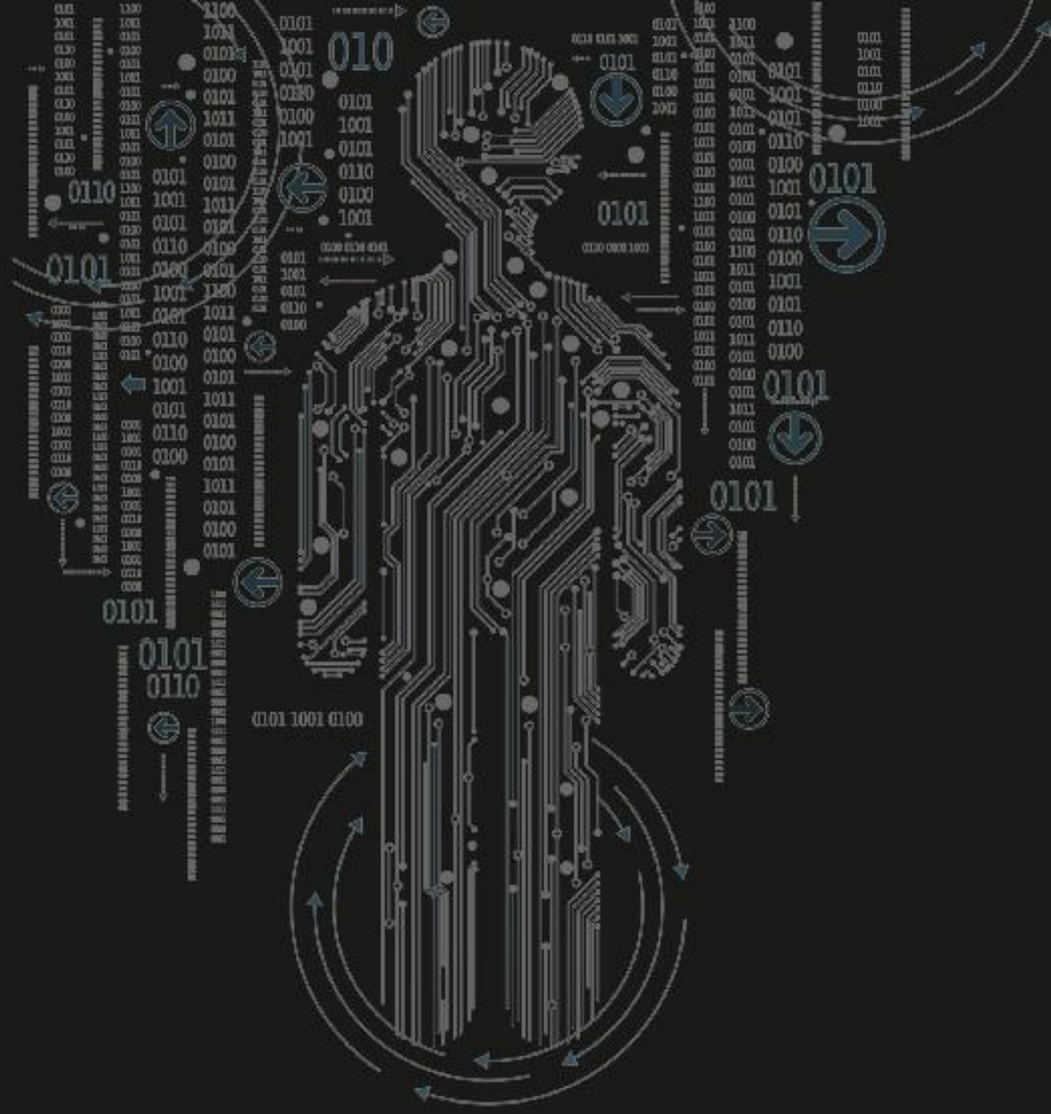
BYJU'S EXAM PREP

ALGORITHMS

Time and Space Complexity



ABOUT ME

- Teaching Experience: 14+ Years
- Achievements: B.Tech, M.Tech, (Ph.D.)
- Achievements:
  - CS Senior Faculty
  - Mentored More Than 30,000+ Students
  - GATE Qualified
  - UGC-NET Qualified
  - SET Qualified
  - CISCO Certified

SATYA NARAYANA KAITEPALLI

BYJU'S EXAM PREP

Computer Science & Engineering

Algorithms

Searching, sorting, hashing. Asymptotic worst-case time and space complexity. Algorithm design techniques: greedy, dynamic programming and divide-and-conquer. Graph traversals, minimum spanning trees, shortest paths.

Last 5 Year's Trend (2018-2022) for GATE

BYJU'S EXAM PREP

ALGORITHMS	
No. of Questions	4 to 5
Marks	6 to 8
Frequently Asked Topics	Recurrence Relations, Greedy Algorithms, Minimum Spanning Trees, Sorting Algorithms

Weightage

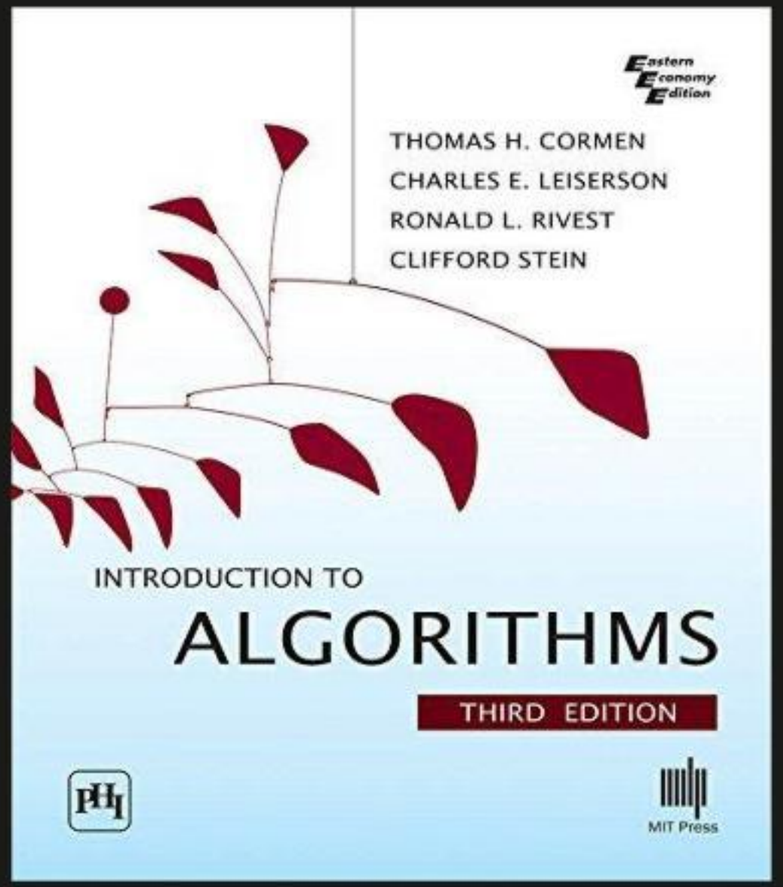
BYJU'S EXAM PREP

Algorithms

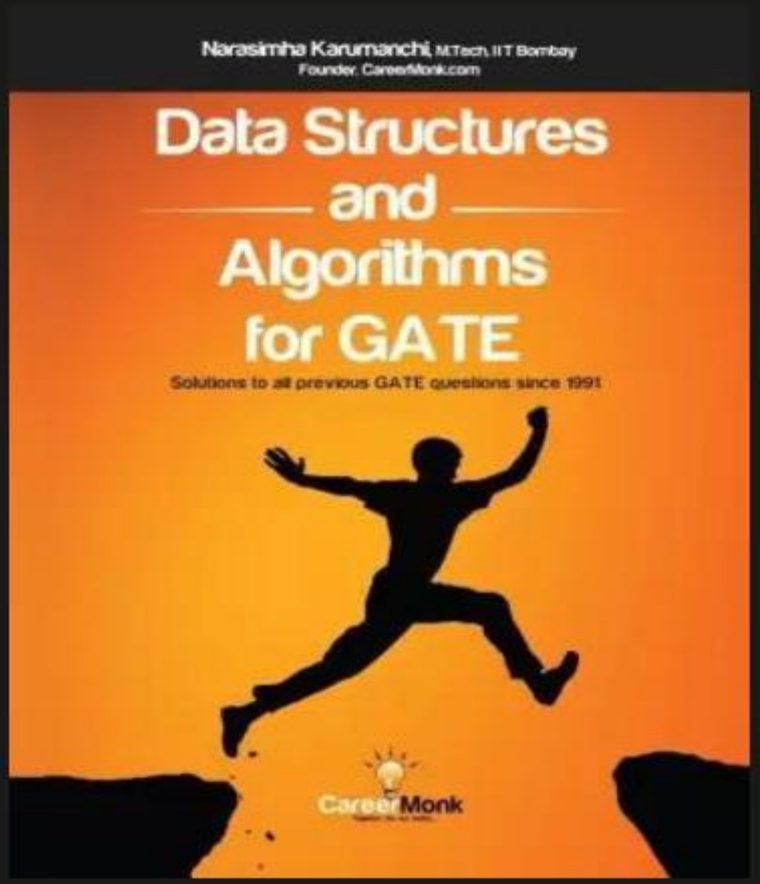
YEAR	2022	2021	2020	2019	2018	2017	2016	2015	2014	2013
MARKS	7	8	6	7	7	5	11	10	9	8

Expected weightage : 6+ marks

Reference Books



Book Name: Introduction to Algorithms
Author: Thomas H. Cormen
Publisher: PHI Publications



Book Name: Data Structures & Algorithms for GATE
Author: Narasimha Karumanchi
Publisher: Career Monk Publications

Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

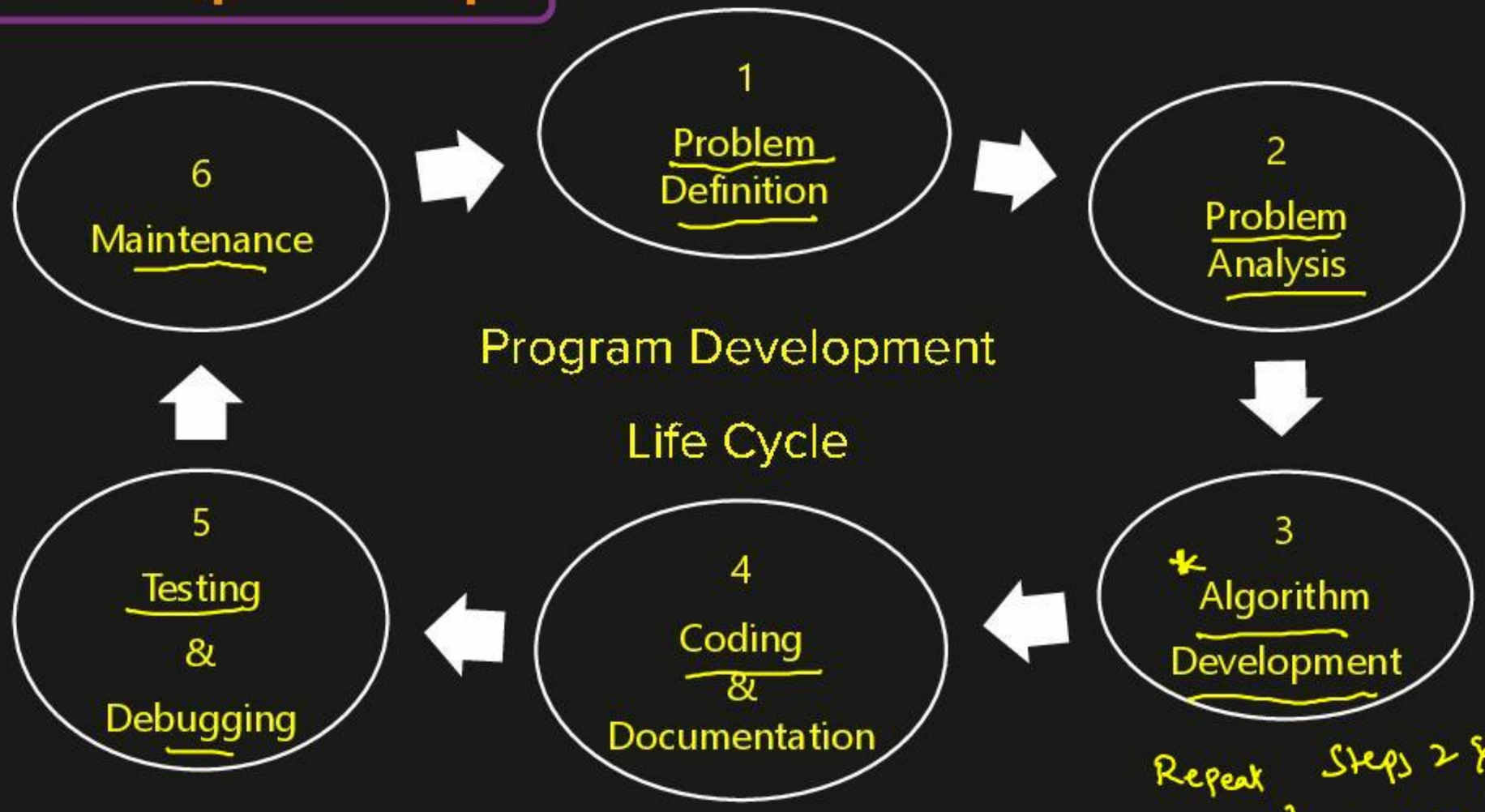
Computer

- It is an electronic device, which accepts input, process it, store and produces results.
- A Computer is Combination of Hardware and Software.
- Computer Hardware include CPU, Memory Devices, I/O devices and other peripherals.
- A software is set of instructions expressed using any programming language.

BYJU'S EXAM PREP

"for i=1 to n, do

Program Development Steps



Repeat Steps 2 & 3 until 'i' reaches 'n'

Program Development Steps

BYJU'S EXAM PREP

Algorithm

- Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.
- Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

BYJU'S EXAM PREP

Characteristics

i/p: n = 542671

- Unambiguous – Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.
- Input – An algorithm should have 0 or more well-defined inputs.
- Output – An algorithm should have 1 or more well-defined outputs, and should match the desired output.
- Finiteness – Algorithms must terminate after a finite number of steps.
- Feasibility – Should be "feasible" with the available resources.
- Independent – An algorithm should have step-by-step directions, which should be independent of any programming code.

BYJU'S EXAM PREP

Algorithm Analysis

Efficiency of an algorithm can be analyzed at two different stages, before implementation and after implementation. They are the following –

- A Priori Analysis – This is a theoretical analysis of an algorithm. Efficiency of an algorithm is measured by assuming that all other factors (for example, processor speed) are constant and have no effect on the implementation. oximate statistics]
- A Posterior Analysis – This is an empirical analysis of an algorithm. The selected algorithm is implemented using programming language. This is then executed on target computer machine. In this analysis, actual statistics like running time and space required, are collected.

Problem: Visit Delhi from Hyderabad.

5000/- 5 days

Algo 1

1. Start
2. Withdraw money
3. Pack
4. Start walking
5. Keep walking till you reach Delhi

- 3000/-
- 7 days

Algo 2

1
2
3
4. Book Bus

2500/-
20 hours

Algo 3

1
2
3
4. Train

2800/-
16 hours

Algo 4

1
2
3
4. Flight

8000/-
4 hours

- - - -

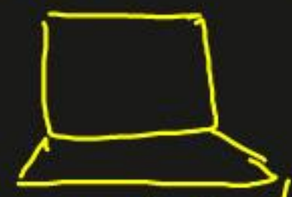
BYJU'S EXAM PREP

Efficiency Of an Algorithm

Suppose X is an algorithm and n is the size of input data, the time and space used by the algorithm X are the two main factors, which decide the efficiency of X.

- Time Factor – Time is measured by counting the number of key operations such as comparisons in the sorting algorithm. [Execution time]
- Space Factor – Space is measured by counting the maximum memory space required by the algorithm.

Machine A



	Algo1	Algo2
i/p	10 sec, Small	7 sec, Small
10 sec very Big i/p		25 sec

Machine B

	Algo1	Algo2
	5 sec, Small	4 sec, Small
	5 sec	20 sec

BYJU'S EXAM PREP

Space Complexity

Space complexity of an algorithm represents the amount of memory space required by the algorithm in its life cycle. The space required by an algorithm is equal to the sum of the following two components –

- A fixed part that is a space required to store certain data and variables, that are independent of the size of the problem. For example, simple variables and constants used, program size, etc.
- A variable part is a space required by variables, whose size depends on the size of the problem. For example, dynamic memory allocation, recursion stack space, etc.

BYJU'S EXAM PREP

Time Complexity

Time complexity of an algorithm represents the amount of time required by the algorithm to run to completion. Time requirements can be defined as a numerical function T(n), where T(n) can be measured as the number of steps, provided each step consumes constant time.

T(n) ⇒ Time Taken by algorithm on input "n"

BYJU'S EXAM PREP

Types Of Statements in an Algorithm / Program

- An Algorithm or Program comprises of various types of expressions or statements. They include:
- Declaration / Definition / Initialization Statements
- I/O Statements
- Computational / Assignment Statements
- Control Statements

int a; void x(void);

include, # define,

int x=5; void x(int i)
{
}

scanf(), Printf(), getch(), Puts()

i=3;
x=y;
x= i+i*i/i;
i = fun(5);

if, if-else, switch
loops, recursive calls, function calls

BYJU'S EXAM PREP

Examples

Declaration | Definitions | Initialization | Assignment | i/o statements

— Constant time. ⇒ $O(1)$

Examples : Declaration : int x; int fun(void);

Definitions : Macro definitions (or) function definitions

Initialization : int x = 5;

Assignment : x = y (or) x = z * y - i/j;

input : gets(str); getchar(), scanf() ---

output : Puts(str), Putchar(), Printf() ---

Time complexity of below code? ⇒ O(1)

```
#include <stdio.h>
void main( )
{   int i=5, j, k;
    Printf("Enter j value");
    Scanf("%d", &j);
    k=i*j+i;
    Printf("k value is %d", k);
}
```

Macro def → 1

Function def → 1

Initialization, Declaration → 1

o/p → 1

o/p → 1

Assignment → 1

o/p → 1

7 units of time

⇒ O(1)

BYJU'S EXAM PREP

Examples

Conditional statements : Max {Times taken by them}

```
if ( )
{
  ---
  ---
}
else
{
  ---
  ---
}
```

n

n^2

$n + n^2 \Rightarrow O(n^2)$ dominating n^{th} term

```
if ( )
{
}
else if ( )
{
}
else
{
}
```

(1)

$O(n^2)$

n^2

n

$= 1 + n + n^2$

```
if (     )
{                    n^2
}
Switch (     )
{ Case 1:
  Case 2:            n^4
  }
}
```

$n^2 + n^4 \Rightarrow O(n^4)$

$7+n^2$

$\Rightarrow 7\cdot n^0 + 1\cdot n^2$

Iterative statements (loops)

The time taken by a single loop, depends up on No. of times statements inside the loop Execute.

Ex:

①
```
for(i=1; i<=n; i++)
{
    // Stmts        n times
}
```

②
```
for (i=n; i>=1; i=i-4)
{
    // Stmts        n-4 times
}
```

$O(n)$

③
```
for(i=1; i<=n; i=i*2)
{ - - -        (log n) times
}
```

$i=1 \quad 2^1 \quad 2^2 \quad 4 \quad 8 \quad \ldots \quad 2^k$

$1<n \quad 2<n \quad 4<n \quad 2^k<n$

$2^k = n \Rightarrow k = \lceil \log_2 n \rceil$

Nested loops :

Multiplication of all loops Execution time.

```
for( i=1; i<=n; i=i+1 )        n times
{
    for( j=1; j<=n; j++)       n times
    {
        - - -
        - - -
    }
}
```

Total $n * n = O(n^2)$

```
for(    )   - n^2 times
{
    for(    )
    {       - n times
        for(    )
        {   - logn times
        }
    }
}
```

$n * n^2 * \log n$

$O(n^3 \log n)$

$i = n;\ i >= 1;\ i = i/2$

$i = n \quad n/2 \quad n/2^2 \quad \text{----} \quad \boxed{n/2^k} = 1 \quad 1/2 = 0$

$n/2^k$

$n >= 1 \quad n/2 >= 1 \quad n/2^2 >= 1 \quad 1 >= 1 \quad 0 >= 1$ false

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

$$\Rightarrow n = 2^k$$

$$\Rightarrow k = \log_2 n$$

BYJU'S EXAM PREP
4.5 Google Play Review
GET IT ON Google Play

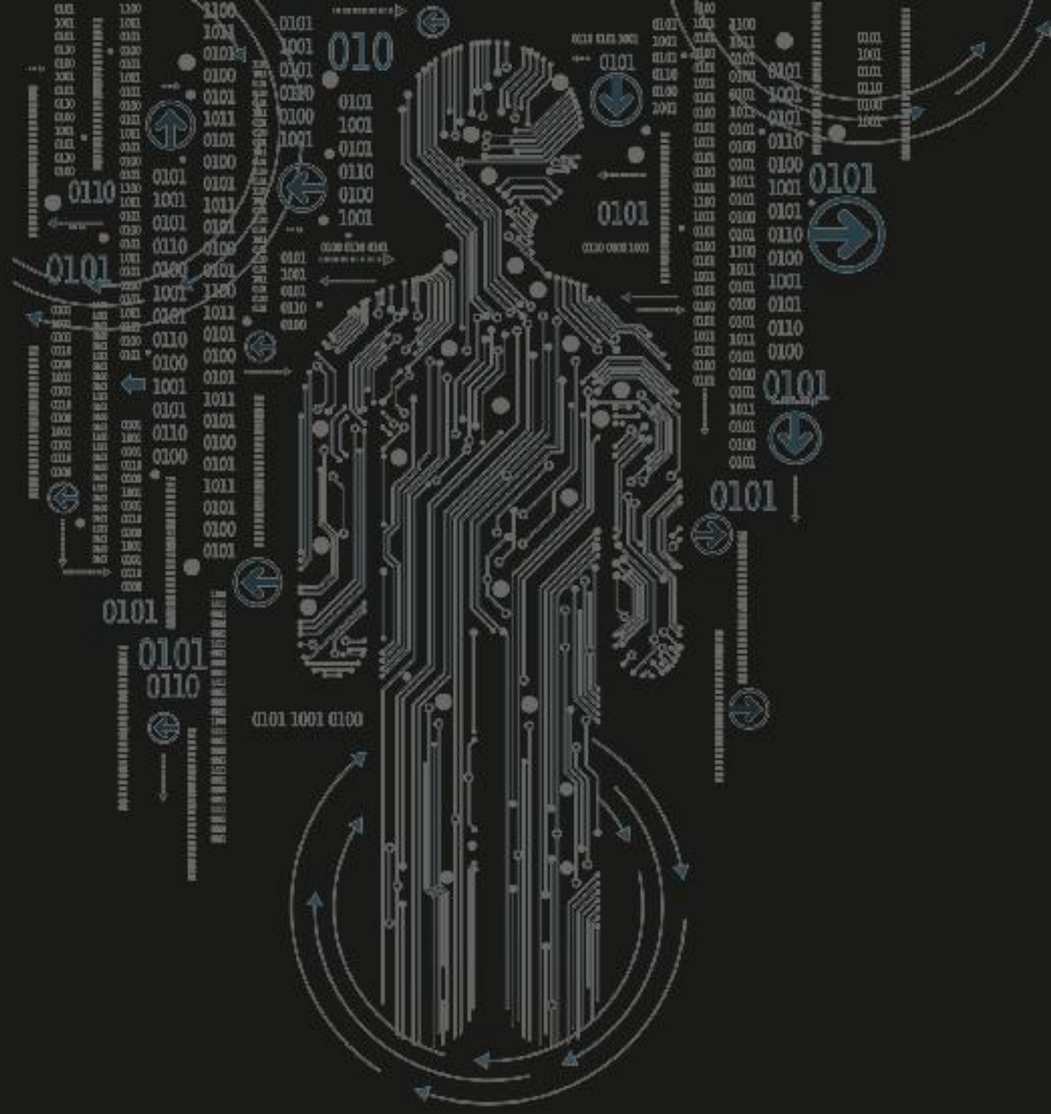
$i = i * j; \qquad i = i/j$

$$\Rightarrow \lceil \log_j n \rceil$$

BYJU'S EXAM PREP

ALGORITHMS

Asymptotic Notations



ABOUT ME

- Teaching Experience: 14+ Years
- Achievements: B.Tech, M.Tech, (Ph.D.)
- Achievements:
  - CS Senior Faculty
  - Mentored More Than 30,000+ Students
  - GATE Qualified
  - UGC-NET Qualified
  - SET Qualified
  - CISCO Certified

SATYA NARAYANA KAITEPALLI

Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

- Program Development Steps
- Algorithm : Step by step Procedure
- Efficiency of an algorithm — Time, Space } input size
- Analysis — Priori analysis*, Posterior analysis [After Execution]
- Time Calculation for different Statements in algorithms/Programs

Asymptotic Notations

a b c

Fibonacci Series : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34---

a b

678 — O(n) ⇒ O(1)

678 — O(1) ⇒ O(1)

Efficiency Of an Algorithm

- The efficiency of an algorithm depends on the amount of time, storage and other resources required to execute the algorithm. The efficiency is measured with the help of asymptotic notations.
- An algorithm "may not" have the "same performance" for different types of inputs. With the increase in the input size, the performance will change.
- The study of change in performance of the algorithm with the change in the order of the input size is defined as asymptotic analysis.

n iterations ↑

O(1) O(n) O(n)

$\boxed{-b \pm \sqrt{b^2-4ac}}$ ↑

O(1) O(1) O(1)

16-bit Processor

unsigned int n;

Range : 0 to 65,535

Asymptotic Notations

- Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.
- For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e. the best case. i/p: 5, 10, 15, 23, 35, 48
- But, when the input array is in reverse condition, the algorithm takes the maximum time (quadratic) to sort the elements i.e. the worst case. i/p: 48, 35, 23, 15, 10, 5
- When the input array is neither sorted nor in reverse order, then it takes average time. These durations are denoted using asymptotic notations.

i/p: 23, 15, 5, 48, 35, 10

n rounds * n times

$\Rightarrow n^2$

BYJU'S EXAM PREP

Asymptotic Notations

O( )

There are mainly three asymptotic notations:

- Big-O notation ⇒ Worst-case : Large input || worst-possible inputs
- Omega notation ⇒ Best-case : Small input
- Theta notation ⇒ Average-case : Medium input

A
AIR 55
- CPU
- Resources
- affordable

Best-Case

B
AIR 70
- College
- Sufficient
- small institute

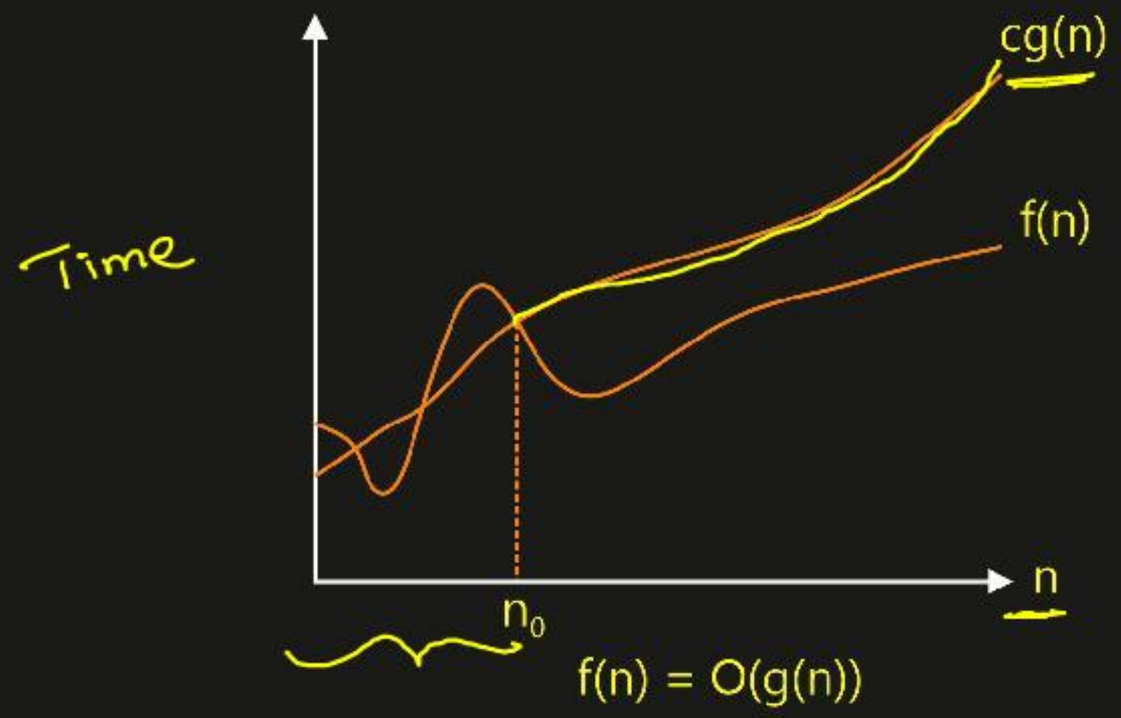
Average

C
AIR 90
- Distance
- No internet, only second hand books
- No coaching.

Worst-Case

Big – O Notation

- Big-O notation represents the upper bound of the running time of an algorithm. Thus, it gives the worst-case complexity of an algorithm.



Big – O Notation

if

$f(n) <= c * g(n)$

$\forall\ n \geq n_0$

$f(n) = O(g(n))$

BYJU'S EXAM PREP

Big – O Notation

- Big-O gives the upper bound of a function
- O(g(n)) = { f(n): there exist positive constants c and n0
- such that $0 \leq f(n) \leq c * g(n)$ for all $n \geq n0$ }
- The above expression can be described as a function f(n) belongs to the set O(g(n)) if there exists a positive constant c such that it lies between 0 and cg(n), for sufficiently large n.
- For any value of n, the running time of an algorithm does not cross the time provided by O(g(n)).
- Since it gives the worst-case running time of an algorithm, it is widely used to analyze an algorithm as we are always interested in the worst-case scenario.

$f(n) = 4n + 5$

Find c, n_0, $g(n)$

Such that, $f(n) <= c * g(n)$

$n \geq n_0$

$c = 5$

$g(n) = n$

$4n+5 <= 5 * n$

$n=1$ $4(1)+5 <= 5(1) \Rightarrow 9 <= 5$ False

$n=2$ $4(2)+5 <= 5(2) \Rightarrow 13 <= 10$ False

$n=3$ $4(3)+5 <= 5(3) \Rightarrow 17 <= 15$ False

$n=4$ $4(4)+5 <= 5(4) \Rightarrow 21 <= 20$ False

$n=5$ $4(5)+5 <= 5(5) \Rightarrow 25 <= 25$ True

$n=6$ $4(6)+5 <= 5(6)$

$29 <= 30$ True

$n=10$ $4(10)+5 <= 5(10)$

$45 <= 50$ True

$n=2000$ $4(2000)+5 <= 5(2000)$

$8005 <= 10000$ True

Algo

Step 1 $\rightarrow$ n times

Step 2 $\rightarrow$ 1

1 3 $\rightarrow$ 1

1 4 $\rightarrow$ n times

1 5 $\rightarrow$ 1

1 6 $\rightarrow$ n times

1 7 $\rightarrow$ 1

1 8 $\rightarrow$ 1

1 9 $\rightarrow$ n times

$4n + 5$

$$\boxed{\begin{array}{ll} f(n) <= c * g(n) & n \geq n_0 \\ 4n+5 <= 5 * & \end{array}} \Rightarrow f(n) == O(g(n))$$

$$== O(n)$$

$4 \cdot n^1 + 5 \cdot n^0$

$f(n) = 4n+5$

$c = 3 \quad g(n) = n^2$

$4n+5 <= 3 * n^2$

$n=1 \quad 9 <= 3$ False

$n=2 \quad 13 <= 12$ False

$n=3 \quad 17 <= 27$ True

$n=5 \quad 25 <= 75$ True

$n=400 \quad 1605 <= 480000$ True

$n_0 = 3. \quad 4n+5 <= 3 \cdot n^2 \ \forall \ n \geq 3 \Rightarrow O(n^2)$

$c = 1, \ g(n) = n^3$

$4n+5 <= 1 \cdot n^3$

$n=1 \quad 9 <= 1$ F

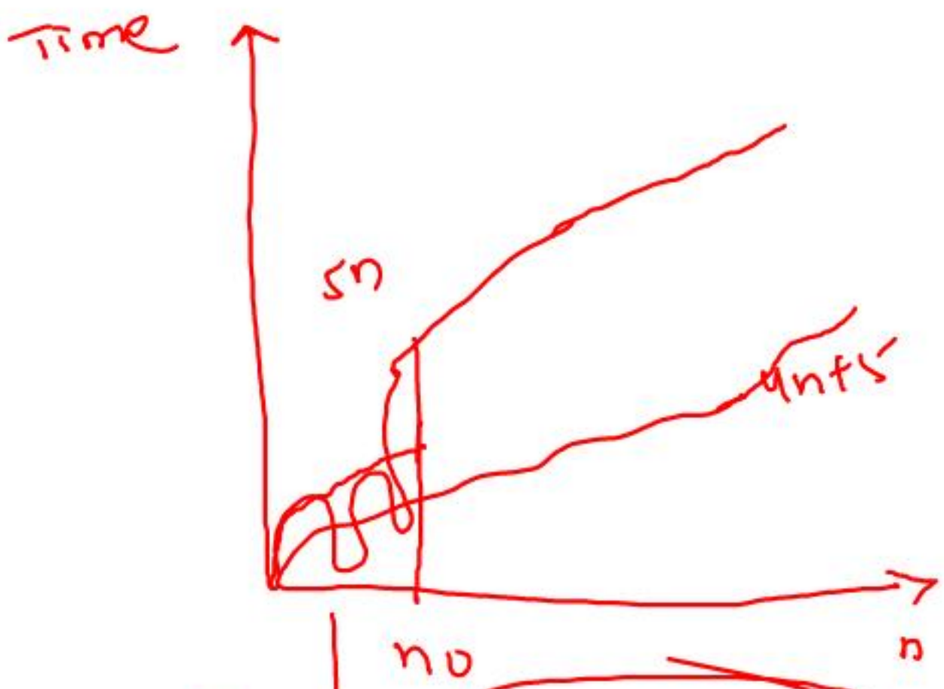
$n=2 \quad 13 <= 8$ F

$n=3 \quad 17 <= 27$ True

$n=5 \quad 25 <= 125$ True

$n_0 = 3$

$4n+5 <= 1 \cdot n^3 \ \forall \ n \geq 3 \Rightarrow O(n^3)$



$f(n) < n$

$< n^2$

$< n^3$

$< n^4$

$\Big\} > n$

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

$f(n) = n^2 + 5n^1$

$c = 100 \quad g(n) = n$

$n^2 + 5n <= 100 \cdot n$

$n=1 \quad 6 <= 100$ True

$n=2 \quad 14 <= 200$ True

$n=3 \quad 24 <= 300$ True

$n=10 \quad 150 <= 1000$ True

$n=50 \quad 2750 <= 5000$ True

$n=100 \quad 10500 <= 10000$ False

$n=300 \quad 91500 <= 30000$ False

$\vdots \quad n \geq 1 \quad n \geq 2 \quad n \geq 50 \Rightarrow n_0$ is not Possible.

$\Rightarrow f(n) \neq O(g(n))$

$c = 2 \quad g(n) = n^2$

$n^2 + 5n <= 2 \cdot n^2$

$n=1 \quad 6 <= 2$ F

$n=2 \quad 14 <= 8$ F

$\vdots$

$n=5 \quad 50 <= 50$ T

$n=6 \quad 66 <= 72$ T

$f(n) <= g(n) \quad \forall n \geq 5$

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

$c = 1 \quad g(n) = n^3$

$f(n) <= g(n) \quad \forall n \geq 3$

$O(n^3)$

$c = 5 \quad g(n) = n^2 \log_2 n$

$n=1 \quad 6 <= 0$ F

$n=2 \quad 14 <= 4$ F

$n=4 \quad 36 <= 32$ F

$n=8 \quad 104 <= 192$ True

$\vdots \quad O(n^2 \log n)$

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

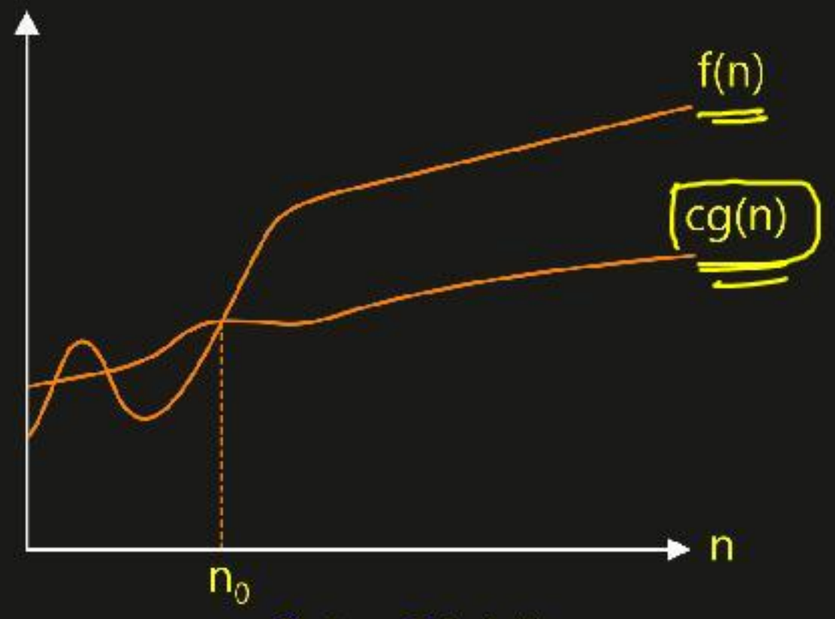
$= O(n^3)$

$= O(n^2 \log n)$

$\vdots$

Omega Notation (Ω-notation)

- Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best - case complexity of an algorithm.



$f(n) = \Omega(g(n))$

Omega Notation (Ω-notation)

BYJU'S EXAM PREP

Omega Notation (Ω-notation)

$f(n) >= c * g(n) \quad \forall \; n \geq n_0$

$\Rightarrow f(n) == \Omega(g(n))$

- Omega gives the lower bound of a function
- Ω(g(n)) = { f(n): there exist positive constants c and n0
- such that 0 ≤ c * g(n) ≤ f(n) for all n ≥ n0 }
- The above expression can be described as a function f(n) belongs to the set Ω(g(n)) if there exists a positive constant c such that it lies above cg(n), for sufficiently large n.
- For any value of n, the minimum time required by the algorithm is given by Omega Ω(g(n)).

$f(n) = 4n+5$

$c = 2 \qquad g(n) = n$

$4n+5 >= 2*n$

$n=1 \quad 9 >= 2$ True

$n=2 \quad 13 >= 4$ True

$n=20 \quad 85 >= 40$ True

$4n+5 >= 2*n \quad \forall n \geq 1$

$\Rightarrow \Omega(n)$

$f(n) = n^2+5n+1$

$c=1 \qquad g(n) = n^2$

$n^2+5n+1 >= n^2 \Rightarrow \Omega(n^2)$

$c=2 \qquad g(n) = n$

$n^2+5n+1 >= 2n \Rightarrow \boxed{\Omega(n)}$

$\underline{\Omega(1)}$

$c=1 \qquad g(n) = n^3$

$n^2+5n+1 >= 1 \cdot n^3$

$n_0 =$

$f(n) \cancel{=} \Omega(n^3)$

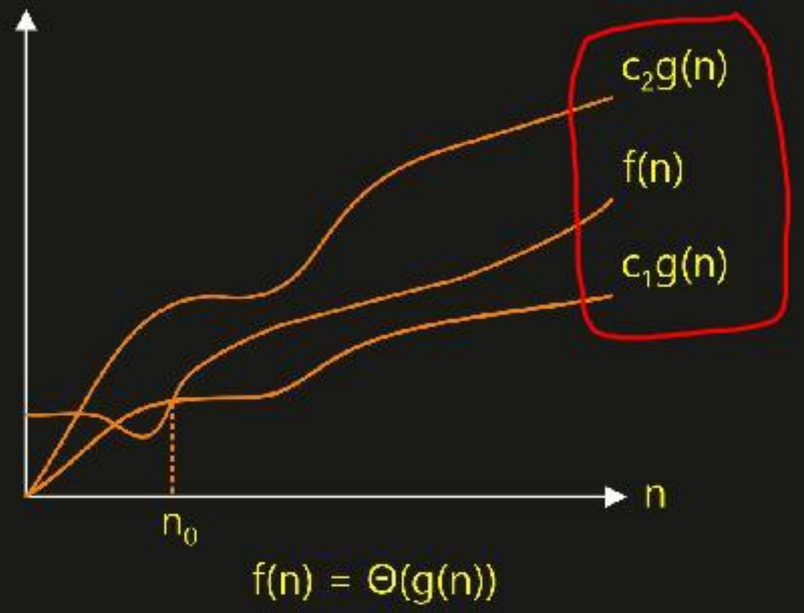
$c=1 \quad g(n) = n\log n$

$n^2+5n+1 >= 1 \cdot n\log n$

$\Omega(n\log n)$

Theta Notation (Θ-notation)

- Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analyzing the average-case complexity of an algorithm.



Theta Notation (Θ-notation)

BYJU'S EXAM PREP

Theta Notation (Θ-notation)

- Theta bounds the function within constants factors
- For a function g(n), Θ(g(n)) is given by the relation:
- Θ(g(n)) = { f(n): there exist positive constants c1, c2 and n0
- such that $0 \le c1g(n) \le f(n) \le c2g(n)$ for all $n \ge n0$ }
- The above expression can be described as a function f(n) belongs to the set Θ(g(n)) if there exist positive constants c1 and c2 such that it can be sandwiched between c1g(n) and c2g(n), for sufficiently large n.
- If a function f(n) lies anywhere in between c1g(n) and c2g(n) for all $n \ge n0$, then f(n) is said to be asymptotically tight bound.

$f(n) = n^3 + 4n^2 + 2$

$c_1 = 1 \qquad c_2 = 3 \qquad g(n) = n^3$

$n=1 \qquad 1 \cdot n^3 <= n^3 + 4n^2 + 2 <= 3 \cdot n^3$

$1 <= 7 <= 3$ False

$n=2 \qquad 8 <= 26 <= 24$ False

$n=3 \qquad 27 <= 65 <= 81$ True

$\vdots$

$1 \cdot n^3 \leq n^3 + 4n^2 + 2 <= 3 \cdot n^3 \quad \forall n \geq 3$

$\Rightarrow \Theta(n^3)$

Avg-Case

$\boxed{c_1 \cdot g(n) <= f(n)} <= c_2 \cdot g(n)$

Worst-Case

$f(n) <= c \cdot g(n)$

Asymptotic Notations

Time Complexities

Minimum

O(1)	Constant time
O(log n)	Logarithmic time
O(n)	Linear time
O(n log n)	Linearithmic time
$O(n^2)$	Quadratic time
$O(n^c)$	Polynomial time
$O(c^n)$	Exponential time
O(n!)	Factorial time
O(∞)	Infinite time

Maximum

Image Subtitle Goes Here

BYJU'S EXAM PREP

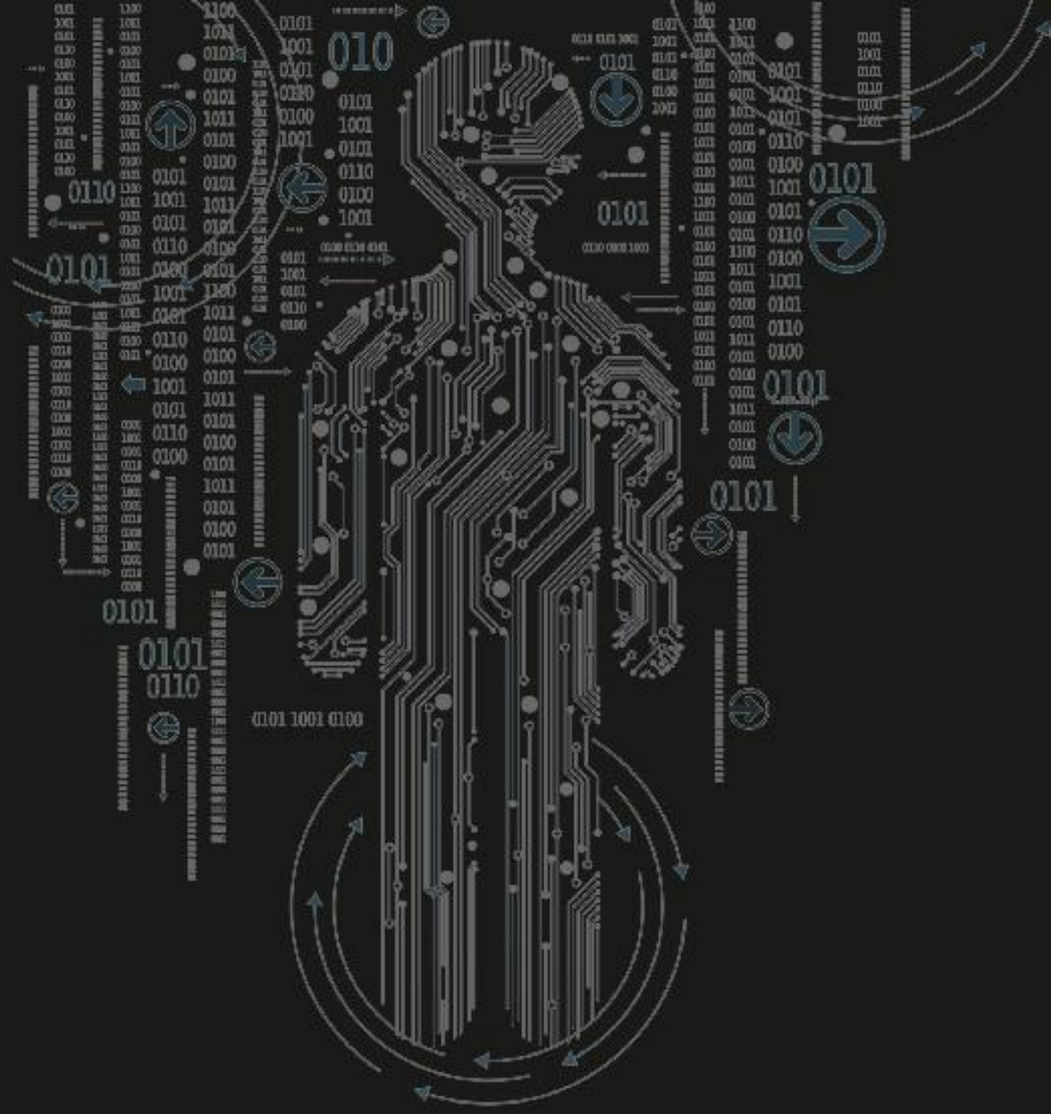
Examples

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Recurrence Relations



ABOUT ME

- Teaching Experience: 14+ Years
- Achievements: B.Tech, M.Tech, (Ph.D.)
- Achievements:
  - CS Senior Faculty
  - Mentored More Than 30,000+ Students
  - GATE Qualified
  - UGC-NET Qualified
  - SET Qualified
  - CISCO Certified

SATYA NARAYANA KAITEPALLI

Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

	$f(n) == \Omega(g(n))$	$f(n) >= c \cdot g(n)$
Best Case - Omega Notation :		
Avg Case - Theta Notation :	$\Theta(g(n))$	$c_1 \cdot g(n) <= f(n) <= c_2 \cdot g(n)$
Worst Case - Big-oh notation :	$O(g(n))$	$f(n) <= c \cdot g(n)$

$O(1) < \log n < n < n\log n < n^2 < n^3 < n^4 < \ldots < 2^n < n!$

$f_1 = 2^n \qquad f_2 = n\log n \qquad f_3 = n^3$

$n\log n < n^3 < 2^n$

$f_2 < f_3 < f_1$

$x \quad y \quad x \quad y$

$f_1 == O(f_2)$ ✗

$f_2 = O(f_3)$ ✓

$f_3 = \Omega(f_1)$ ✗

$f_2 = \Omega(f_3)$ ✗

$f_1 = \Omega(f_3)$ ✓

$\vdots$

$x < y$
$x = O(y)$

$x > y$
$x = \Omega(y)$

$f_2 == O((f_3))$

$f_2 == O((f_1))$

$f_3 = O((f_1))$

$f_1 == \Omega(f_3)$

$f_1 == \Omega(f_2)$

$f_3 = \Omega(f_2)$

$f_3 > f_2$

$f(n) > g(n)$

$f(n) <= c\,g(n) \Rightarrow f(n) == O(g(n))$

$f(n) >= c \cdot g(n) \Rightarrow f(n) == \Omega(g(n))$

$f(n) = \Omega(n) \Rightarrow n$

$g(n) = O(n) \Rightarrow n$

$h(n) = \Theta(n) \Rightarrow n$

$x = \Theta(y)$

$c_1 \cdot y <= x <= c_2 \cdot y$

$f(n) \cdot g(n) + h(n)$

(a)

$n \cdot n + n \Rightarrow \underbrace{n^2+n}_{x} \Rightarrow \underset{y}{\Theta(n)}$ ✗

$\boxed{c_1 \cdot n <= n^2+n <= c_2 \cdot n}$ $\forall n \geq n_0$

$n^2+n >= c \cdot n$

$n^2+n <= c \cdot n$

$\underset{x}{n^2+n} \Rightarrow \underset{y}{\Omega(n)}$ ✓

$\underset{x}{n^2+n} \Rightarrow \underset{y}{O(n)}$ ✗

$n^2+n = \Omega(n)$

BYJU'S EXAM PREP

Types of Algorithms

- Brute Force algorithm
- Greedy algorithm
- Recursive algorithm
- Divide & Conquer algorithm
- Dynamic programming algorithm
- Randomized algorithm
- Backtracking algorithm

BYJU'S EXAM PREP

f(5)
↳ f(4)
↳ f(3) ↳ f(2) ↳ f(1)

Recursive Algorithms

- This is one of the simplest to devise algorithms as it does not require specifically think about every subproblem. This means we just need to think about the existing cases and the solution of the simplest subproblem, all other complexity will be handled by it automatically.
- Recursion is a very powerful tool, but we should always take care of memory management here as recursion works using recursive stack which is called every time recursion function is invoked.

BYJU'S EXAM PREP

Recurrence Relation

recursive

- The model that uses mathematical concepts to calculate the time complexity of an algorithm is known as the recurrence relation.
- A recursive relation, T(n), is a recursive function of integer n. Every recursive function consists of both recursive and base cases.
- The section of the definition that does not contain T is called a base case of the recurrence relation, and the portion that contains T is called recursive or recursive relation.
- Example: A Fibonacci sequence is defined by the recurrence relation – f(n) = f(n-1) + f(n-2) with initial conditions f1=1 and f2 = 1.

Recurrence Relations

Let $T(n)$ be time taken by function $f(n)$ on input 'n'.

Example - 1

```
void f(int n)
{
if(n>0)
 {
   printf("%d",n);
   f(n-1);
 }
}
```

$T(n)$

$\rightarrow$ f $\rightarrow$ False

$\downarrow$ True

$\rightarrow 1$

$\rightarrow T(n-1)$

Recurrence Relation

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

$$T(n) = 1+1+T(n-1) \qquad n>0$$

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

$$= 1 \qquad \text{Otherwise } (n<=0)$$

BYJU'S EXAM PREP

Example - 2

```
void f(int n)
{
if(n>0)
 {
 for(i=0;i<n;i++)
 printf("%d",n);
 f(n-1);
 }
}
```

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

$i < n$

$i=0 \quad 0<n$

$i=1 \quad 1<n$

$i=n-1 \quad (n-1)<n$

$i=n \quad n<n$ — False

(n times) True

+1 time

$\longrightarrow T(n)$

$\longrightarrow 1 \longrightarrow$ FALSE End

TRUE

$(n+1)$ times

n times

$T(n-1)$

n times

0 to (n-1) True

n False

last case

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

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

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

BYJU'S EXAM PREP

Example - 3

```
void f(int n)
{
if(n>0)
 {
 for(i=1;i<n;i=i*2)
 printf("%d",n);
 f(n-1);
 }
}
```

$\longrightarrow T(n)$

$\longrightarrow 1$

$\longrightarrow (\log n + 1)$ times $\begin{cases} \log n \text{ times TRUE} \\ 1 \text{ time False} \end{cases}$

$\longrightarrow \log n$

$\longrightarrow T(n-1)$

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

$i = i*2$ (or) $i/2$ $\quad \log_2 n$

$i = i*3$ (or) $i/3$ $\quad \log_3 n$

$\vdots$

$i = i*m$ (or) i/m $\quad \log_m n$

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

$$T(n) = T(n-1) + \underbrace{2\log n + 2}$$

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

BYJU'S EXAM PREP

Example - 4

```
void f(int n)
{
if(n>0)
 {
  for(i=n; i>1; i=i/2)
  printf("%d",n);
  f(n-1);
 }
}
```

$T(n)$

1

$\log n + 1$

$\log n$

$T(n-1)$

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

$$\boxed{\begin{aligned} T(n) &= T(n-1) + \log n \quad && n > 0 \\ &= 1 && \text{otherwise} \end{aligned}}$$

BYJU'S EXAM PREP

Example - 5

```
void f(int n)
{
if(n>0)
 {
 printf("%d",n);
 f(n-1);
  f(n-1);
 }
}
```

$$T(n) = \begin{cases} 2T(n-1)+1 & n>0 \\ 1 & \text{otherwise} \end{cases}$$

$T(n)$

1

1

$T(n-1)$

$T(n-1)$

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

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

$2 \cdot n^2 \Rightarrow n^2$

$4 \cdot \log n \rightarrow \log n$

$2 \Rightarrow 2 \cdot n^0 \Rightarrow n^0$

BYJU'S EXAM PREP

Example - 6

```
void f(int n)
{
if(n>0)
{
for(i=1;i<n;i=i*2)
 {
   printf("%d",n);
   f(n-1);
 }
}
}
```

$\longrightarrow T(n)$

$\longrightarrow 1$

$\longrightarrow \log n + 1$

$\longrightarrow \log n$ times

$\longrightarrow \log n$ times $* T(n-1)$

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

$$T(n) = 1 + \log n + 1 + \log n + \log n * T(n-1)$$
$$= \log n \cdot T(n-1) + 2\log n + 2$$
$$= \log n \cdot T(n-1) + \log n$$

Recurrence Relations

BYJU'S EXAM PREP

Examples

Derive Recurrence Relations of :

1. Linear Search
2. Binary Search $T(n) = T(\frac{n}{2}) + 1$
3. Merge Sort $T(n) = 2 \cdot T(\frac{n}{2}) + n$

False

63 == 20 → 1

True

Print Success → 1

End

Linear Search: Given array of elements and Search for some element, key.

Compare with every element one by one.

20 | 40 | 15, 5, 17, 19, 21, 63

Worst-Case: getting at last or none

Best-Case: First element is key

Average-Case: Middle element is key.

Let Key = 63

Let $T(n)$ be time taken for performing linear search with 'n' elements ⇒

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

Recurrence Relations

BYJU'S EXAM PREP

How to Solve RR ? ⇒ Derive Time complexity from given Recurrence Relation

1. Recursion Tree method
2. Substitution method (Induction method)
3. Master method

BYJU'S EXAM PREP

Sum of 'n' natural numbers
Sum of squares
Sum of cubes
G.P

Recursive Tree Method

In the recurrence tree method, a recurrence tree is drawn and calculates the time taken by every level of a tree. After analyzing each level, we sum up the work done at all levels. In order to draw a recurrence tree, start from the given recurrences and keep drawing till we find a pattern among all the levels. The pattern follows arithmetic or geometric series.

BYJU'S EXAM PREP

Recursive Tree Method

Consider the recurrence relation:

T(n) = 2 T(n/2) + 1

$\boxed{T(1) = 1}$ $\boxed{2 \cdot T\left(\frac{n/2}{2}\right) + 1}$

$T(n)$ 개수

$2 \cdot T\left(\frac{n}{2}\right) + \boxed{1}$ $= 2^1$

$2 \cdot T\left(\frac{n}{4}\right) + \boxed{1}$ $= 2^2$

$2 \cdot T\left(\frac{n}{8}\right) + \boxed{1}$ $= 2^3$

$2 \cdot T\left(\frac{n}{16}\right) + \boxed{1}$ $= 2^4$

$T\left(\frac{n}{2^k}\right) + \boxed{1}$

$2\left[T\left(\frac{n}{2}\right) + T\left(\frac{n}{2^2}\right) + T\left(\frac{n}{2^3}\right) + \cdots T\left(\frac{n}{2^k}\right)\right] + k$

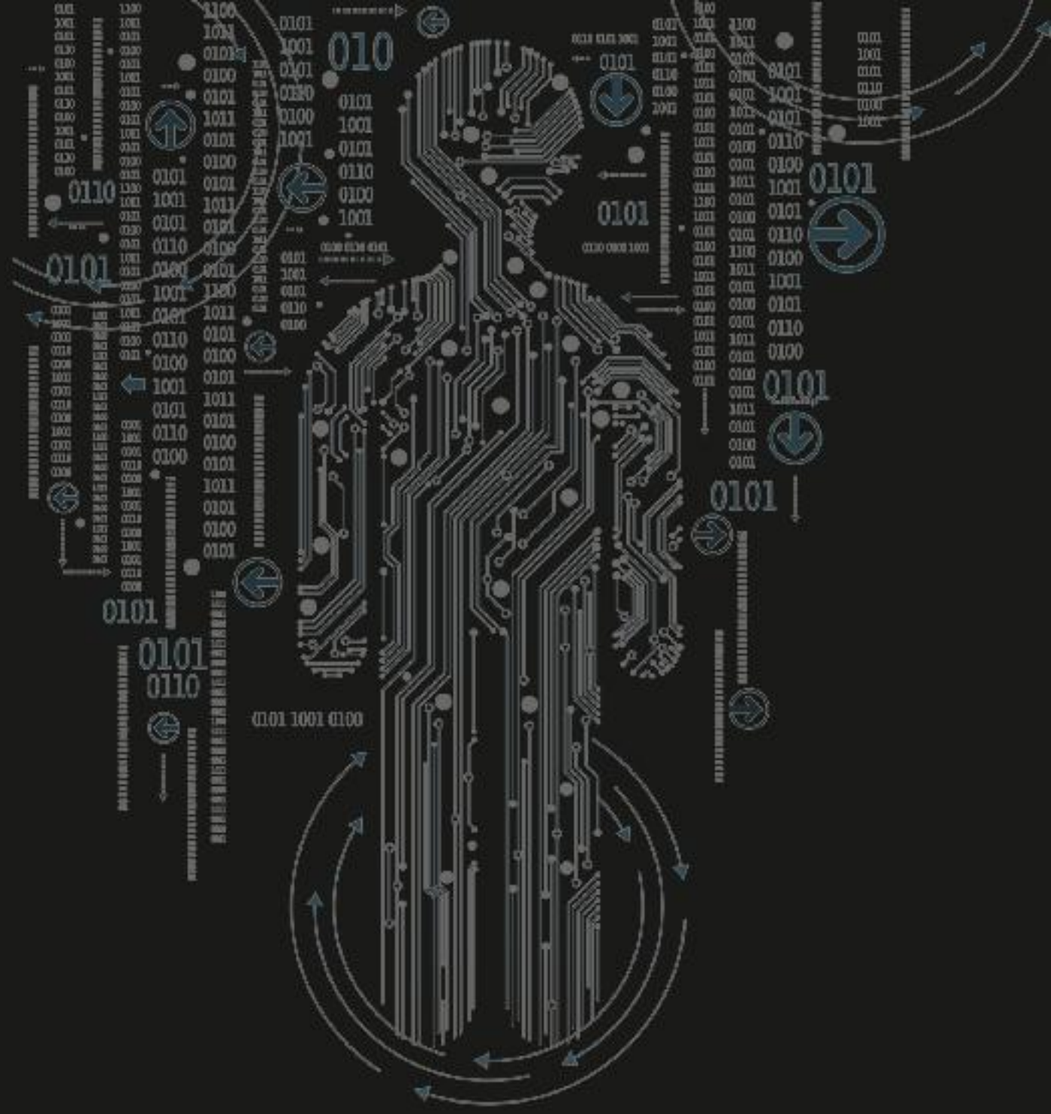
$\frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \cdots \frac{1}{2^k}$

$1 - 2^{-k} + k$

BYJU'S EXAM PREP

ALGORITHMS

Master Theorem



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

Master Theorem

Last Class – Quick Revision

```
for (i=1; i<n; i=i*i)
  i=1   i=1   i=1
  i<n   i<n   i<n  - - - - - ∞
```

How to form Recurrence Relation?

↳ Recursive algorithms
- Base case → C
- Recursive calls → T(- - -)

$$T(n) = \begin{cases} T(\text{- - - -}) + C & \text{Non Base Case} \\ 1 & \text{Base Case} \end{cases}$$

How to find time from RR ⇒ Solving RR
- Recursive Tree
- Substitution
- Master Theorem

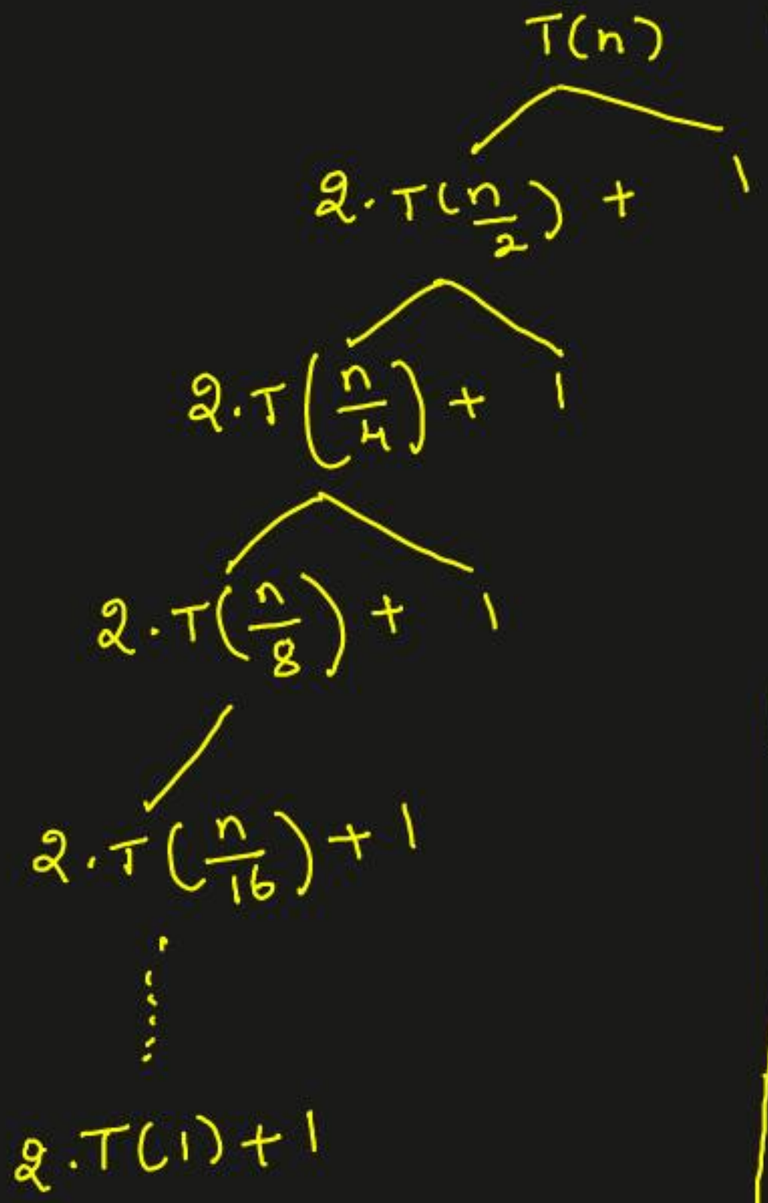
Recursive Tree Method

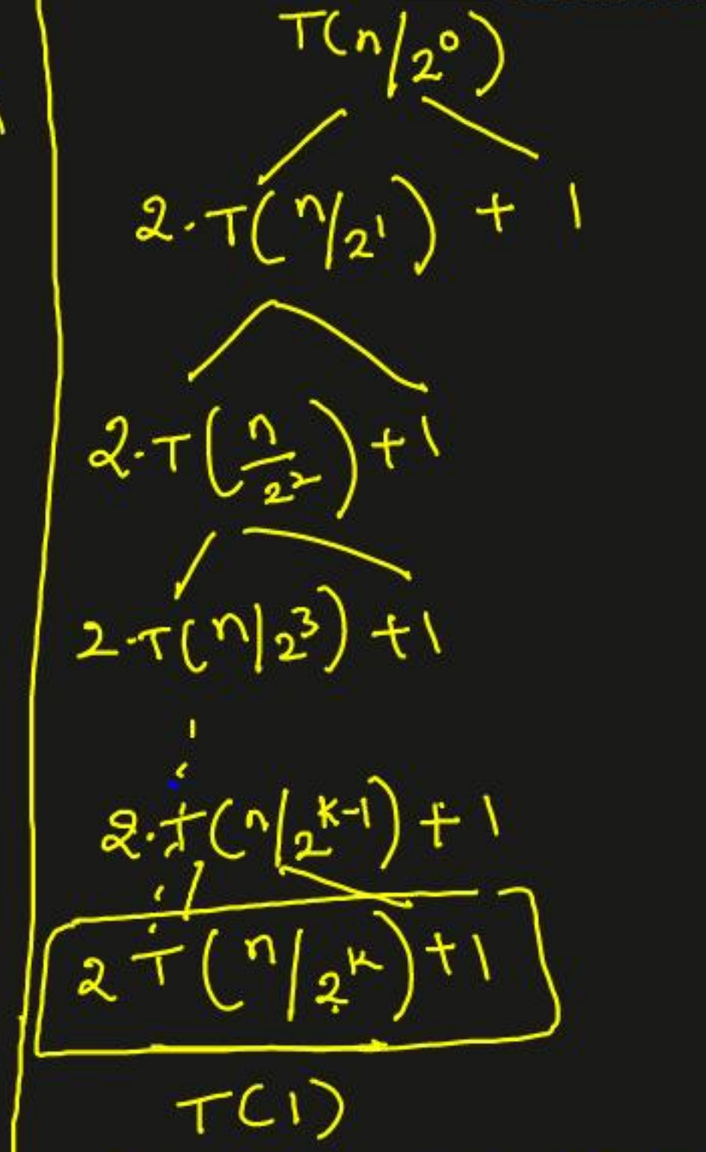
Consider the recurrence relation:

$T(n) = 2\,T(n/2) + 1$

$T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{\left(\frac{n}{2}\right)}{2}\right) + 1$

$$\boxed{T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{2^2}\right) + 1}$$





$2 \cdot T(1) + 1 \Rightarrow 2 \cdot (1) + 1$

$$T\left(\frac{n}{2^{k-1}}\right) = \underline{2(1)+1} = 3$$

$$T\left(\frac{n}{2^{k-2}}\right) = 2 \cdot T\left(\frac{n}{2^{k-1}}\right) + \underline{1}$$

$$= 2\left[2(1)+1\right] + \underline{1}$$

$$\left[2(1)+1\right] + \left[2\left[2(1)+1\right]+1\right] + 2\left[2\left[2(1)+1\right]+1\right]+1 + \cdots$$

$$(2+1) + 2(2+1)+1 + 2(2(2+1)+1) + \cdots$$

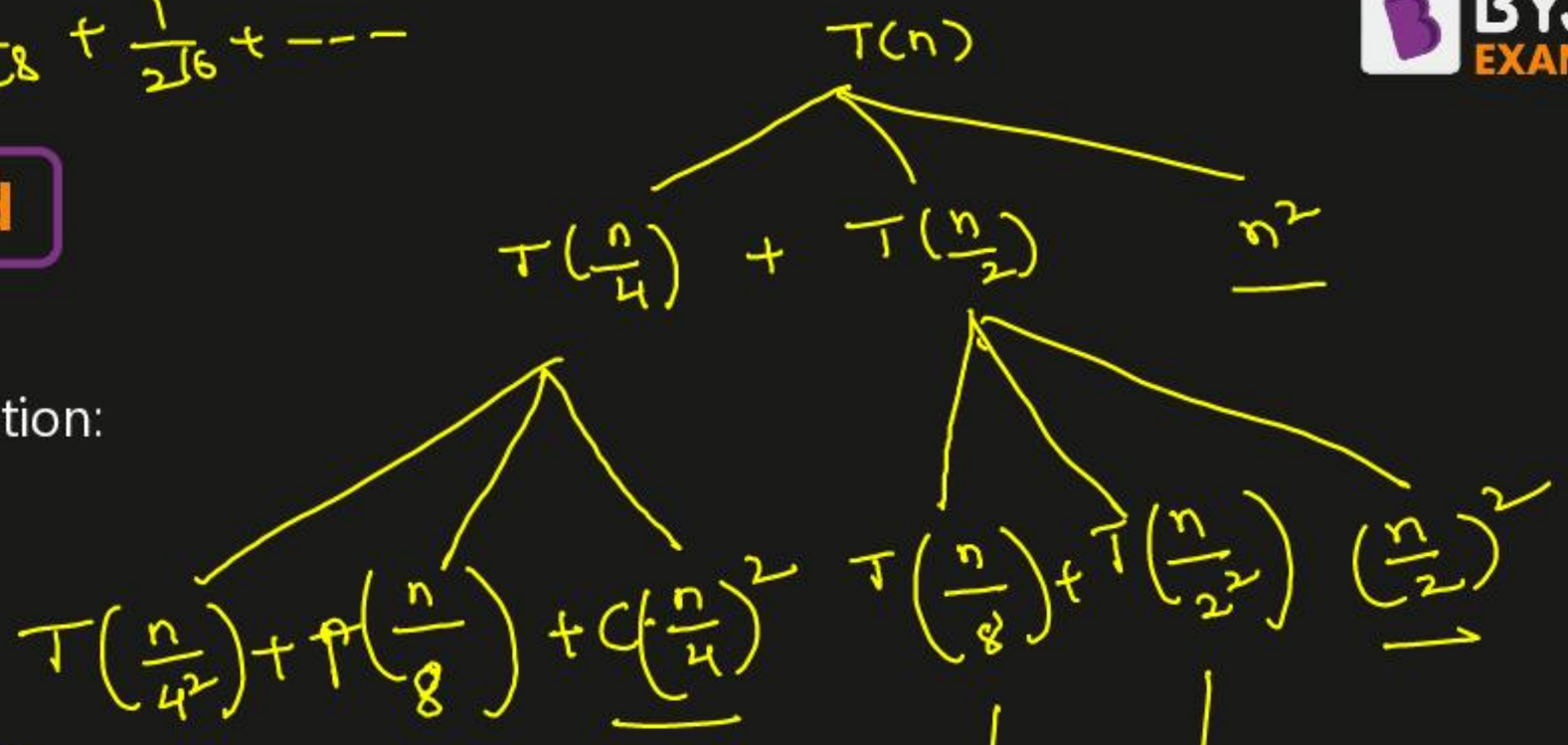
$$\underline{2^n - 2 - n} \Rightarrow \underline{O(2^n)}$$

$$1 + \frac{1}{2^2} + \frac{1}{2^4} + \frac{1}{2^8} + \frac{1}{2^{16}} + \cdots$$

Recursive Tree Method

Consider the recurrence relation:

$T(n) = T(n/4) + T(n/2) + cn^2$



$$1 + n^2 + \frac{n^2}{2^2} + \frac{n^2}{4^2} + \frac{n^2}{16^2} + \frac{n^2}{32^2}$$

$$n + n^2 \left[1 + \frac{1}{2^2} + \frac{1}{4^2} + \frac{1}{16^2} + \frac{1}{256^2} + \cdots\right] \Rightarrow O(\quad)$$

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) + c \cdot n^2$$

$$T\left(\frac{n}{2}\right) > T\left(\frac{n}{4}\right) \Rightarrow T\left(\frac{n}{4}\right) + T\left(\frac{n}{2}\right) \Rightarrow T\left(\frac{n}{2}\right)$$

$$T(n) = T\left(\frac{3n}{4}\right) + T\left(\frac{n}{3}\right) + c$$

$$T(0.75n) + T(0.33n)$$

$$T(n) = T\left(\frac{3n}{4}\right) + c$$

BYJU'S EXAM PREP

Substitution Method

- In this method, mathematical induction is used to solve the recurrence relation.
- The substitution method is a powerful approach that can prove upper bounds for almost all recurrences.

BYJU'S EXAM PREP

Substitution Method

- Consider the recurrence relation T(n) = T(n-1) + 1

$T(0) = 1$,

$T(n-k) = T(0)$

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

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

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

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

$= T(n-4)+4$

k times $\left\{ \begin{array}{l} \vdots \\ T(n-k)+k \end{array} \right.$

Substitute 'n' for k

$T(n-n)+n$

$= T(0)+n$

$= 1+n \Rightarrow O(n)$

Master Theorem

Induction Method.

Substitution Method

$2^{x+y} = \frac{2^x}{2^y} = \frac{2^{\log_2 n}}{2}$

- Consider the recurrence relation T(n) = 2T(n/2) + 1

$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + 1$

$= 2\left[2 \cdot T\left(\frac{n}{2^2}\right) + 1\right] + 1$

$= 2^2 \cdot \left[2 \cdot T\left(\frac{n}{2^3}\right) + 1\right] + 2 + 1$

$= 2^3 \cdot T\left(\frac{n}{2^3}\right) + 2^2 + 2 + 1$

$\vdots$

k times $\left\{ 2^k \cdot T\left(\frac{n}{2^k}\right) + \right.$

$2^{k-1} + 2^{k-2} + \cdots + 2^2 + 2^1 + 2^0$

$T(1) = 1$

Let, $T\left(\frac{n}{2^k}\right) = T(1)$

$\Rightarrow \frac{n}{2^k} = 1$

$\Rightarrow 2^k = n \Rightarrow k = \log_2 n$

Substitute $\log_2 n$ for k

$\Rightarrow 2^{\log_2 n} \cdot 1 + 2^{\log_2 n - 1} + 2^{\log_2 n - 2} + \cdots 2^2 + 2^1 + 2^0$

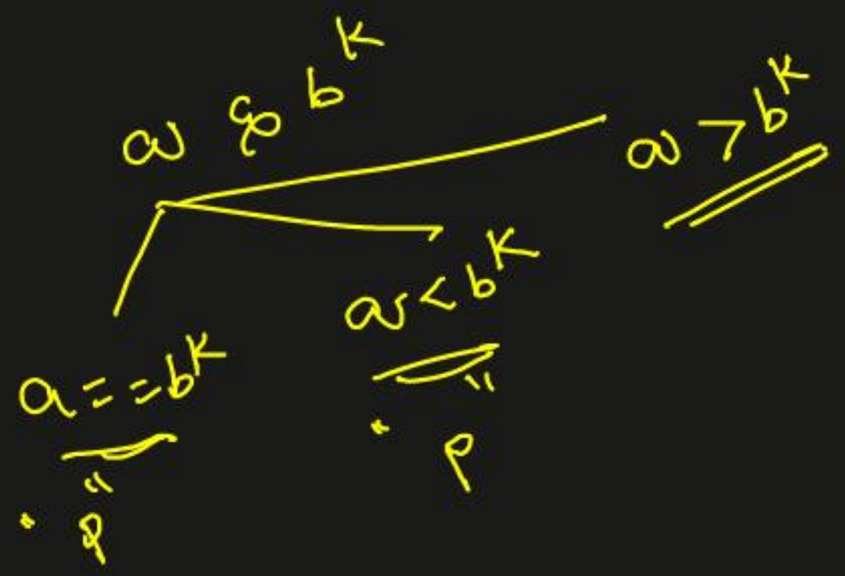
$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots + 1$

$\Rightarrow 2^n + \cdots \Rightarrow O(2^n)$

BYJU'S EXAM PREP

Master Theorem

Jon Bentley 1980



log Power p

$$T(n) = a\,T(n/b) + \Theta(n^k \log^p n)$$

a>=1, b>1, k>=0 and p is a real number

Case 1: $a > b^k$ then , $T(n) = \Theta(n^{\log_b a})$

Case 2: $a = b^k$ and
- a) if $p > -1$, $T(n) = \Theta(n^{\log_b a} \log^{(p+1)} n)$
- b) if $p = -1$, $T(n) = \Theta(n^{\log_b a} \log \log n)$
- c) if $p < -1$, $T(n) = \Theta(n^{\log_b a})$

Case 3: $a < b^k$ and
- a) if $p >= 0$, $T(n) = \Theta(n^k \log^p n)$
- b) if $p < 0$, $T(n) = \Theta(n^k)$

Master Theorem

$\because n$ is decreasing Exponentially. $\frac{n}{2}$ $\frac{n}{4}$ $\frac{n}{8}$ ---

BYJU'S EXAM PREP

Example - 1

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

$T(n) = a \cdot T(\frac{n}{b}) + n^k \log^p n$

$T(n) = 1 \cdot T(\frac{n}{2}) + n^2 \cdot \log^0 n$

$a = 1 \quad b = 2 \quad k = 2 \quad p = 0$

$a = 1, \quad b^k = 2^2 = 4$

$\Rightarrow a < b^k \Rightarrow$ Case 3

$p >= 0 \Rightarrow$ Case 3 a

$O(n^2 \log^0 n) = O(n^2)$

$$T(n) = a\,T(n/b) + \theta(n^k \log^p n)$$

$a >= 1$, $b > 1$, $k >= 0$ and p is a real number

Case 1: $a > b^k$ then, $T(n) = \theta(n^{\log_b a})$

Case 2: $a = b^k$ and
- a) if $p > -1$, $T(n) = \theta(n^{\log_b a} \log^{(p+1)} n)$
- b) if $p = -1$, $T(n) = \theta(n^{\log_b a} \log \log n)$
- c) if $p < -1$, $T(n) = \theta(n^{\log_b a})$

Case 3: $a < b^k$ and
- a) if $p >= 0$, $T(n) = \theta(n^k \log^p n)$
- b) if $p < 0$, $T(n) = \theta(n^k)$

Master Theorem

Example - 2

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

$a=2 \quad b=4 \quad k=3 \quad p=0$

$a < b^k, \quad p=0 \qquad$ Case 3(a)

$O(n^k \log^p n)$

$O(n^3)$

$$T(n) = a\,T(n/b) + \theta(n^k \log^p n)$$

$a>=1$, $b>1$, $k>=0$ and p is a real number

Case 1: $a>b^k$ then , $T(n)= \theta(n^{\log_b a})$

Case 2: $a=b^k$ and
- a) if $p>-1$, $T(n)= \theta(n^{\log_b a} \log^{(p+1)} n)$
- b) if $p=-1$, $T(n)= \theta(n^{\log_b a} \log \log n)$
- c) if $p<-1$, $T(n)= \theta(n^{\log_b a})$

Case 3: $a<b^k$ and
- a) if $p>=0$, $T(n)= \theta(n^k \log^p n)$
- b) if $p<0$, $T(n)= \theta(n^k)$

Master Theorem

Example - 3

$T(n) = 16T(n/2) + n^2\log^2 n$

$a=16 \quad b=2 \quad k=2 \quad p=2$

$a > b^k$ Case 1

$O(n^{\log_b a})$

$= O(n^{\log_2 16})$

$= O(n^4)$

$$T(n) = a\,T(n/b) + \theta(n^k \log^p n)$$

$a>=1$, $b>1$, $k>=0$ and p is a real number

Case 1: $a>b^k$ then , $T(n)= \theta(n^{\log_b a})$

Case 2: $a=b^k$ and
- a) if $p>-1$, $T(n)= \theta(n^{\log_b a} \log^{(p+1)} n)$
- b) if $p=-1$, $T(n)= \theta(n^{\log_b a} \log \log n)$
- c) if $p<-1$, $T(n)= \theta(n^{\log_b a})$

Case 3: $a<b^k$ and
- a) if $p>=0$, $T(n)= \theta(n^k \log^p n)$
- b) if $p<0$, $T(n)= \theta(n^k)$

$$T(n) = 2 \cdot T\left(\frac{3n}{4}\right) + n^2 \log^{-1} n$$

$$= 2 \cdot T\left(\frac{n}{(4/3)}\right) + n^2 \log^{-1} n$$

$a = 2$ $\quad b = \frac{4}{3}$ $\quad k = 2$ $\quad p = -1$

$a >= 1$ $\quad b > 1$ $\quad k >= 0$ $\quad p$ is real number

$b^k = \left(\frac{4}{3}\right)^2 = \frac{16}{9}$ $\quad a = 2$ $\quad \Rightarrow a > b^k \Rightarrow$ Case 1

$$O\left(n^{\log_b a}\right)$$

$$= O\left(n^{\log_{(4/3)} 2}\right)$$

BYJU'S EXAM PREP

"Linear decrease of 'n'"

Master Theorem

$b=1$ $n, n-1, n-2, n-3, n-4, ----$

$f(n) = n^k \cdot g(n)$

$$T(n) = a\,T(n-b) + f(n)$$

$a>0$, $b>0$, $f(n)=n^k$, $k>=0$

Case 1: $a<1$ $O(f(n))$

Case 2: $a=1$ $O(n*f(n))$

Case 3: $a>1$ $O(f(n)\ a^{n/b})$

Master Theorem

BYJU'S EXAM PREP

Example - 1

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

n^k

$T(n) = a \cdot T(n-b) + \boxed{f(n)}$

$T(n) = 1 \cdot T(n-1) + 1 \cdot n^0$

$a = 1 \quad b = 1 \quad k = 0$

Case 2

$O(n * f(n))$

$= O(n \cdot n^0)$

$= O(n)$

$$T(n) = a\,T(n-b) + f(n)$$

$a>0,\ b>0,\ f(n)=n^k,\ k>=0$

Case 1: $a<1$ $O(f(n))$

Case 2: $a=1$ $O(n*f(n))$

Case 3: $a>1$ $O(f(n)\ a^{n/b})$

Master Theorem

Example - 2

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

$a=2 \quad b=2 \quad k=1 \quad f(n)=n^1$

$a>1$ Case 3

$O\left(f(n) \cdot a^{\frac{n}{b}}\right)$

$= O\left(n \cdot 2^{\frac{n}{2}}\right) = O\left(n \cdot \sqrt{2^n}\right)$

$T(n) = a\,T(n-b) + f(n)$

$a>0,\ b>0,\ f(n)=n^k,\ k>=0$

Case 1: $a<1$ $O(f(n))$

Case 2: $a=1$ $O(n*f(n))$

Case 3: $a>1$ $O(f(n)\ a^{n/b})$

Master Theorem

Example - 3

$T(n) = T(n-2) + n / \log n$

$a=1 \quad b=2 \quad f(n) = n^k g(n)$

$k=1 \quad g(n) = \log^{-1} n$

$a==1$ Case 2

$O(n * f(n)) == O\left(n * \frac{n}{\log n}\right)$

$= O(n^2/\log n)$

$T(n) = a\,T(n-b) + f(n)$

$a>0,\ b>0,\ f(n)=n^k,\ k>=0$

Case 1: $a<1$ $O(f(n))$

Case 2: $a=1$ $O(n*f(n))$

Case 3: $a>1$ $O(f(n)\ a^{n/b})$

Practice Problems

1. $T(n) = 8\,T\left(\frac{n}{3}\right) + n^2 \log^2 n$

2. $T(n) = 7\,T(n-5) + \frac{n}{\log^{+2} n}$

3. $T(n) = 4 \cdot T\left(\frac{5n}{7}\right) + n^3 \log^{-2} n$

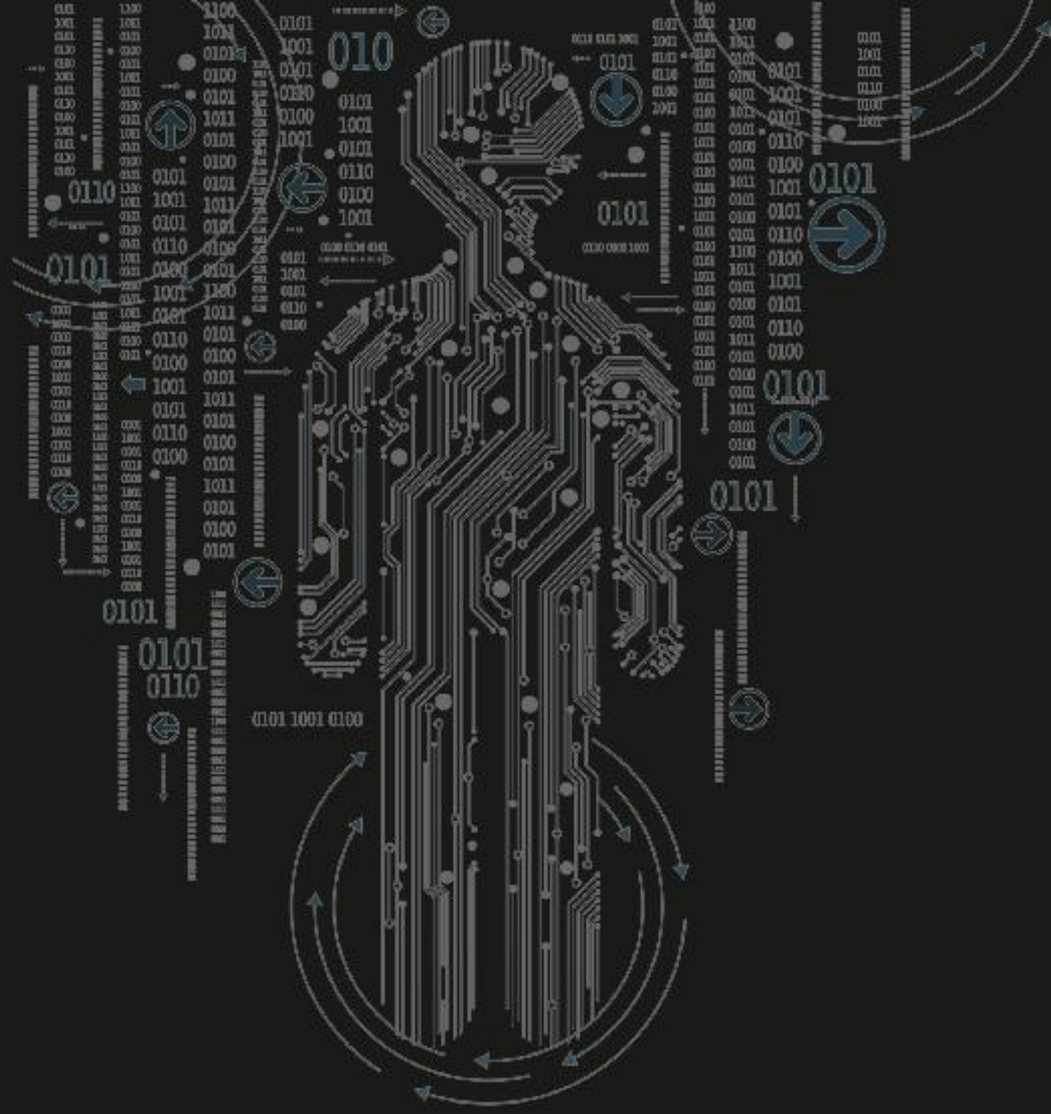
4. $T(n) = 2 \cdot T\left(\frac{3n}{2}\right) + 3 \cdot T\left(\frac{2n}{3}\right) + n^2 \log n$

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Collision Resolution Techniques



ABOUT ME

- Teaching Experience: 14+ Years
- Achievements: B.Tech, M.Tech, (Ph.D.)
- Achievements:
  - CS Senior Faculty
  - Mentored More Than 30,000+ Students
  - GATE Qualified
  - UGC-NET Qualified
  - SET Qualified
  - CISCO Certified

SATYA NARAYANA KAITEPALLI

Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Solving Recurrence Relations
- Substitution (Induction) Method
- Master's Theorem

input size decreases Exponentially

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + n^k \log^p n$$

a, b^k
- $a > b^k$
- $a == b^k$
- $a < b^k$

} p

input size decreases Linearly

$$T(n) = a \cdot T(n-b) + f(n)$$

$a > 1$

$a = 1$

$a < 1$

$\boxed{i=1 \quad i<n \quad i=i*i}$

infinite

int a[8] = {000, 001, 010, 011, 100, 101, 110, 111};

Octal values — Decimal values

'0' Prefix ⇒ octal number

Decimal: {0, 1, 8, 9, 100, 101, 110, 111}

option: C

1000 + 2*1 − 4*1

char s[] = "GATE2022";

0 1 2 3 4 5 6 7

s + (int)(s[4] − '0') − 2 * (int)(s[6] − '0')

1000

('2' − '0') → 2

'2' − '0' → 2

1000 + 2(1)

s + 2 − 4(1)

s + 2 − 2 * 2 ⇒ 1002 − 4 ⇒ 998

Garbage value.

String at address 998

Collision Resolution Techniques

Hashing == Mapping Technique

- Hashing is a technique in which a given key field value is converted into the address of the storage location of the record by applying the same operation to it.

Hash Table

- The advantage of hashing is that it allows the execution time of basic operation to remain constant even for the larger side.

Collision Resolution Techniques

allocated : 4000 characters space

Used : 90 characters space

Hashing

- Suppose we have 50 employees, and we have to give 4 digit key to each employee (as for security), and we want after entering a key, direct user map to a particular position where data is stored.
- If we give the location number according to 4 digits, we will have to reserve 0000 to 9999 addresses because anybody can use anyone as a key. There is a lot of wastage.
- In order to solve this problem, we use hashing which will produce a smaller value of the index of the hash table corresponding to the key of the user.

Collision Resolution Techniques

BYJU'S EXAM PREP

Collision Resolution

Modulo function

key % size

- In this, the hash function is used to compute the index of the array.
- The hash function can return the same hash value for two or more keys.
- When two or more keys are given the same hash value, it is called a collision. To handle this collision, we use collision resolution techniques.

Ex: Size = 10

73 % 10 ⇒ 3

51 % 10 ⇒ 1

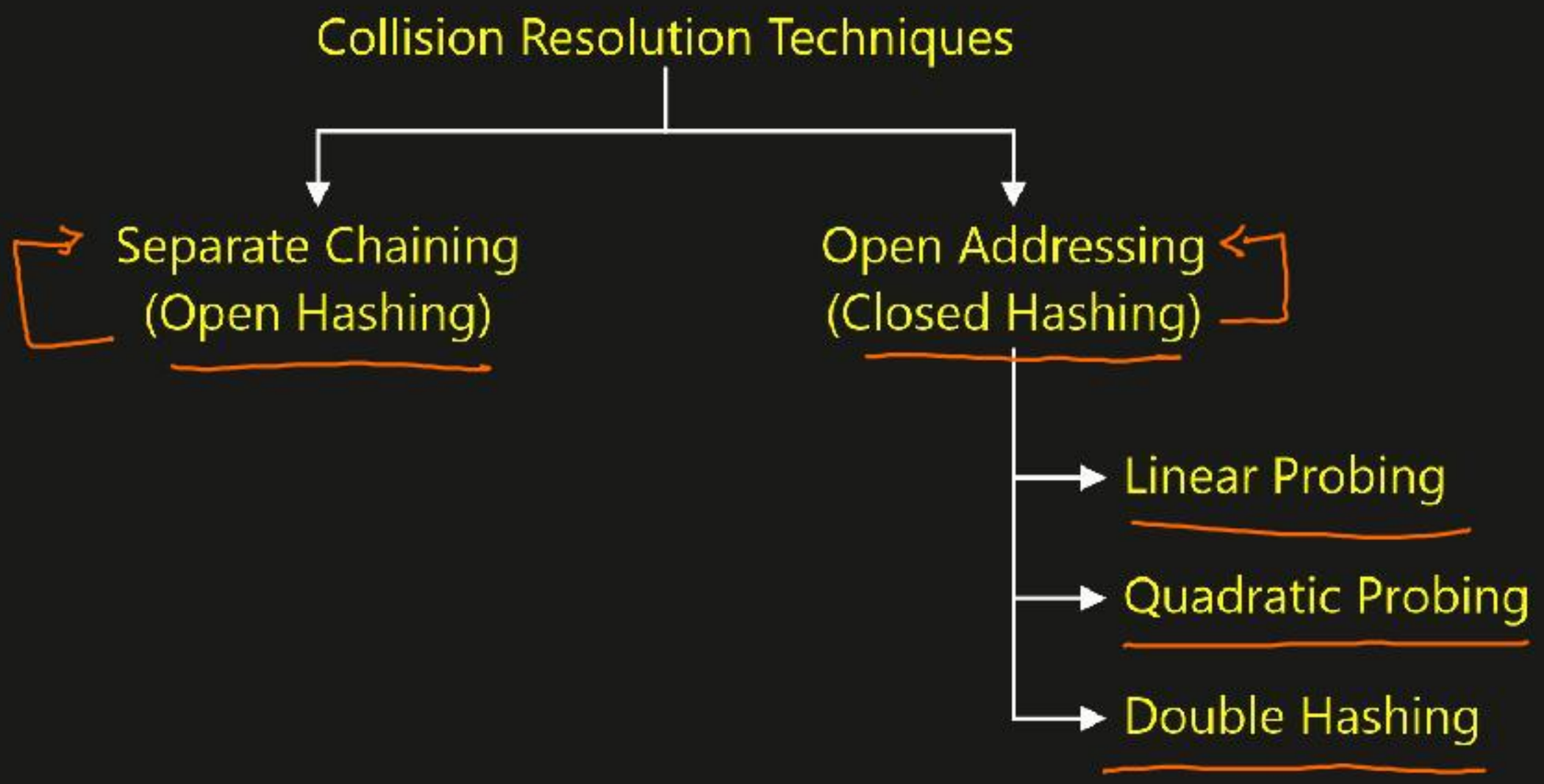
89 % 10 ⇒ 9

keys : 73, 62, 51, 89, 93, 22, 69, 19, 31, 42

73/3, 62/2, 51/1, 89/9, 93/3, 22/2, 69/9, 19/9, 31/1, 42/2

Collision Resolution Techniques

Collision Resolution



Collision Resolution

BYJU'S EXAM PREP

Chaining Method

0 ⇐ 9, 18, 27, 36, 45, 54, 63, 72, 81, 90, 99, - - -

1 ⇐ 10, 19, 28, - - - -

Keys

- Let us consider the insertion of elements 5, 28, 19,15,20,33,12,17,10 into a chained hash table. Let us suppose the hash table has 9 slots and the hash function be h (k) =k mod 9.

5 % 9 = 5

28 % 9 = 1 = 10 % 9

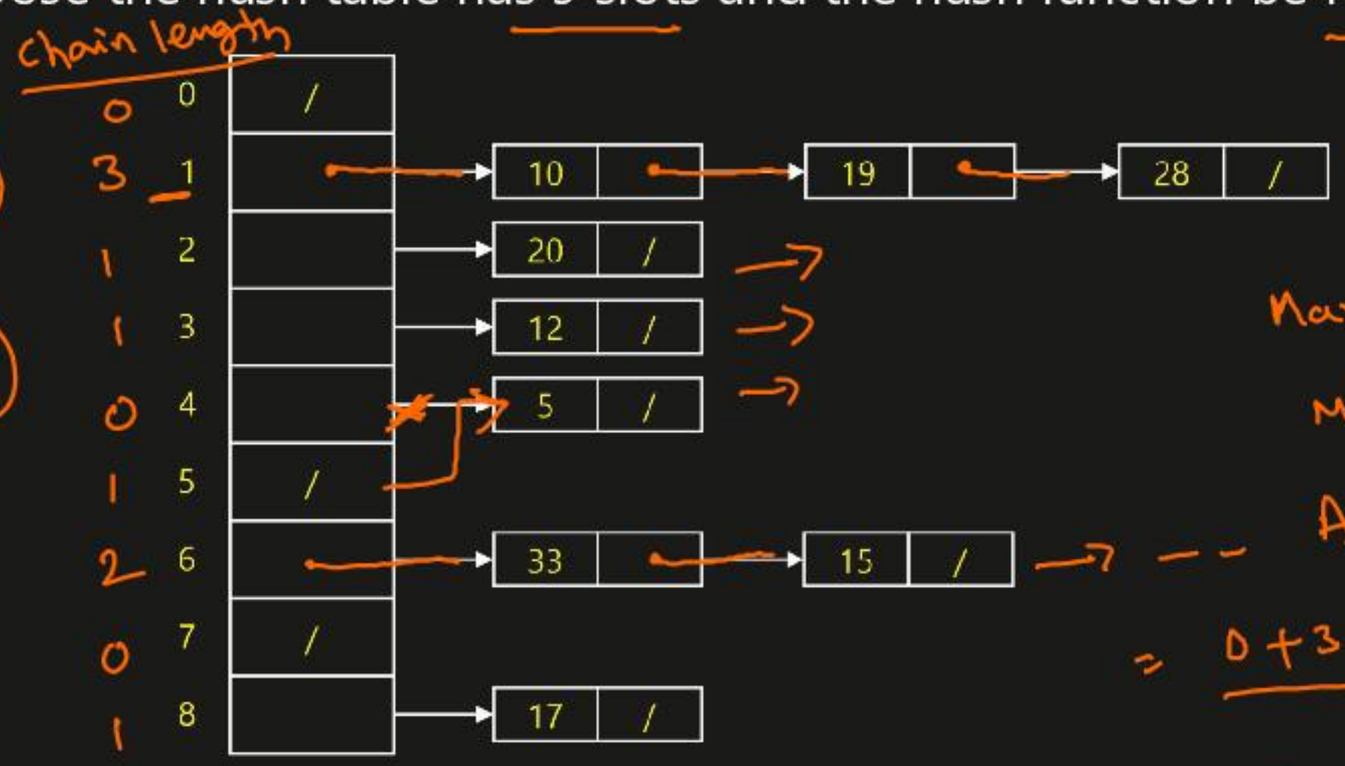
19 % 9 = 1

15 % 9 = 6 = 33 % 9

20 % 9 = 2

12 % 9 = 3

17 % 9 = 8



Chaining Method

Maximum length = 3

Minimum length = 0

Average length = 1

$= \frac{0+3+1+1+0+1+2+0+1}{9}$

$= \frac{9}{9} = 1$

BYJU'S EXAM PREP

Chaining Method

Advantages:

1) Simple to implement.

2) Hash table never fills up, we can always add more elements to the chain.

3) Less sensitive to the hash function or load factors.

4) It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.

BYJU'S EXAM PREP

x/2 Cache (High speed RAM)

CPU x ↔ Cache ↔ RAM x/20 ← HD

Random Memory allocation

Chaining Method

Disadvantages:

1) Cache performance of chaining is not good as keys are stored using a linked list. Open addressing provides better cache performance as everything is stored in the same table.

2) Wastage of Space (Some Parts of hash table are never used)

3) If the chain becomes long, then search time can become O(n) in the worst case.

4) Uses extra space for links.

Register 1 word

CPU 1 word at a time

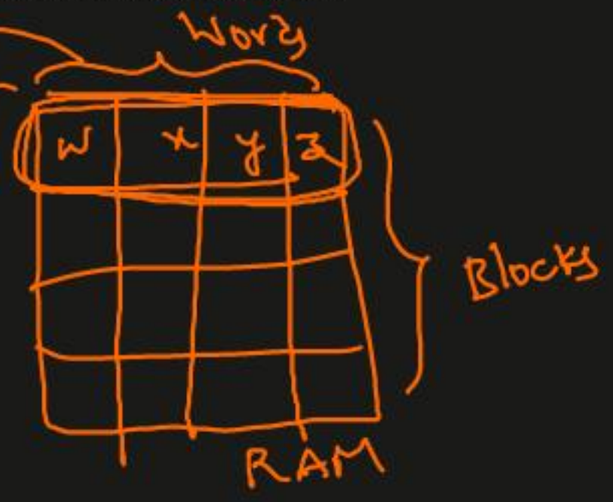
word 'x'

Cache

Miss

"x" — Same "x" — Temporal

"x" → Near by "x" — bring 4 words — spatial



BYJU'S EXAM PREP

Linear Probing

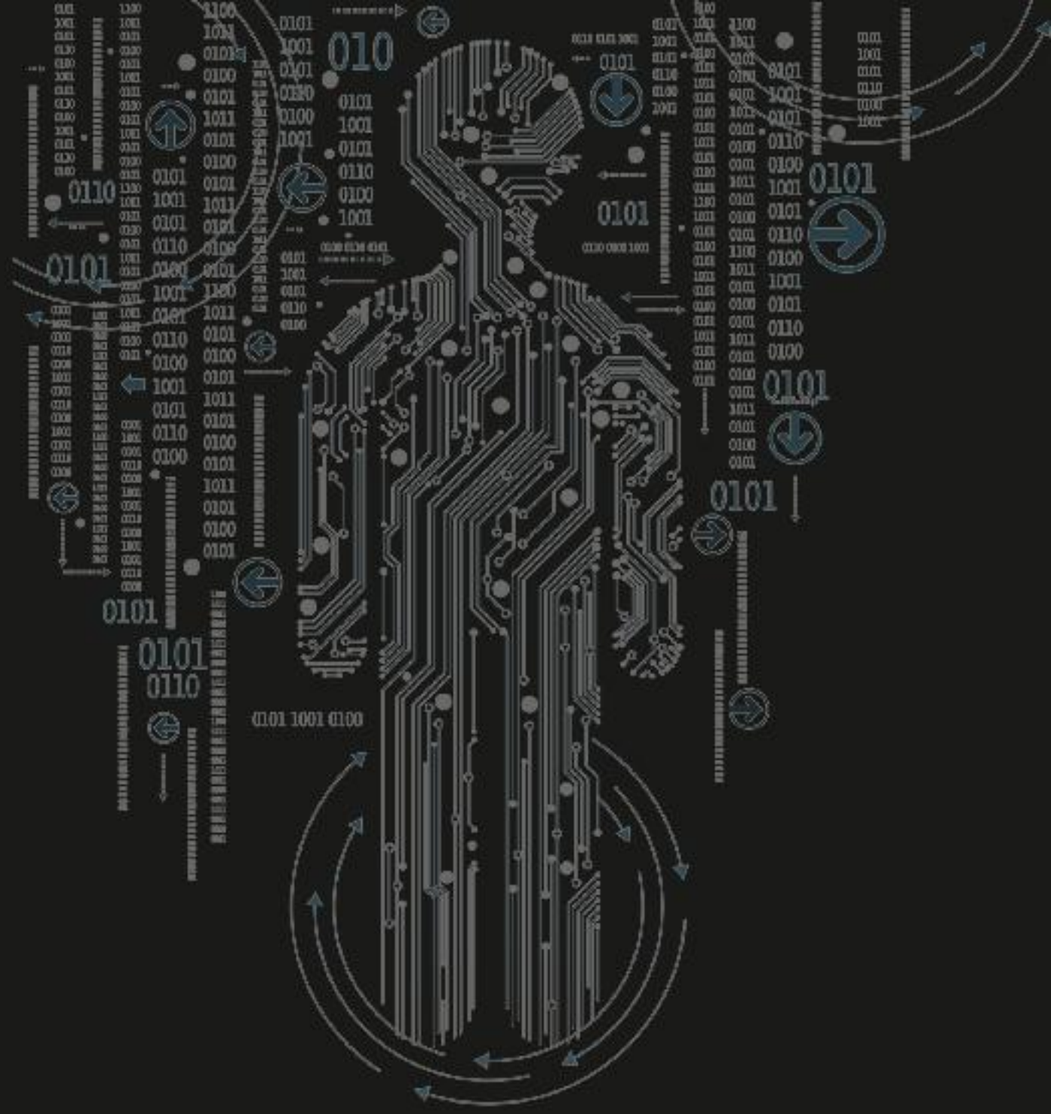
Probe == Attempt / Poin

Consider inserting the keys 24, 36, 58,65,62,86 into a hash table of size m=11 using linear probing, consider the primary hash function is h' (k) = k mod m.

BYJU'S EXAM PREP

ALGORITHMS

Four Problems of Class Divide and Conquer



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Hashing : The Process of Mapping / Converting key to hash value in the hash Table.

⇒ Hash Function : Modulo function : (Key % size)

→ If 2 or More keys maps to same hash value → Collision.

Collision Resolution →
- → Open Hashing : Seperate chaining Method
- → Closed Hashing : Open addressing Method
  - → *Linear Probing
  - → Quadratic
  - → Double Hashing

$h(k, i) = (i + k) \% m$

For Every 'k', 'i' Starts with '0' $\Rightarrow$ Probe 0

if $h(k, 0) \Rightarrow$ Collision $\Rightarrow$ i++

$h(55, 3)$

$(3 + 55) \% 10 = 58 \% 10$

$= 8$

Let $m = 10$ Keys = 22, 45, 67, 32, 85, 12, 55

$h(22, 0) = (0 + 22) \% 10 = 22 \% 10 = 2$

$h(45, 0) = (0 + 45) \% 10 = 45 \% 10 = 5$

$h(67, 0) = 7$

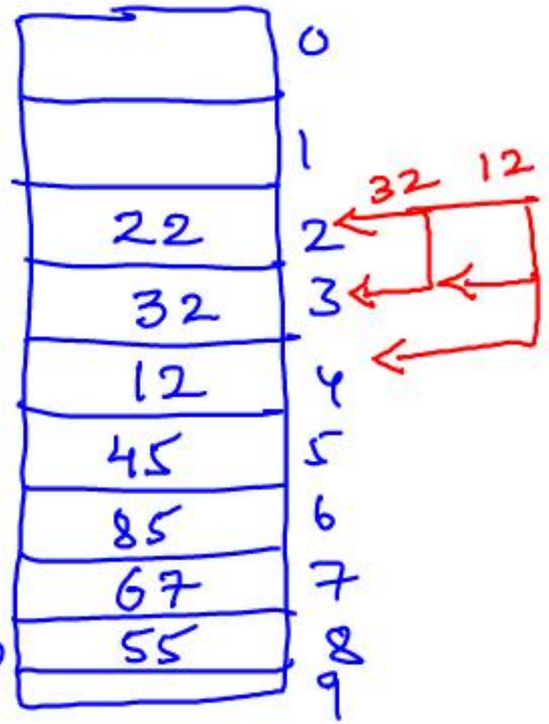
$h(32, 0) = 2$ collision $\Rightarrow h(32, 1) = (1 + 32) \% 10 = 33 \% 10 = 3$

$h(85, 0) = 5$ collision $\Rightarrow h(85, 1) = 86 \% 10 = 6$

$h(12, 0) = 2$ Collision $\Rightarrow h(12, 1) = 13 \% 10 = 3$ Collision

$h(12, 2) = 14 \% 10 = 4$

$h(55, 0) = 5$ collision $\Rightarrow h(55, 1) = 6$ collision $\Rightarrow h(55, 2) = 7$ Collision



BYJU'S EXAM PREP

Linear Probing

$$h(k, i) = (k + i) \% m$$

k = key

i = Probe Number

m = Size of Hash Table.

Consider inserting the keys 24, 36, 58,65,62,86 into a hash table of size m=11 using linear probing, consider the primary hash function is h' (k) = k mod m.

$$h(k, i) = [i + h'(k)] \% m$$

$[k \% m] \% m = k \% m$

$h(24,0) = [0 + h'(24)] \% 11 = (h'(24)) \% 11 = [(24 \% 11)] \% 11 = 2 \% 11 = 2$

$h(36,0) = 36 \% 11 = 3$

$h(58,0) = 58 \% 11 = 3$ collision $\Rightarrow h(58,1) = [1 + (58 \% 11)] \% 11$

$= (1+3) \% 11 = 4 \% 11 = 4$

$h(65,0) = 65 \% 11 = 10$

$h(86,0) = 9$

$h(62,0) = 7$

	0
	1
24	2
36	3
58	4
	5
	6
62	7
	8
86	9
65	10

58

BYJU'S EXAM PREP

Quadratic Probing

For Every key start with Probe '0' $\Rightarrow i=0$

if collision with "i", Then only $i=i+1$

$$h(k, i) = (h'(k) + c_1 i^1 + c_2 i^2) \bmod m$$

$h'(k) = k \% m$

$h(k,0)$ == Linear Probing hash function == Primary hash function =

$h(k,0) = (h'(k) + \underbrace{c_1 * 0^1}_{0} + \underbrace{c_2 * 0^2}_{0}) \% m = [h'(k)] \% m = (k \% m) \% m == k \% m$

On Probe '0' for any key 'k' if collision occurs, Then $i=i+1$

$h(k,1) = [h'(k) + c_1 \cdot 1^1 + c_2 \cdot 1^2] \% m$ [c_1 & c_2 are Pre-defined]

$\Rightarrow$ if collision $\Rightarrow i = i+1 = i=2$

$h(k,2) = [h'(k) + c_1 \cdot 2^1 + c_2 \cdot 2^2] \% m$

BYJU'S EXAM PREP

Linear Probing

Advantages-

- It is easy to compute.

Disadvantages-

- The main problem with linear probing is clustering.
- Many consecutive elements form groups.
- Then, it takes time to search an element or to find an empty bucket.

BYJU'S EXAM PREP

Quadratic Probing

$h(k) = [h'(k) + c_1 \cdot i^1 + c_2 \cdot i^2] \% m$

84, 96, 47

Consider inserting the keys 74, 28, 36,58,21,64 into a hash table of size m =11 using quadratic probing with c1=1 and c2=3. Further consider that the primary hash function is h' (k) = k mod m.

$h(74,0) = h'(k) = 74 \% 11 = 8$

$h(28,0) = 28 \% 11 = 6$

$h(36,0) = 36 \% 11 = 3$

$h(58,0) = 58 \% 11 = 3$ Collision → if LP ⇒ 4 → QP ⇒ $h(58,1) \Rightarrow 7$

$h(58,1) = [(58 \% 11) + 1 \cdot 1^1 + 3 \cdot 1^2] \% 11 = (3+1+3) \% 11 = 7$

$h(21,0) = 10$ | $h(64,0) = 9$

84	0
96	1
	2
36	3
	4
	5
28	6
58	7
74	8
64	9
21	10

47

$h(84,0) = 84 \% 11 = 7$ collision $\rightarrow$ if LP $\Rightarrow$ 8 $\Rightarrow$ 9 $\Rightarrow$ 10 $\Rightarrow$ 0

Probe 0 — Probe 1 — Probe 2 — Probe 3 — Probe 4

$Qp: h(84,1) = [(84 \% 11) + 1 \cdot 1^1 + 3 \cdot 1^2] \% 11 = [7+1+3] \% 11 = 11 \% 11 = 0$

LP: 5 Probes

Qp: 2 Probes to find free slot.

$h(96,0) = 96 \% 11 = 8$ Collision $\xrightarrow{LP}$ slot 9 $\rightarrow$ slot 10 $\rightarrow$ slot 0 $\rightarrow$ slot 1 $\Rightarrow$ 5 Probes

Probe 0 — Probe 1 — Probe 2 — Probe 3 — Probe 4

$Qp: h(96,1) = [(96 \% 11) + 1 \cdot 1^1 + 3 \cdot 1^2] \% 11 = (8+1+3) \% 11 = 12 \% 11 = 1 \Rightarrow$ 2 Probes

---

$h(47,0) = 47 \% 11 = 3$ collision $\xrightarrow{LP}$ slot 4 $\Rightarrow$ 2 Probes

Qp: $h(47,1) = [(47 \% 11) + 1 \cdot 1^1 + 3 \cdot 1^2] \% 11 \Rightarrow (3+1+3) \% 11 = 7 \% 11 = 7$ Collision

$h(47,2) = [(47 \% 11) + 1 \cdot 2^1 + 3 \cdot 2^2] \% 11 = (3+2+12) \% 11 = 17 \% 11 = 6 =$ Collision

$h(47,3) = [3 + 1 \cdot 3^1 + 3 \cdot 3^2] \% 11 = (3+3+27) \% 11 = 33 \% 11 = 0$ collision

$h(47,4) = (47 \% 11 + 1 \cdot 4^1 + 3 \cdot 4^2) \% 11$

$= (3 + 4 + 48) \% 11 = 55 \% 11 = 0$ Collision

$h(47,5) = (3 + 1 \cdot 5^1 + 3 \cdot 5^2) \% 11 = (3 + 5 + 75) \% 11 = 83 \% 11 = 6$ Collision

$h(47,6) = (3 + 1 \cdot 6^1 + 3 \cdot 6^2) \% 11 = (3 + 6 + 108) \% 11 = 117 \% 11 = 7$ Collision

Cluster : $\{6, 7, 0\}$

BYJU'S EXAM PREP

Double Hashing

$h_1(k)$ = Primary hash function = $k \% m$

$h_2(k)$ = Secondary hash function = $k \% m'$

Double hashing uses a hash function of the form

$$h(k, i) = (h_1(k) + i\, h_2(k)) \bmod m$$

when $i=0 \Rightarrow [h_1(k)] \% m = (k \% m) \% m$
$= k \% m$ == Primary hash function

Where h_1 and h_2 are auxiliary hash functions and m is the size of the hash table.

$h_1(k) = k \bmod m$ or $h_2(k) = k \bmod m'$.

Here m' is slightly less than m (say m-1 or m-2). ← Given [NOTE: if not given, take $m' = m-1$]

BYJU'S EXAM PREP

Double Hashing

On Probe 0, Every hash function is equivalent to Primary hash function.

Consider inserting the keys 76, 26, 37,59,21,65 into a hash table of size m = 11 using double hashing. Consider that the auxiliary hash functions are $h_1(k)=k \bmod 11$ and $h_2(k) = k \bmod 9$.

$h(76,0) = 76 \% 11 = 10$

$h(26,0) = 26 \% 11 = 4$

$h(37,0) = 37 \% 11 = 4$ collision

LP 5

QP $(4+1\cdot1+3\cdot1)\% 11 = 8 \% 11 = 8$

$c_1=1$ $c_2=3$

DH 5

$h(37,1) = [h_1(37) + 1\cdot h_2(37)] \% 11 = [4 + 1\cdot(37\%9)] \% 11$

$= [4 + 1\cdot1] \% 11 = 5$

	0
	1
21	2
	3
26	4
37	5
	6
	7
	8
59	9
76	10

$h(59,0) = 59 \% 11 = 4$ collision

- LP: $5 \rightarrow 6 \Rightarrow$ 3 Probes
- QP, $c_1 = 1, c_2 = 3$: $(4+1+3) \% 11 = 8 \rightarrow$ 2 Probes
- DH: $9 \rightarrow$ 2 Probes

DH

$h(59,1) = [4 + 1 \cdot (59 \% 9)] \% 11 = (4+5) \% 11 = 9$

$h(21,0) = 21 \% 11 = 10$ collision

- LP $\rightarrow$ slot 0
- QP $\rightarrow 14 \% 11 =$ slot 3
- DH $\rightarrow$ slot 2

key = 87, 32, 43

DH

$h(21,1) = [10 + 1 \cdot (21 \% 9)] \% 11$

$= (10+3) \% 11 = 13 \% 11 = 2$

$h(65,0) = 65 \% 11 = 10$ Collision

- LP $\rightarrow$ slot 0 [21] $\Rightarrow$ slot 1
- QP $\rightarrow$ 21 in slot 3 $\Rightarrow h(65,2) = (10+2+12) \% 11$
  $= 24 \% 11 =$ slot 2 collision
  $h(65,3) = 7$
- DH $\rightarrow$ slot 1

DH

$h(65,1) = [10 + 1 \cdot (65 \% 9)] \% 11$

$= (10+2) \% 11 = 12 \% 11$

$=$ slot 1

BYJU'S EXAM PREP

Load Factor

$$\boxed{Load\ factor\ (\alpha) = \frac{Number\ of\ Keys\ to\ be\ Inserted\ in\ the\ hash\ table}{Total\ size\ of\ the\ hash\ table}}$$

BYJU'S EXAM PREP

Comparison

	Linear Probing	Quadratic Probing	Double Hashing
Primary Clustering	Yes	No	No
Secondary Clustering	Yes	Yes	No
Number of Probe Sequence (m = size of table)	m	m	m^2
Cache performance	Best	Lies between the two	Poor

Comparison

BYJU'S EXAM PREP

Lp: $k_1 = 0 \to 1 \to 2 \to 3 \to 4 \to \cdots$

$k_2 = 0 \to 1 \to 2 \to 3 \to 4 \to \cdots$

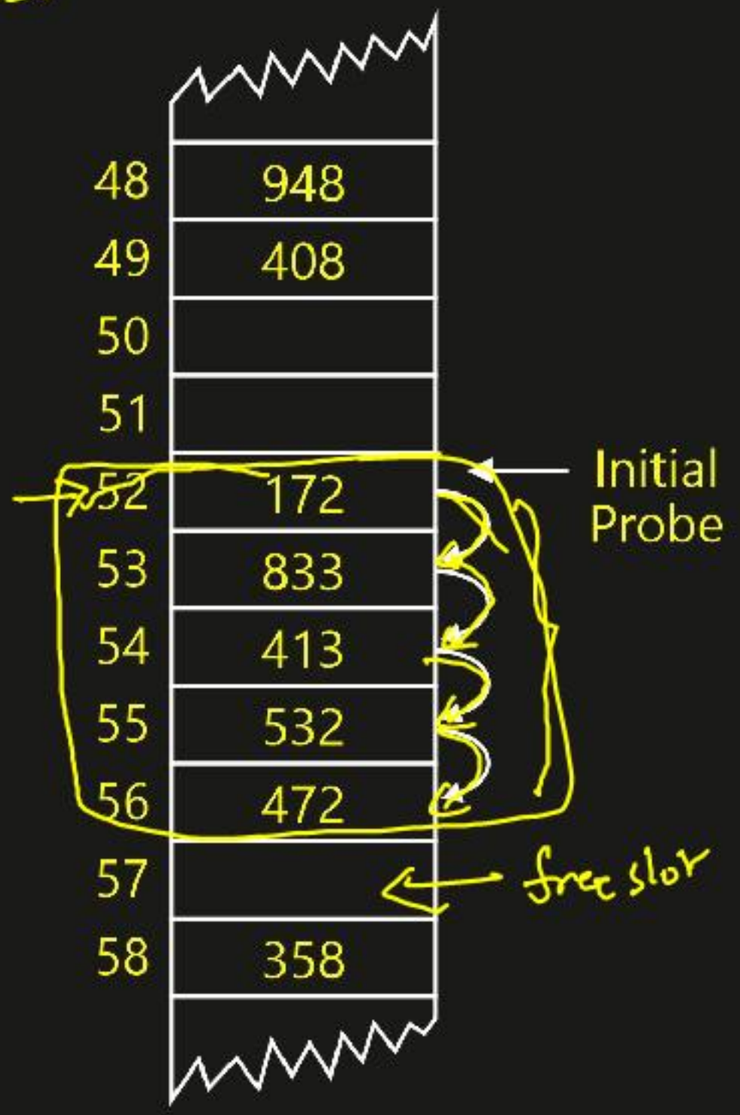
Primary Clustering

↳ Linear Probing.

Primary clustering is the tendency for a collision resolution scheme such as linear probing to create long runs of filled slots near the hash position of keys.

If the primary hash index is x, subsequent probes go to x+1, x+2, x+3 and so on, this results in Primary Clustering.

Once the primary cluster forms, the bigger the cluster gets, the faster it grows. And it reduces the performance.



Primary Clustering

BYJU'S EXAM PREP

Secondary Clustering

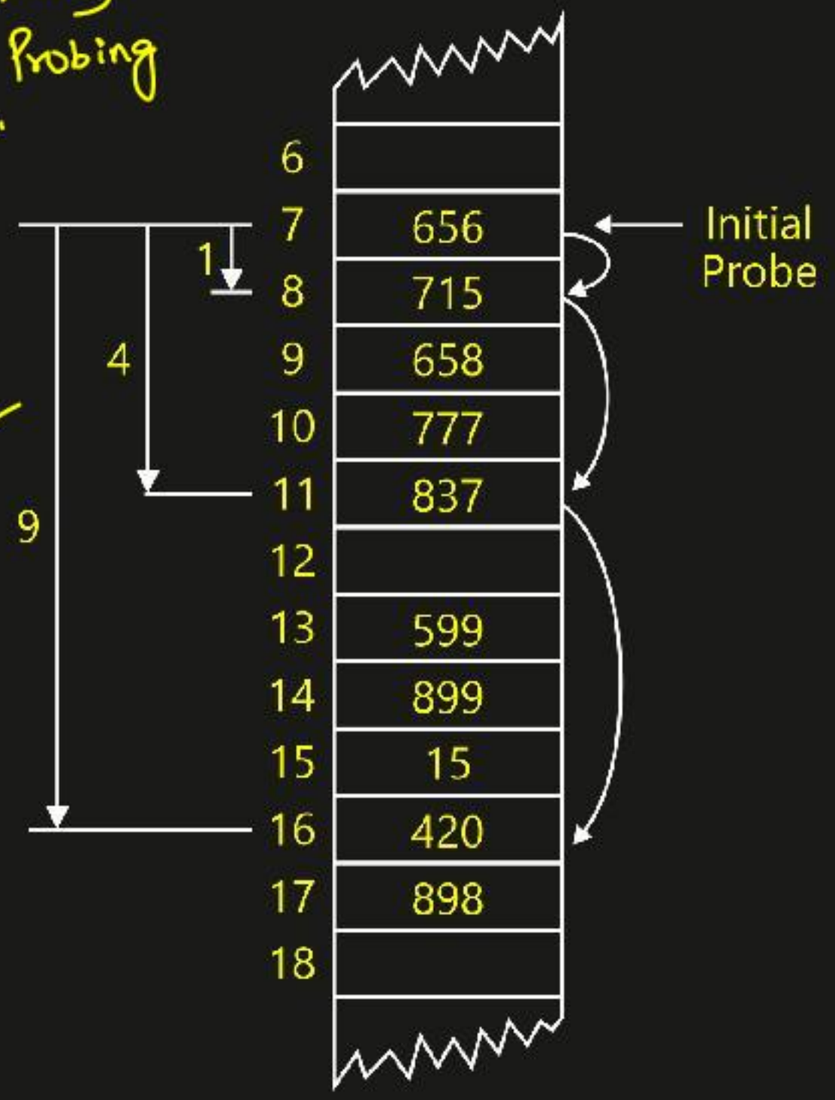
→ Quadratic Probing | sometimes linear Probing also.

Secondary clustering is the tendency for a collision resolution scheme such as quadratic probing to create long runs of filled slots away from the hash position of keys.

$x+1^2$ $x+2^2$ $x+3^2$ $x+4^2$

If the primary hash index is x, probes go to x+1, x+4, x+9, x+16, x+25 and so on, this results in Secondary Clustering.

Secondary clustering is less severe in terms of performance hit than primary clustering and is an attempt to keep clusters from forming by using Quadratic Probing. The idea is to probe more widely separated cells, instead of those adjacent to the primary hash site.



Secondary Clustering

$k_1 = 21$

DH: $m = 11$ Let slot 10 is occupied:

$h(21, 0) = 21 \% 11 = 10$ ✗

$h_2(k) = k \% 9$

DH: $h(21, 1) = (10 + 3) \% 11 = 2$ ✓

Qp: $c_1 = 1, c_2 = 3$ $(10 + 4) \% 11 = 3$

Lp: $10 \rightarrow 0$

$k_2 = 32$

$h(32, 0) = 32 \% 11 = 10$ ✗

DH: $h(32, 1) = [32 + 1 \cdot (32 \% 9)] \% 11 = (32 + 5) \% 11 = 37 \% 11 = 4$ ✓

Qp: $h(32, 1) = ((32 \% 11) + 1 \cdot 1^1 + 3 \cdot 1^2] \% 11 = (10 + 1 + 3) \% 11 = 14 \% 11 = 3$

$c_1 = 1, c_2 = 3$

Lp: $10 \rightarrow 0$

$k_3 = 43$

$h(43, 0) = 43 \% 11 = 10$ ✗

DH: $h(43, 1) = [10 + 1 \cdot (43 \% 9)] \% 11 = (10 + 7) \% 11 = 17 \% 11 = 6$ ✓

Qp: $h(43, 1) = (10 + 1 \cdot 1^1 + 3 \cdot 1^2) \% 11 = 14 \% 11 = 3$

Lp: $10 \rightarrow 0$

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

Primary Clustering

S.NO.	SEPARATE CHAINING	OPEN ADDRESSING
1.	Chaining is Simpler to implement.	Open Addressing requires more computation.
2.	In chaining, Hash table never fills up, we can always add more elements to chain.	In open addressing, table may become full.
3.	Chaining is Less sensitive to the hash function or load factors.	Open addressing requires extra care for to avoid clustering and load factor.
4.	Chaining is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.	Open addressing is used when the frequency and number of keys is known.
5.	Cache performance of chaining is not good as keys are stored using linked list.	Open addressing provides better cache performance as everything is stored in the same table.
6.	Wastage of Space (Some Parts of hash table in chaining are never used).	In Open addressing, a slot can be used even if an input doesn't map to it.
7.	Chaining uses extra space for links.	No links in Open addressing

Primary Clustering

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

Q1. Given a hash table T with 40 slots that stores 2000 elements, the load factor α for T is _____.

[GATE 2015]

A. 50

B. 0.02

C. 500

D. 2

$$\alpha = \frac{2000}{40} = 50$$

BYJU'S EXAM PREP

Q2. A hash table contains 10 buckets and uses linear probing to resolve collisions. The key values are integers and the hash function used is key % 10. If the values 43, 165, 62, 123, 142 are inserted in the table, in what location would the key value 142 be inserted ?

[GATE 2005]

A. 2

B. 3

C. 4

D. 6

43 ⇒ 3

165 ⇒ 5

62 ⇒ 2

123 ⇒ 3 ⇒ 4

142 ⇒ 2 → 3 → 4 → 5 → 6

	0
	1
62	2
43	3
123	4
165	5
142	6
	7
	8
	9

BYJU'S EXAM PREP

Q3. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

* [Linear Probing]

[GATE 2007]

A. 8, _ , _ , _ , _ , _ , 10

B. 1, 8, 10, _ , _ , _ , 3

C. 1, _ , _ , _ , _ , _ ,3

D. 1, 10, 8, _ , _ , _ , 3

$(3x+4) \% 7$

0	1	2	3	4	5	6
1	8	10	_	_	_	3

BYJU'S EXAM PREP

Q4. Consider a hash table of size 11 that uses open addressing with "linear probing". Let h(k) = k mod 11 be the hash function used. A sequence of records with keys

43 36 92 87 11 4 71 13 14

10, 3, 4, 0, 1, 5, 6, 2 7

is inserted into an initially empty hash table, the bins of which are indexed from zero to ten. What is the index of the bin into which the last record is inserted ?

[GATE 2008]

A. 2

B. 4

C. 6

D. 7

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

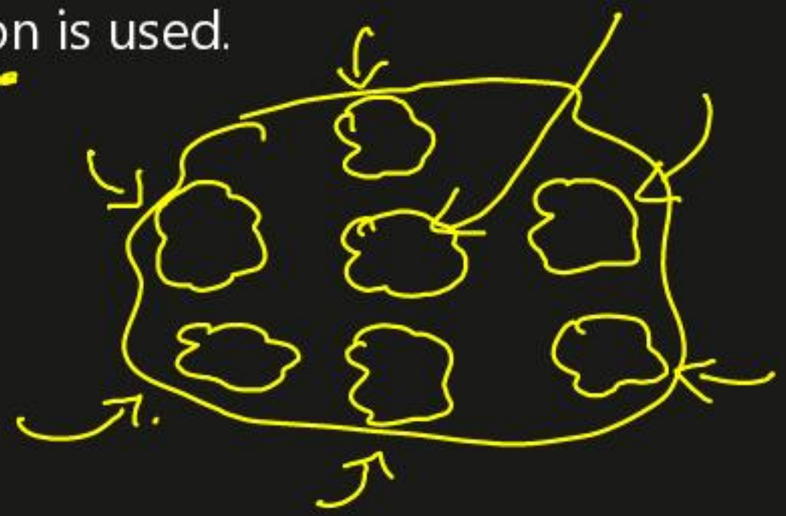
Divide and Conquer

Algorithm Design Techniques
- Divide & conquer
- Dynamic Programming
- Greedy method
- Backtracking
- Branch & Bound

} out of syllabus

- A divide and conquer algorithm is a strategy of solving a large problem by
  1. Breaking the problem into smaller sub-problems (Divide)
  2. Solving the sub-problems, and
  3. Combining them to get the desired output.

  } Conquer
- To use the divide and conquer algorithm, recursion is used.



BYJU'S EXAM PREP

Applications Of Divide and Conquer

Divide and conquer strategy has various application areas such as:

1. Binary search
2. Finding the maximum and minimum in an array
3. Merge sort
4. Quicksort
5. Strassen's matrix multiplication
6. Towers of Hanoi [Recursion Based]

and many more.

Four Problems of Class Divide and Conquer

Search

To be Contd ...

The Process Of finding an element (key) from the given list of elements is called as searching.

There are 2 ways to perform searching:

- Linear Search
- Binary Search

BYJU'S EXAM PREP

Linear Search

```
int RecursiveLS(int arr[], int value, int index, int n) {
    int pos = 0;
     if(index > = n)
    {
        return 0;
    }
     else if (arr[index] == value)
    {
        pos = index + 1;
        return pos;
    }
     else
    {
        return RecursiveLS(arr, value, index+1, n);
    }
    return pos;
}
```

BYJU'S EXAM PREP

Binary Search

While implementing binary search, we divide the array into 2 halves and check if the number to be searched could be on the left half or right half. Then, we go to that half and again divide the array into further two halves. This process goes on until the number to be searched is found.

BYJU'S EXAM PREP

Binary Search

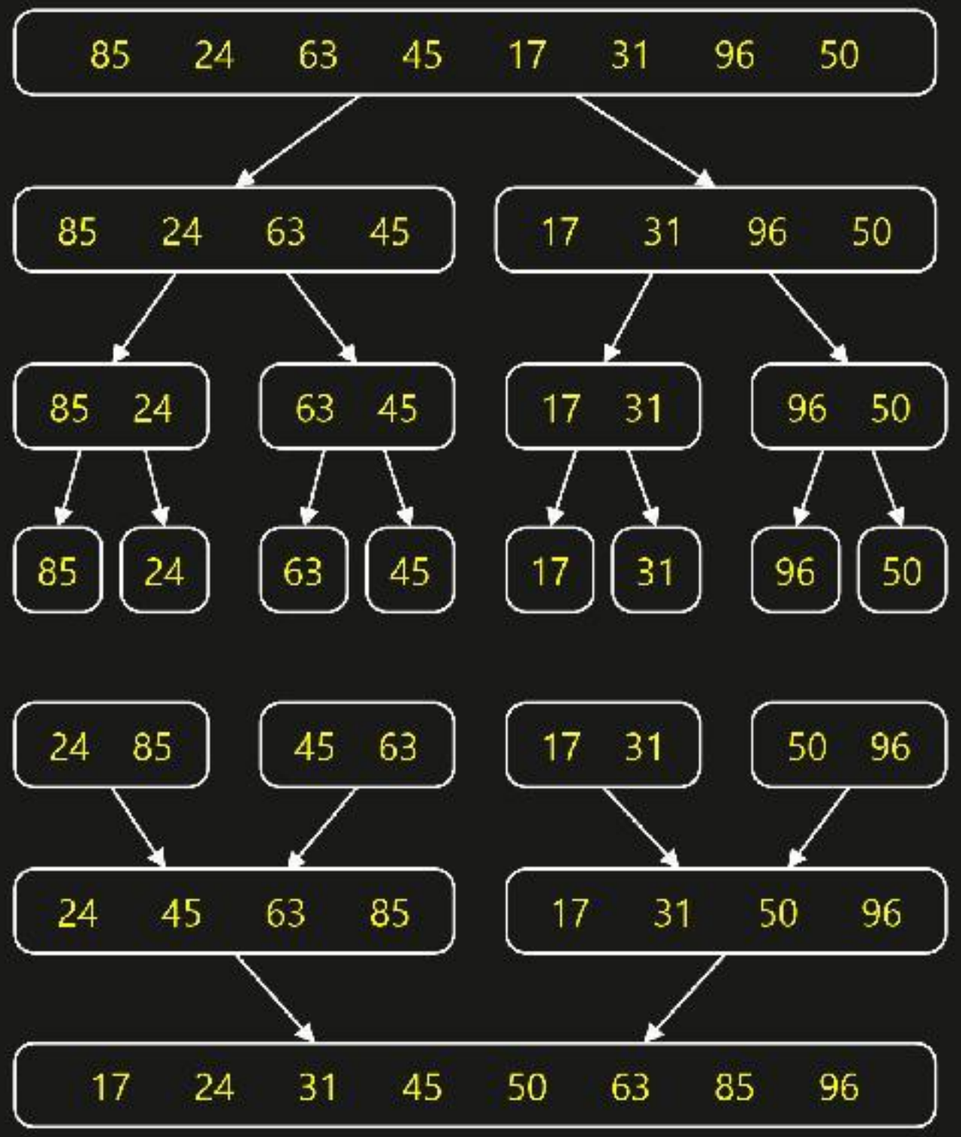
Given a sorted array 'A' of size 'n' we need to find the position of an element, say 'x' in A. Let i be the starting index and j be the last index.

```
BSearch(A, i, j, x){
  if(i == j){
    if(A[i] == x)
      return i;
    else
      return -1;
  }
  mid = (i + j)/2;
if(A[mid] == x)
    return mid;
  else if(x < A[mid])
    return BSearch(A, i, mid-i, x)
  else
    return BSearch(A, mid+1, j, x)
}
```

Binary Search Example

Four Problems of Class Divide and Conquer

Merge Sort



Merge Sort

BYJU'S EXAM PREP

Merge Sort

```
void mergeSort(int arr[ ], int l, int r)
{
if (l < r)
{
// Finding mid element
int m = 1+(r-l)/2;
// Recursively sorting both the halves
mergeSort(arr, l, m);
mergeSort(arr, m+1, r);

// Merge the array
merge(arr, l, m, r);
}
}
```

```
void merge(int arr[ ], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;
// Create temp arrays
int L[n1], R[n2];
// Copy data to temp array
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1+ j];
// Merge the temp arrays
i = 0;
j = 0;
k = l;
```

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

Merge Sort

```
while (i < n1 && j < n2)
{
if (L[i] <= R[j])
{
arr[k] = L[i];
i++;
}
else
{
arr[k] = R[j];
j++;
}
k++;
}
// Copy the remaining elements of L[ ]
while (i < n1){
arr[k] = L[i];
i++;
k++;}
```

```
// Copy the remaining elements of
R[]
while (j < n2)
{
arr[k] = R[j];
j++;
k++;
}
}
```

Four Problems of Class Divide and Conquer

Quick Sort

- This is one of the most efficient sorting algorithms.
- It selects a pivot value from an array and then divides the rest of the array elements into two subarrays one on the right side of the pivot and the other on the left side.
- It then compares the pivot element to each element and sorts the array recursively.

Quick Sort Example

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

Quick Sort

```
void quicksort(int number[25],int first,int last){
int i, j, pivot, temp;
if(first<last){
pivot=first;
i=first;
j=last;
while(i<j){
while(number[i]<=number[pivot]&&i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j){
temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}
}
```

BYJU'S EXAM PREP

Quick Sort

Worst Case Complexity [Big-O]: $O(n^2)$ It occurs when the pivot element picked is either the greatest or the smallest element. ...

- Best Case Complexity [Big-omega]: O(n*log n)
- Average Case Complexity [Big-theta]: O(n*log n)

Recurrence Relations:

- T(n) = T(n-1) +O(n) : Worst Case
- T(N) = 2T(n/2) + n : Average and Best Case

Towers Of Hanoi

- Tower of Hanoi game is a puzzle invented by French mathematician Édouard Lucas in 1883.
- Tower of Hanoi consists of three pegs or towers with n disks placed one over the other.
- The objective of the puzzle is to move the stack to another peg following these simple rules.

1. Only one disk can be moved at a time.
2. No disk can be placed on top of the smaller disk.

TOH with 2 discs

- Move a Disc from A to B using C
- Move a Disc from A to C
- Move a Disc from B to C using A

TOH with 3 discs

- Move 2 Discs from A to B using C
- Move a Disc from A to C
- Move 2 Discs from B to C using A

TOH with n discs

- Move n-1 Discs from A to B using C
- Move a Disc from A to C
- Move n-1 Discs from B to C using A

Four Problems of Class Divide and Conquer

Q1. Consider a hash table with 9 slots. The hash function is $h(k) = k \bmod 9$. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

[GATE 2014]

A. 3, 0, and 1

B. 3, 3, and 3

C. 4, 0, and 1

D. 3, 0, and 2

#/n9

Four Problems of Class Divide and Conquer

BYJU'S EXAM PREP

Q2. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020 ?

[GATE 2015]

A. $h(i) = i^2 \bmod 10$

B. $h(i) = i^3 \bmod 10$

C. $h(i) = (11 * i^2) \bmod 10$

D. $h(i) = (12 * i) \bmod 10$

4/20

Four Problems of Class Divide and Conquer

Q3. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 4 slots are unfilled after the first 4 insertions ?

[GATE 2014]

A. $(96 \times 96 \times 96 \times 96)/100^4$

B. $(99 \times 98 \times 97 \times 96)/100^4$

C. $(97 \times 96 \times 95 \times 94)/100^4$

D. $(97 \times 96 \times 95 \times 94)/(4! \times 100^4)$

$\frac{4}{n}$

Four Problems of Class Divide and Conquer

Q4. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true ?

1. 9679, 1989, 4199 hash to the same value
2. 1471, 6171 hash to the same value
3. All elements hash to the same value
4. Each element hashes to a different value

4/10

[GATE 2004]

A. 1 only

B. 2 only

C. 1 and 2 only

D. 3 or 4

BYJU'S EXAM PREP

Q5. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

Which one of the following choices gives a possible order in which the key values could have been inserted in the table ? [GATE 2010]

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

A. 46, 42, 34, 52, 23, 33

B. 34, 42, 23, 52, 33, 46

C. 46, 34, 42, 23, 52, 33

D. 42, 46, 33, 23, 34, 52

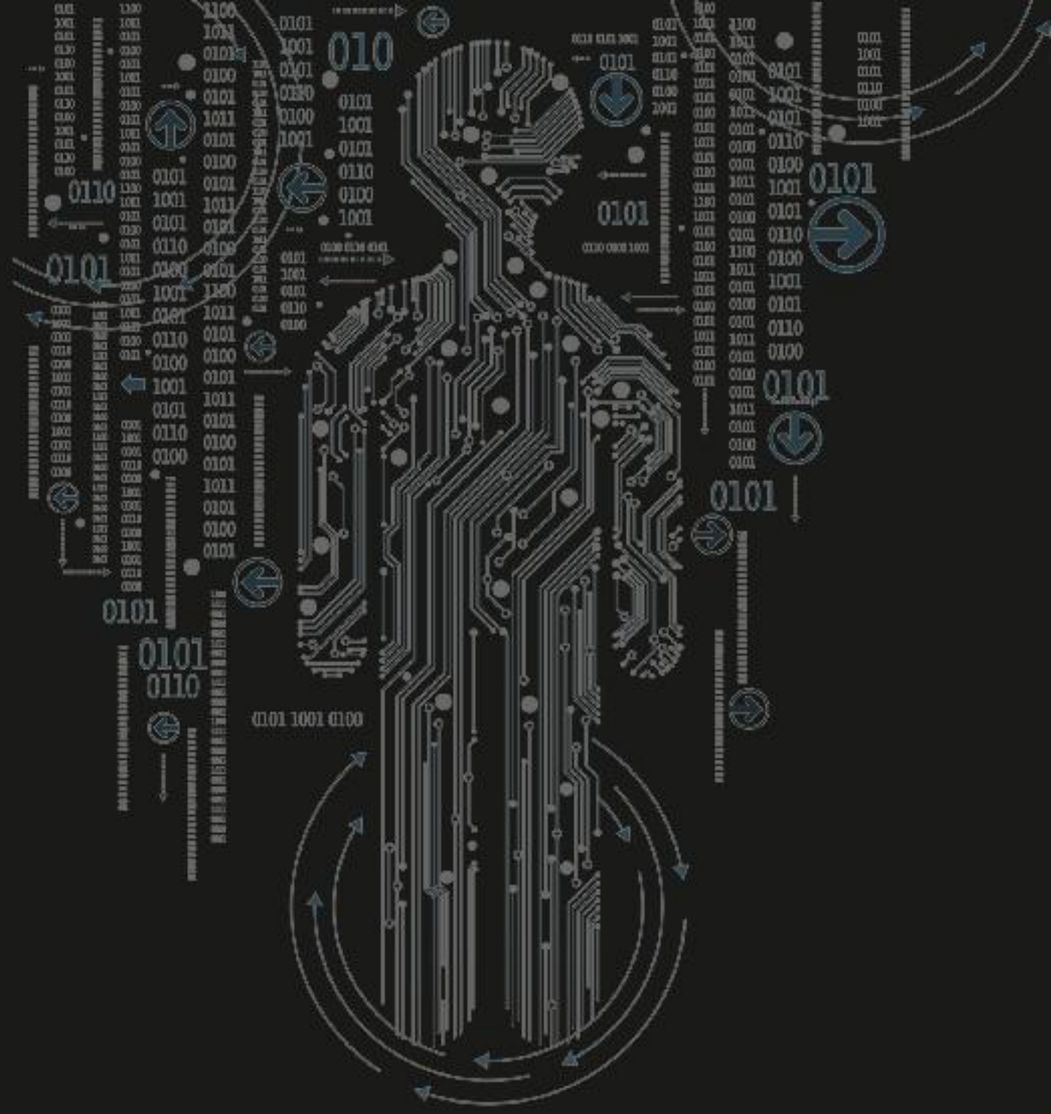
H/W

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Selection, Insertion, Bubble and Heap Sort



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

Applications Of Divide and Conquer

Divide and conquer strategy has various application areas such as:

1. Binary search
2. Finding the maximum and minimum in an array
3. Merge sort
4. Quicksort
5. Strassen's matrix multiplication

and many more.

Search

The Process Of finding an element (key) from the given list of elements is called as searching.

There are 2 ways to perform searching:

- Linear Search : Compare with Each Element in the list
- Binary Search

Best - case : First Element : Key $O(1)$

Average - Case : Middle Element : Key : $\frac{n}{2}$

$\Rightarrow O(\frac{n}{2}) \Rightarrow O(n)$

Worst - Case : Last Element / Not Found : key $\Rightarrow O(n)$

$T(n) = T(n-1) + 1 \Rightarrow O(n)$

Linear Search

Using Recursion

key

```
int RecursiveLS(int arr[], int value, int index, int n) {
    int pos = 0;                                  =1
    if(index >= n)                                =1
    {
        return 0;                                 =1
    }
    else if (arr[index] == value)                 =1
    {
        pos = index + 1;                          =1
        return pos;                               =1
    }
    else                                          =1
    {
        return RecursiveLS(arr, value, index+1, n);
    }
    return pos;
}
```

$T(n)$

$T(n-1)$

Example

	0	1	2	3	4
arr	10	25	32	15	17

int RecursiveLS(arr, 15, 0, 5)

Pos = 0

if 0 >= 5 False

else if (arr[0] == 15) 10 == 15 False

else

RLS(arr, 15, 1, 5)

Pos = 0

1 >= 5 False

arr[1] == 15 25 == 15 False

↳ RLS(arr, 15, 2, 5) → RLS(arr, 15, 3, 5)

Pos = 4

Return 4

low = 49501
high = 65000
mid = 57250

low = 57251
high = 65000
mid = 61125

low = 61126
high = 65000 ⇒ mid = 63063,

low = 63064
high = 65000
mid = 64032

BYJU'S EXAM PREP

Binary Search

Divide & Conquer approach

***Assumption: input is sorted list.

If key ≥ A[mid]

$mid = \frac{32501 + 65500}{2} = \frac{98001}{2} = 49500$

While implementing binary search, we divide the array into 2 halves and check if the number to be searched could be on the left half or right half. Then, we go to that half and again divide the array into further two halves. This process goes on until the number to be searched is found.

	low=0	1	2	3	4	5	6	7 = high
A:	15	25	35	45	55	60	75	80

key

$mid == \frac{low + high}{2}$

$mid = \frac{0+7}{2} = \frac{7}{2} = 3$

if key == A[mid] return mid;

else if key < A[mid] high = mid − 1, repeat:

else low = mid + 1, repeat

int arr[65000];

unsigned integer (16-bit Processor) ⇒ 0 to $(2^{16} - 1)$ ⇒ 0 to 65,535

low = 0 high = 65000

key > A[mid]

low = mid + 1

$mid = \frac{0 + 65000}{2} = 32500$

low = 32501

Binary Search

$n \to n/2 \to (n/2)/2 \to (n/2^2)/2 \to (n/2^3)/2 \to \cdots \; n/2^k = 1$

$\frac{n}{2^k} = 1 \Rightarrow 2^k = n \Rightarrow k = \log_2 n$

$\Rightarrow T.C : O(\log_2 n)$

Given a sorted array 'A' of size 'n' we need to find the position of an element, say 'x' in A. Let i be the starting index and j be the last index.

```
BSearch(A, i, j, x){
  if(i == j){
    if(A[i] == x)
      return i;
    else
      return -1;
  }
  mid = (i + j)/2;
if(A[mid] == x)
    return mid;
  else if(x < A[mid])
    return BSearch(A, i, mid-1, x)
  else
    return BSearch(A, mid+1, j, x)
}
```

Constant time

→ search unsuccessful

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

$\Rightarrow T.C : O(\log_2 n)$

Binary Search Example

$A[i], B[j]$

$i < j$

	A	B	C
$i = 0$	85	$j = 0$: 24	24
		$j = 1$ ✗	85

Merge Sort

Sorting : Arrangement of elements either in Ascending or descending order.

- → Selection sort
- → Insertion sort
- → Bubble sort
- → Quick sort
- → Merge sort
- → Heap sort

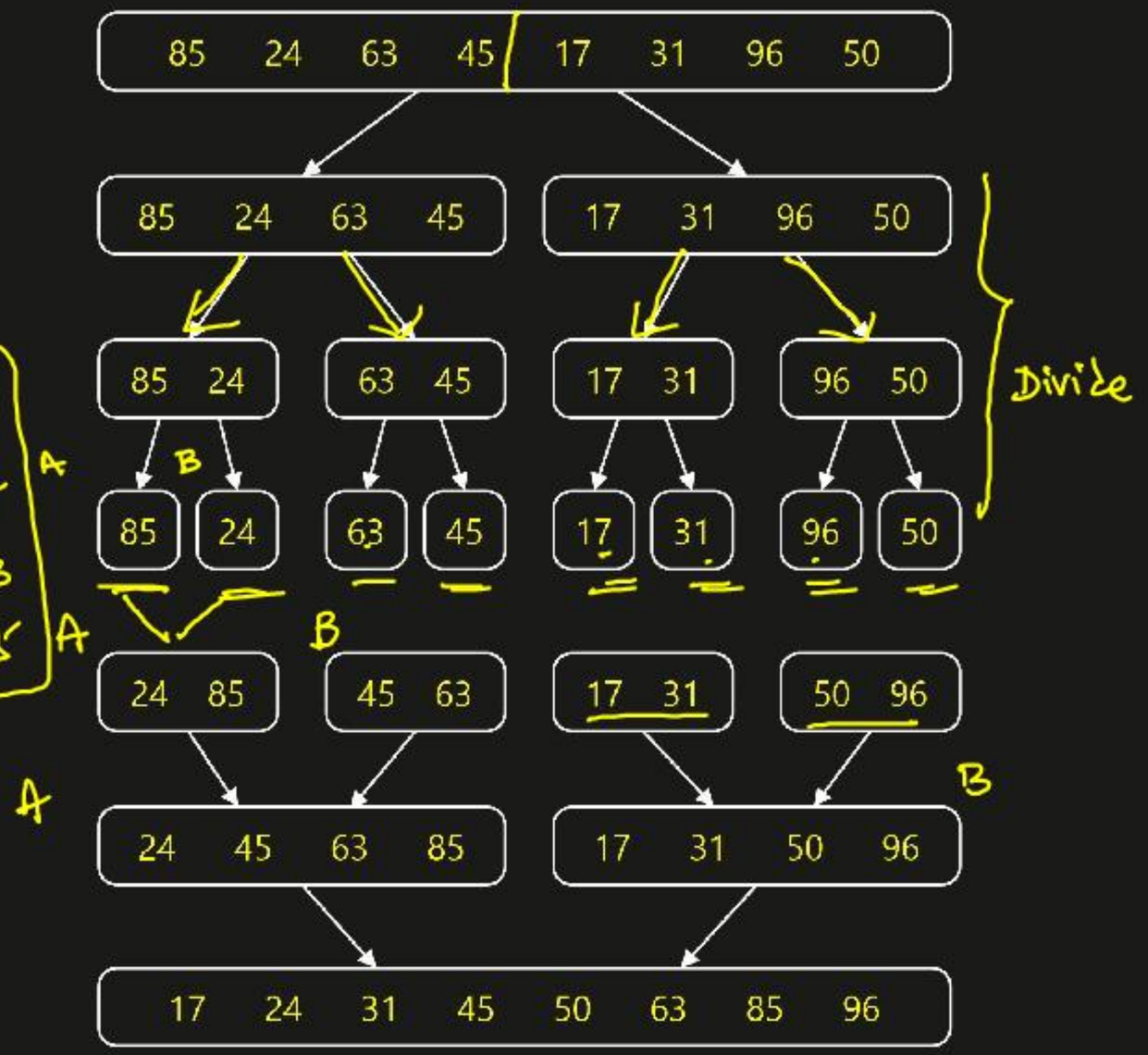
} D & C

- → Radix sort
- → Bucket sort ---

	A	B	j	C
$i = 0$	24	45	0	24
$i = 1$	85	63	$j = 1$	45
			$j = 2$	63
				85

A

A



Merge Sort

	A
$i=0$	24
$i=1$	45
$i=2$	63
$i=3$	85
$i=4$	

	B
$j=0$	17
$j=1$	31
$j=2$	50
$j=3$	96

C
17
24
31
45
50
63
85
96

Divide = 1

Conquer : Merge.

n comparisions

$\Rightarrow O(n)$

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + n + 1 = 2T\left(\frac{n}{2}\right) + n$$

$$\Rightarrow O(n \cdot \log n)$$

$$\boxed{l + \frac{(r-l)}{2}} = \frac{2l + r - l}{2} = \frac{r+l}{2}$$

BYJU'S EXAM PREP

Merge Sort

```
void mergeSort(int arr[ ], int l, int r)
{
if (l < r)
{
// Finding mid element
int m = l+(r-l)/2;
// Recursively sorting both the halves
mergeSort(arr, l, m);
mergeSort(arr, m+1, r);

// Merge the array
merge(arr, l, m, r);
}
}
```

```
void merge(int arr[ ], int l, int m, int r)
{
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;
// Create temp arrays
int L[n1], R[n2];
// Copy data to temp array
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1+ j];
// Merge the temp arrays
i = 0;
j = 0;
k = l;
```

$$\frac{(low + high)}{2}$$

1. $\boxed{low + high}$ = Accum

2. $\frac{\text{Accumulator}}{2}$

low = 65534

high = 65535

out-of range

1,31,069

65533

Merge Sort

```
while (i < n1 && j < n2)
{
if (L[i] <= R[j])
{
arr[k] = L[i];
i++;
}
else
{
arr[k] = R[j];
j++;
}
k++;
}
// Copy the remaining elements of L[ ]
while (i < n1){
arr[k] = L[i];
i++;
k++;}
```

```
// Copy the remaining elements of
R[]
while (j < n2)
{
arr[k] = R[j];
j++;
k++;
}
}
```

Quick Sort

- This is one of the most efficient sorting algorithms.
- It selects a pivot value from an array and then divides the rest of the array elements into two subarrays one on the right side of the pivot and the other on the left side.
- It then compares the pivot element to each element and sorts the array recursively.

Quick Sort Example

A

0	1	2	3	4	5
45	65	25	15	95	35

$i = 1$ $j = 5$

pivot

$A[0] \rightarrow$ pivot

$i \rightarrow 1$

$j \rightarrow n-1$

$A[i] >$ pivot do nothing.

else $i{+}{+}i$

$A[j] \leq$ pivot do nothing

else $j{-}{-}j$

Compare i & j indices

$i \leq j$ Swap $A[i]$ & $A[j]$

$i > j$ Swap $A[j]$ & pivot.

$A[1] >$ pivot True

$A[5] <$ pivot True

$i = 1$ $j = 5$ $i < j$ swap $A[i]$ & $A[j]$

45	35	25	15	95	65

$i=1$ $\quad$ $j=3$ $\quad$ $j=4$ $\quad$ $j=5$

45 35 25 15 95 65

Pivot $\quad$ $i=2$ $\quad$ $i=3$ $\quad$ $i=4$

15 25 35 45 65 95

$i=4$ $\quad$ $j=3$

$i>j$ $\quad$ A[j] & Pivot are Swapped.

$i=1$ $\quad$ $j=2$

15 35 25 $\quad$ 45

Pivot $\quad$ $j=1$

$j=0$

$i=1, j=1$ $\quad$ $i=2$

95 65 $\quad$ 45

pivot

$i=1, j=0$

$i>j$

Swap a[j] & pivot

15 $\quad$ 35 25 $\quad$ 45

$i=1, j=1$

Pivot $\quad$ $i=2$

$i=2, j=1$ $\quad$ $i>j$

15 25 35 $\quad$ 45

$i=2, j=1$

$i>j$ swap

a[j] & Pivot

65 $\quad$ 95

$$T(n)=2T\left(\frac{n}{2}\right)+n$$

65 $\quad$ 95

Average | Worst-case

$T(n) = 2T(\frac{n}{2}) + n$ list is already sorted.

$j = 5$

$i = 1$

15 25 35 45 55 65

Pivot

$j = 0$

$i = 1, j = 0$ $i > j$ Swap a[j] & pivot

$$\boxed{T(n) = T(n-1) + n}$$

$\boxed{15}$ $\boxed{25 \quad 35 \quad 45 \quad 55 \quad 65}$

(n-1) elements

$j = 5$

$i = 1$

$\boxed{65}$ 55 45 35 25 15

$i = 6$

Pivot

(n-1) elements

$i = 6$ $j = 5$ $i > j$ 15 55 45 35 25 $\boxed{65}$

Quick Sort

```
void quicksort(int number[25],int first,int last){
int i, j, pivot, temp;
if(first<last){
pivot=first;
i=first;
j=last;
while(i<j){
while(number[i]<=number[pivot]&&i<last)
i++;
while(number[j]>number[pivot])
j--;
if(i<j){
temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
temp=number[pivot];
number[pivot]=number[j];
number[j]=temp;
quicksort(number,first,j-1);
quicksort(number,j+1,last);
}
}
```

Quick Sort

Worst Case Complexity [Big-O]: $O(n^2)$ It occurs when the pivot element picked is either the greatest or the smallest element. ...

- Best Case Complexity [Big-omega]: O(n*log n)
- Average Case Complexity [Big-theta]: O(n*log n)

Recurrence Relations:

- $T(n) = T(n-1) + O(n)$: Worst Case
- $T(N) = 2T(n/2) + n$: Average and Best Case

To be Contd ...

Towers Of Hanoi

- Tower of Hanoi game is a puzzle invented by French mathematician Édouard Lucas in 1883.
- Tower of Hanoi consists of three pegs or towers with n disks placed one over the other.
- The objective of the puzzle is to move the stack to another peg following these simple rules.

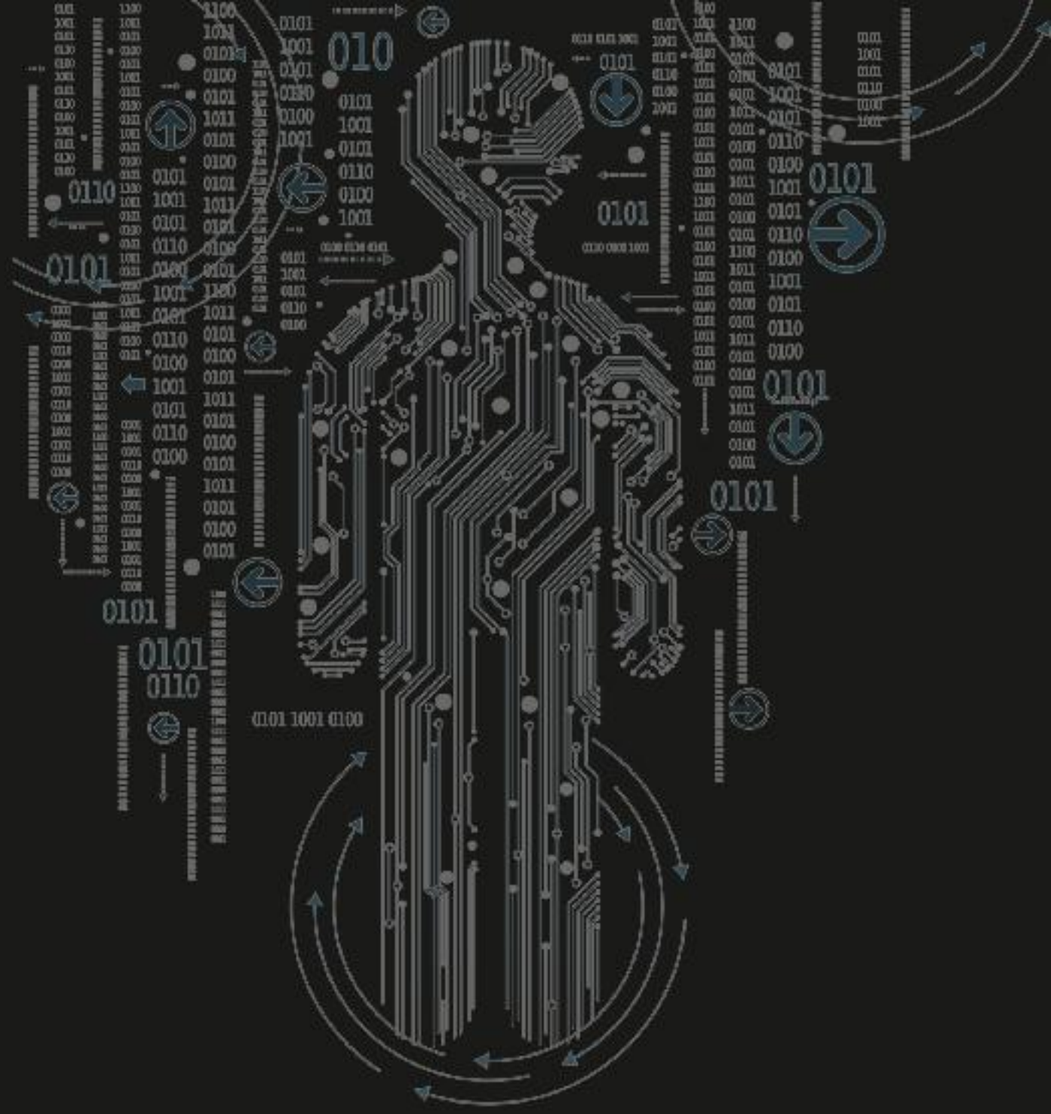
1. Only one disk can be moved at a time.
2. No disk can be placed on top of the smaller disk.

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Interactive Quiz - 1



Guidelines to Attend Live Class

BYJU'S EXAM PREP

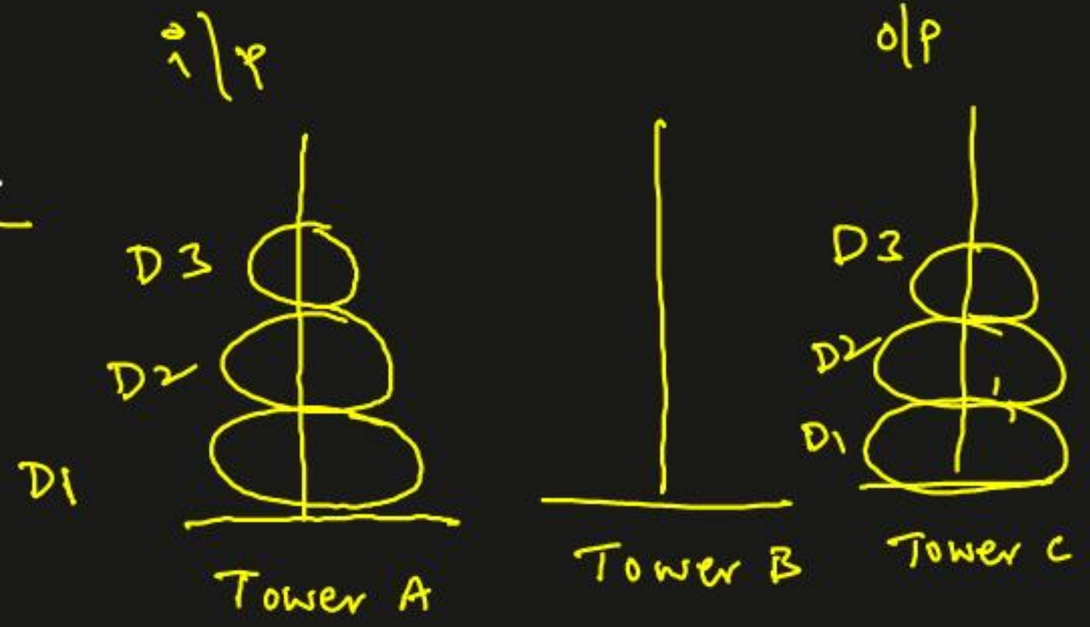
- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Towers Of Hanoi

- Tower of Hanoi game is a puzzle invented by French mathematician Édouard Lucas in 1883.
- Tower of Hanoi consists of three pegs or towers with n disks placed one over the other.
- The objective of the puzzle is to move the stack to another peg following these simple rules.

1. Only one disk can be moved at a time.
2. No disk can be placed on top of the smaller disk.

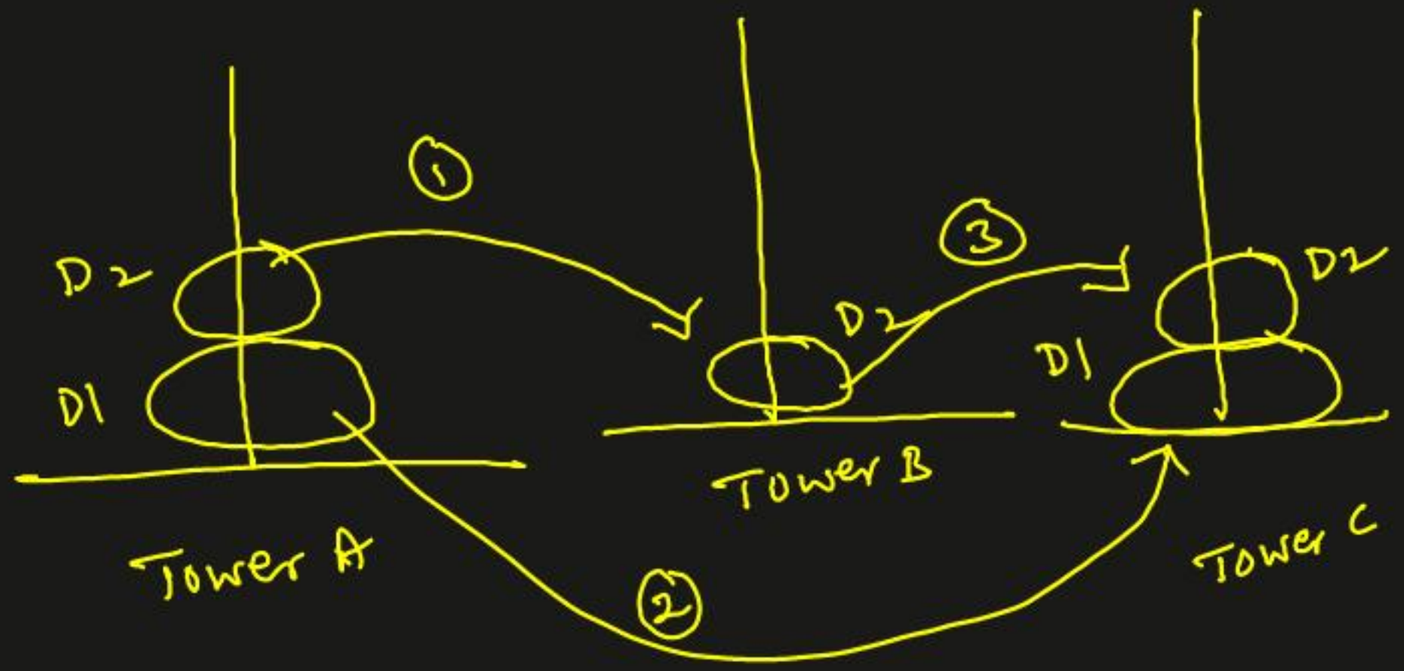


Interactive Quiz - 1

TOH with 2 discs

- Move a Disc from A to B using C
- Move a Disc from A to C
- Move a Disc from B to C using A

Only 1 disk



3 Movements.

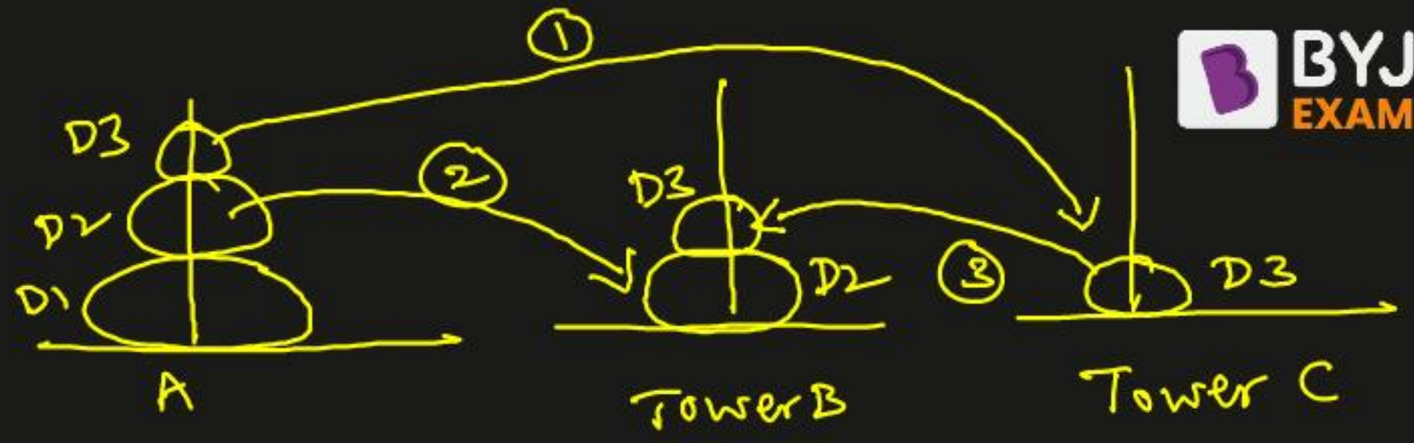
Interactive Quiz - 1

TOH with 3 discs

7 Movements

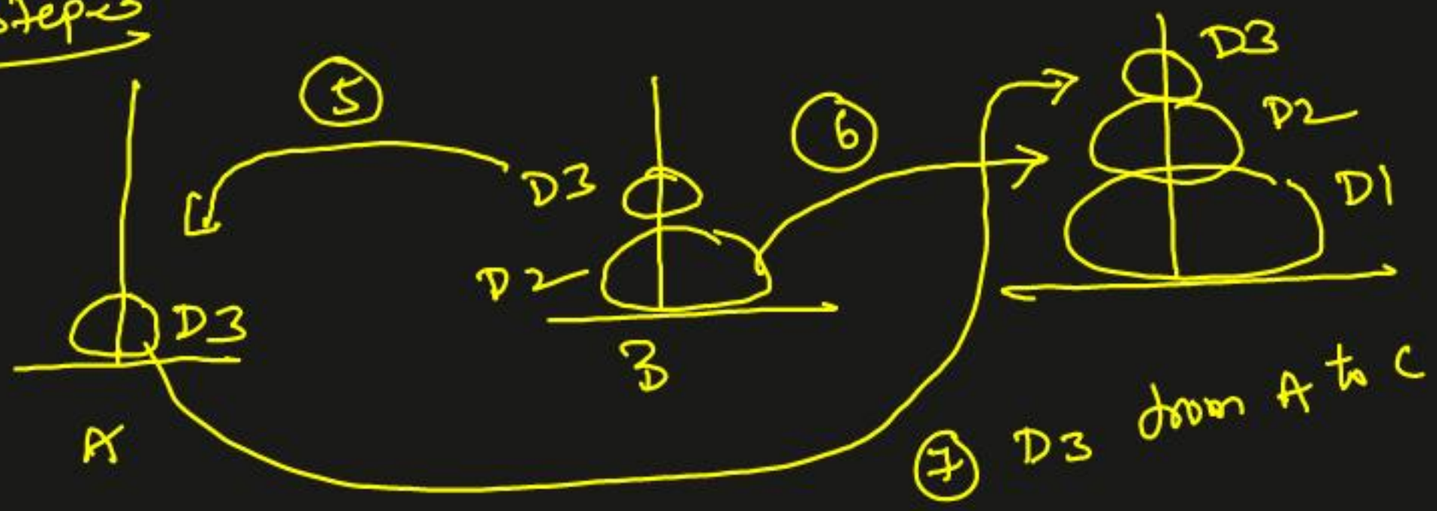
- Move 2 Discs from A to B using C
- Move a Disc from A to C
- Move 2 Discs from B to C using A

Step-1

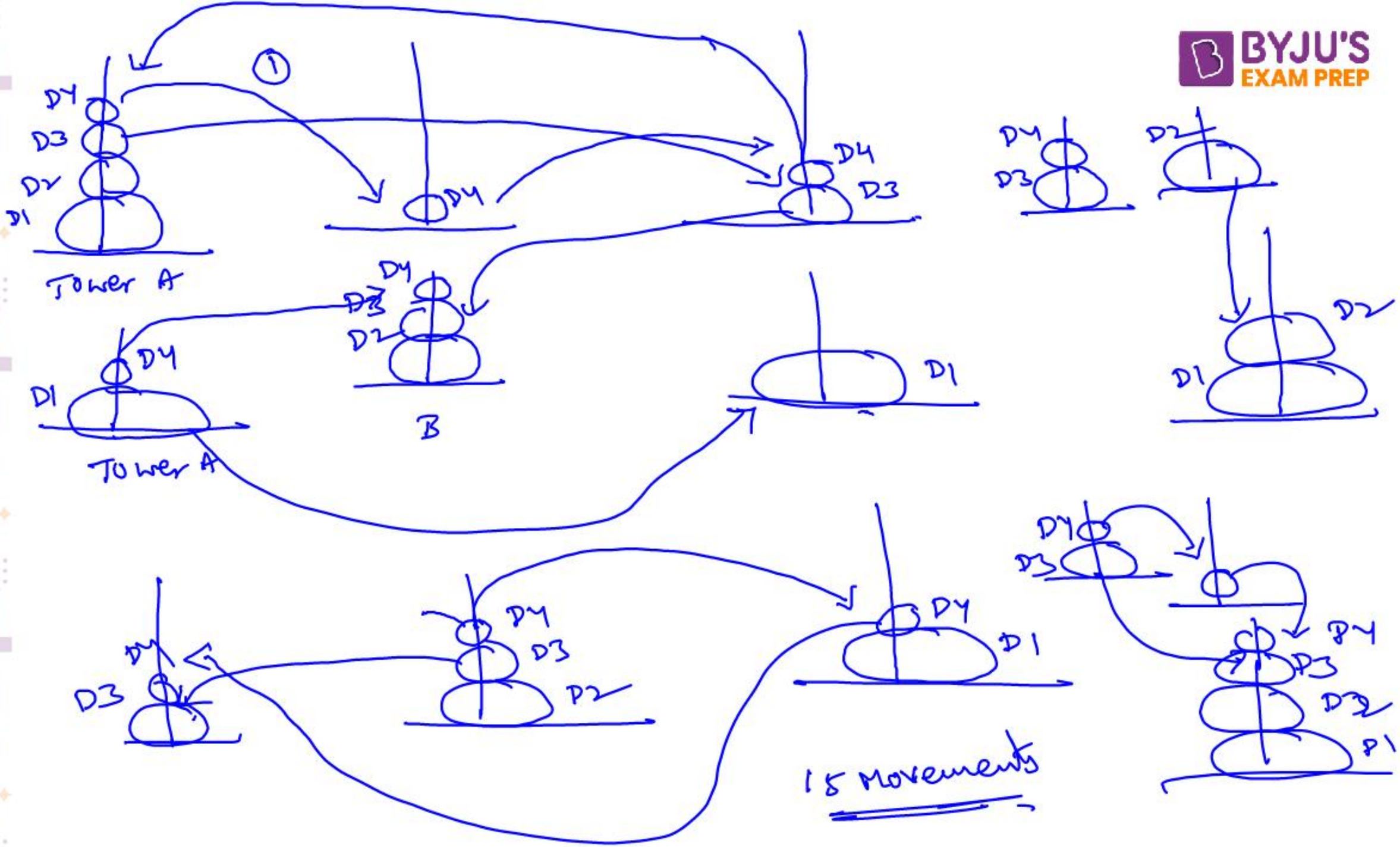


Step-2

Step-3



1. D3 from A to C
2. D2 from A to B
3. D3 from C to B
4. D1 from A to C
5. D3 from B to A
6. D2 from B to C
7. D3 from A to C


BYJU'S EXAM PREP
1
D4
D3
D2
D1
Tower A
D4
D4
D3
D4
D3
D2
D1
D2
D1
D4
D3
D2
B
D4
D1
Tower A
D1
D4
D3
D3
D4
D3
D2
D4
D1
D4
D3
D2
D1
15 Movements

Interactive Quiz - 1

BYJU'S EXAM PREP

TOH with n discs

$T(n-1)$

- Move n-1 Discs from A to B using C
- Move a Disc from A to C -1
- Move n-1 Discs from B to C using A

$T(n-1)$

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

T.C $O(2^n)$

	Movements
1 disk	1
2 discs	3
3 discs	7
4 discs	15
⋮	
n discs	$(2^n - 1)$ Movements

Interactive Quiz - 1

BYJU'S EXAM PREP

Q1. Consider a hash table with 9 slots. The hash function is $h(k) = k \bmod 9$. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

[GATE 2014]

A. 3, 0, and 1

B. 3, 3, and 3

C. 4, 0, and 1

D. 3, 0, and 2

Interactive Quiz - 1

BYJU'S EXAM PREP

$n \% 10 \Rightarrow$ LSB digit [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Q2. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020 ?

[GATE 2015]

A. $h(i) = i^2 \bmod 10$

B. $h(i) = i^3 \bmod 10$

C. $h(i) = (11 * i^2) \bmod 10$

D. $h(i) = (12 * i) \bmod 10$

uniform distribution

i	i^2	i^3	$(11*i^2)$	$(12*i)$	$i^2 \% 10$	$11*i^2 \% 10$	$12*i \% 10$	$i^3 \% 10$
0	0	0	0	0	0	0	0	0
1	1	1	11	2	1	11	2	1
2	4	8	44	6	4	44	4	8
3	9	27	99	8	9	99	6	7
4	16	64	176	0	6	176	8	4
5	25	125	275	2	5	275	0	5
6	36	216	396	4	6	396	2	6
7	49	343	539	6	9	539	4	3
8	64	512	704	8	4	704	6	2
9	81	729	891	8	1	891	8	9

2, 3, 8 missing.

missing 1, 3, 5, 7, 9

Interactive Quiz - 1

BYJU'S EXAM PREP

Q3. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 4 slots are unfilled after the first 4 insertions ?

[GATE 2014]

A. $(96 \times 96 \times 96 \times 96)/100^4$

B. $(99 \times 98 \times 97 \times 96)/100^4$

C. $(97 \times 96 \times 95 \times 94)/100^4$

D. $(97 \times 96 \times 95 \times 94)/(4! \times 100^4)$

$\frac{96}{100} * \frac{96}{100} * \frac{96}{100} * \frac{96}{100}$

if open-addressing

$\frac{96}{100} * \frac{95}{100} * \frac{94}{100} * \frac{93}{100}$

Interactive Quiz - 1

BYJU'S EXAM PREP

Slot: 2 4 1 9 9 1 3 9

Q4. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true ?

1. 9679, 1989, 4199 hash to the same value
2. 1471, 6171 hash to the same value
3. All elements hash to the same value
4. Each element hashes to a different value

[GATE 2004]

A. 1 only

B. 2 only

C. 1 and 2 only

D. 3 or 4

BYJU'S EXAM PREP

Linear Probing.

Q5. A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

Which one of the following choices gives a possible order in which the key values could have been inserted in the table ? (A)

[GATE 2010]

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

A. 46, 42, 34, 52, 23, 33

B. 34, 42, 23, 52, 33, 46

C. 46, 34, 42, 23, 52, 33

D. 42, 46, 33, 23, 34, 52

0	
1	
2	42
3	52
4	34
5	23
6	46
7	33
8	
9	

BYJU'S EXAM PREP

Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared, and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst-case complexity are of $O(n^2)$ where n is the number of items.

Interactive Quiz - 1

Bubble Sort Example

End = n-1

0 45 15 35 5 25 16

Pass 1:

15 45 35 5 25 16

15 35 45 5 25 16

15 35 5 45 25 16

15 35 5 25 45 16

15 35 5 25 16 45

5 (n-1) comparisons

Pass-2 0 to n-2 (End--)

i j

15 35 5 25 16 45

(n-2)

i j

15 5 35 25 16 45

15 5 25 35 16 45

15 5 25 16 35 45

Pass 3: 0 to n-3 (End--)

15 5 25 16 35 45

5 15 25 16 35 45

5 15 16 25 35 45

(n-3)

Pass-4

5 15 16 25 35 45

No. of swaps == zero, There return result array as Sorted.

Best-Case : Already sorted i/p

i/p : 10 20 30 40 50 60

Number of comparisions = (n-1)

⇒ O(n)

Pass 1: Compare 10, 20 NO swap
20, 30 NO swap
30, 40 NO swap
40, 50 NO swap
50, 60 NO swap.

Number of swaps == 0
return result array as Sorted list.

Worst-Case | Average-Case

R.R $\boxed{T(n) = (n-1) + T(n-1)}$

OK $\boxed{T(n-1) + n}$

$$(n-1) + (n-2) + (n-3) + (n-4) + \cdots + 1$$

$$= 1 + 2 + 3 + 4 + 5 + \cdots + (n-1)$$

$$= \frac{n(n-1)}{2} \Rightarrow \frac{(n-1)(n-2)}{2} \text{ (or) } \frac{n(n+1)}{2} \Rightarrow \frac{n^2+n}{2}$$

$$O\left(\frac{n^2}{2}\right) + O\left(\frac{n}{2}\right)$$

$$\Rightarrow O(n^2)$$

Interactive Quiz - 1

Bubble Sort

```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

BYJU'S EXAM PREP

Selection Sort

To find smallest element in the random list of 'n' numbers $\Rightarrow (n-1)$ comparisions.

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of $O(n^2)$, where n is the number of items.

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

$\Rightarrow \boxed{T(n) = T(n-1) + n} \Rightarrow O(n^2)$

BYJU'S EXAM PREP

Selection Sort Example

unsorted

Sorted:

72 35 15 42 56 62 23

Pass 1:

unsorted

15 35 72 42 56 62 23

Sorted

Pass 2:

15 23 72 42 56 62 35

Pass 3:

15 23 35 42 56 62 72

No. of swaps : (n-1)

BYJU'S EXAM PREP

Selection Sort

Algorithm :

Step 1 – Set MIN to location 0.

Step 2 – Search the minimum element in the list.

Step 3 – Swap with value at location MIN.

Step 4 – Increment MIN to point to next element.

Step 5 – Repeat until list is sorted.

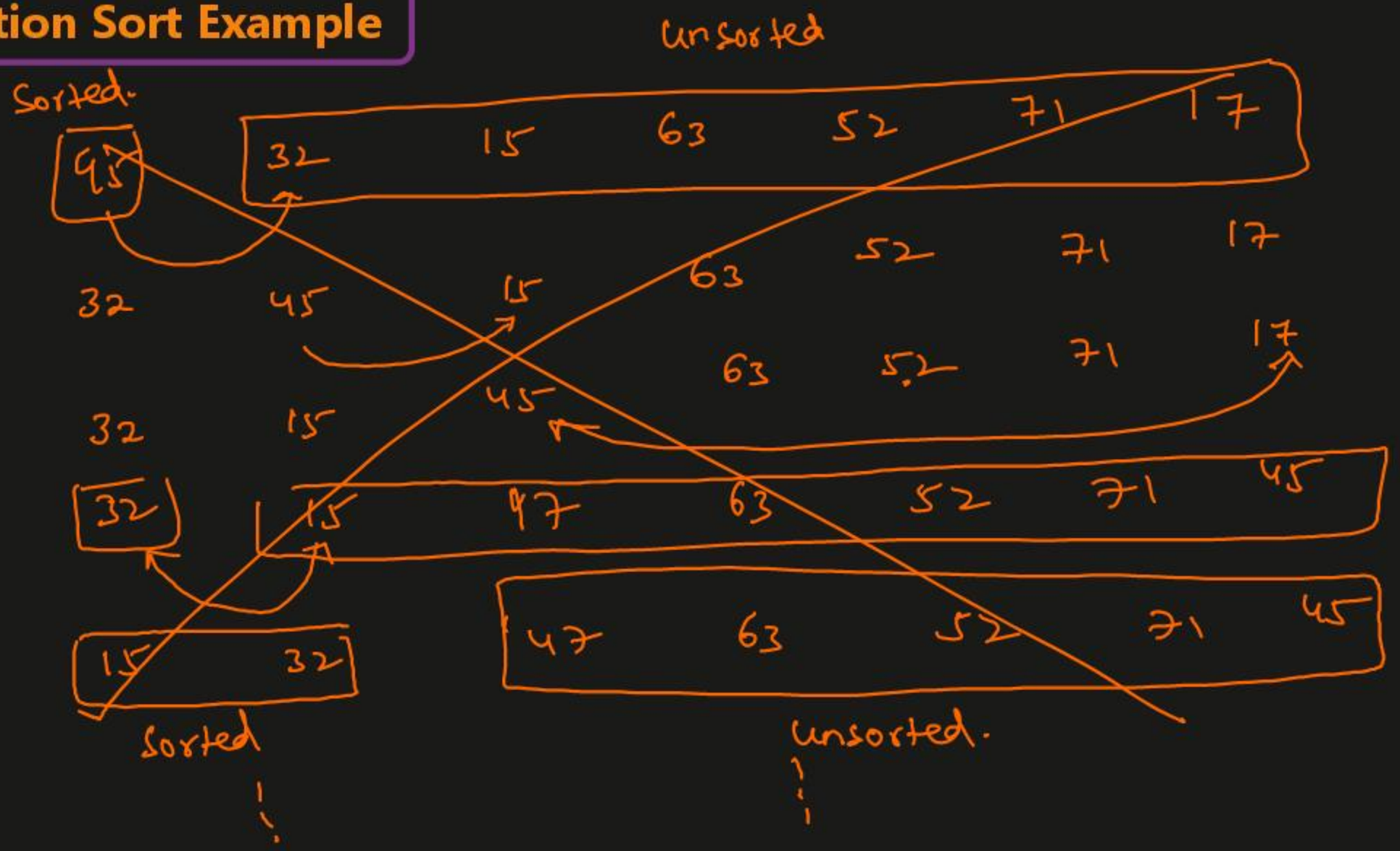
BYJU'S EXAM PREP

Insertion Sort

- This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.
- The array is searched sequentially, and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst-case complexity are of $O(n^2)$, where n is the number of items.

BYJU'S EXAM PREP

Insertion Sort Example



Sorted array

Unsorted

45 25 32 15 5 70 10

Pass 1:

Sorted 5

45 25 32 15 70 10 unsorted.

Pass 2:

Sorted

5 10

45 25 32 15 70 unsorted

Pass 3: Sorted

5 10 15

45 25 32 70 unsorted

Pass 4: Sorted

5 10 15 25

45 32 70 unsorted

Pass 5:

5 10 15 25 32 45 70

Pass 6:

5 10 15 25 32 45 70

BYJU'S EXAM PREP

Insertion Sort

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

$O(n^2)$ Average | worst

$O(n)$ Best-case

Algorithm :

Step 1 – If it is the first element, it is already sorted. return 1;

Step 2 – Pick next element.

Step 3 – Compare with all elements in the sorted sub-list.

Step 4 – Shift all the elements in the sorted sub-list that is greater than the value to be sorted.

Step 5 – Insert the value.

Step 6 – Repeat until list is sorted.

BYJU'S EXAM PREP

Heap Sort

Heap sort processes the elements by creating the min-heap or max-heap using the elements of the given array. Min-heap or max-heap represents the ordering of array in which the root element represents the minimum or maximum element of the array.

Heap sort basically recursively performs two main operations -

- Build a heap H, using the elements of array.
- Repeatedly delete the root element of the heap formed in 1st phase.

1. Max Heap
2. Min Heap

Max-Heap: Parent > All it's children.

BYJU'S EXAM PREP

Heap Sort Example

Sorted array → n elements

0	1	2	3	4	5		n-1
						- - -	

Ex: 7 elements Sorted array:

0	1	2	3	4	5	6
20	23	45	47	50	60	70

i/p: 70, 60, 47, 23, 50, 45, 20

Delete and place in (n-1) Position of array.

Delete and place in (n-2) position.

Delete and place in (n-3) position

- - - -

Interactive Quiz - 1

Heap Sort Algorithm

```
HeapSort(arr)
BuildMaxHeap(arr)
for i = length(arr) to 2
   swap arr[1] with arr[i]
      heap_size[arr] = heap_size[arr] ? 1
      MaxHeapify(arr,1)
End
```

R.R.: $T(n) = T(n-1) + \log n + 1$

$\Rightarrow \boxed{T(n) = T(n-1) + \log n}$

```
BuildMaxHeap(arr)
   heap_size(arr) = length(arr)
   for i = length(arr)/2 to 1
MaxHeapify(arr,i)
End
```

T.C $O(n \log n)$

```
MaxHeapify(arr,i)
L = left(i)
R = right(i)
if L ? heap_size[arr] and arr[L] > arr[i]
largest = L
else
largest = i
if R ? heap_size[arr] and arr[R] > arr[largest]
largest = R
if largest != i
swap arr[i] with arr[largest]
MaxHeapify(arr,largest)
End
```

BYJU'S EXAM PREP

Comparison

Sorting Algorithms	Time Complexity			Space Complexity
	Best Case	Average Case	Worst Case	Worst Case
Bubble Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Selection Sort	$\Omega(N^2)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Insertion Sort	$\Omega(N)$	$\Theta(N^2)$	$O(N^2)$	$O(1)$
Quick sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N^2)$	$O(N)$
Merge Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(N)$
Heap Sort	$\Omega(N \log N)$	$\Theta(N \log N)$	$O(N \log N)$	$O(1)$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q1. The Time Complexity Of the Recurrence Relation T(n) = T(n-2)+n/logn is -

A. O(nlogn)

B. O(n^2 logn)

C. O(n^2 / logn)

D. O(logn)

$$T(n) = a \cdot T(n-b) + \frac{n}{\log n}$$

$$a = 1 \qquad O(n \cdot f(n)) = O\left(\frac{n^2}{\log n}\right)$$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q2. What is recurrence for worst case of Quick Sort and what is the time complexity in Worst case ?

A. Recurrence is $T(n) = T(n-1) + O(\log n)$ and time complexity is $O(n^2)$

B. Recurrence is $T(n) = T(n-1) + O(n)$ and time complexity is $O(n^2)$

C. Recurrence is $T(n) = 2T(n-1) + O(n)$ and time complexity is $O(n^2)$

D. Recurrence is $T(n) = T(n-1) + O(1)$ and time complexity is $O(n^2)$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q3. If one uses bubble sort algorithm to sort the following elements in ascending order

5, 1, 4, 2, 8

then the order of these elements after the first pass of the algorithm is :

A. 1, 4, 5, 2, 8

B. 1, 2, 4, 5, 8

C. 1, 4, 2, 5, 8

D. 1, 2, 5, 4, 8

Pass 1:

5, 1, 4, 2, 8

1, 5, 4, 2, 8

1, 4, 5, 2, 8

1, 4, 2, 5, 8

Interactive Quiz - 1

BYJU'S EXAM PREP

Q4. Consider the following functions from positives integers to real numbers -

$$10, \sqrt{n}, n, \log_2 n, 100/n$$

Let $n = 65536$

The CORRECT arrangement of the above functions in increasing order of asymptotic complexity is :

$10, \frac{100}{65536}, n = 65536$

A. $\log_2 n, 100/n, 10, \sqrt{n}, n$

B. $100/n, 10, \log_2 n, \sqrt{n}, n$

C. $10, 100/n, \sqrt{n}, \log_2 n, n$

D. $100/n, \log_2 n, 10, \sqrt{n}, n$

$\sqrt{65536} = 256$

$\log_2 2^{16} = 16$

$\frac{100}{n}, 10, \log_2 n, \sqrt{n}, n$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q5. Match the algorithms with their time complexities :

	Algorithm		Time Complexity
(P)	Towers of Hanoi with n disks	(i)	$O(n^2)$
(Q)	Binary search given n sorted numbers	(ii)	O(nlogn)
(R)	Heap sort given n numbers at the worst case	(iii)	$O(2^n)$
(S)	Addition of two n×n matrices	(iv)	O(logn)

A. P→(iii),Q→(iv),R→(i),S→(ii)

B. P→(iv),Q→(iii),R→(i),S→(ii)

C. P→(iii),Q→(iv),R→(ii),S→(i)

D. P→(iv),Q→(iii),R→(ii),S→(i)

Interactive Quiz - 1

BYJU'S EXAM PREP

Q6. Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X

I. (n^4)

II. $\theta(n^5)$

III. $O(n^5)$

IV. $\Omega(n^3)$

The equality above remains correct if X is replaced by

H/w

A. Only I

B. Only II

C. I or III or IV but not II

D. II or III or IV but not I

Interactive Quiz - 1

BYJU'S EXAM PREP

Q7. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is ?

4/w

A. $\Theta(n\log n)$

B. $\Theta(n)$

C. $\Theta(\log n)$

D. $\Theta(1)$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q8. Let $f(n) = n$ and $g(n) = n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct ?

I. $f(n) = O(g(n))$

II. $f(n) = \Omega(g(n))$

4/10

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

BYJU'S EXAM PREP

Q9. Which of the given options provides the increasing order of asymptotic complexity of functions f1, f2, f3 and f4 ?

$f1(n) = 2^n$

$f2(n) = n^{(3/2)}$

$f3(n) = n\log n$

$f4(n) = n^{\log n}$

4/10

A. f3, f2, f4, f1

B. f3, f2, f1, f4

C. f2, f3, f1, f4

D. f2, f3, f4, f1

Interactive Quiz - 1

BYJU'S EXAM PREP

Q10. Consider the following functions :

$f(n) = 2^n$

$g(n) = n!$

$h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of f(n), g(n), and h(n) is true ?

A. $f(n) = O(g(n))$; $g(n) = O(h(n))$

B. $f(n) = \Omega(g(n))$; $g(n) = O(h(n))$

C. $g(n) = O(f(n))$; $h(n) = O(f(n))$

D. $h(n) = O(f(n))$; $g(n) = \Omega(f(n))$

Interactive Quiz - 1

BYJU'S EXAM PREP

Q11. Suppose we want to arrange the 'n' numbers stored in an array such that all negative values occur before all positive ones. Minimum number of exchanges required in the worst case is :

A n-1

B n

C n+1

D n/2

A/w

Interactive Quiz - 1

BYJU'S EXAM PREP

Q12. Consider elements 17, 25, 5, 10, 32, 45, 50, 29 are sorted using selection sort. The order of elements after 5th pass is :

H/w

A. 5, 10, 17, 25, 29, 32, 45, 50

B. 5, 10, 17, 25, 32, 45, 29, 50

C. 5, 10, 17, 25, 29, 45, 32, 50

D. 5, 10, 17, 25, 29, 45, 50, 32

Interactive Quiz - 1

BYJU'S EXAM PREP

Q13. Consider the functions $f_1 = 10^n$, $f_2 = n^{\log n}$ and $f_3 = n^{\sqrt{n}}$.

Which of the following will be the ascending order of funcjtions ?

4/10

A. f_1, f_2, f_3

B. f_2, f_3, f_1

C. f_2, f_1, f_3

D. f_3, f_1, f_2

BYJU'S EXAM PREP

Q14. Let P be a Quick Sort Program to sort numbers in ascending order using the first element as pivot. Let t1 and t2 be the number of comparisons made by P for the inputs {1, 2, 3, 4, 5} and {4, 1, 5, 3, 2} respectively. Which one of the following holds ?

H/w

A. t1 = 5

B. t1 < t2

C. t1 > t2

D. t1 = t2

Interactive Quiz - 1

BYJU'S EXAM PREP

Q15. Quicksort is run on two inputs shown below to sort in ascending order taking first element as pivot,

(i) 1, 2, 3,....., n

(ii) n, n-1, n-2,....., 2, 1

Let C1 and C2 be the number of comparisons made for the inputs (i) and (ii) respectively. Then,

A. C1 < C2

B. C1 > C2

C. C1 = C2

D. (D) None

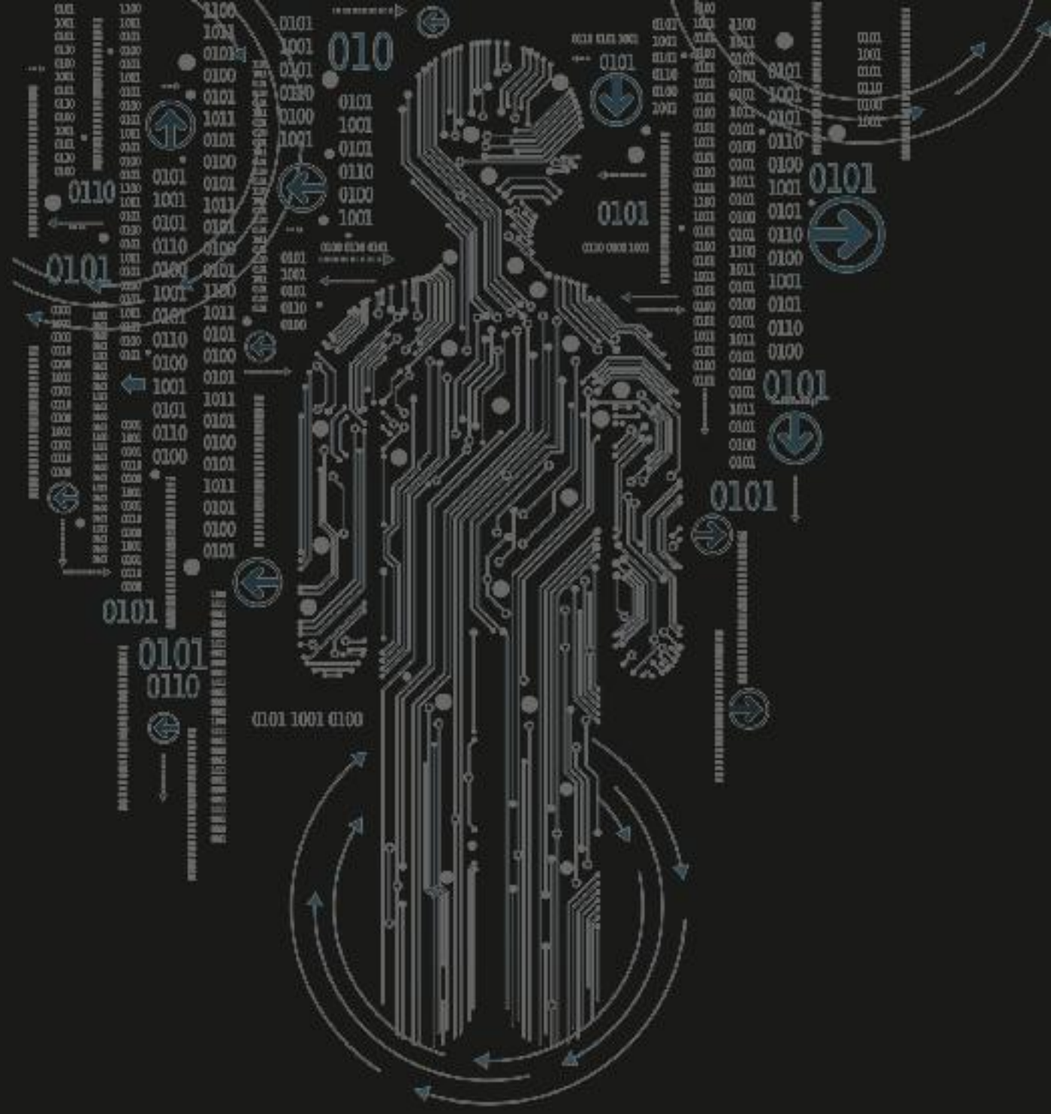
BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Job Sequencing with Deadlines,
Optimal Merge Pattern

GREEDY METHOD



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Greedy Method

Maximization

Minimization

This method is used for solving optimization problems. An optimization problem is a problem that demands either maximum or minimum results. The main function of this approach is that the decision is taken based on the currently available information. Whatever the current information is present, the decision is made without worrying about the effect of the current decision in future.

This technique is basically used to determine the feasible solution that may or may not be optimal. The feasible solution is a subset that satisfies the given criteria. The optimal solution is the solution which is the best and the most favorable solution in the subset. In the case of feasible, if more than one solution satisfies the given criteria then those solutions will be considered as the feasible, whereas the optimal solution is the best solution among all the solutions.

BYJU'S EXAM PREP

Applications Of Greedy Method

- It is used in finding the shortest paths. (minimization)
- It is used to find the minimum spanning tree using the prim's algorithm or the Kruskal's algorithm. (minimization)
- It is used in a job sequencing with a deadline. (Maximization)
- It is used in merging of two subsets or files into one. (Minimization)
- It is used in Huffman encoding. (Minimization)
- This algorithm is also used to solve the fractional knapsack problem. (Maximization)

etc---

BYJU'S EXAM PREP

Job Sequencing With Deadlines

In job sequencing problem, the objective is to find a sequence of jobs, which is completed within their deadlines and gives maximum profit.

Let us consider, a set of n given jobs which are associated with deadlines and profit is earned, if a job is completed by its deadline. These jobs need to be ordered in such a way that there is maximum profit.

It may happen that all of the given jobs may not be completed within their deadlines.

Assume, deadline of ith job J_i is d_i and the profit received from this job is p_i. Hence, the optimal solution of this algorithm is a feasible solution with maximum profit.

Thus, $D(i) > 0$ for $1 \leqslant i \leqslant n$.

BYJU'S EXAM PREP

Job Sequencing With Deadlines

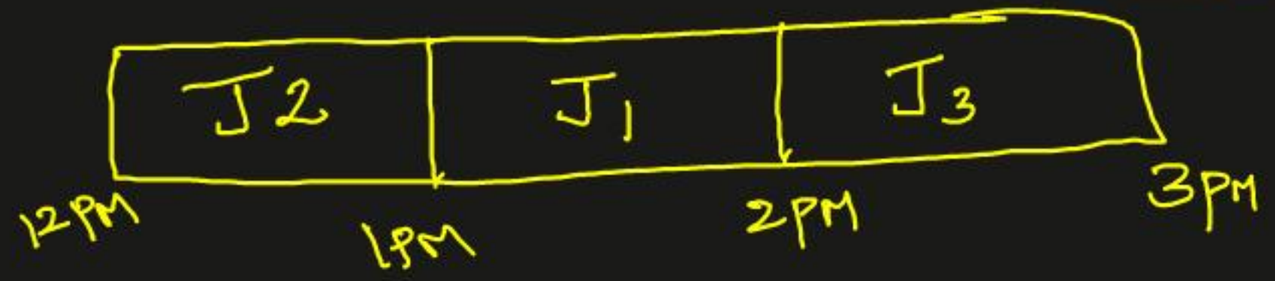
Initially, these jobs are ordered according to profit, i.e. p1⩾p2⩾p3⩾...⩾pn.

Algorithm: Job-Sequencing-With-Deadline (D, J, n, k)

```
D(0) := J(0) := 0
k := 1
J(1) := 1   // means first job is selected
for i = 2 ... n do
   r := k
   while D(J(r)) > D(i) and D(J(r)) ≠ r do
      r := r – 1
   if D(J(r)) ≤ D(i) and D(i) > r then
      for l = k ... r + 1 by -1 do
         J(l + 1) := J(l)
         J(r + 1) := i
         k := k + 1
```

BYJU'S EXAM PREP

Job Sequencing With Deadlines



Let us consider a set of given jobs as shown in the following table. We have to find a sequence of jobs, which will be completed within their deadlines and will give maximum profit. Each job is associated with a deadline and profit.

Unit time (Let 1 hour)

Job	J1	J2	J3	J4	J5
Deadline	2 PM	1 PM	3 PM	2 PM	1 PM
Profit	60	100	20	40	20

J2 ✓ J1 ✓ J4 ✗ J3 ✓ J5 ✗

BYJU'S EXAM PREP

Job Sequencing With Deadlines

To solve this problem, the given jobs are sorted according to their profit in a descending order. Hence, after sorting, the jobs are ordered as shown in the following table.

Job	J_2	J_1	J_4	J_3	J_5
Deadline	1	2	2	3	1
Profit	100	60	40	20	20

BYJU'S EXAM PREP

Job Sequencing With Deadlines

From this set of jobs, first we select J2, as it can be completed within its deadline and contributes maximum profit.

Next, J1 is selected as it gives more profit compared to J4.

In the next clock, J4 cannot be selected as its deadline is over, hence J3 is selected as it executes within its deadline.

The job J5 is discarded as it cannot be executed within its deadline.

Thus, the solution is the sequence of jobs (J2, J1, J3), which are being executed within their deadline and gives maximum profit.

Total profit of this sequence is 100 + 60 + 20 = 180.

BYJU'S EXAM PREP

Job Sequencing With Deadlines

Example - 2 :

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Profits :	10	50	70	70	50	30	50	10	70	30
Deadline :	4	3	5	2	4	5	2	4	1	3

$70 + 70 + 50 + 50 + 70 = 310$

Optimal solution:

J_9	J_4	J_2	J_5	J_3

0 1 2 3 4 5

J_3, J_4, J_9, J_2, J_5, J_7, J_6, J_{10}, J_1, J_8

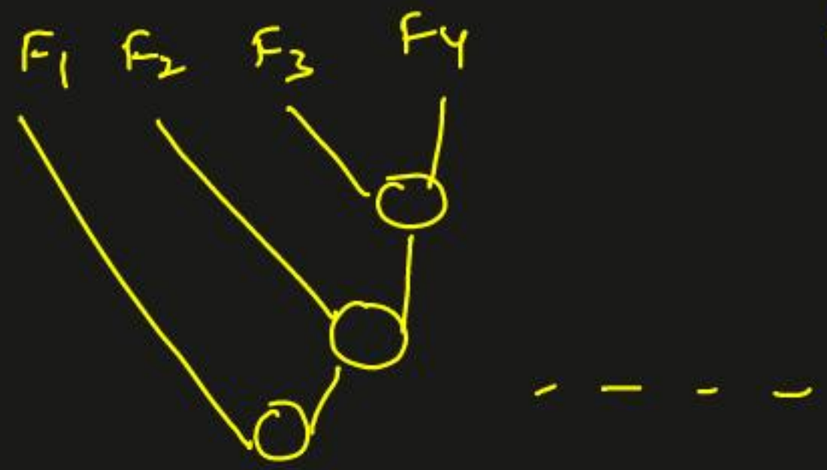
70 70 70 50 50 50 30 30 10 10

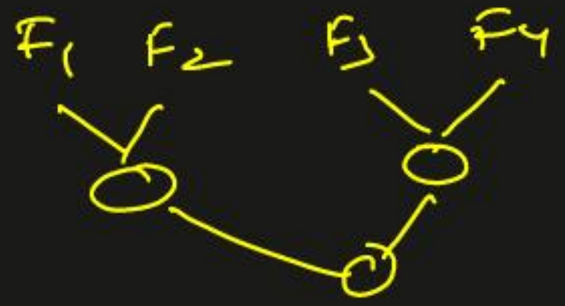
Maximum deadline

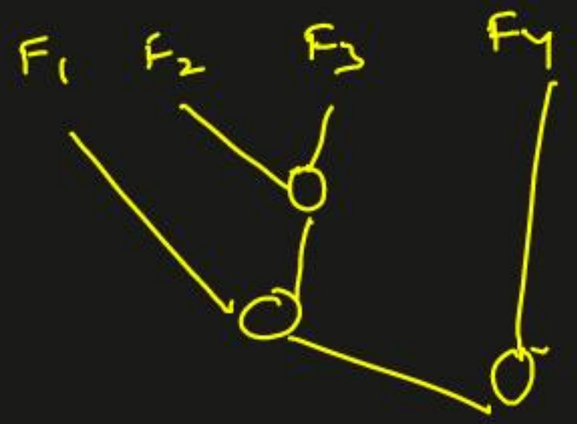
BYJU'S EXAM PREP

Optimal Merge Pattern

- Merge a set of sorted files of different length into a single sorted file. We need to find an optimal solution, where the resultant file will be generated in minimum time. minimum No-of comparisions.
- If the number of sorted files are given, there are many ways to merge them into a single sorted file. This merge can be performed pair wise. Hence, this type of merging is called as 2-way merge patterns.







BYJU'S EXAM PREP

f_1 f_2
m n elements $\Rightarrow$ $\boxed{m+n}$ comparisions.

Optimal Merge Pattern

Let us consider the given files, f1, f2, f3, f4 and f5 with 20, 30, 10, 5 and 30 number of elements respectively.

If merge operations are performed according to the provided sequence, then

M1 = merge f1 and f2 => 20 + 30 = 50

M2 = merge M1 and f3 => 50 + 10 = 60

M3 = merge M2 and f4 => 60 + 5 = 65

M4 = merge M3 and f5 => 65 + 30 = 95

Hence, the total number of operations is

50 + 60 + 65 + 95 = 270

F1: 15, 20, 25

F2: 8, 22

18, 22

8, 15

22, 15
22, 20
22, 25

4

15, 20, 25

18, 15
18, 20
22, 15
22, 20
22, 25

5

BYJU'S EXAM PREP

Optimal Merge Pattern

Sorting the numbers according to their size in an ascending order, we get the following sequence f4, f3, f1, f2, f5

Hence, merge operations can be performed on this sequence

M1 = merge f4 and f3 => 5 + 10 = 15

M2 = merge M1 and f1 => 15 + 20 = 35

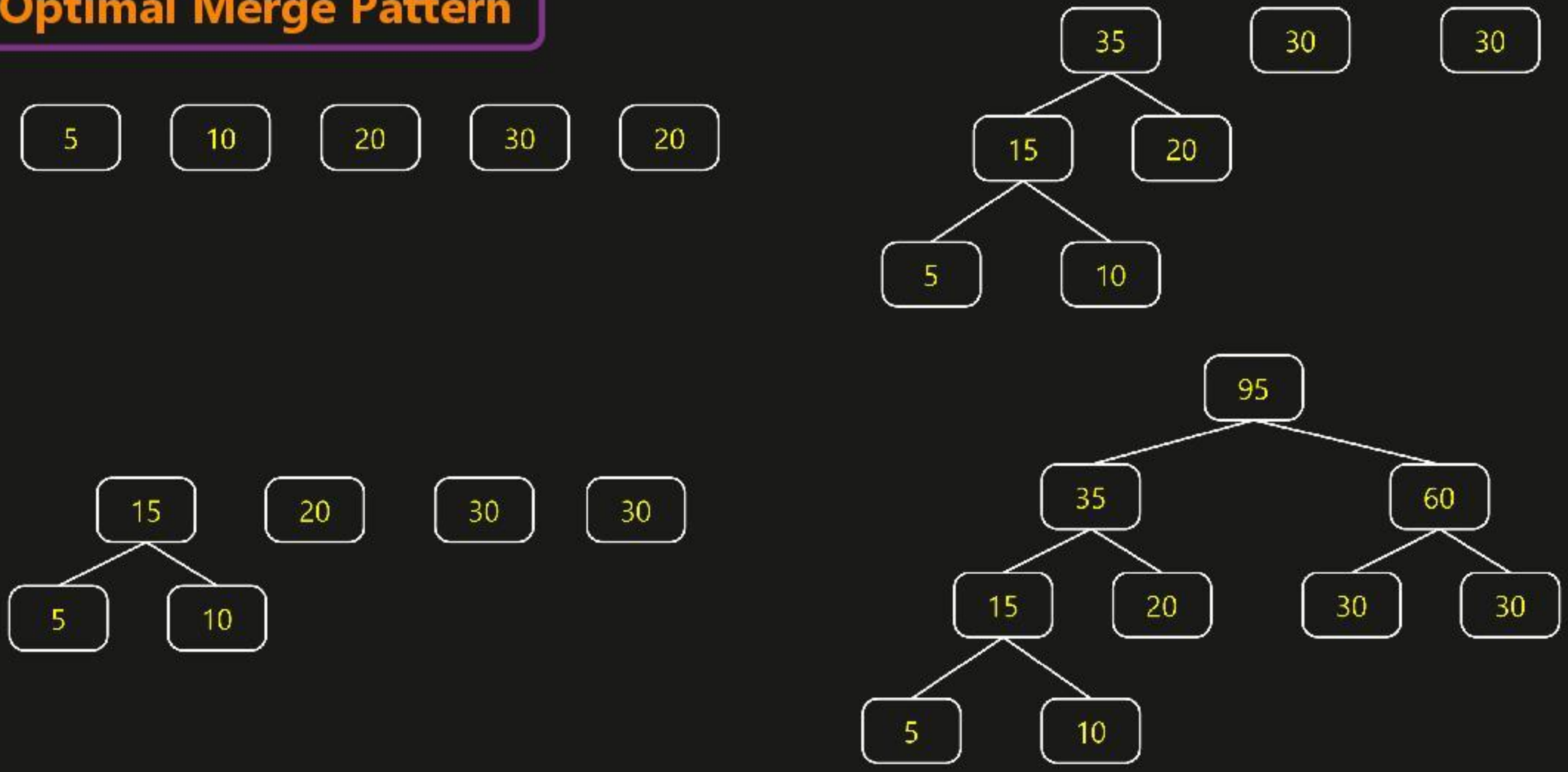
M3 = merge M2 and f2 => 35 + 30 = 65

M4 = merge M3 and f5 => 65 + 30 = 95

Therefore, the total number of operations is

15 + 35 + 65 + 95 = 210

Optimal Merge Pattern



Optimal Merge Pattern

To be contd ---

Q6. Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X

I. (n^4)

II. $\theta(n^5)$

III. $0(n^5)$

IV. $\Omega(n^3)$

The equality above remains correct if X is replaced by

A. Only I

B. Only II

C. I or III or IV but not II

D. II or III or IV but not I

Q7. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is ?

A. $\Theta(n\log n)$

B. $\Theta(n)$

C. $\Theta(\log n)$

D. $\Theta(1)$

Q8. Let $f(n) = n$ and $g(n) = n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct ?

I. $f(n) = O(g(n))$

II. $f(n) = \Omega(g(n))$

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

Q9. Which of the given options provides the increasing order of asymptotic complexity of functions f1, f2, f3 and f4 ?

$f1(n) = 2^n$

$f2(n) = n^{(3/2)}$

$f3(n) = n\log n$

$f4(n) = n^{\log n}$

A. f3, f2, f4, f1

B. f3, f2, f1, f4

C. f2, f3, f1, f4

D. f2, f3, f4, f1

Q10. Consider the following functions :

$f(n) = 2^n$

$g(n) = n!$

$h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of f(n), g(n), and h(n) is true ?

A. $f(n) = O(g(n)); g(n) = O(h(n))$

B. $f(n) = \Omega(g(n)); g(n) = O(h(n))$

C. $g(n) = O(f(n)); h(n) = O(f(n))$

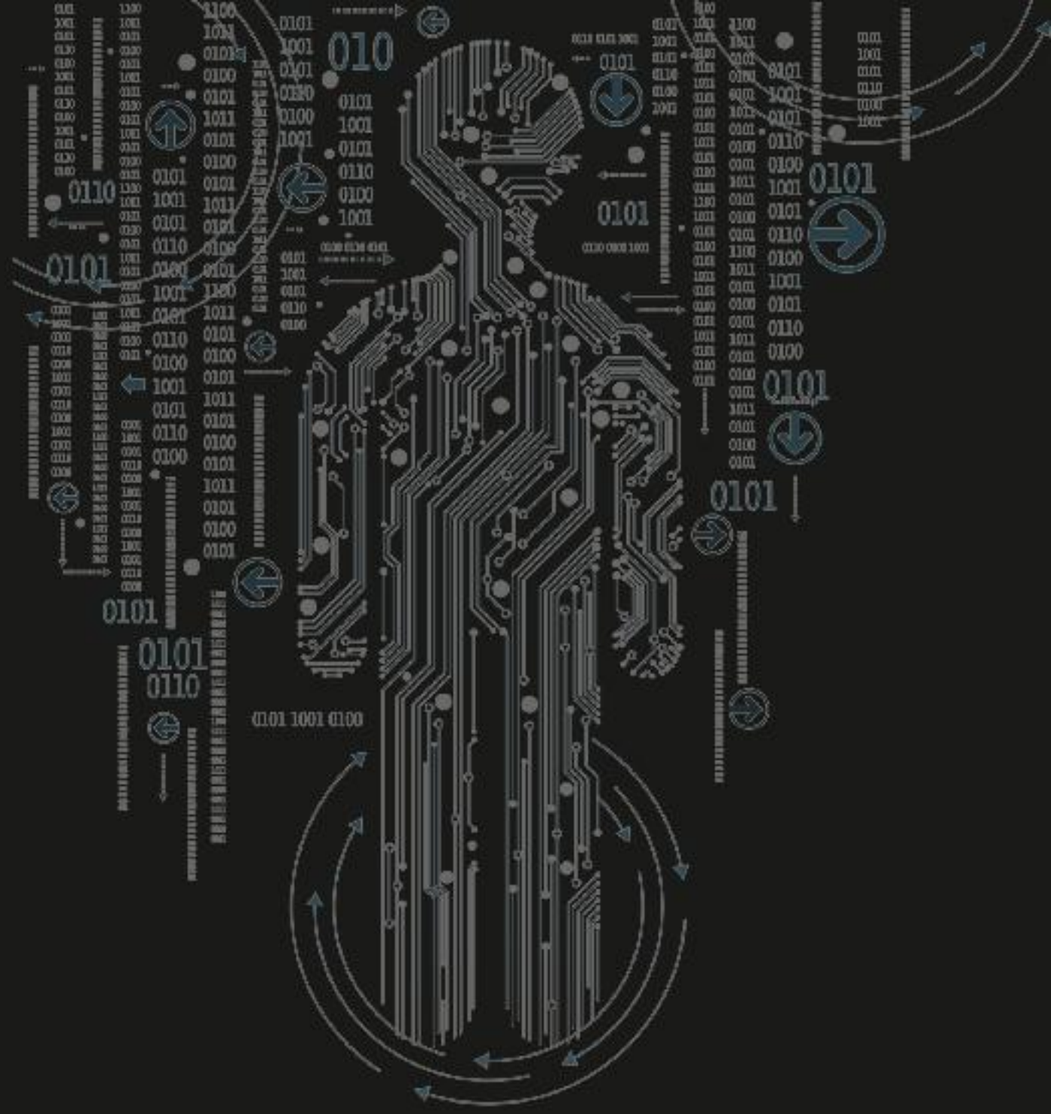
D. $h(n) = O(f(n)); g(n) = \Omega(f(n))$

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Huffman Coding



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

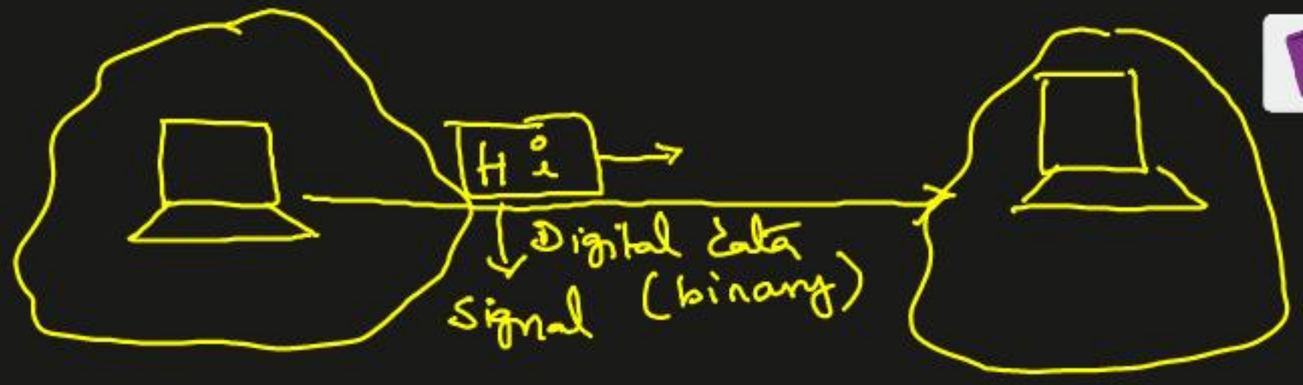
Last Class – Quick Revision

- Greedy Method ⇒ Optimization Problems
  - Maximization
  - Minimisation

- Job Sequencing with Deadlines (Maximization)

- Optimal Merge Pattern (Minimization)

Huffman Coding



Huffman Codes

[Minimization Problem]

(i) Data can be encoded efficiently using Huffman Codes.

(ii) It is a widely used and beneficial technique for compressing data.

200 characters
200 * 8 = 1600 bits

(iii) Huffman's greedy algorithm uses a table of the frequencies of occurrences of each character to build up an optimal way of representing each character as a binary string.

D → 69 → [0100 0101] 8-bit ASCII value

0110 0100

100 d, 97 → 01100001 a

20 times 30 times

Huffman Coding

Fixed Length Coding

(ASCII coding)

For every charaters
— 8 bits are used.

- Suppose we have 10^5 characters in a data file. Normal Storage:
- 8 bits per character (ASCII) – 8×10^5 bits in a file.
- But we want to compress the file and save it compactly.
- Suppose only six characters appear in the file:

Example :

	a	b	c	d	e	f	**Total**
Frequency	45	13	12	16	9	5	**100**

$100 \times 8 \Rightarrow 800$ bits

BYJU'S EXAM PREP

Fixed Length Coding

Each letter is represented by an equal number of bits. With a fixed length code, at least 3 bits per character:

For Example:

	3 bits
a	000
b	001
c	010
d	011
e	100
f	101

6 distinct charaters

100 charaters

$\Rightarrow 100 \times 3$

$\Rightarrow 300$ bits

ASCII coding

$\Rightarrow 800$ bits.

2 charaters $\begin{cases} 0 \\ 1 \end{cases}$ single bit

3 charaters $\rightarrow$ 2 bits $\begin{cases} 00 \\ 11 \\ 01 \end{cases}$

4 charaters $\rightarrow$ 2 bits

'n' charaters $\rightarrow \lceil \log_2 n \rceil$

For a file with 10^5 characters, we need 3×10^5 bits.

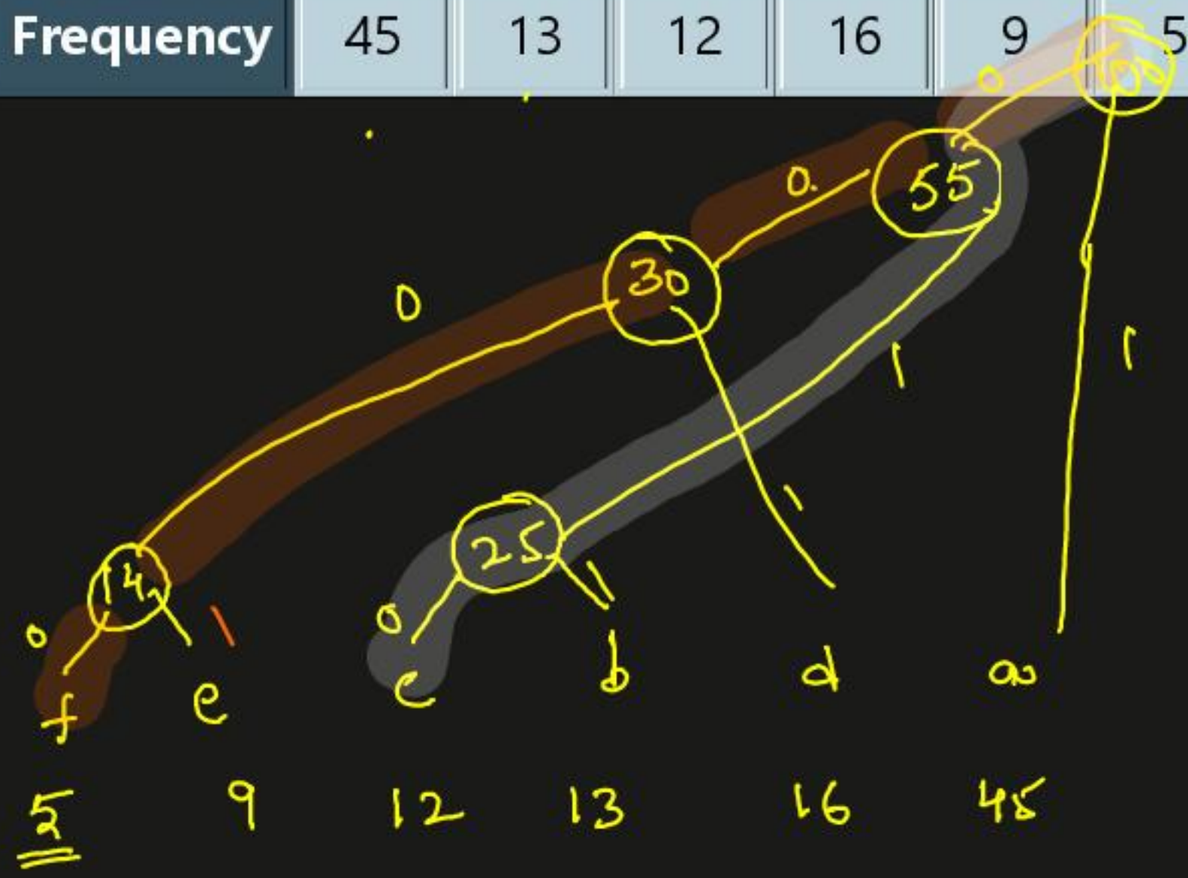
ASCII coding : 8×10^5 bits

Huffman Coding

"Variable-length" Coding - [optimal merge pattern]

Huffman Tree

	a	b	c	d	e	f	Total
Frequency	45	13	12	16	9	5	100



0000 – f (4 bits)

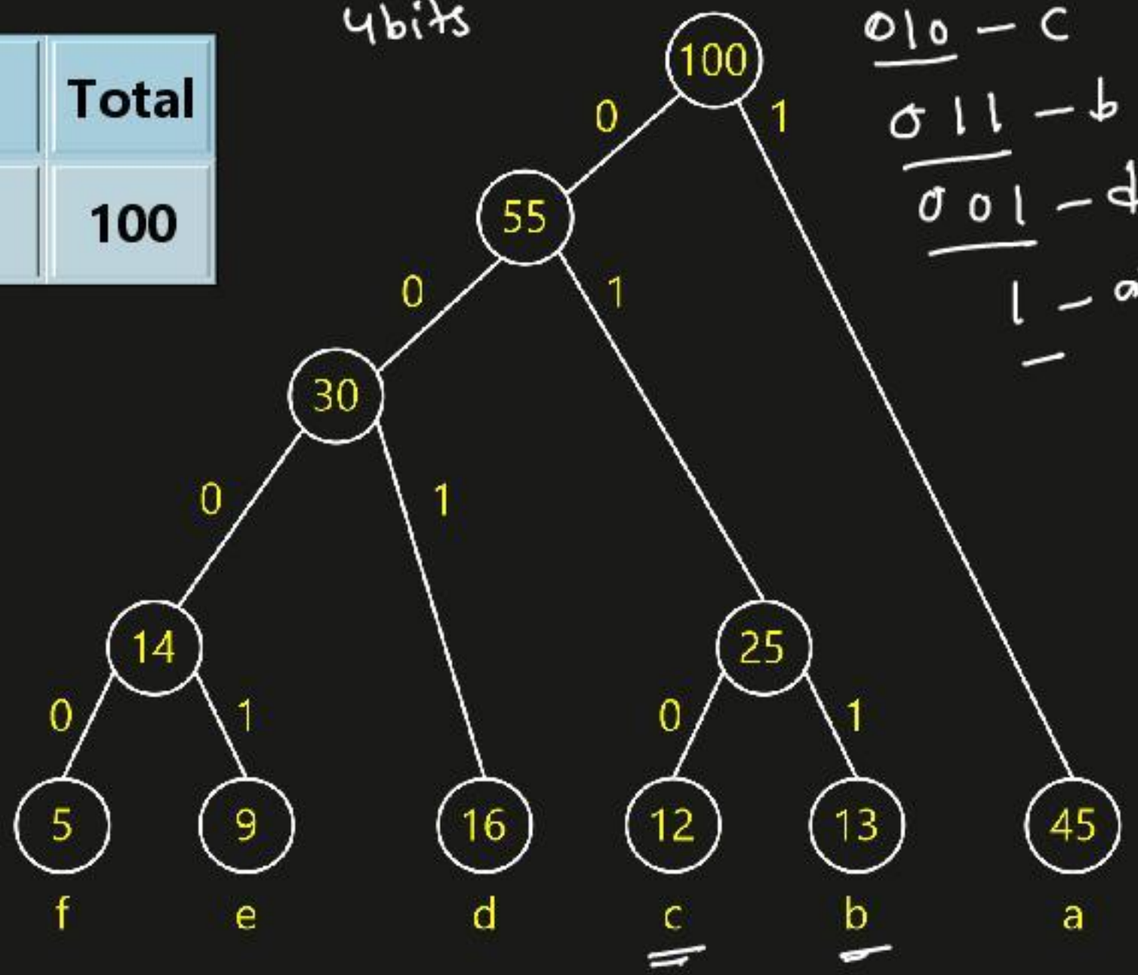
0001 – e (4 bits)

010 – c

011 – b

001 – d

1 – a



Huffman Tree

Huffman Coding

BYJU'S EXAM PREP

Σ (frequency * no. of bits)

Variable Length Coding

It can do considerably better than a fixed-length code, by giving many characters short code words and infrequent character long codewords.

For Example :

a	1
b	011
c	010
d	001
e	0001
f	0000

	a	b	c	d	e	f	Total
Frequency	45	13	12	16	9	5	**100**

1x 45 13x3

ASCII → 800 bits
Fixed length → 300 bits
Huffman coding → 224 bits

Number of bits = (45 x 1 + 13 x 3 + 12 x 3 + 16 x 3 + 9 x 4 + 5 x 4) / 1000 = 2.24 x 105bits

Thus, 224,000 bits represent the file, a saving of approximately 25%. This is an optimal character code for this file.

Huffman Coding

BYJU'S EXAM PREP

ASCII coding : 8bits × 11 = 88bits

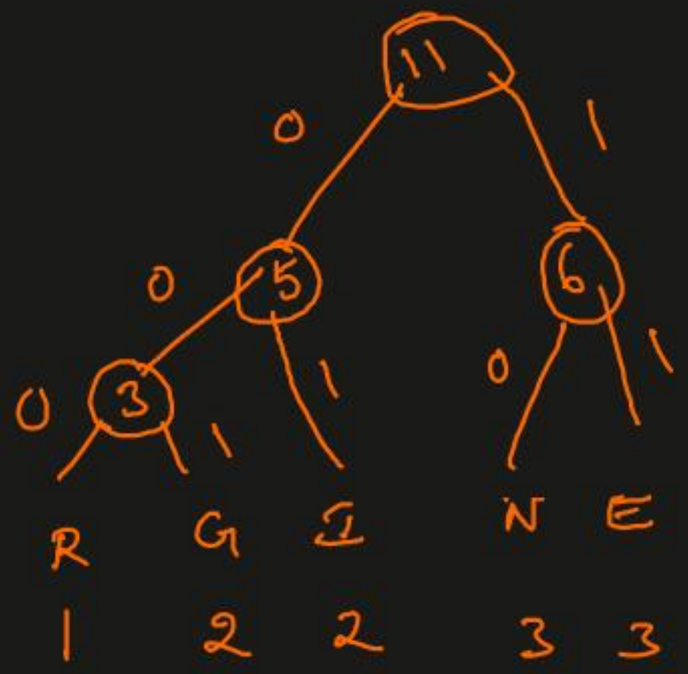
Fixed length coding : 3bits * 11 = 33 bits

Huffman coding : 25bits.

Example - 1

Calculate the number of bits for encoding the message 'ENGINEERING" using Huffman Coding technique ?

Character	frequency	Fixed length	Huffman bits.
E	3	000	11 (2bits)
N	3	111	10 (2bits)
G	2	011	001 (3bits)
I	2	101	01 (2bits)
R	1	110	000 (3bits)



Total No. of bits = $(3\times2)+(3\times2)+(2\times3)+(2\times2)+(1\times3) = 6+6+6+4+3$
$= 25$ bits.

Huffman Coding

Example - 2

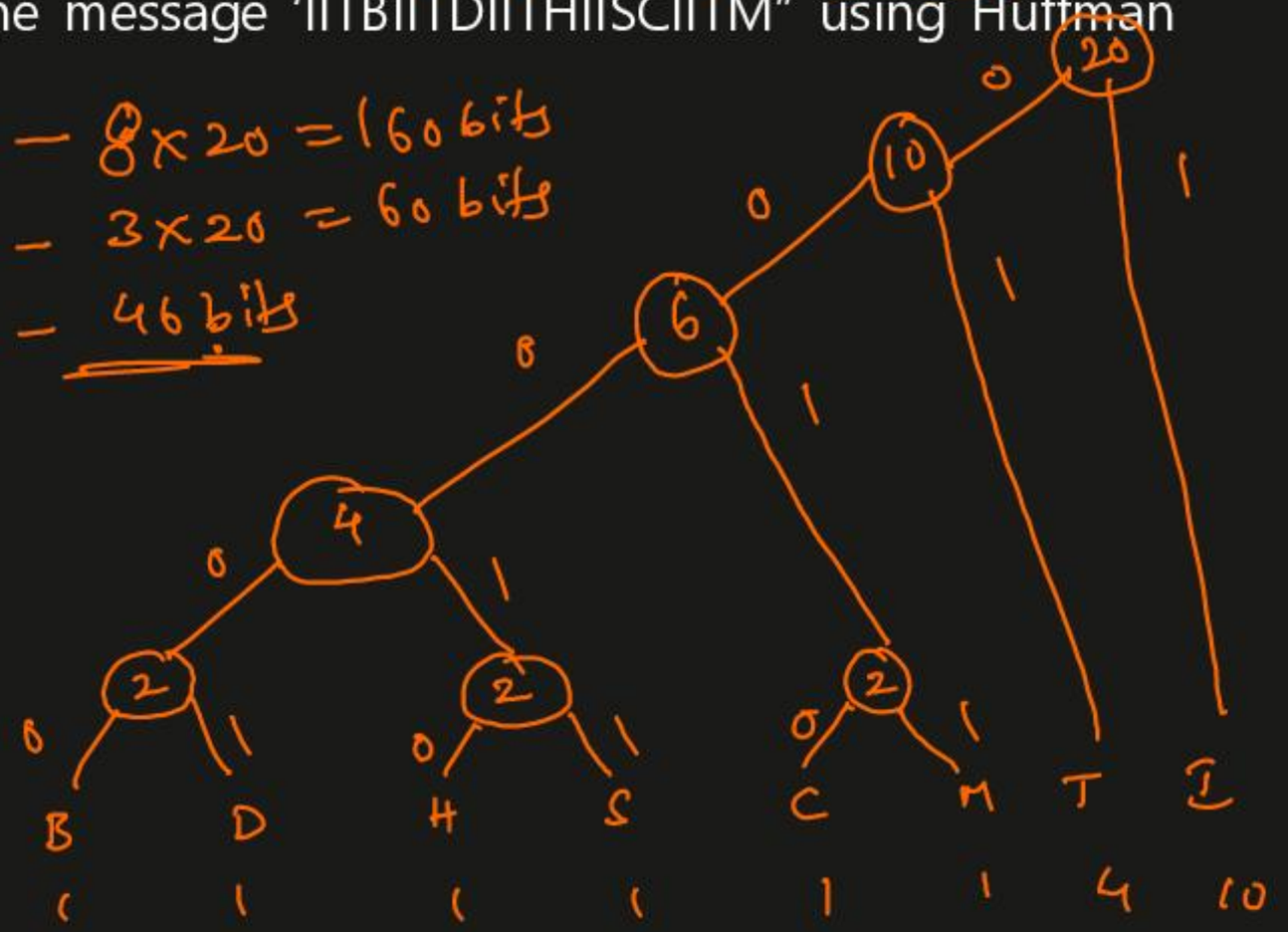
$(10\times1)+(4\times2)+(1\times5)+(1\times5)+(1\times5)+(1\times5)$
$+(1\times4)+(1\times4) = 10+8+5+5+5+5+4+4$
$= 46$

Calculate the number of bits for encoding the message 'IITBIITDIITHIISCIITM" using Huffman Coding technique ?

Character		frequency	
I	000	10	1
T	001	4	01
B	010	1	00000
D	011	1	00001
H	100	1	00010
S	101	1	00011
C	110	1	0010
M	111	1	0011

8 distinct characters

ASCII – $8\times20 = 160$ bits
Fixed – $3\times20 = 60$ bits
Huffman – 46 bits



Huffman Coding

Algorithm

Huffman (C)

```
n=|C|
Q ← C
for i=1 to n-1
do
z= allocate-Node ()
x= left[z]=Extract-Min(Q)
y= right[z] =Extract-Min(Q)
f [z]=f[x]+f[y]
Insert (Q, z)
return Extract-Min (Q)
```

BYJU'S EXAM PREP

Algorithm

Find an optimal Huffman Code for the following set of frequencies:

a: 50 b: 25 c: 15 d: 40 e: 75

Given that: C = {a, b, c, d, e}

f(C) = {50, 25, 15, 40, 75)}

n = 5

Q ← c

i.e.

c	15

b	25

d	40

a	50

e	75

BYJU'S EXAM PREP

Algorithm

Find an optimal Huffman Code for the following set of frequencies:

a: 50 b: 25 c: 15 d: 40 e: 75

for i ← 1 to 4

i = 1 Z ← Allocate node

x ← Extract-Min (Q)

y ← Extract-Min (Q)

40

c	15

b	25

d	40

a	50

e	75

BYJU'S EXAM PREP

Algorithm

Find an optimal Huffman Code for the following set of frequencies:

a: 50 b: 25 c: 15 d: 40 e: 75

Left [z] ← x

Right [z] ← y

f(z) ← f(x) + f(y) = 15 + 25

f(z) = 40

d	40

a	50

e	75

BYJU'S EXAM PREP

Algorithm

Find an optimal Huffman Code for the following set of frequencies:

a: 50 b: 25 c: 15 d: 40 e: 75

z ← Allocate node

x ← 40

y ← 40

d	40

a	50

e	75

left [z] ← x

right [z] ← y

f(z) = 40 + 40 = 80

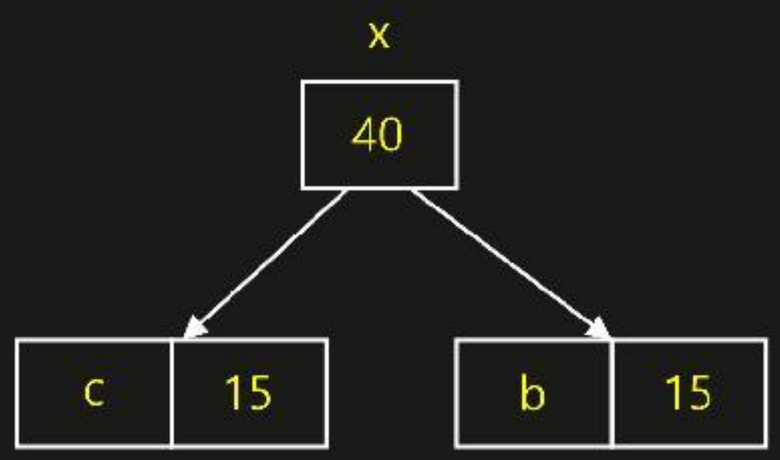
a	50

e	75

Algorithm

Find an optimal Huffman Code for the following set of frequencies:

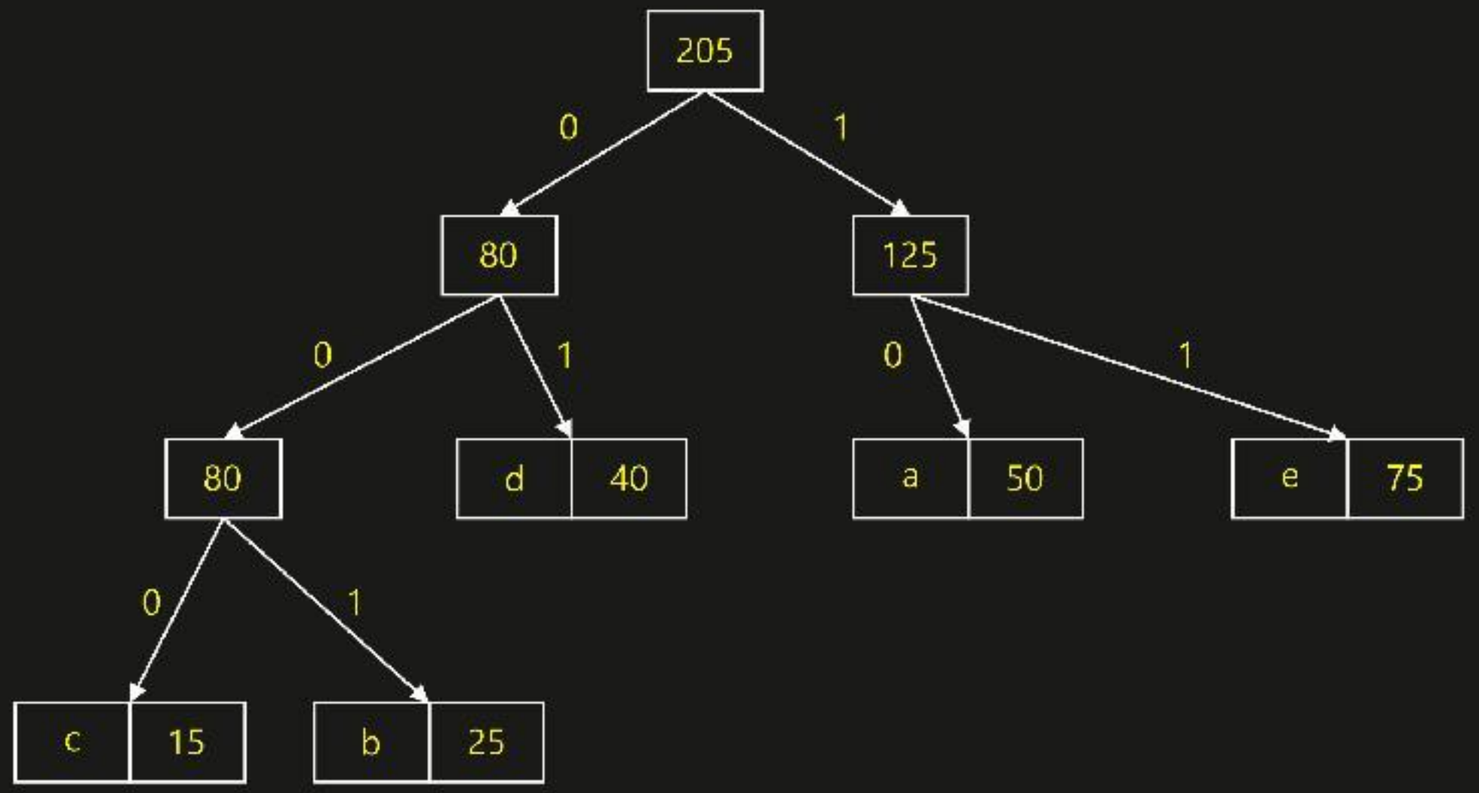
a: 50 b: 25 c: 15 d: 40 e: 75



Algorithm

Following same procedure, final output will be like:

Huffman Tree

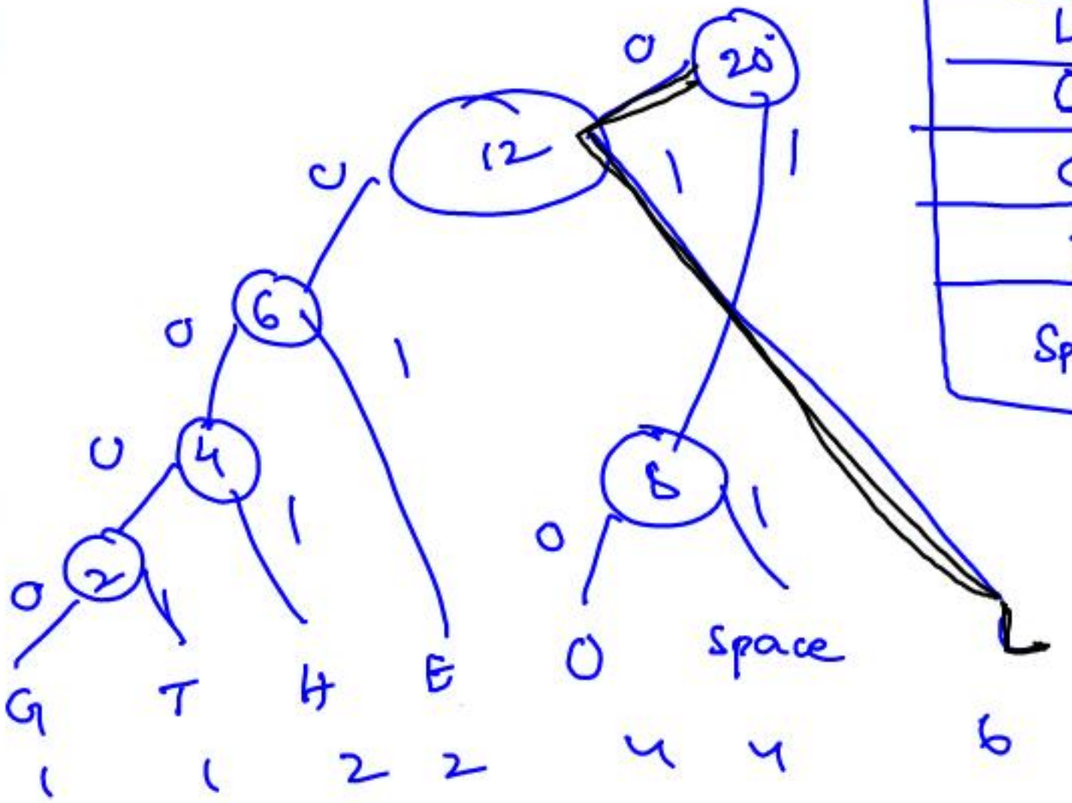


Algorithm

HELLO _ LOL _ GO _ TO _ HELL

Sender

0001001 0101101 01101



Character	frequency	bits
H	2	0001
E	2	001
L	6	01
O	4	10
G	1	00000
T	1	00001
Space	4	11

Receiver

Received message

ETLH_OLEGEL

0010000101 0001 11 10 01 001

E T L H space O L E

00000001 01

G E L

Q6. Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X

I. $O(n^4)$

II. $\Theta(n^5)$

III. $O(n^5)$

IV. $\Omega(n^3)$

The equality above remains correct if X is replaced by

$0^3 + 1^3 + 2^3 + 3^3 + \cdots + n^3 = \frac{n^2(n+1)^2}{4}$

$\Rightarrow = n^4 + \cdots$

$\underset{f(x)}{n^4} = O(\underset{g(x)}{n^4})$

$f(x) <= c \cdot g(x)$

$n^4 <= 2 \cdot n^4$

$n^4 <= c_1 \cdot n^5$

$n^4 >= c \cdot n^3$

$c_1 \cdot n^5 <= n^4 <= c_2 \cdot n^5$

False True

A. Only I

B. Only II

C. I or III or IV but not II

D. II or III or IV but not I

18 > 8
=
18 > 7

18 > 8 < 7 70 90 60 50 40

Q7. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is ?

A. Θ(nlogn)

B. Θ(n)

C. Θ(logn)

D. Θ(1)

Max = 50
min = 10
neither max, nor min = 20, 30, 40

3 comparisions | 3 comparisions

20 10 30 40 50

20 > 10 ⇒ 20 is not minimum

20 < 30 ⇒ 20 is not maximum

⇒ 20 is neither max nor min

Q8. Let $f(n) = n$ and $g(n) = n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct ?

I. $f(n) = O(g(n))$

II. $f(n) = \Omega(g(n))$

A. Only I

B. Only II

C. Both I and II

D. Neither I nor II

Q9. Which of the given options provides the increasing order of asymptotic complexity of functions f1, f2, f3 and f4 ?

$f1(n) = 2^n$

$f2(n) = n^{(3/2)}$

$f3(n) = n\log n$

$f4(n) = n^{\log n}$

A. f3, f2, f4, f1

B. f3, f2, f1, f4

C. f2, f3, f1, f4

D. f2, f3, f4, f1

Q10. Consider the following functions :

$f(n) = 2^n$

$g(n) = n!$

$h(n) = n^{\log n}$

Which of the following statements about the asymptotic behavior of f(n), g(n), and h(n) is true ?

A. $f(n) = O(g(n)); g(n) = O(h(n))$

B. $f(n) = \Omega(g(n)); g(n) = O(h(n))$

C. $g(n) = O(f(n)); h(n) = O(f(n))$

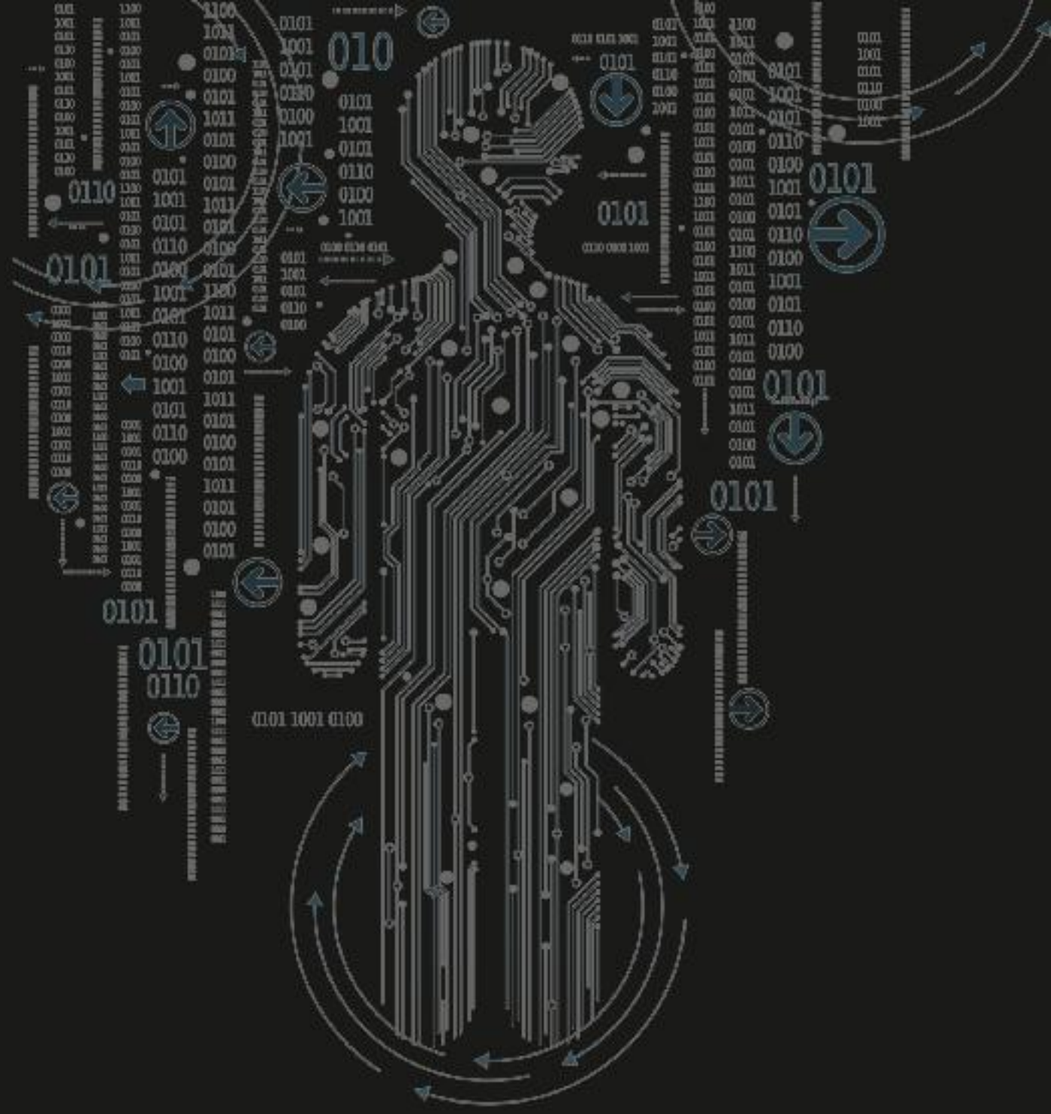
D. $h(n) = O(f(n)); g(n) = \Omega(f(n))$

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Minimum Cost Spanning Tree



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

ASCII Encoding : Every character is represented with it's Equivalent ASCII value → [8 bits] 'n' characters ⇒ (n*8) bits

Fixed length Encoding : Identify number of distinct characters ⇒ 'i'
Every character is represented with $\lceil \log_2 i \rceil$ bits.
'n' characters ⇒ $\lceil \log_2 i \rceil * n$ bits

Variable length Encoding [HUFFMAN Encoding] :
1. identify No. of distinct char with frequency
2. Arrange in increasing order of freq.
3. optimal Merge
4. left child – 0, right child – 1

Minimum Cost Spanning Tree

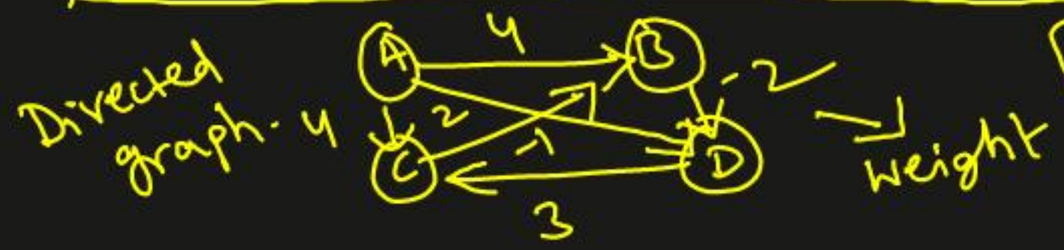
Every Tree is graph but Not vice-versa

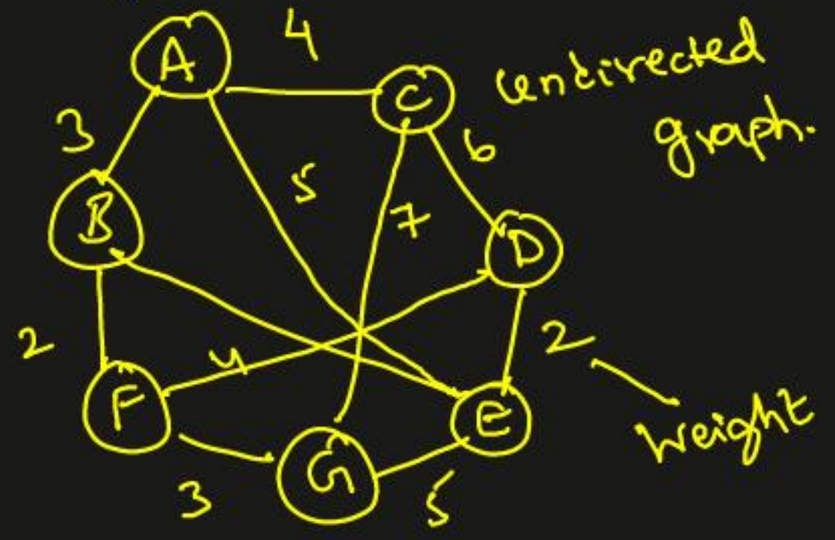
Graph : Non-linear Data Structure
Vertices | Nodes, links | Edges ⇒ May or May not have cycle.

Spanning Tree

- A spanning tree is a subset of an undirected Graph that has all the vertices connected by minimum number of edges. [n-1]
- If all the vertices are connected in a graph, then there exists at least one spanning tree. In a graph, there may exist more than one spanning tree.
- Properties -

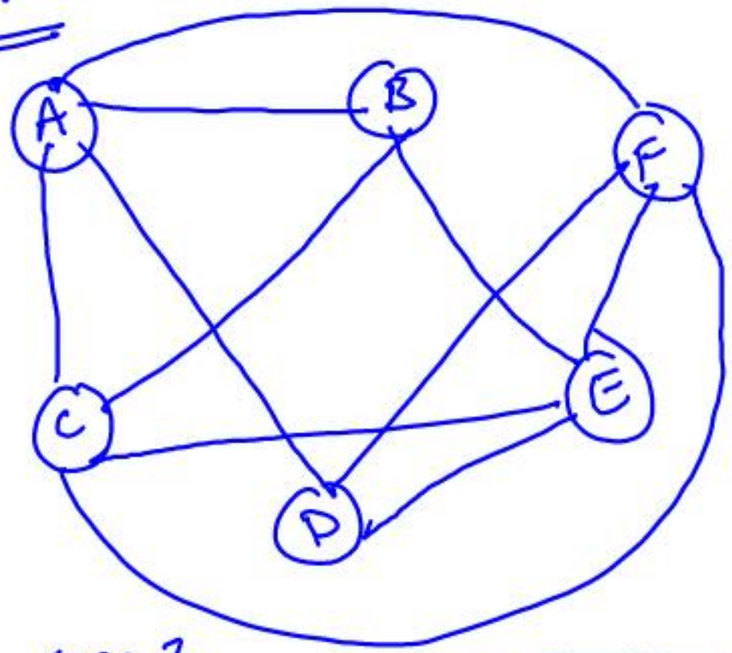
1. A spanning tree does not have any cycle.
2. Any vertex can be reached from any other vertex. [Connected]



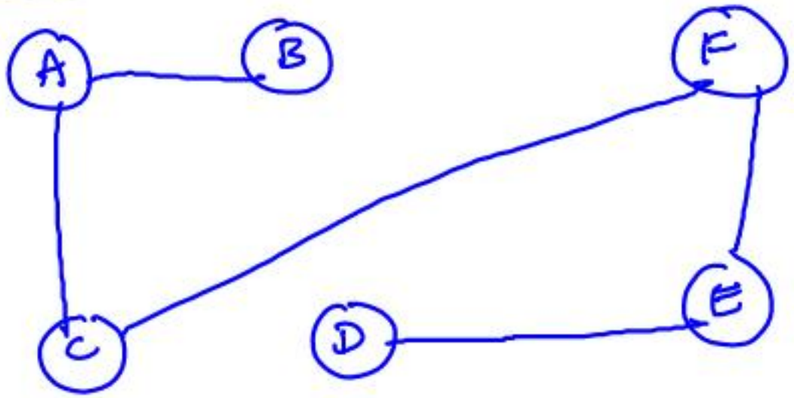


Ex:

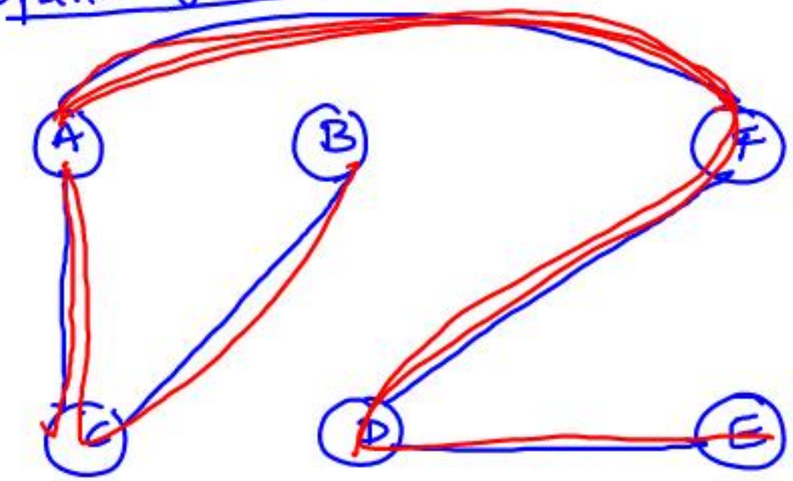
Graph, G



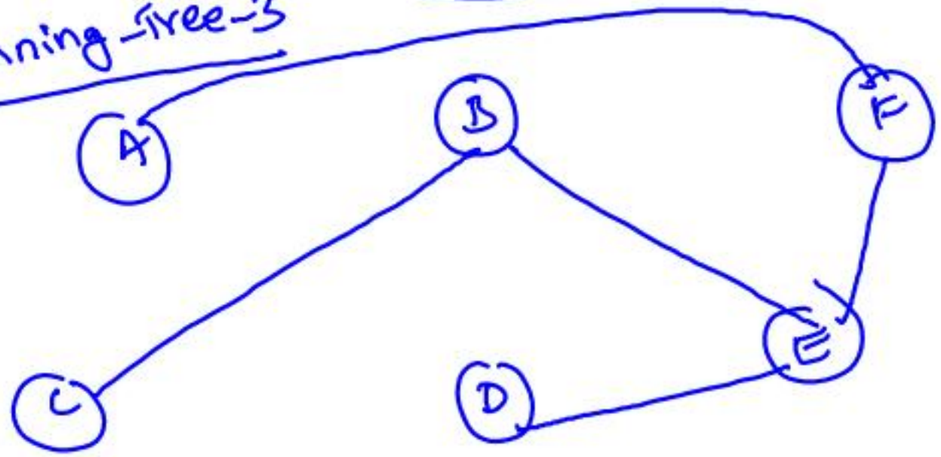
Spanning Tree -1



Spanning-Tree-2



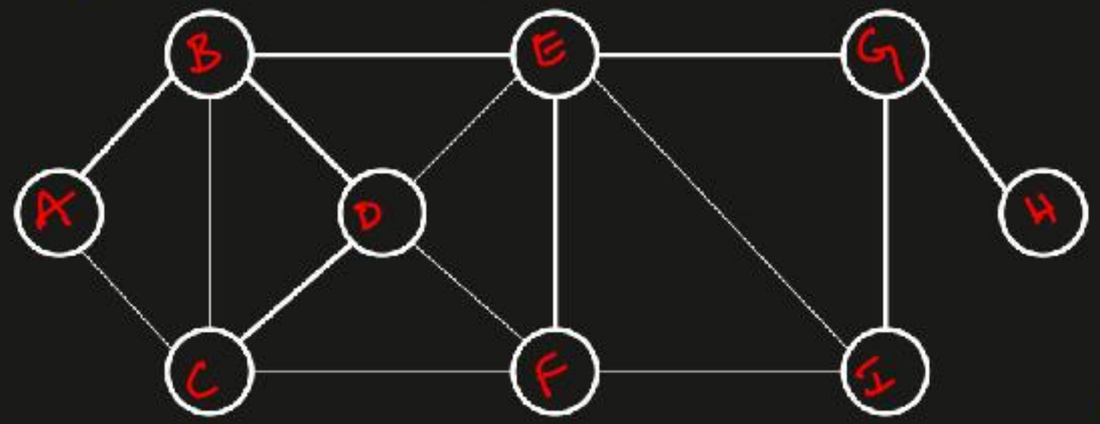
Spanning-Tree-3

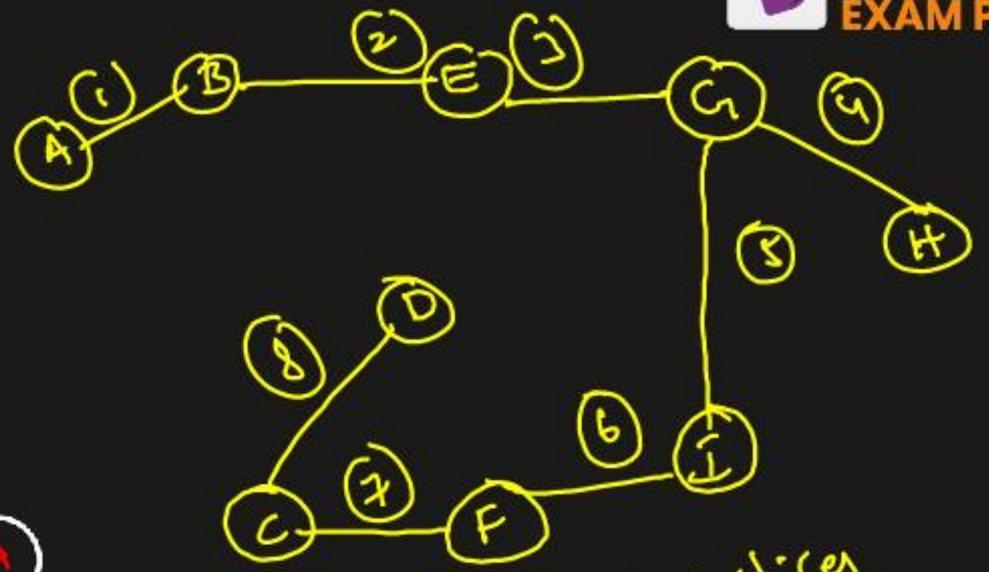


Minimum Cost Spanning Tree

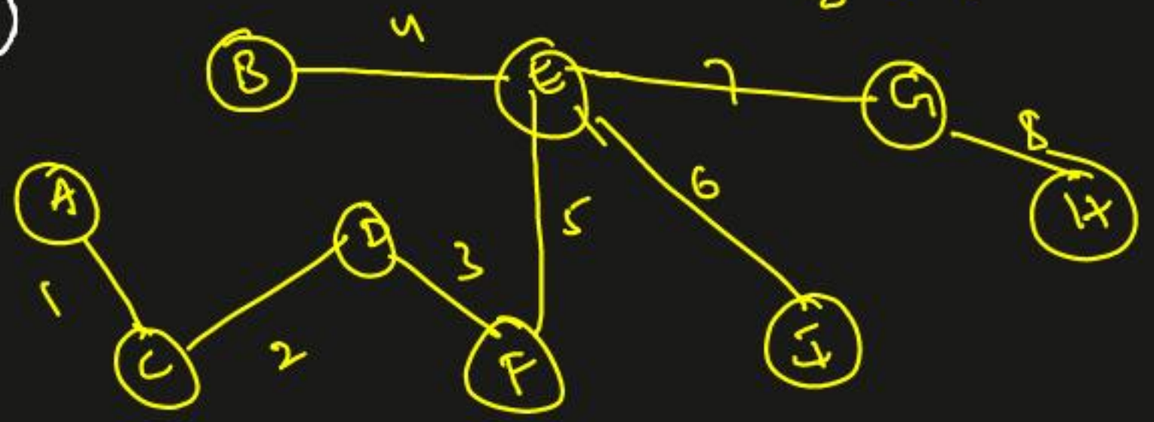
Example

'n' vertices in graph
with 'x' no. of edges.





9 vertices
8 edges.



Spanning Tree: n vertices
(n-1) edges

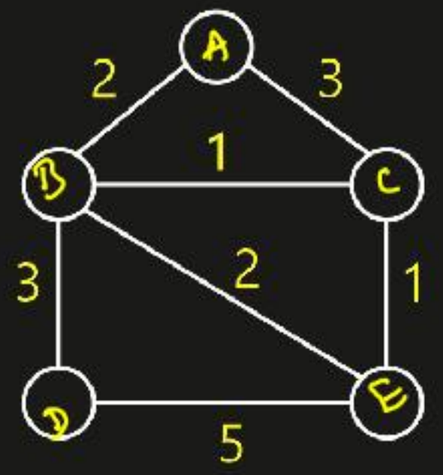
BYJU'S EXAM PREP

Minimum Spanning Tree

- A Minimum Spanning Tree (MST) is a subset of edges of a connected weighted undirected graph that connects all the vertices together with the minimum possible total edge weight.
- To derive an MST, Prim's algorithm or Kruskal's algorithm can be used.

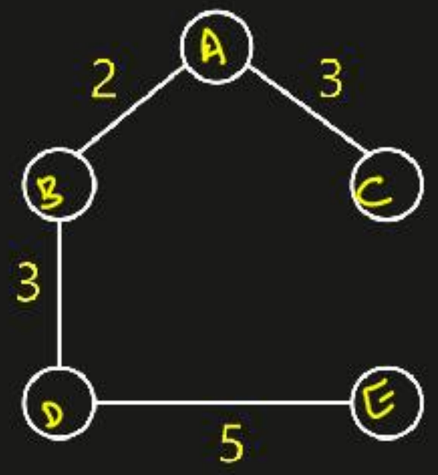
Minimum Cost Spanning Tree

Example - 1 Of MST



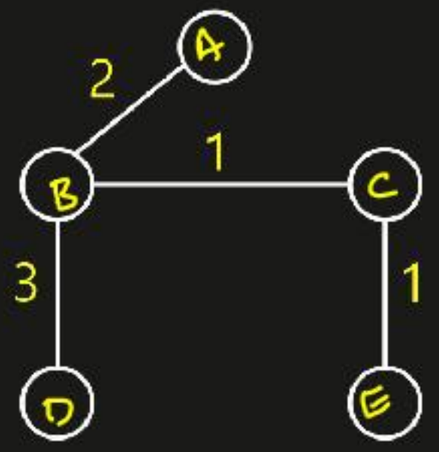
Graph

(1)



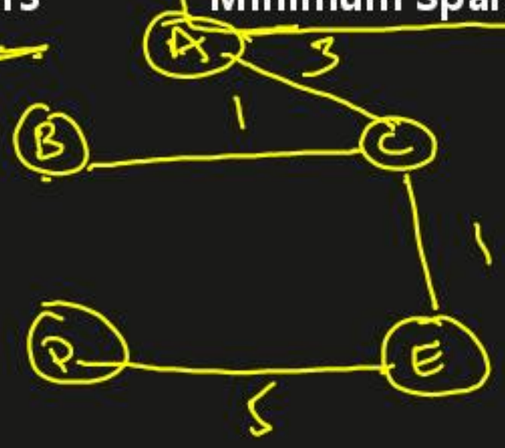
Spanning Tree, Cost = 13

(2)



Minimum Spanning Tree, Cost = 7

(3)

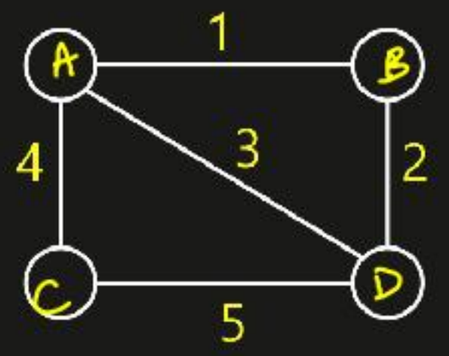


Cost = 3+1+1+5

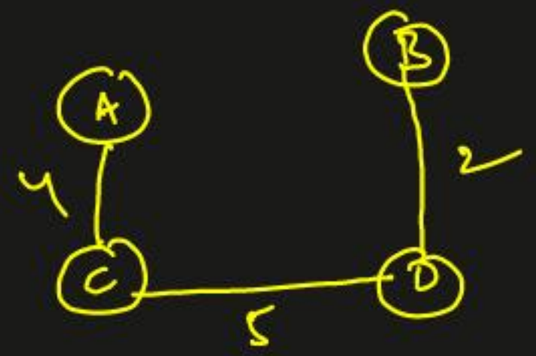
= 10

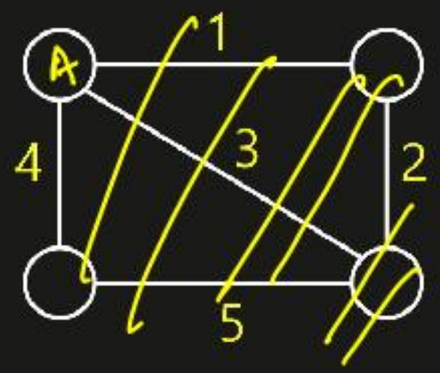
Minimum Cost Spanning Tree

Example - 2 Of MST



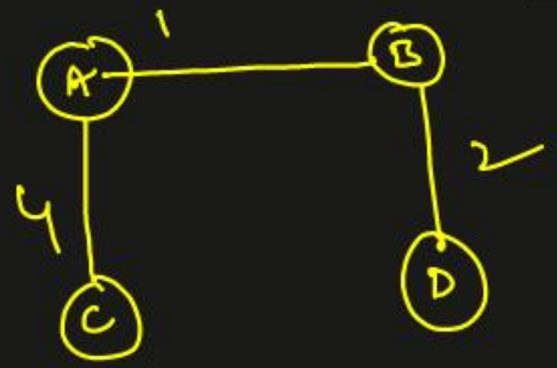
Undirected Graph

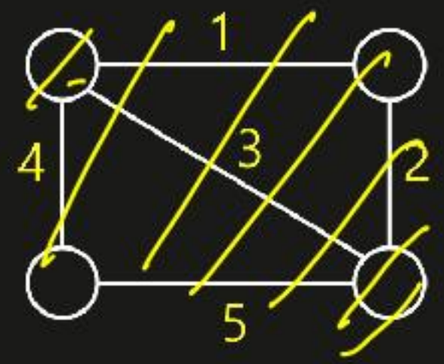




Spanning Tree

Cost = 11(= 4 + 5 + 2)





Minimum Spanning Tree

Cost = 7 (=4 + 1 + 2)

For a Complete graph : [Every vertex directly connected to every other vertex]

'n' vertices ⇒ Total Spanning Trees Possible = n^{n-2}

Minimum Cost Spanning Tree

Let Complete graph with 8 vertices, ${}^{8}C_{2}$ edges $= 28$ edges.

$\hookrightarrow 8^{8-2} = 8^{6} = 262144$ Spanning Trees

Prim's Algorithm

- Prim's algorithm is a greedy approach to find the minimum spanning tree. In this algorithm, to form a MST we can start from an arbitrary vertex.
- The algorithm may informally be described as performing the following steps:

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree. * Connected Edges
3. Repeat step 2 (until all vertices are in the tree).

BYJU'S EXAM PREP

Pseudocode – Prim's Algorithm

Cost[v]

- Associate with each vertex v of the graph a number C[v] (the cheapest cost of a connection to v) and an edge E[v] (the edge providing that cheapest connection).
- To initialize these values, set all values of C[v] to +∞ (or to any number larger than the maximum edge weight) and set each E[v] to a special flag value indicating that there is no edge connecting v to earlier vertices.
- Initialize an empty forest F and a set Q of vertices that have not yet been included in F (initially, all vertices).

  D.M
- Repeat the following steps until Q is empty:

BYJU'S EXAM PREP

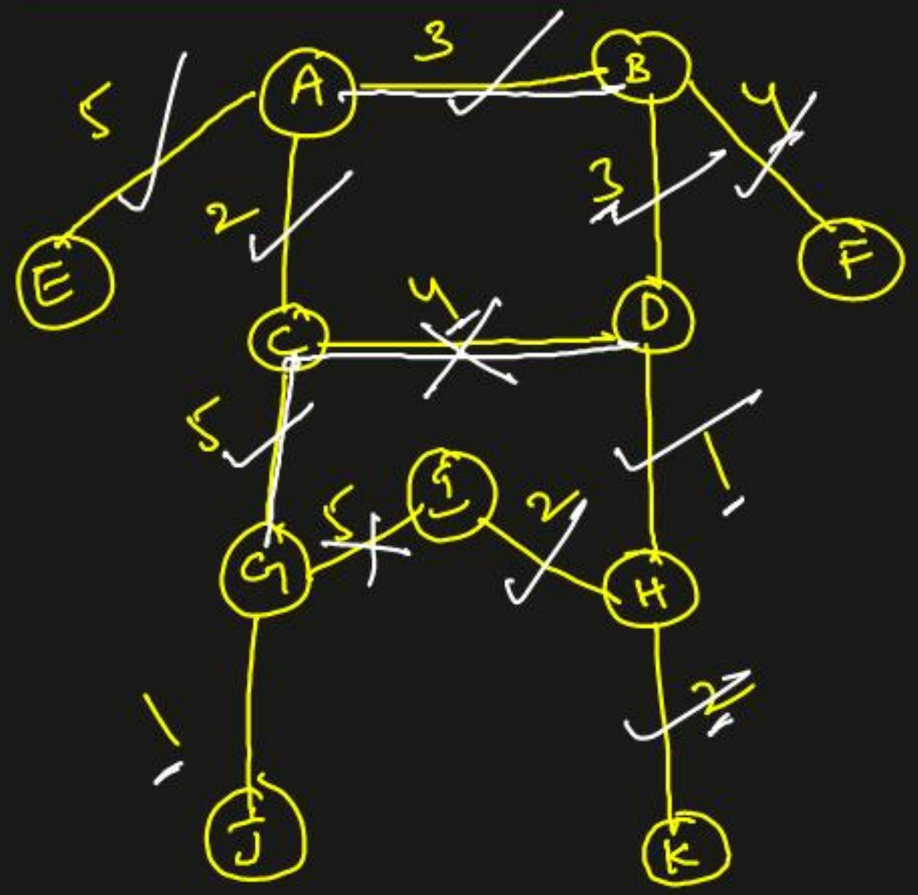
Pseudocode – Prim's Algorithm

- Find and remove a vertex v from Q having the minimum possible value of C[v]
- Add v to F and, if E[v] is not the special flag value, also add E[v] to F
- Loop over the edges vw connecting v to other vertices w. For each such edge, if w still belongs to Q and vw has smaller weight than C[w], perform the following steps:
- Set C[w] to the cost of edge vw
- Set E[w] to point to edge vw.
- Return F

Minimum Cost Spanning Tree

BYJU'S EXAM PREP

Practice Problem



MST.

There may be multiple MSTs Possible. but All MST cost will be always Same.

Minimum Cost

$5+2+3+5+3+4$

$+1+1+2+2$

$= 28$

BYJU'S EXAM PREP

Kruskal's Algorithm

Prim's algo : only connected edges only can be picked up.

- It also follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum.
- The steps to implement Kruskal's algorithm are listed as follows -

1. First, sort all the edges from low weight to high.
2. Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle, then reject the edge.
3. Continue to add the edges until we reach all vertices, and a minimum spanning tree is created.

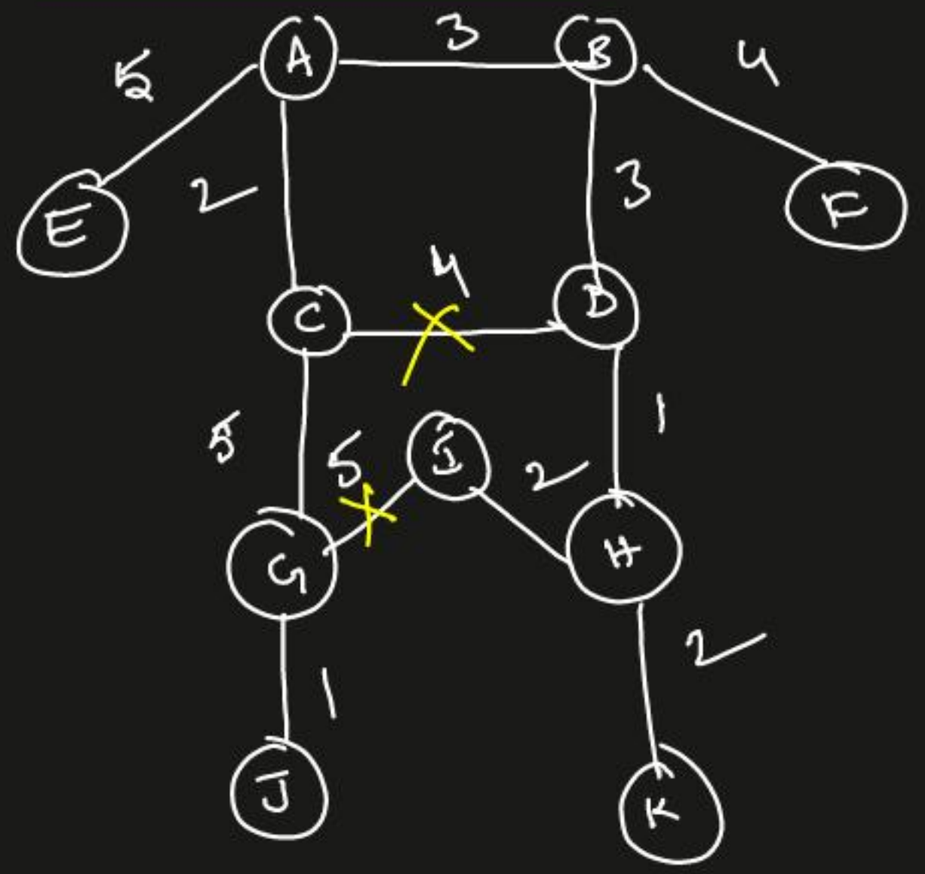
BYJU'S EXAM PREP

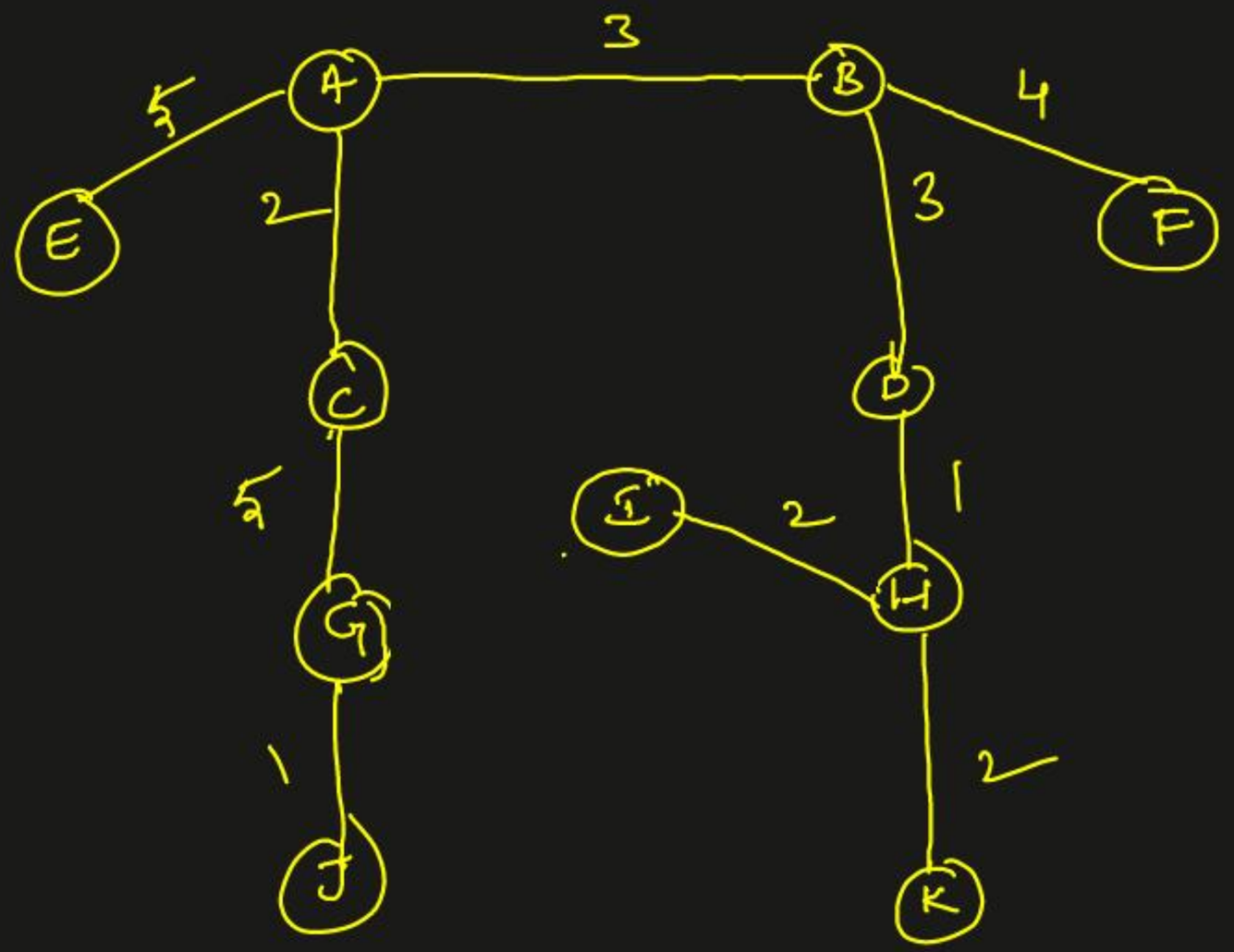
Pseudocode – Kruskal's Algorithm

```
Algorithm Kruskal(G) is
    F:= Ø
    for each v ∈ G.V do
        MAKE-SET(v)
    for each (u, v) in G.E ordered by weight(u, v), increasing do
        if FIND-SET(u) ≠ FIND-SET(v) then
            F:= F ∪ {(u, v)} ∪ {(v, u)}
            UNION(FIND-SET(u), FIND-SET(v))
    return F
```

Minimum Cost Spanning Tree

Practice Problem





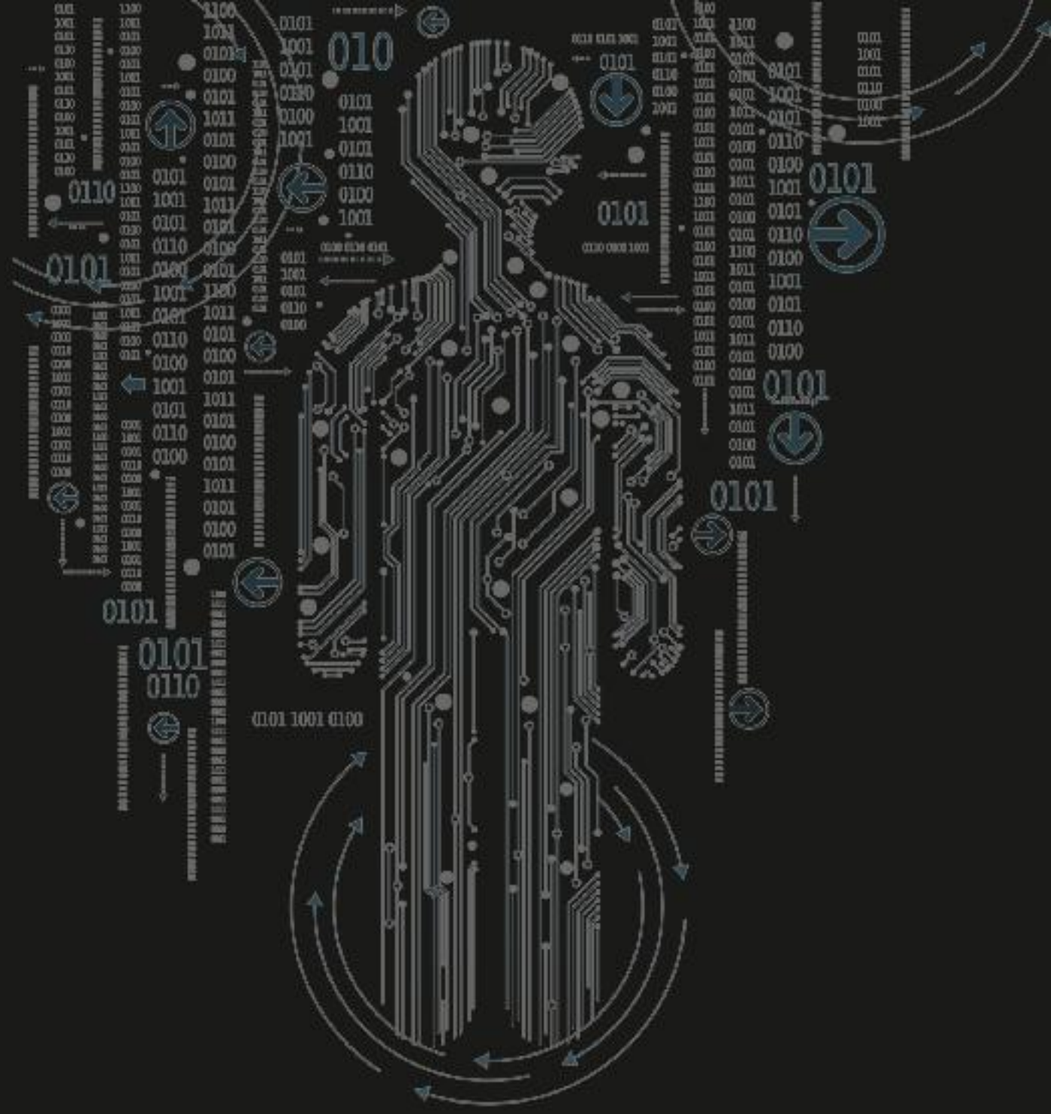
Minimum cost = 28

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Dijkstra's Algorithm



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

For a complete graph with 'n' vertices
$\therefore n^{n-2}$ spanning trees

Minimum Spanning Tree (MST)

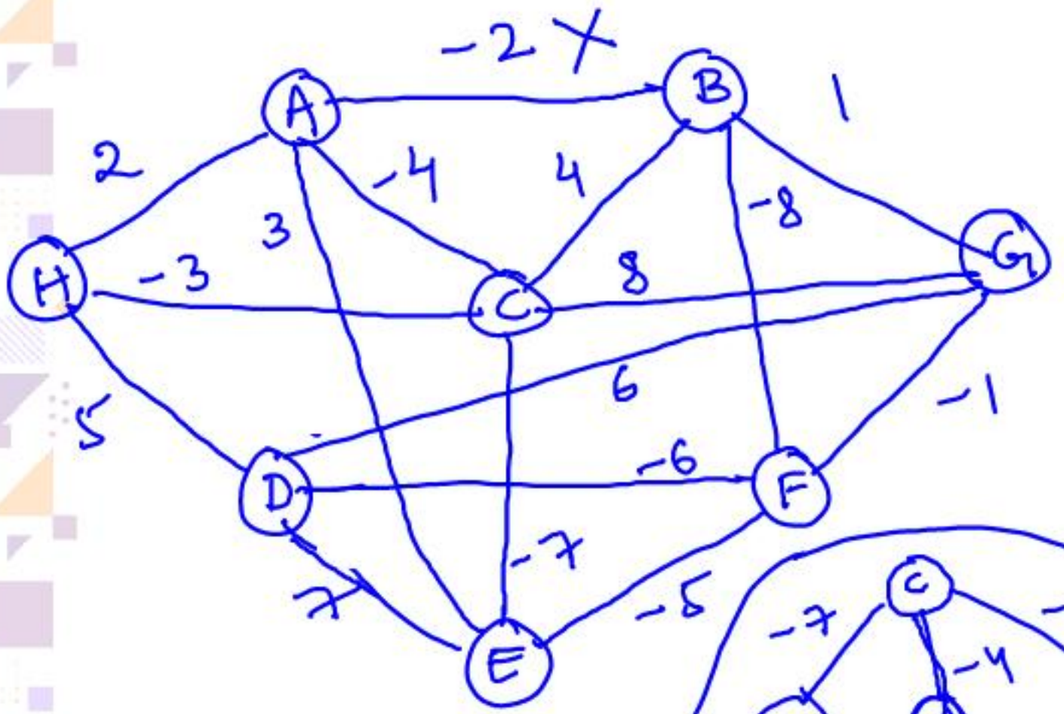
- Spanning Tree : Subset of graph 1) without cycle
  2) $(n-1)$ Edges with 'n' vertices

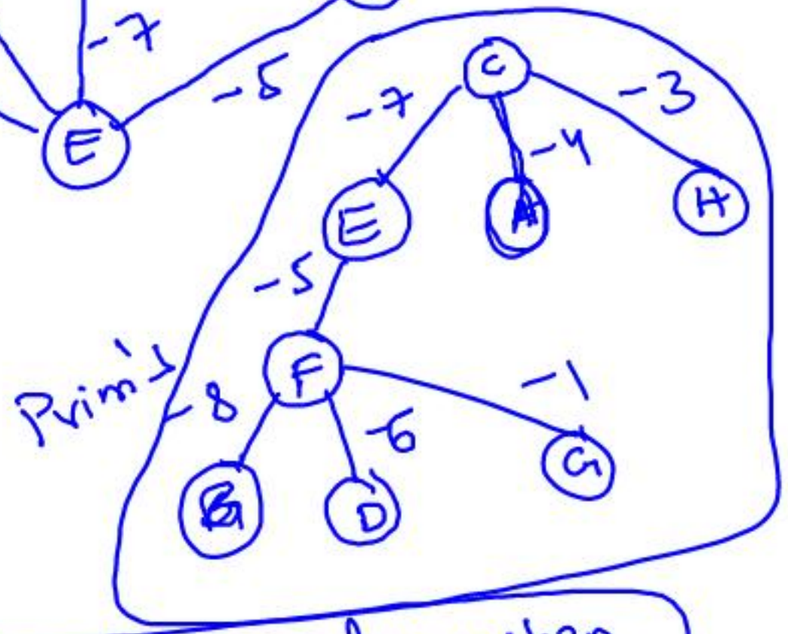
- For a given graph, Multiple Spanning Trees Possible/feasible.

MST: The spanning Tree with minimum Cost

Greedy algorithms
- Prim's algo : Connected Edges to selected vertices
- Kruskal's algo : Pick an edge with minimum cost

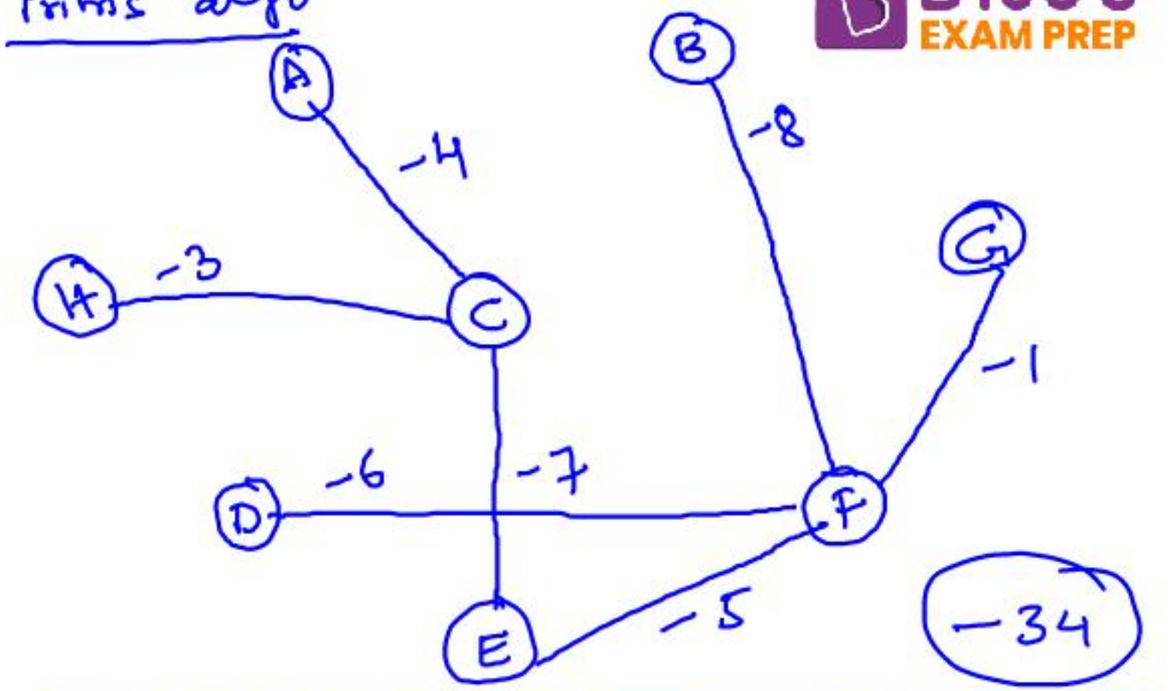
Same min. cost



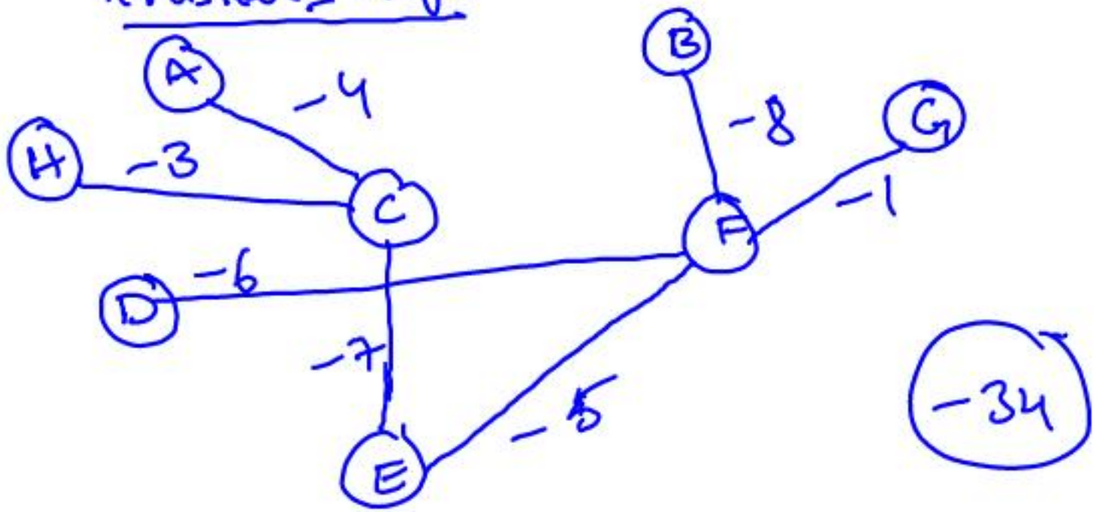


Unique MST with both algo when all edges weight is distinct.

Prim's algo



Kruskal's algo



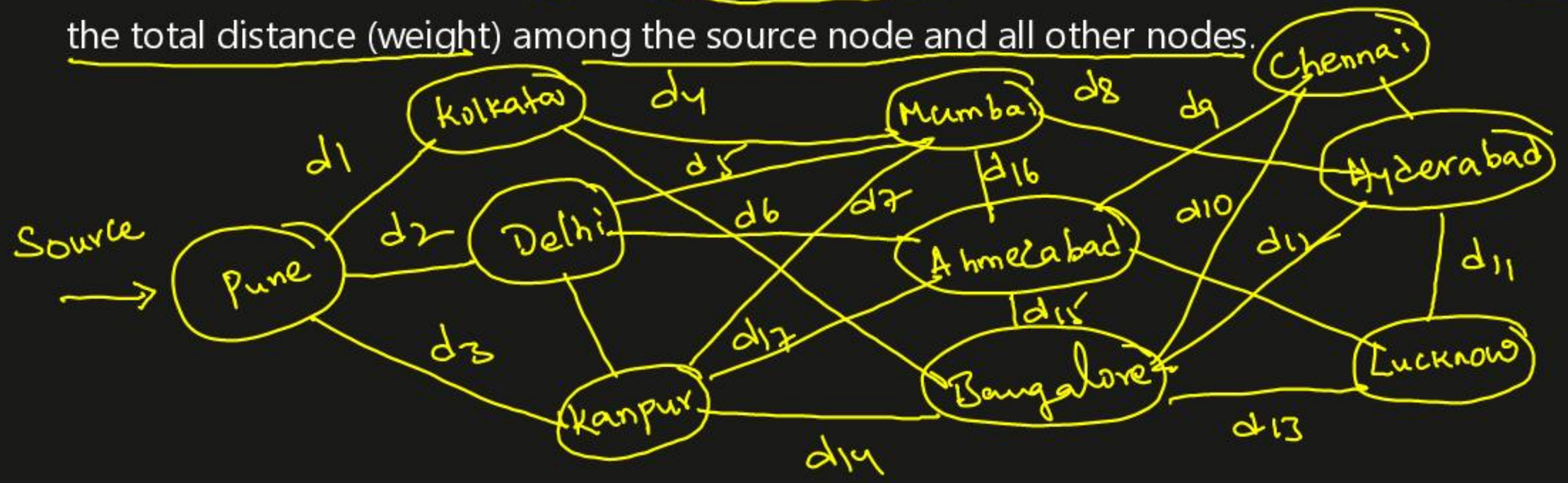
→ Greedy based Problem

BYJU'S EXAM PREP

Dijkstra's Algorithm

[minimization Problem] minimal feasible solution

- Dijkstra algorithm is a single-source shortest path algorithm. Here, single-source means that we need to find the shortest path from the given source to all the nodes.
- Dijkstra's algorithm makes use of weights of the edges for finding the path that minimizes the total distance (weight) among the source node and all other nodes.



BYJU'S EXAM PREP

Dijkstra's Algorithm

- Dijkstra's algorithm is the iterative algorithmic process to provide us with the shortest path from one specific starting node to all other nodes of a graph. It is different from the minimum spanning tree as the shortest distance among two vertices might not involve all the vertices of the graph.
- Generally, Dijkstra's algorithm works on the principle of relaxation where an approximation of the accurate distance is steadily displaced by more suitable values until the shortest distance is achieved.

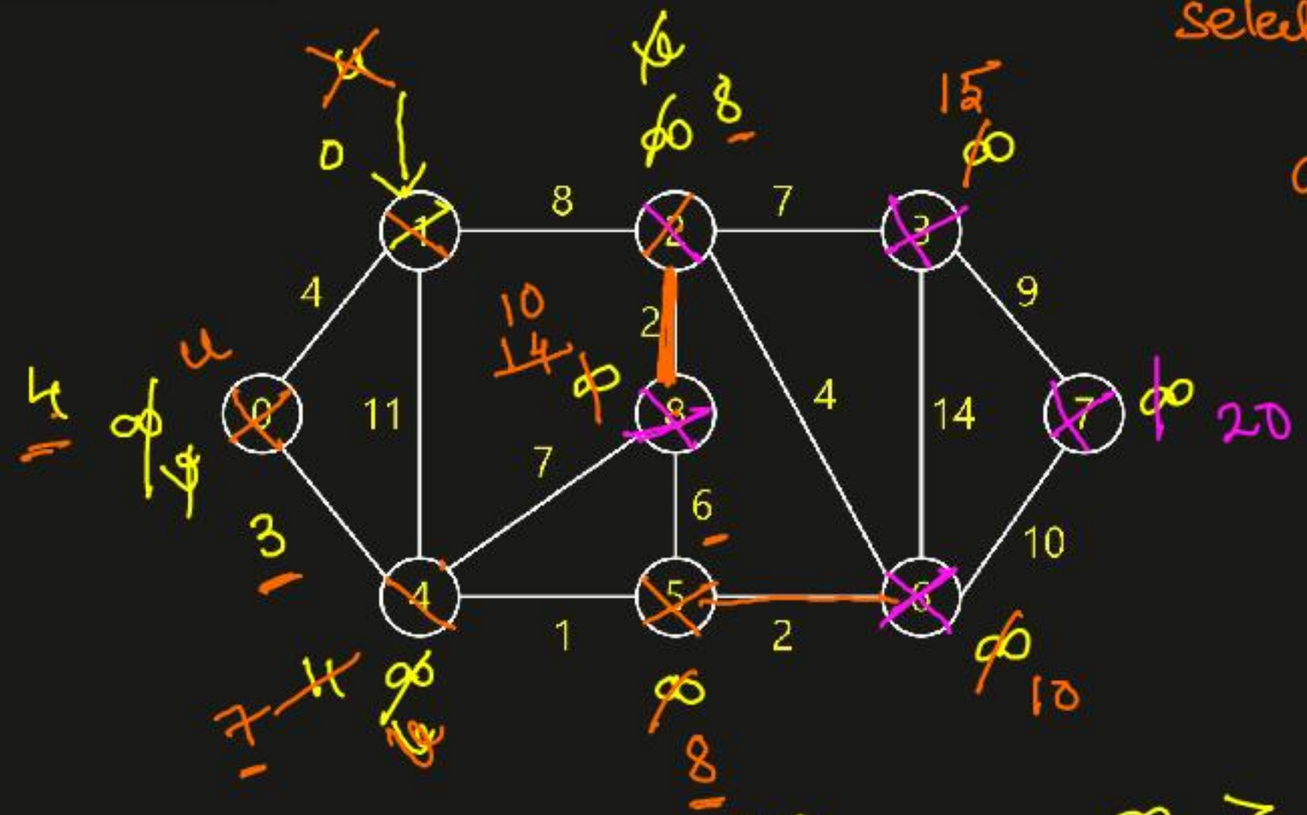
Dijkstra's Algorithm

BYJU'S EXAM PREP

Example

Node 1 to

Node	Cost
0	4
1	0
2	8
3	15
4	7
5	8
6	10
7	20
8	10



Select Node with min. Cost

$Cost(v) > 4+3$

$11 > 7$

$\infty > 0+4 \Rightarrow \infty > 4$

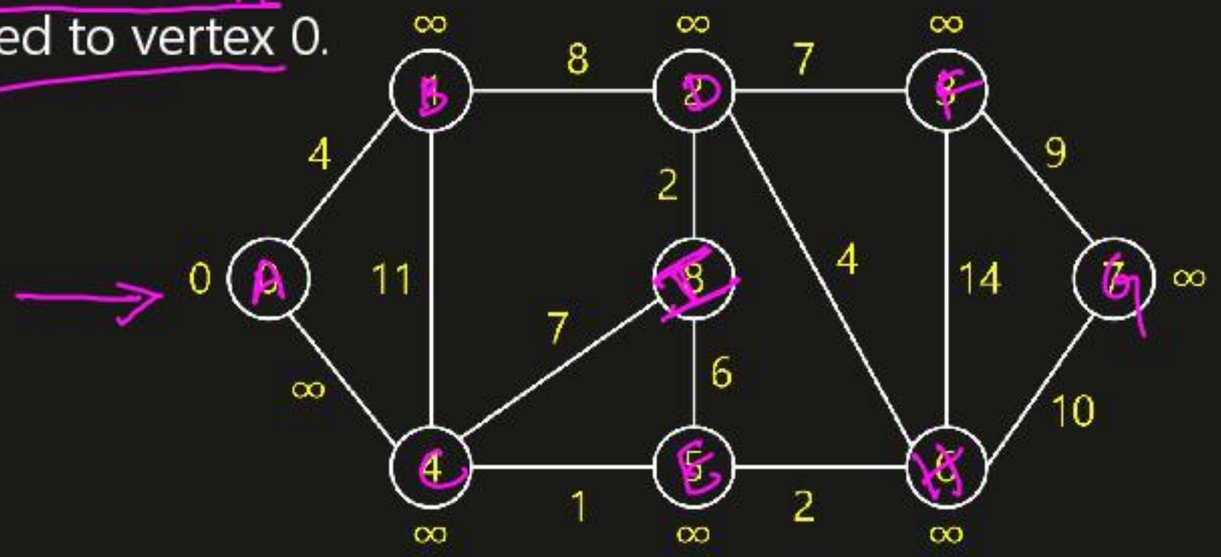
$Cost(v) = 4$

if $(Cost(v) > Cost(u) + d\{u,v\})$

Relaxation $\Rightarrow$ $\boxed{Cost(v) = Cost(u) + d\{u,v\}}$

Dijkstra's Algorithm

- First, we have to consider any vertex as a source vertex. Suppose we consider vertex 0 as a source vertex.
- Here we assume that A as a source vertex, and distance to all the other vertices is infinity. Initially, we do not know the distances. First, we will find out the vertices which are directly connected to the vertex A. As we can observe in the above graph that two vertices are directly connected to vertex 0.



Dijkstra's Algorithm

Dijkstra's Algorithm

Let's assume that the vertex 0 is represented by 'x' and the vertex 1 is represented by 'y'. The distance between the vertices can be calculated by using the below formula:

$$d(x, y) = d(x) + c(x, y) < d(y)$$

$$= (0 + 4) < \infty$$

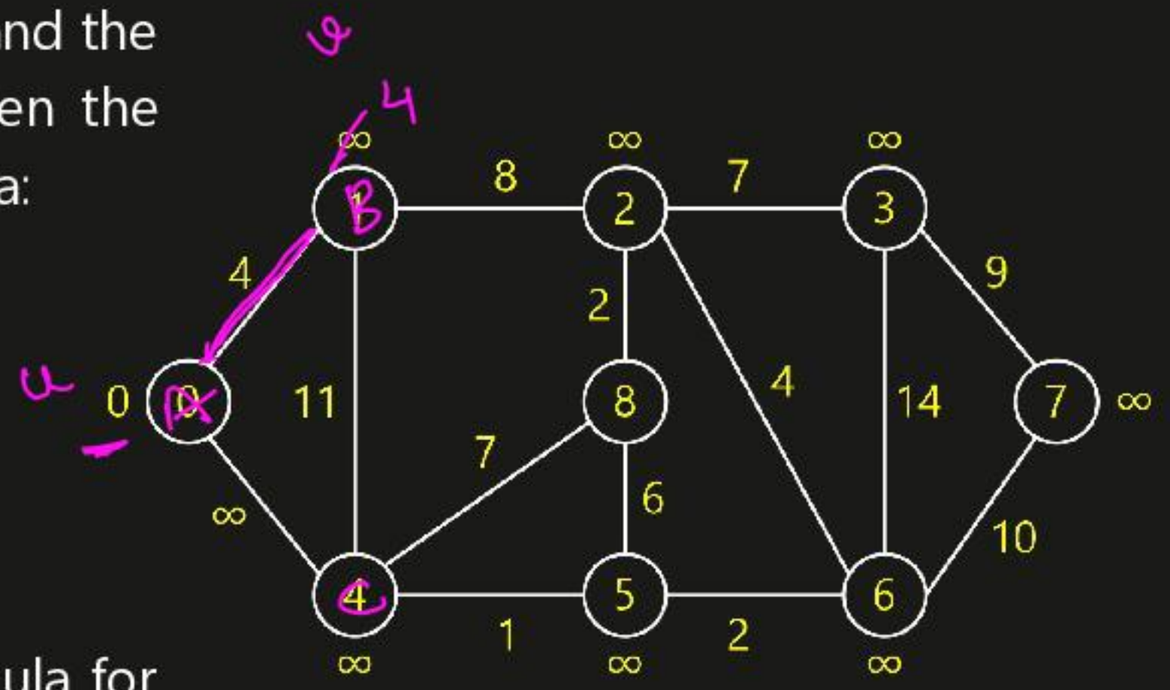
$$= 4 < \infty$$

Since $4 < \infty$ so we will update d(y) from ∞ to 4.

Therefore, we come to the conclusion that the formula for calculating the distance between the vertices:

$$\{ \text{ if}(d(u) + c(u, v) < d(v))$$

$$d(v) = d(u) + c(u, v) \}$$



Dijkstra's Algorithm

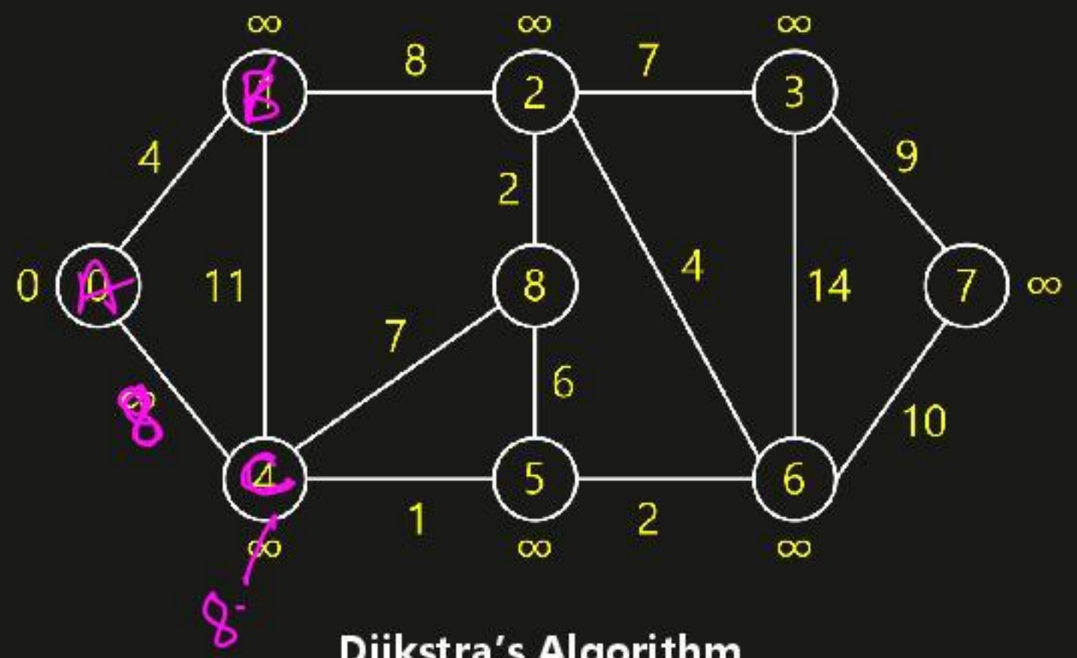
Dijkstra's Algorithm

Now we consider vertex 0 same as 'x' and vertex 4 as 'y'.

$d(x, y) = d(x) + c(x, y) < d(y)$

$= (0 + 8) < \infty$

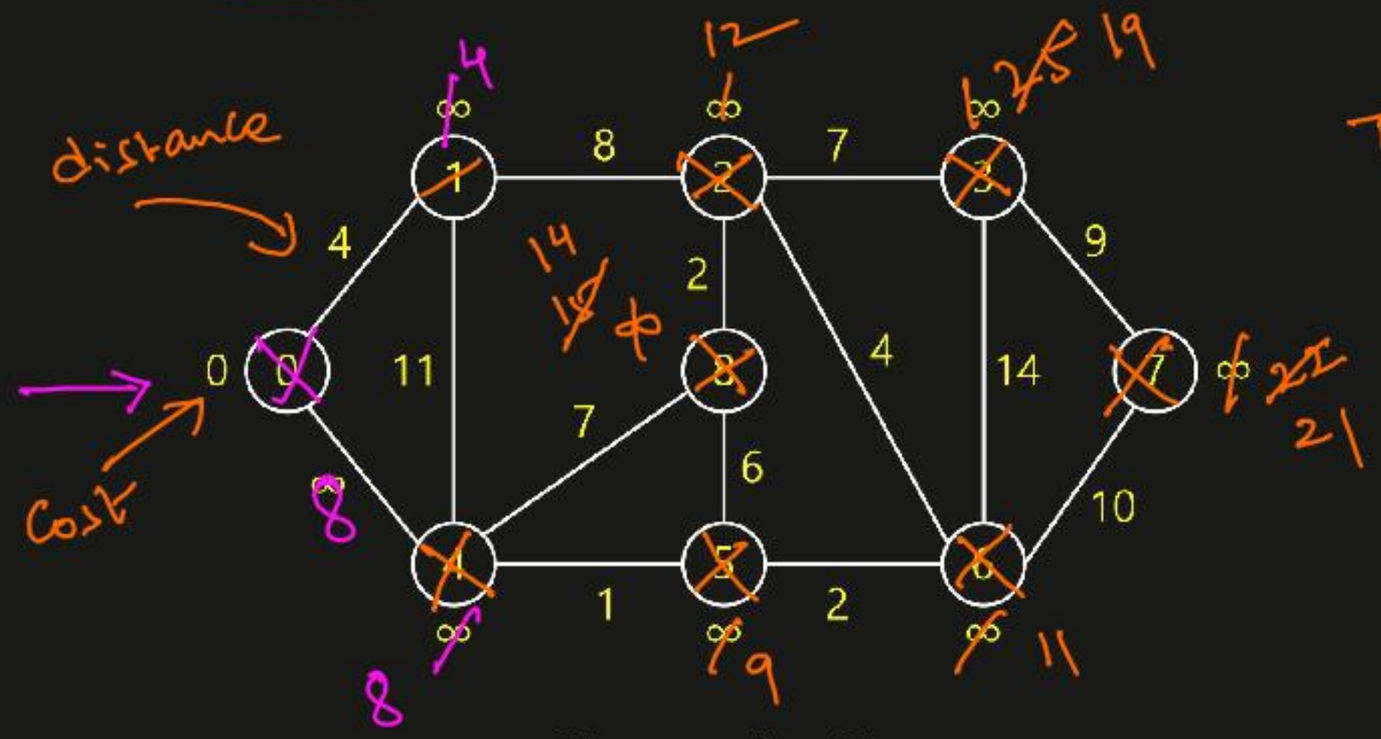
$= 8 < \infty$



Dijkstra's Algorithm

Dijkstra's Algorithm

Repeat the process until all vertices get relaxed or until we get shortest distance from source to all other vertices.



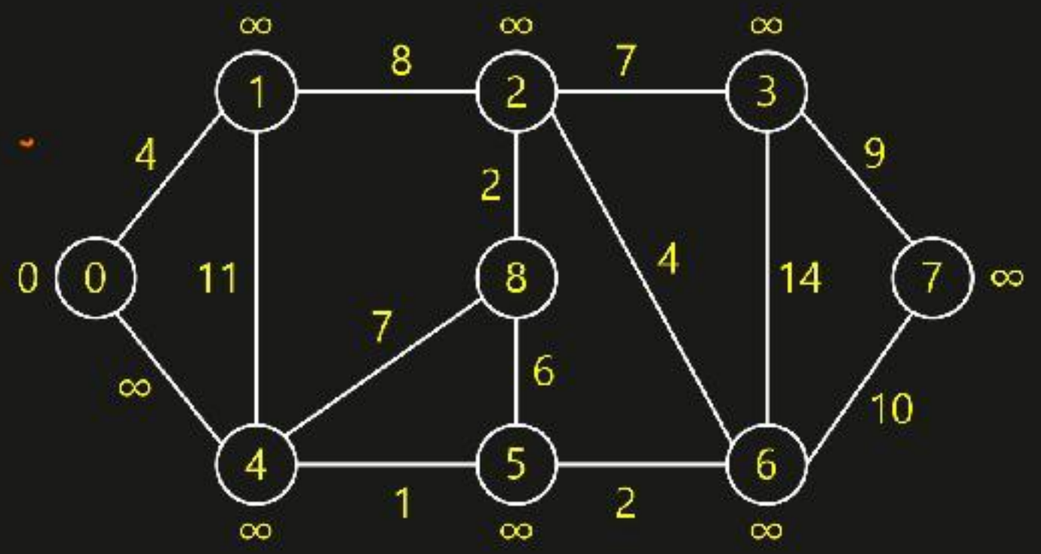
Dijkstra's Algorithm

Node 0

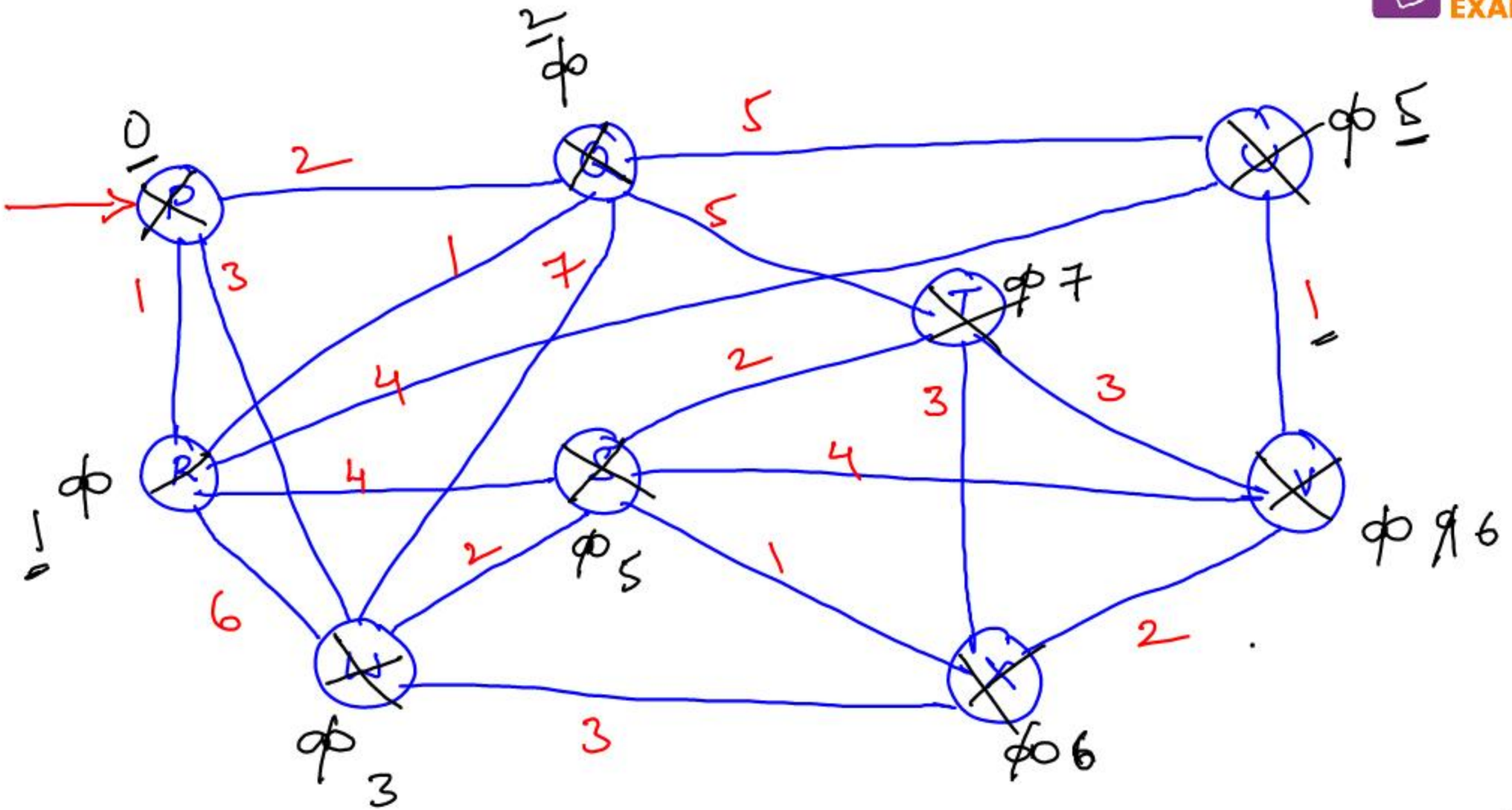
To	Cost
0	0
1	4
2	12
3	19
4	8
5	9
6	11
7	21
8	14

BYJU'S EXAM PREP

Dijkstra's Algorithm



Dijkstra's Algorithm

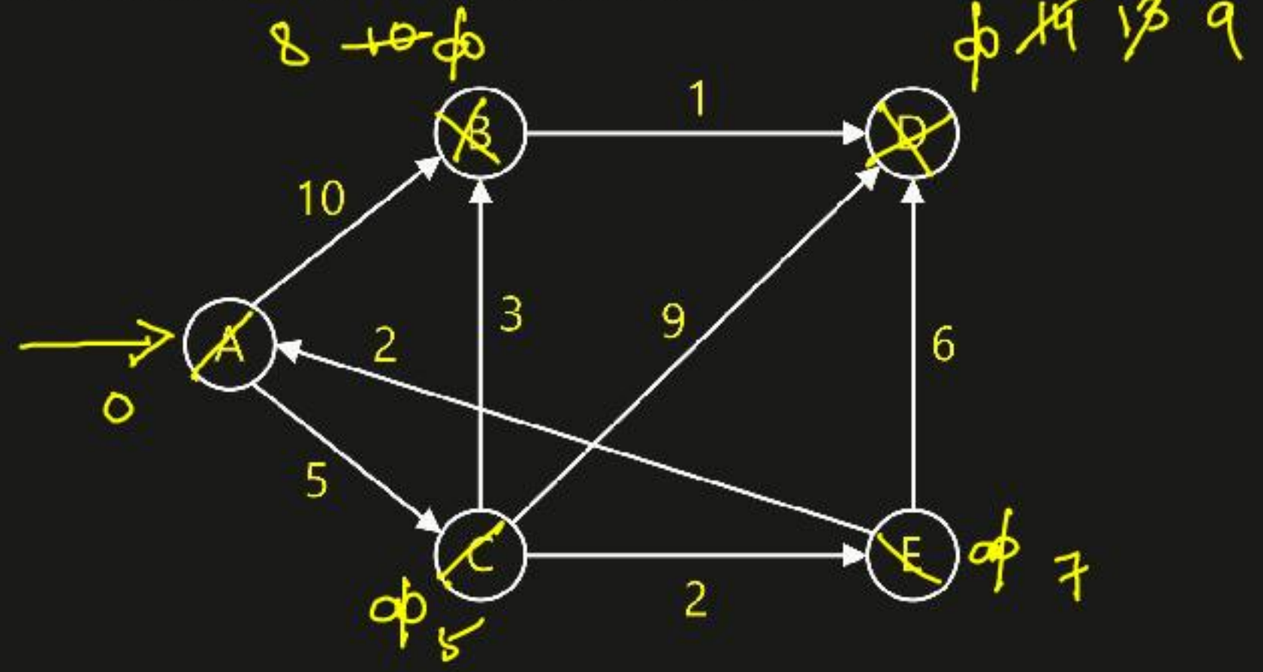


Nodes Selection Sequence : P, R, Q, W, S, U, V, X, T

Dijkstra's Algorithm – Directed Graph

Here, we consider A as a source vertex. A vertex is a source vertex so entry is filled with 0 while other vertices filled with ∞. The distance from source vertex to source vertex is 0, and the distance from the source vertex to other vertices is ∞.

A, C, E, B, D



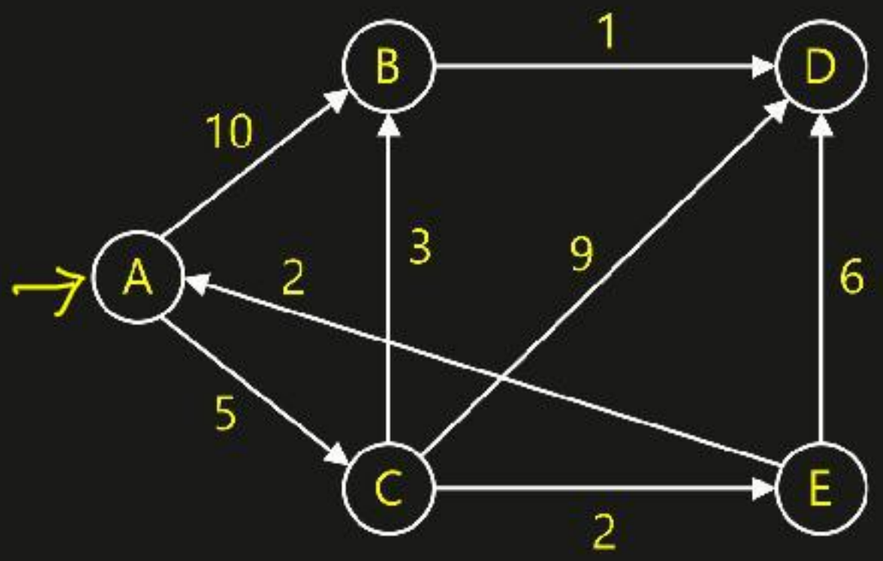
Dijkstra's Algorithm – Directed Graph

Dijkstra's Algorithm – Directed Graph

We will solve this problem using the below table:

	A	B	C	D	E
A	0	∞	∞	∞	∞

Dijkstra's Algorithm – Directed Graph



Dijkstra's Algorithm – Directed Graph

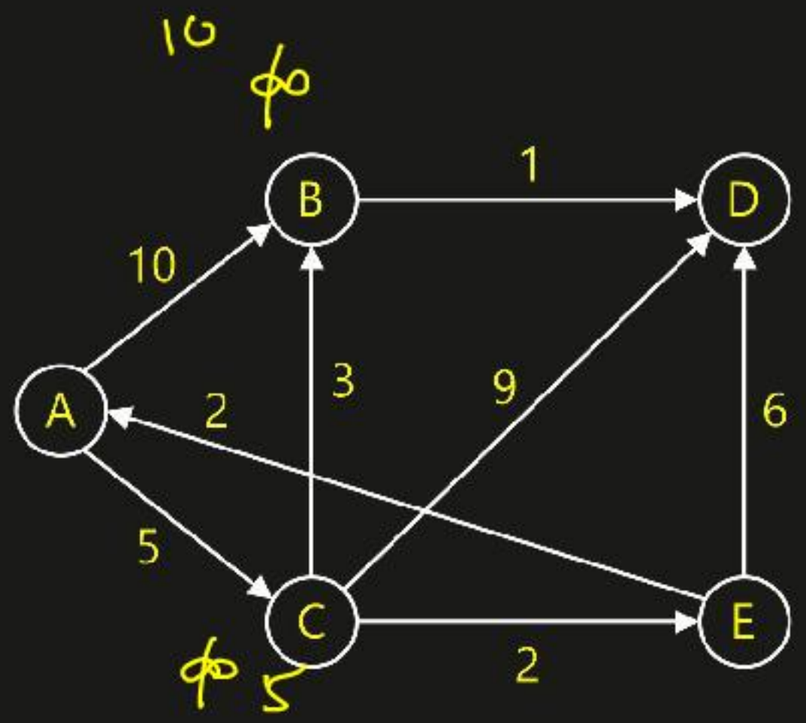
Dijkstra's Algorithm – Directed Graph

If(d(x) + c(x, y) < d(y))

Then we update d(y) = d(x) + c(x, y)

	A	B	C	D	E
A	0	∞	∞	∞	∞
		10	5	∞	∞

Dijkstra's Algorithm – Directed Graph



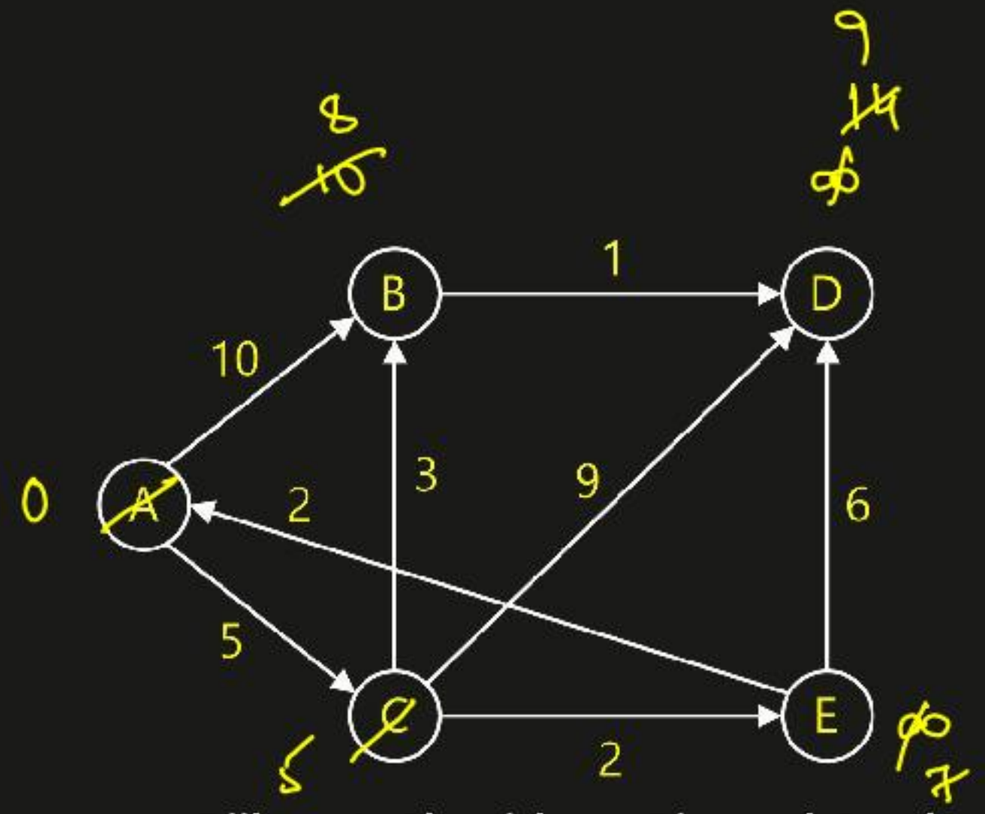
Dijkstra's Algorithm – Directed Graph

BYJU'S EXAM PREP

Dijkstra's Algorithm – Directed Graph

Final table would be :

	A	B	C	D	E
A	0	∞	∞	∞	∞
C		8 ~~10~~	5	~~∞~~ 14	~~∞~~ 7
E		8		~~14~~ 13	7
B		8		~~13~~ 9	
D	0	8	5	9	7



Dijkstra's Algorithm – Directed Graph

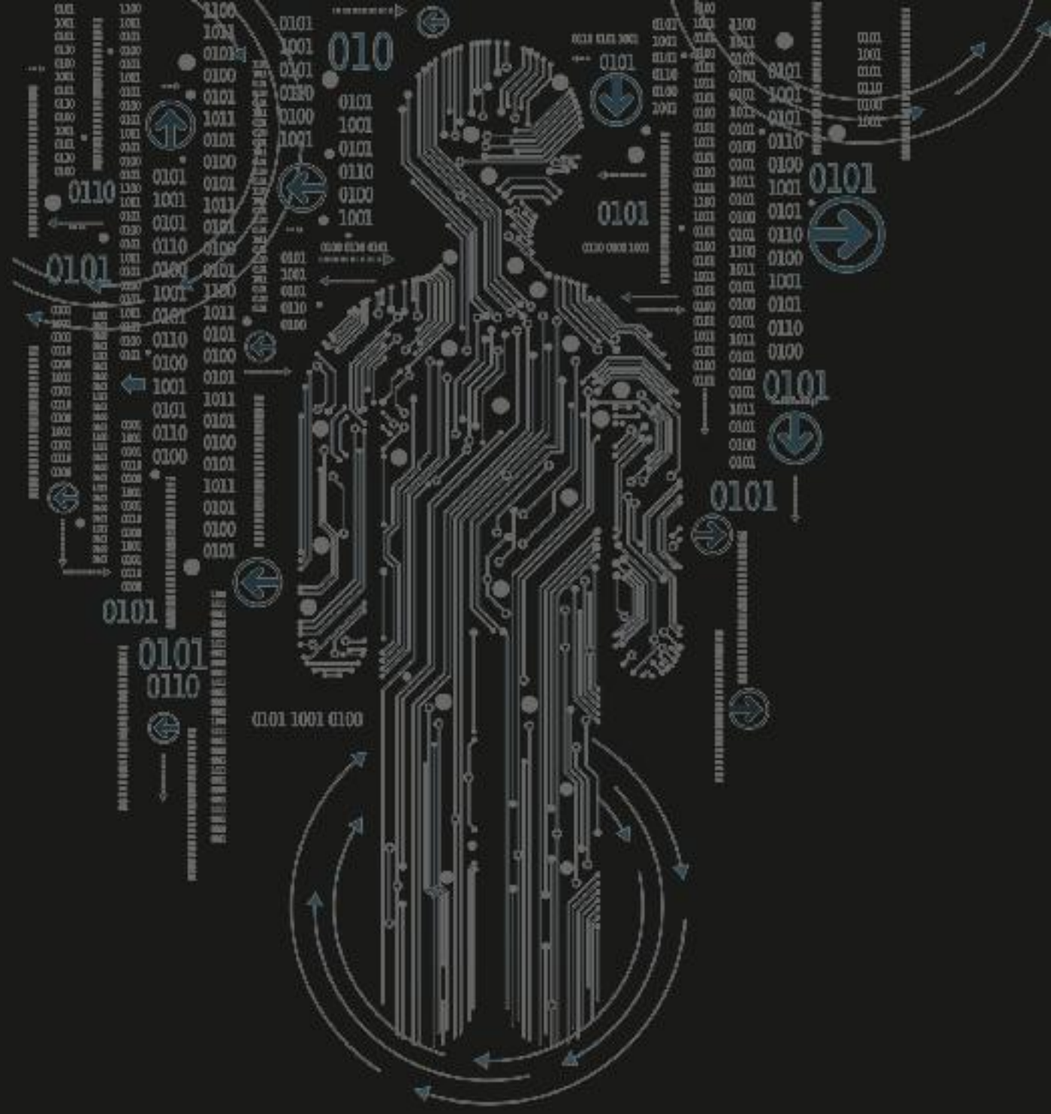
Dijkstra's Algorithm – Directed Graph

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

DP Problem Characteristics, Examples



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Dijkstra's algorithm : N/w graph : shortest Path/ minimum distance from one Node to all other Nodes.

- Single Source shortest Path algorithm.
- Greedy - based Problem [minimization]

⇒ Select a Vertex as Source : it's cost = 0, other Nodes cost = ∞

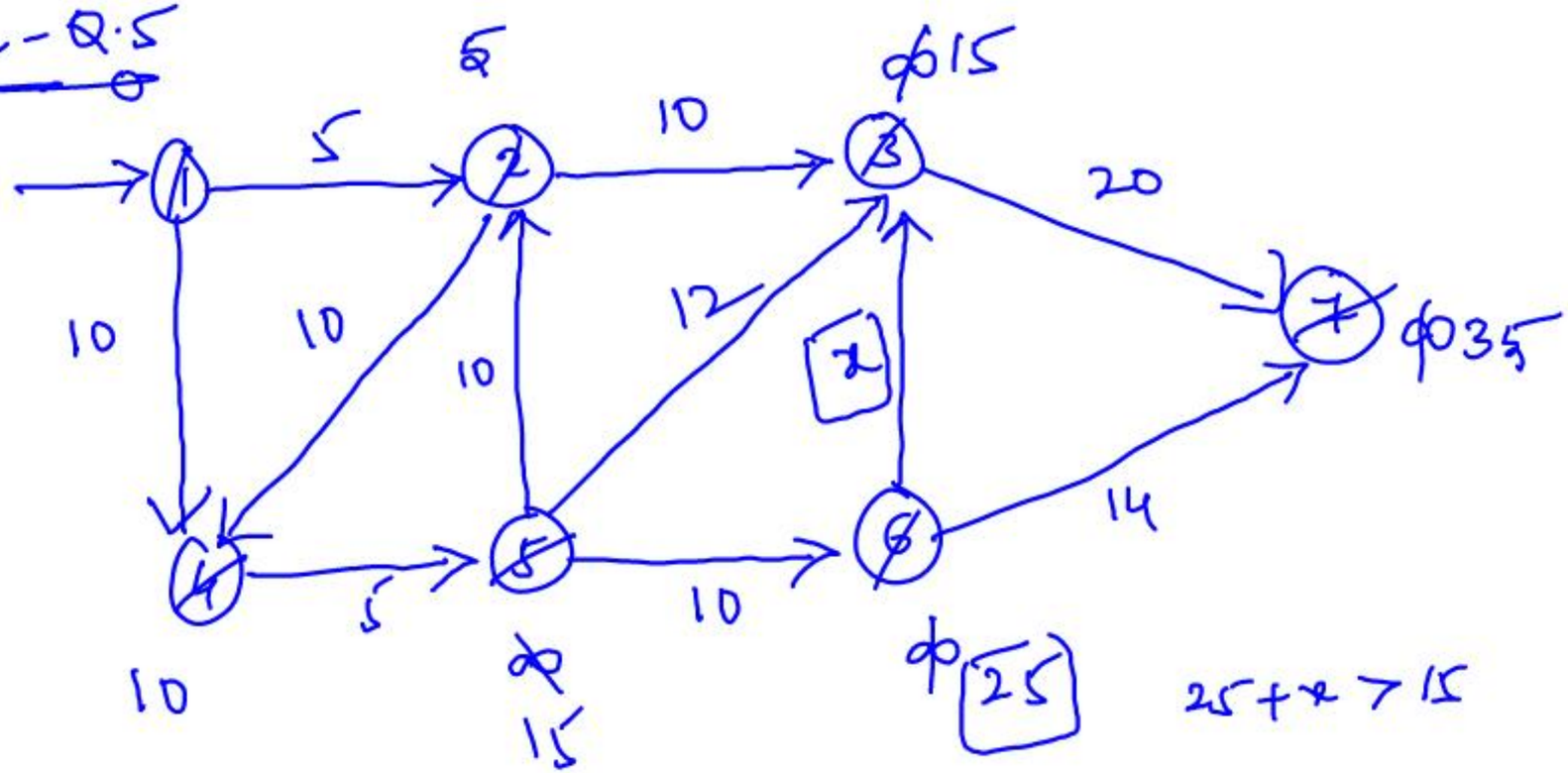
Immediate connected Nodes : $d\{u, v\}$.

Iterate and update $cost(v) = cost\{u\} + d\{u, v\}$

when $\boxed{cost(v) > cost\{u\} + d\{u, v\}}$

in the order of selecting minimum cost Nodes.

Quiz - Q.5



$25 + x > 15$

1, 2, 4, 5, $\boxed{3}$, 6, 7

1, 2, 4, 3, 5, 6, 7

a) 1, 2, 3, 4, 5, 6, 7

b) 1, 3, 4, 5, 6, 2, 7

c) 1, 2, 4, 3, 5, 6, 7

d) 1, 2, 4, 5, $\boxed{6}$, 3, 7

BYJU'S EXAM PREP

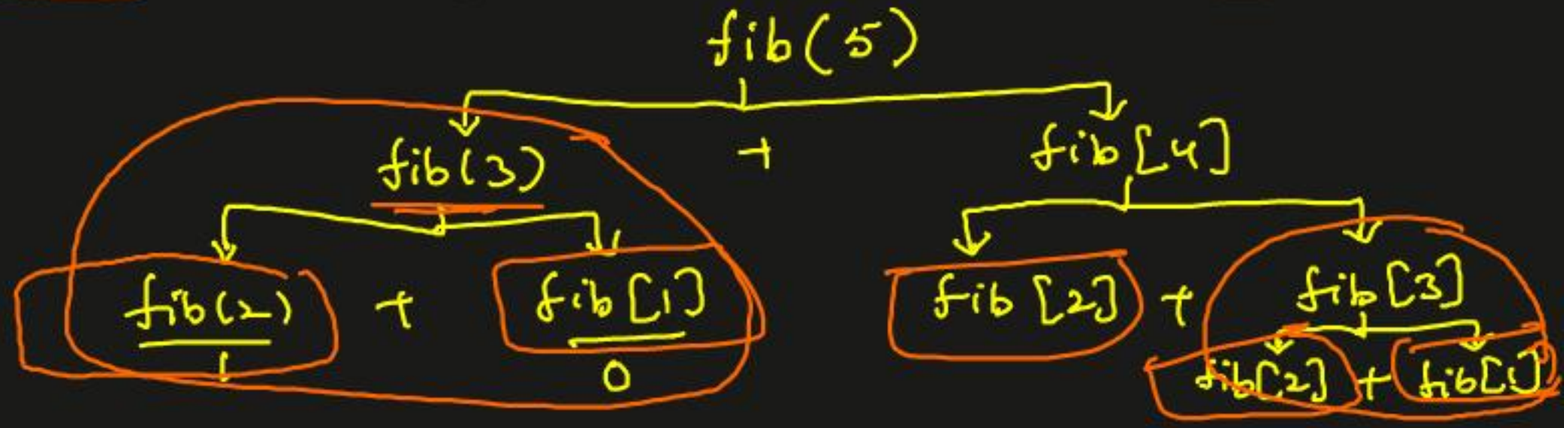
Binary Search

1, 2, 3, 4, 5, 6, 7, 8

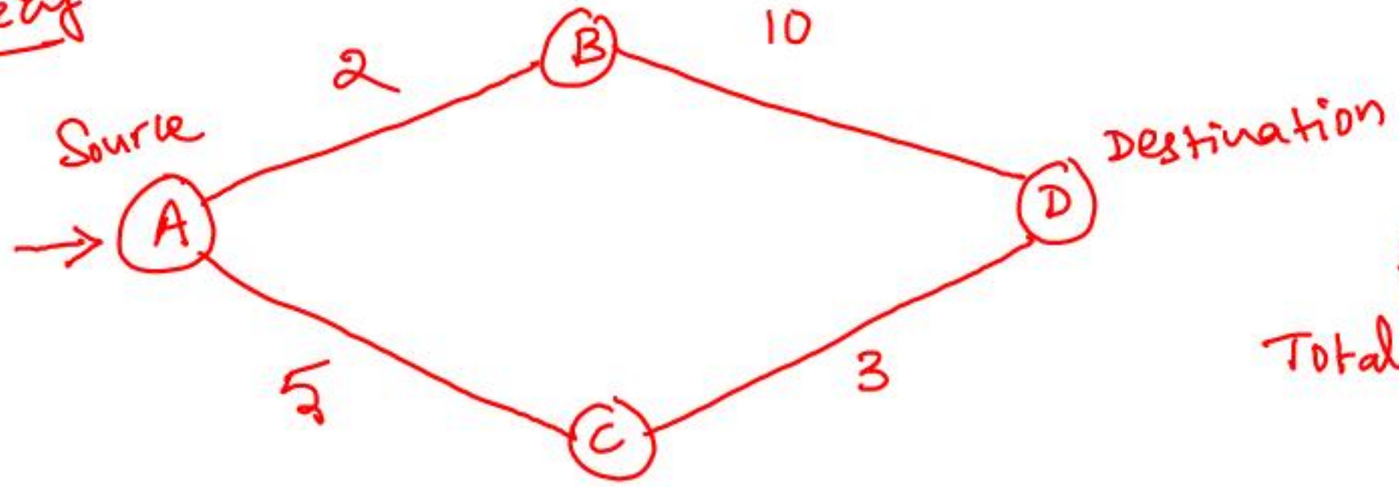
Dynamic Programming

Fibo : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ---

- Dynamic programming is a technique that breaks the problems into sub-problems and saves the result for future purposes so that we do not need to compute the result again.
- The main use of dynamic programming is to solve optimization problems. Here, optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem.
- The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.



Greedy



A — B — D

Total Cost : 2 + 10 = 12

Dynamic Prog

Compare ({(A − B) + (B − D)}, {(A − C) + (C − D)}) and choose optimal solution.

(2+10, 5+3) = (12, 8)

↓

Select A − C − D optimal sol.

BYJU'S EXAM PREP

Characteristics Of a DP Problem

- Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are overlapping sub-problems and optimal substructure.

Repetition of Sub Problem Solving.

D & C : optimal Sub structure only

Dp : overlapping & optimal Structure both.

BYJU'S EXAM PREP

Overlapping Of Sub Problems

- Like Divide-and-Conquer approach, Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.
- For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

BYJU'S EXAM PREP

Example

Consider an example of the Fibonacci series. The following series is the Fibonacci series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ,...

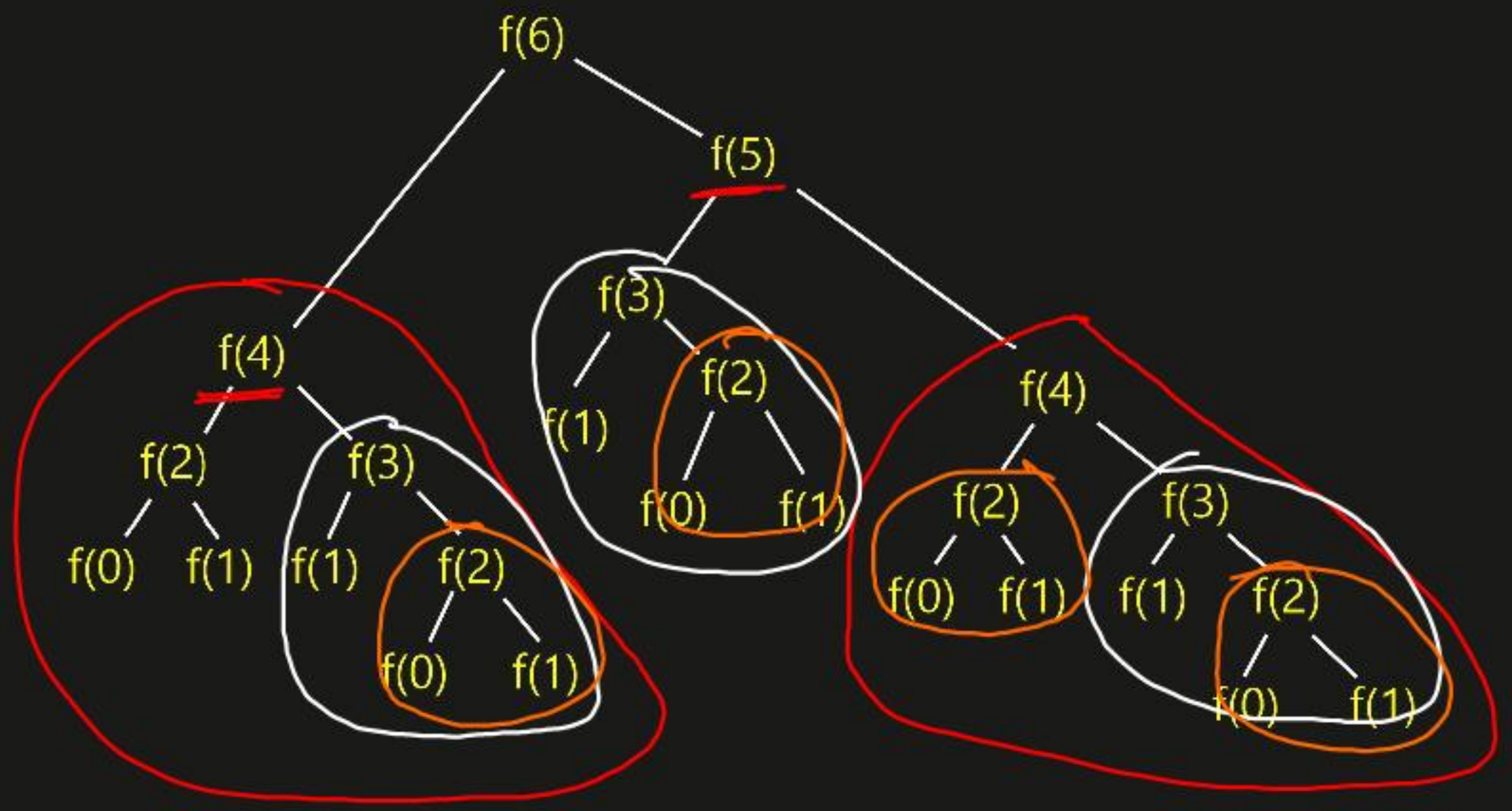
The numbers in the above series are not randomly calculated. Mathematically, we could write each of the terms using the below formula:

$$F(n) = F(n-1) + F(n-2),$$

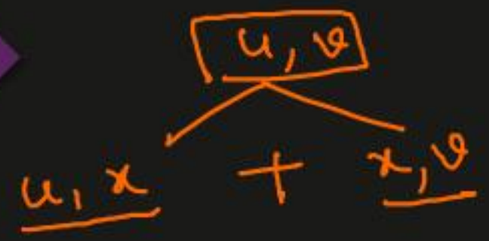
DP Problem Characteristics, Examples

Overlapping Of Sub Problems

: Repetition of Sub Problem Solving.



BYJU'S EXAM PREP



Optimal Sub Structure Property

A given problem has Optimal Substructure Property, if the optimal solution of the given problem can be obtained using optimal solutions of its sub-problems.

For example, the Shortest Path problem has the following optimal substructure property –

If a node x lies in the shortest path from a source node u to destination node v, then the shortest path from u to v is the combination of the shortest path from u to x, and the shortest path from x to v.

BYJU'S EXAM PREP

Dynamic Programming

Dynamic Programming algorithm is designed using the following four steps –

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from the computed information.

BYJU'S EXAM PREP

2 Methods to solve a DP Problem.

Memoization (Top-Down)

Tabulation (Bottom-UP)

Top - Down Vs Bottom – Up Approach

Top-Down Approach:

- In the top-down approach, a complex algorithm is broken down into smaller fragments, better known as 'modules.'
- These modules are then further broken down into smaller fragments until they can no longer be fragmented. This process is called 'modularization.' Base-Case
- However, during the modularization process, you must always maintain the integrity and originality of the algorithm.
- Moreover, a top-down approach is more suitable when the software needs to be designed from scratch and very specific details are unknown.

BYJU'S EXAM PREP

Top - Down Vs Bottom – Up Approach

Advantages of Top-down approach:-

- Each module of code is to be tested separately.
- Breaking a problem down into smaller chunks makes it far easier to understand, solve and manage.
- Testing and debugging are efficient and easier.
- Project implementation is smoother and shorter.

Drawbacks Top-down approach :-

- Specification tends to change over time and in a top-down approach, all decisions made from the beginning of the project depend directly or indirectly on the high-level specification.
- In Dynamic Programming, the top-down approach is slow as compared to the bottom-up approach, as it involves recursion.

BYJU'S EXAM PREP

Top - Down Vs Bottom – Up Approach

Bottom-Up Approach :

- Bottom-up programming focuses on designing an algorithm by beginning at the very basic level and building up as it goes upside.
- In this approach, the modules are designed individually and are then integrated together to form a complete algorithmic design.
- Moreover, the bottom-up approach is more suitable when a system needs to be created from some existing components.

factorial of a given number

function Calling Sequence

$f(5) \rightarrow f(4) \rightarrow f(3) \rightarrow f(2) \rightarrow f(1)$

$\boxed{\text{Top-Down approach}}$

fact(5) = 5 * fact(4)
↳ 4 * fact(3)
↳ 3 * fact(2)
↳ 2 * fact(1)

= 5 × 4 × 3 × 2 × 1

---

fact(5).

n==1 fact(1) = 1

function Calling Sequence

$f(1) \rightarrow f(2) \rightarrow f(3) \rightarrow f(4) \rightarrow f(5)$

$\boxed{\text{Bottom-up approach}}$

fact(1) * 2

(1 * 2) * 3

((1 * 2) * 3) * 4

(((1 * 2) * 3) * 4) * 5

$\boxed{n = 5}$

= 1 * 2 * 3 * 4 * 5

BYJU'S EXAM PREP

Top - Down Vs Bottom – Up Approach

Advantages of Bottom-Up Approach:-

- Test conditions are easier to create.
- Observation of test results is easier.
- Contains less redundancy due to the presence of data encapsulation and data-hiding.
- Reusability of the code.

Drawbacks of Bottom-Up Approach :-

- In the Bottom-Up approach, we solve all sub-problems (even though some of the solutions of the subproblems aren't needed to solve), which requires additional calculations.
- In the Bottom-Up approach, sometimes it is difficult to identify the overall functionality of the system in the initial stages.

DP Problem Characteristics, Examples

Applications Of DP

(Bottom-up approach, Tabulation Method).

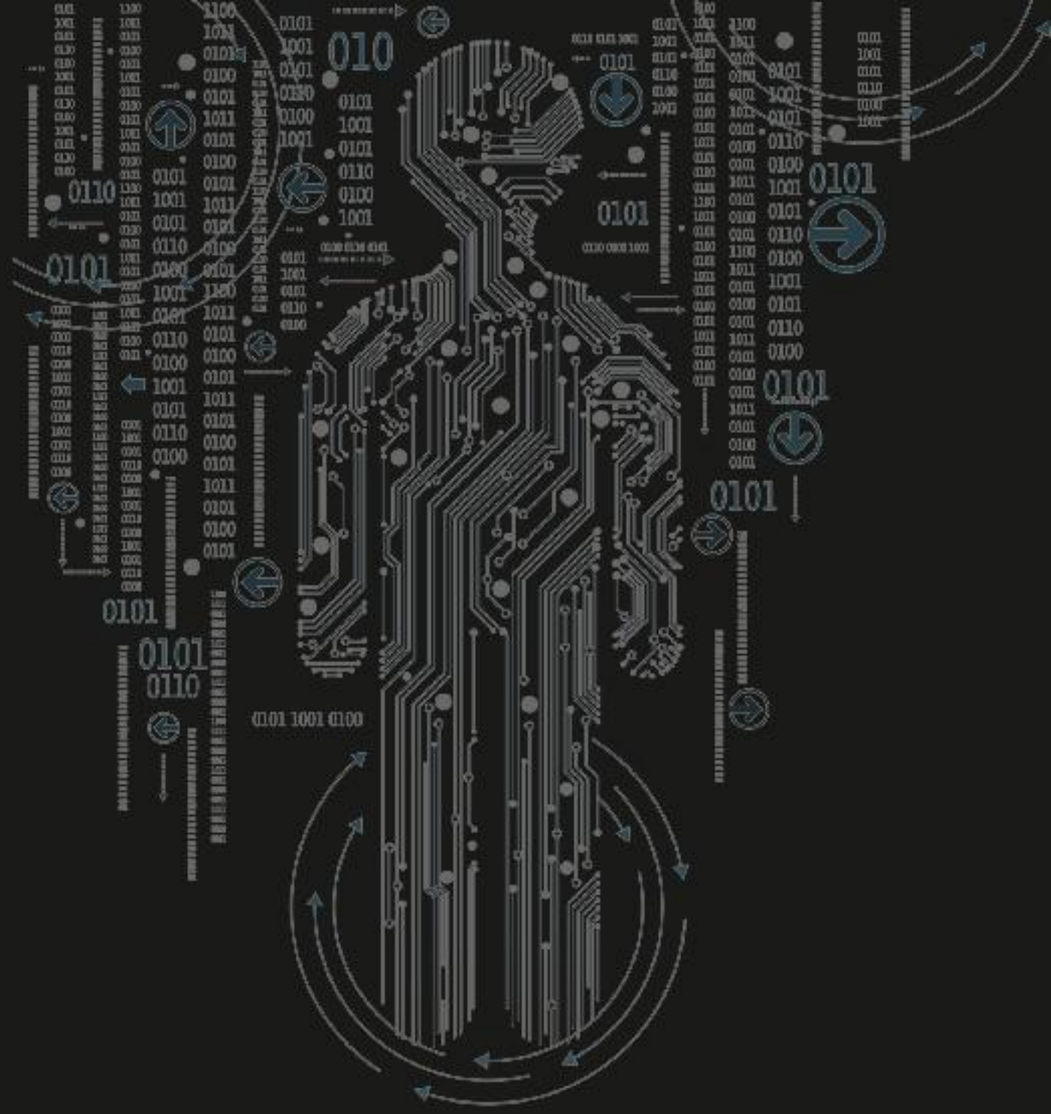
- Matrix Chain Multiplication
- Longest Common Subsequence Problem
- Travelling Salesman Problem
- Single Source Shortest Path Problem [Bellman-Ford]
- All – Pair Shortest Path Problem (Floyd-Warshall algo)
- Knapsack Problem and many more

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Matrix Chain Multiplication



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Dynamic Programming : Optimization Problems with 2 characteristics

a) overlapping of Sub Problems

b) optimal Sub structure

Dp Technique in 2 ways
- Memoization (Top-Down approach)
- Tabulation (Bottom-up approach)

Preferred by DP as it guarantees optimal solution

BYJU'S EXAM PREP

Matrix Chain Multiplication

(MCM) ⟹ minimization Problem.

- Given a sequence of matrices, find the most efficient way to multiply these matrices together. The problem is not actually to perform the multiplications, but merely to decide in which order to perform the multiplications.
- We have many options to multiply a chain of matrices because matrix multiplication is associative, means that, no matter how we parenthesize the product, the result will be the same.

$$A \cdot (B \cdot C) = (A \cdot B) \cdot C$$

→ minimum number of multiplications

Matrix Chain Multiplication

BYJU'S EXAM PREP

$A_1 \quad A_2 \quad A_3 \quad A_4$

$2\times3 \quad 3\times4 \quad 4\times5 \quad 5\times6$

$A_1 \cdot A_2 = 2\times3\times4 = 24$

$A_2 \cdot A_3 = 3\times4\times5 = 60$

$A_3 \cdot A_4 = 4\times5\times6 = 120$

Matrix Multiplication

- For multiplying any two matrices of orders r1 * c1 and r2 * c2 respectively, the condition c1 == r2 must be satisfied.

$$\underset{A}{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}} \underset{B}{\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}} = \underset{C}{\begin{bmatrix} c_{11} = (a_{11} * b_{11} + a_{12} * b_{21}) & c_{12} = (a_{11} * b_{12} + a_{12} * b_{22}) \\ c_{21} = (a_{21} * b_{11} + a_{22} * b_{21}) & c_{22} = (a_{21} * b_{12} + a_{22} * b_{22}) \end{bmatrix}}$$

1 2 3 4 5 6 7 8

$\underset{r_1}{2} \times \underset{c_1}{2} \quad \underset{r_2}{2} \times \underset{c_2}{2}$

2×2

$r_1 \times c_2$

Total multiplications $= 8 = 2\times2\times2 \Rightarrow r_1 \times c_1 \times c_2$

$r_1 \times r_2 \times c_2$

A B C

$(A \cdot B) \cdot C$ $\quad$ $A \cdot (B \cdot C)$

10×5 $\quad$ 5×60 $\quad$ 10×30 $\quad$ 30×60

Matrix Chain Multiplication

- For example, if we had four matrices A, B, C, and D, we would have:
- (ABC)D = (AB)(CD) = A(BCD) = ...
- However, the order in which we parenthesize the product affects the number of simple arithmetic operations needed to compute the product, or the efficiency.
- For example, suppose A is a 10 × 30 matrix, B is a 30 × 5 matrix, and C is a 5 × 60 matrix. Then,

(AB)C = (10×30×5) + (10×5×60) = 1500 + 3000 = 4500 operations minimum No. of operations

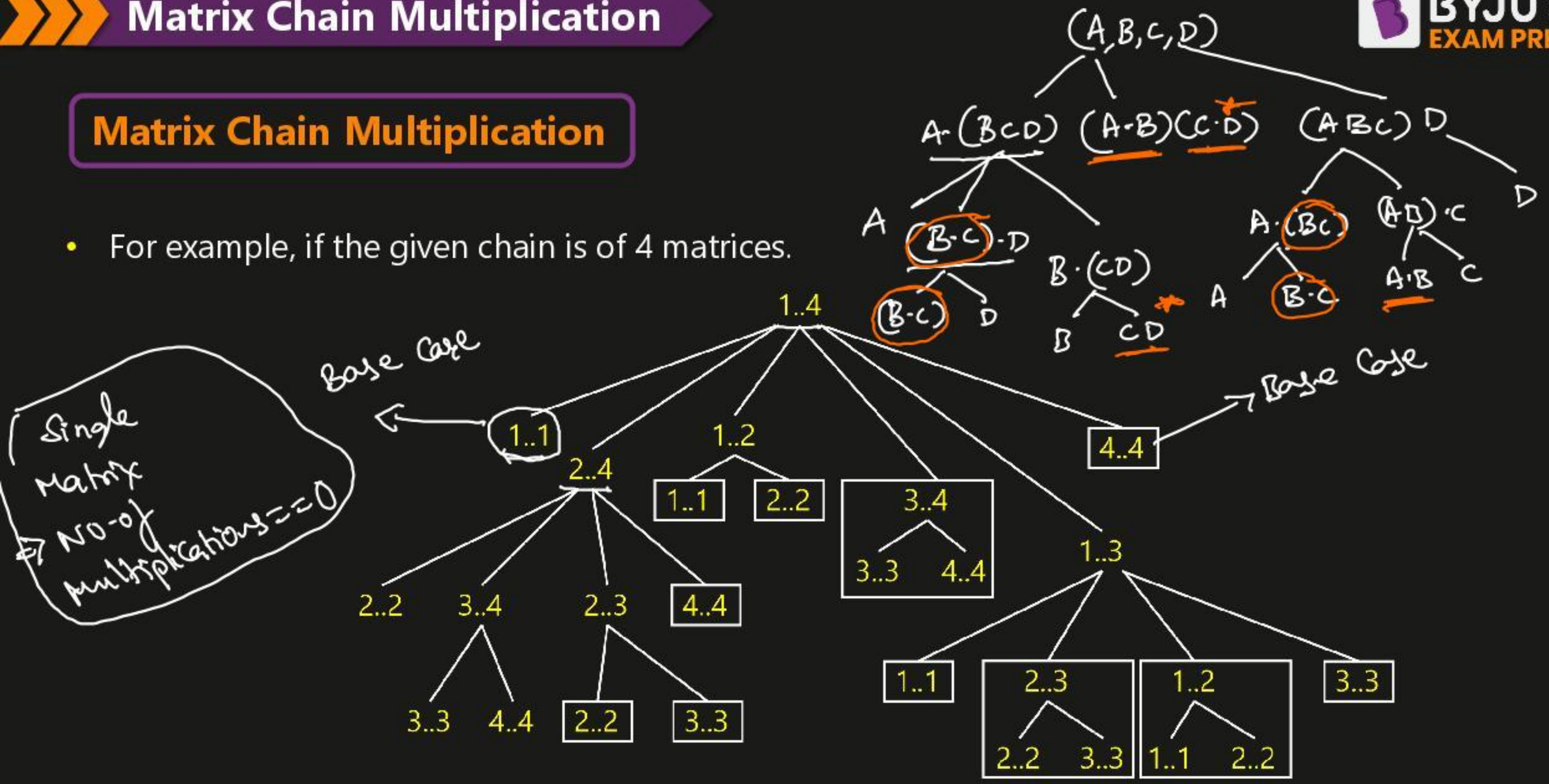
A(BC) = (30×5×60) + (10×30×60) = 9000 + 18000 = 27000 operations.

Clearly the first parenthesization requires less number of operations.

BYJU'S EXAM PREP

Matrix Chain Multiplication

- For example, if the given chain is of 4 matrices.



BYJU'S EXAM PREP

Matrix Chain Multiplication

- In Dynamic Programming, We initialize the diagonal element with equal i,j value with '0'.
- After that second diagonal is sorted out and we get all the values corresponded to it
- Now the third diagonal will be solved out in the same way.

4 Matrices

A, B, C, D

2x3 3x4 4x5 5x6

1. (A·B)·(C·D)
2. (A(B·C))·D
3. ((A·B)·C)·D
4. A·(B·(C·D))
5. A·((B·C)·D)

Result

	A	B	C	D
A	0			
B	X	0		
C	X	X	0	
D	X	X	X	0

Let 3 Matrices A_1 A_2 A_3

4×5 5×7 7×9

d_0 d_1 d_1 d_2 d_2 d_3

$(A_1 \times A_2) \times A_3$

$A_1 \times A_2 = 4 \times 5 \times 7 = 140 \Rightarrow [\ ]_{4\times 7}$

$A_3 =$

$[A_1 \times A_2] \times A_3 = \underset{d_0}{4} \times \underset{d_2}{7} \times \underset{d_3}{9} = 252$

4×7 7×9

$140 + 252 = 392$

$M[1,3] = M[1,2] + M[3,3] + \underset{d_0 \times d_2 \times d_3}{4 \times 7 \times 9}$

i j k $k+1$ i j $d_{(i-1)} \times d_k \times d_j$

$A_1 \times (A_2 \times A_3)$

$A_2 \times A_3 = 5 \times 7 \times 9 = 315 \Rightarrow [\ ]_{5\times 9}$

$A_1 = 0$

$A_1 \cdot [\ ] = \underset{d_0}{4} \times \underset{d_1}{5} \times \underset{d_3}{9} = 180$

4×5 5×9

$315 + 180 = 495$

$M[1,3] = M[1,1] + M[2,3] + \underset{d_0\ d_1\ d_3}{4 \times 5 \times 9}$

i j k $k+1$ i j $d_{(i-1)} \times d_k \times d_j$

$$\boxed{M[i,j] = \min_{i \le k < j} \left\{ M[i,k] + M[k+1,j] + d_{i-1} * d_k * d_j \right\}}$$

$i = 1$

$j = 3$

$1 \le k < 3$ min

$$\left.\begin{array}{l} k=1 \quad \left\{ M[1,1] + M[2,3] + d_0 \times d_1 \times d_3 \right. = 495 \\ k=2 \quad \left\{ M[1,2] + M[3,3] + d_0 \times d_2 \times d_3 \right. = 392 \end{array}\right.$$

min = 392

Matrix Chain Multiplication

$$M[i,j] = \min_{i \le k < j} \{ M[i,k] + M[k+1,j] + d_{i-1} \times d_k \times d_j \}$$

MCM Example

$A_1 \quad A_2 \quad A_3 \quad A_4$

Consider 4 matrices ~~A, B, C and D~~ of orders 2 * 3, 3 * 4, 4 * 5 and 5 * 6 respectively.

$d_0 \quad d_1 \quad d_2 \quad d_3 \quad d_4$

$$M[1,4] = \min_{1 \le k < 4} \begin{cases} k=1 & \overset{0}{M[1,1]} + M[2,4] + d_0 \times d_1 \times d_4 \\ k=2 & M[1,2] + M[3,4] + d_0 \times d_2 \times d_4 \\ k=3 & M[1,3] + \underset{0}{M[4,4]} + d_0 \times d_3 \times d_4 \end{cases}$$

$$M[\overset{i}{1},\overset{j}{2}] = \min_{1 \le k < 2} \{ k=1 \quad M[1,1] + M[2,2] + d_0 \times d_1 \times d_2 \}$$

$$= 0 + 0 + 2 \times 3 \times 4$$

$$= 24$$

Construct 2 Matrices M and K.

|row number − column number| = 0, 1, 2, 3

Matrix — M

	A_1	A_2	A_3	A_4
A_1	0	24	64	124
A_2	×	0	60	150
A_3	×	×	0	120
A_4	×	×	×	0

Matrix — K

	A_1	A_2	A_3	A_4
A_1	0	1	2	3
A_2	×	0	2	3
A_3	×	×	0	3
A_4	×	×	×	0

Minimum No. of Multiplications = 124

A_1 2×3, A_2 3×4, A_3 4×5 → 2×4 (2×5)

A_4 5×6

$((A_1 A_2) A_3) A_4$

24 + 40 + 60 = 124

BYJU'S EXAM PREP

MCM Example

$d_0 \quad d_1 \quad d_2 \quad d_3 \quad d_4$

Consider 4 matrices A, B, C and D of orders 2 * 3, 3 * 4, 4 * 5 and 5 * 6 respectively.

$$M[\overset{i}{2},\overset{j}{3}] = \underset{2 \le k < 3}{\overset{i \quad j}{}} \{k=2 \quad M[2,2] + M[3,3] + d_1 \times d_2 \times d_3\}$$

$$= 0 + 0 + 3 \times 4 \times 5 = 60$$

$$M[\overset{i}{3},\overset{j}{4}] = 3 \le k < 4 \{k=3 \quad M[3,3] + M[4,4] + d_2 \times d_3 \times d_4\}$$

$$= 0 + 0 + 4 \times 5 \times 6 = 120$$

$$M[\overset{i}{1},\overset{j}{3}] = \min_{1 \le k < 3} \begin{cases} k=1 & M[1,1] + M[2,3] + d_0 \times d_1 \times d_3 \\ k=2 & M[1,2] + M[3,3] + d_0 \times d_2 \times d_3 \end{cases} = \begin{cases} k=1 & 0 + 60 + 2 \times 3 \times 5 \\ k=2 & 24 + 0 + 2 \times 4 \times 5 \end{cases}$$

$$= \min\{90, 64\} = \underline{64} \ (k=2)$$

BYJU'S EXAM PREP

MCM Example

$d_0 \quad d_1 \quad d_2 \quad d_3 \quad d_4$

Consider 4 matrices A, B, C and D of orders 2 * 3, 3 * 4, 4 * 5 and 5 * 6 respectively.

$$M[\overset{i}{2},\overset{j}{4}] = \min_{2 \le k < 4} \begin{cases} k=2, & M[2,2] + M[3,4] + d_1 \times d_2 \times d_4 \\ k=3 & M[2,3] + M[4,4] + d_1 \times d_3 \times d_4 \end{cases} = \min \begin{cases} k=2 & 0 + 120 + 3\times4\times6 \\ k=3 & 60 + 0 + 3\times5\times6 \end{cases}$$

$$= \min\{192, 150\} = 150 \ (k=3)$$

$$M[\overset{i}{1},\overset{j}{4}] = \min_{1 \le k < 4} \begin{cases} k=1 & M[1,1] + M[2,4] + d_0 \times d_1 \times d_4 \\ k=2 & M[1,2] + M[3,4] + d_0 \times d_2 \times d_4 \\ k=3 & M[1,3] + M[4,4] + d_0 \times d_3 \times d_4 \end{cases} = \min \begin{cases} k=1 & 0 + 150 + 2\times3\times6 \\ k=2 & 24 + 120 + 2\times4\times6 \\ k=3 & 64 + 0 + 2\times5\times6 \end{cases}$$

$$= \min\{186, 192, 124\} = \underline{124} \ (k=3)$$

BYJU'S EXAM PREP

MCM Algorithm

MATRIX-CHAIN-ORDER (p)

```
n   length[p]-1
for i ← 1 to n
do m [i, i] ← 0
for l ← 2 to n    // l is the chain length
do for i ← 1 to n-l + 1
do j ← i+ l -1
m[i,j] ← ∞ => X
for k  ← i to j-1
do q  ← m [i, k] + m [k + 1, j] + p_{i-1} p_k p_j
If q < m [i,j]
then m [i,j] ← q
s [i,j] ← k
return m and s.
```

BYJU'S EXAM PREP

MCM Parenthesizing

We will use table s to construct an optimal solution.

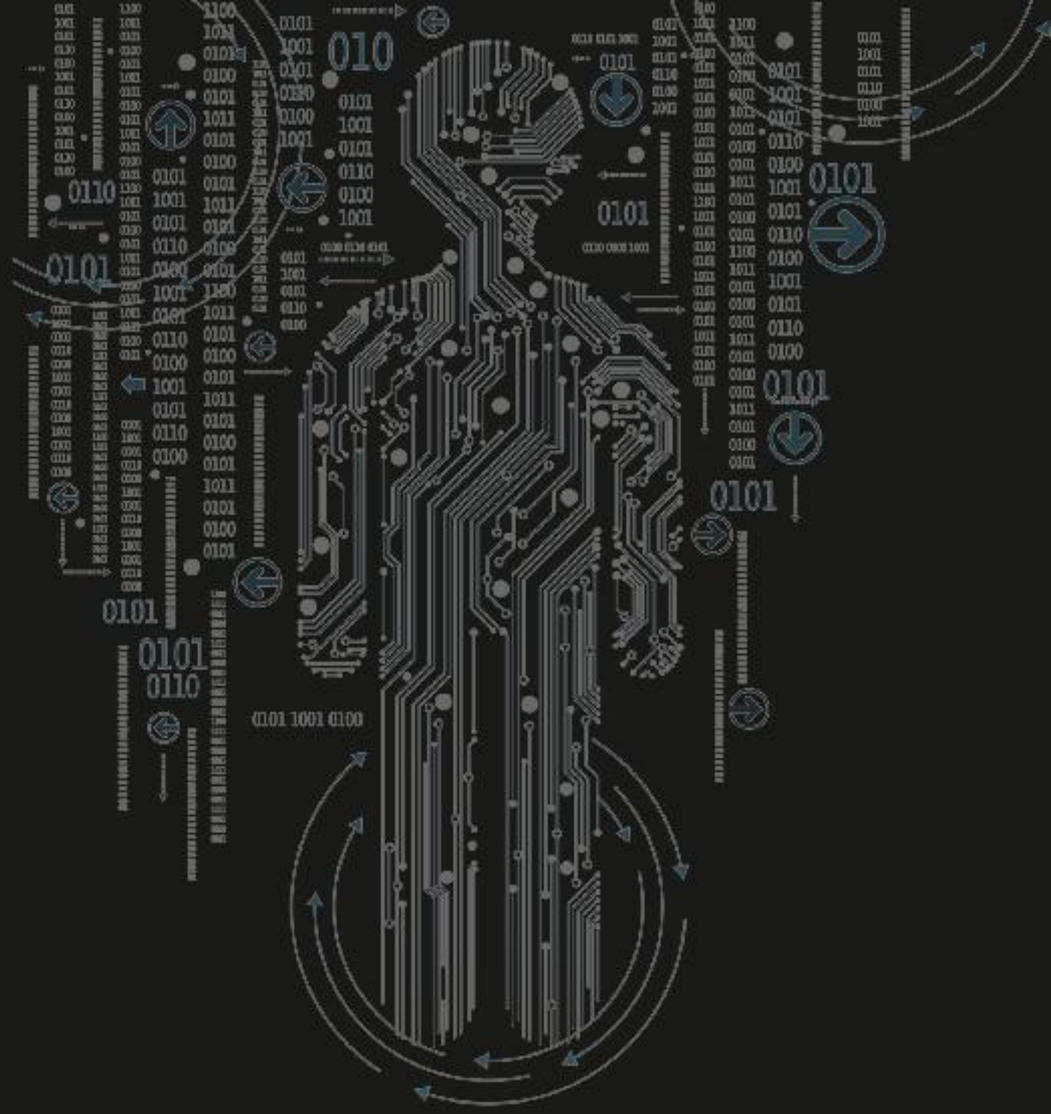
```
PRINT-OPTIMAL-PARENS (s, i, j)
 1. if i=j
 2. then print "A"
 3. else print "("
 4. PRINT-OPTIMAL-PARENS (s, i, s [i, j])
 5. PRINT-OPTIMAL-PARENS (s, s [i, j] + 1, j)
 6. print ")"
```

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Travelling Salesman Problem



Guidelines to Attend Live Class

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Matrix chain Multiplication

- Series of Matrices : optimal Sequence to multiply : minimal multiplications
- Brute-Force : Try all Possibilities : Compare and find minimum
- Dp : Bottom-up approach : Tabulation Method

2 Matrices : Matrix 'M', Matrix 'K' / 'P'

Number of multiplications — optimal Sequence

$$M[i, j] = \min_{i \le k < j} \left\{ M[i, k] + M[k+1, j] + d_{i-1} * d_k * d_j \right\}$$

BYJU'S EXAM PREP

Matrix Chain Multiplication

Example

A_1, A_2, A_3, A_4

K

	A_1	A_2	A_3	A_4
A_1	0	1	1	3
A_2	x	0	2	3
A_3	x	x	0	3
A_4	x	x	x	0

$A_1 \quad A_2 \quad A_3 \quad A_4$

$K[1,4] = 3 \Rightarrow (A_1 \quad A_2 \quad A_3) \quad A_4$

$K[1,3] = 1 \Rightarrow ((A_1)(A_2 \quad A_3)) \quad A_4$

Ex:-2

	A_1	A_2	A_3	A_4
A_1	0	1	2	3
A_2	X	0	2	3
A_3	X	X	0	3
A_4	X	X	X	0

$K[1,4] = 3$

$(A_1 \quad A_2 \quad A_3) \quad A_4$

$K[1,3] = 2$

$((A_1 \quad A_2) \quad A_3) \quad A_4$

Ex:3

K	A_1	A_2	A_3	A_4
A_1	0	1	1	2
A_2	X	0	1	2
A_3	X	X	0	2
A_4	X	X	X	0

$K[1,4] = 2$

$A_1 \quad A_2 \quad A_3 \quad A_4$

$(A_1 \quad A_2)(A_3 \quad A_4)$

Travelling Salesman Problem

BYJU'S EXAM PREP

MCM Parenthesizing

We will use table s to construct an optimal solution.

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

Table S

PRINT-OPTIMAL-PARENS (s, i, j)

1. if i=j
2. then print "A"
3. else print "("
4. PRINT-OPTIMAL-PARENS (s, i, s [i, j])
5. PRINT-OPTIMAL-PARENS (s, s [i, j] + 1, j)
6. print ")"

Pop(S, 1, 4) i=1 j=4

if (1=4) False, else Pop(S, 1, s[1,4]) = Pop(S, 1, 3)

Pop(S, 1, 3) = i=1 j=3 1==3 False

Pop(S, 1, 2) i=1, j=2

Pop(S, 1, 1) i=j

Pop(S, 3, 3) i=j

Pop(S, 2, 2)

(((A1 A2) A3) A4)

BYJU'S EXAM PREP

Travelling Salesman Problem

: minimization Problem
- Greedy
- DP
- Branch and Bound.

- The traveling salesman's problems abide by a salesman and a set of cities. The salesman has to visit every city starting from a certain one (e.g., the hometown) and return to the same city. The challenge of the problem is that the traveling salesman needs to minimize the total length of the trip.
- Suppose the cities are x1 x2..... xn where cost cij denotes the cost of traveling from city xi to xj. The traveling salesperson's problem is to find a route starting and ending at x1 that will take in all cities at the minimum cost.

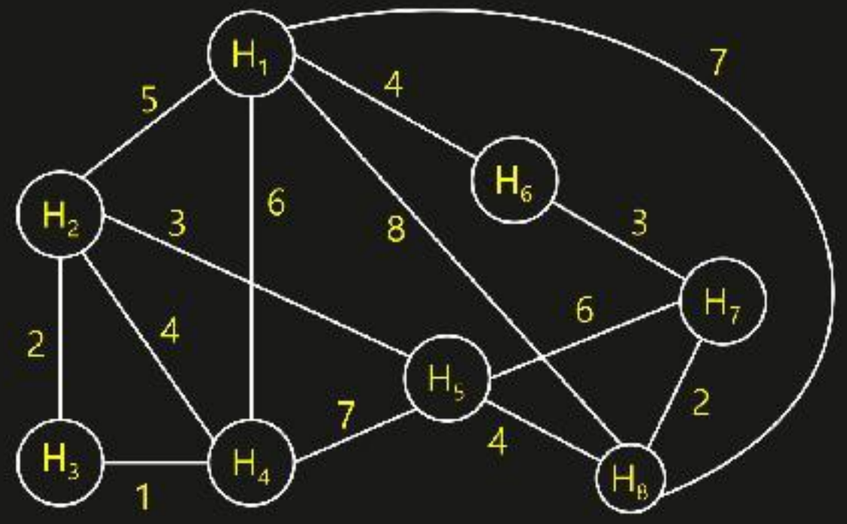
Travelling Salesman Problem

Using Greedy Method.

- Example: A newspaper agent daily drops the newspaper to the area assigned in such a manner that he has to cover all the houses in the respective area with minimum travel cost. Compute the minimum travel cost.

return back to source.

- The area assigned to the agent where he has to drop the newspaper is shown in fig:



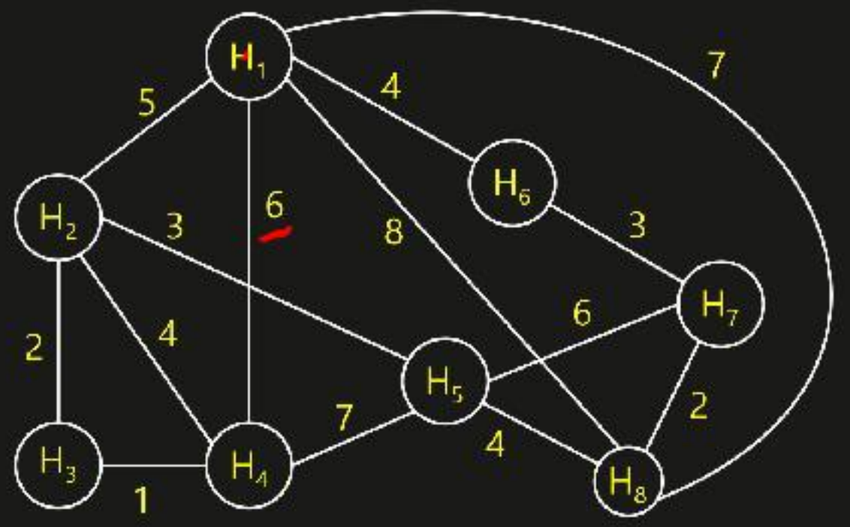
Graph

BYJU'S EXAM PREP

if No Direct Edge : 0 | ∞ | –

Travelling Salesman Problem

- Solution: The cost- adjacency matrix of graph G is as follows:
- $cost_{ij}$ =



Graph

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- The tour starts from area H1 and then select the minimum cost area reachable from H1.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H6 because it is the minimum cost area reachable from H1 and then select minimum cost area reachable from H6.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H7 because it is the minimum cost area reachable from H6 and then select minimum cost area reachable from H7.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H8 because it is the minimum cost area reachable from H7 and then select the minimum cost area reachable from H8.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H5 because it is the minimum cost area reachable from H8 and then select the minimum cost area reachable from H5.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H2 because it is the minimum cost area reachable from H5 and then select the minimum cost area reachable from H2.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

BYJU'S EXAM PREP

Travelling Salesman Problem

- Mark area H3 because it is the minimum cost area reachable from H2 and then select the minimum cost area reachable from H3.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
H_1	0	5	0	6	0	4	0	7
H_2	5	0	2	4	3	0	0	0
H_3	0	2	0	1	0	0	0	0
H_4	6	4	1	0	7	0	0	0
H_5	0	3	0	7	0	0	6	4
H_6	4	0	0	0	0	0	3	0
H_7	0	0	0	0	6	3	0	2
H_8	7	0	0	0	4	0	2	0

Cost adjacency matrix

selected [ ] 1 2

min {(1,2)(1,3)(1,4)}

Travelling Salesman Problem

Mark area H4 and then select the minimum cost area reachable from H4 it is H1.So, using the greedy strategy, we get the following.

i \ j	1	2	3	4
1	0	1	4	2
2	2	0	5	3
3			0	
4				0

4 3 2 4 3 2 1 6

H1 → H6 → H7 → H8 → H5 → H2 → H3 → H4 → H1.

Thus the minimum travel cost = 4 + 3 + 2 + 4 + 3 + 2 + 1 + 6 = 25

Node 1 is already marked and as there are unmarked Nodes still, Compare them only.

	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8
1 → H_1	0	5	∞	6	∞	4	∞	7
H_2		0						
H_3			0					
H_4				0				
H_5					0			
2 H_6	4	∞	∞	∞	∞	0	3	∞
3 H_7	∞	∞	∞	∞	6	3	0	2
4 H_8	7	∞	∞	∞	4	∞	2 ×	0

Cost = 0

Cost 0 + 4

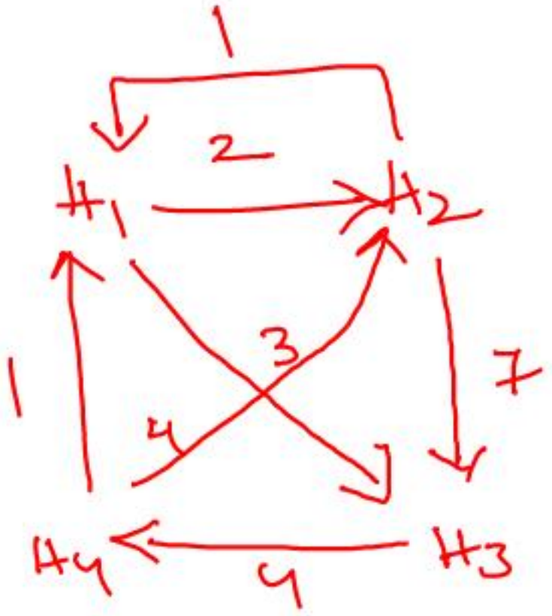
Cost = 4 + 3
= 7

Cost = 7 + 2
= 9

Cost = 9 + 4
= 13

- - -

	H_1	H_2	H_3	H_4
H_1	0	2	3	5
H_2	1	0	7	9
H_3	5	3	0	2
H_4	1	4	9	0



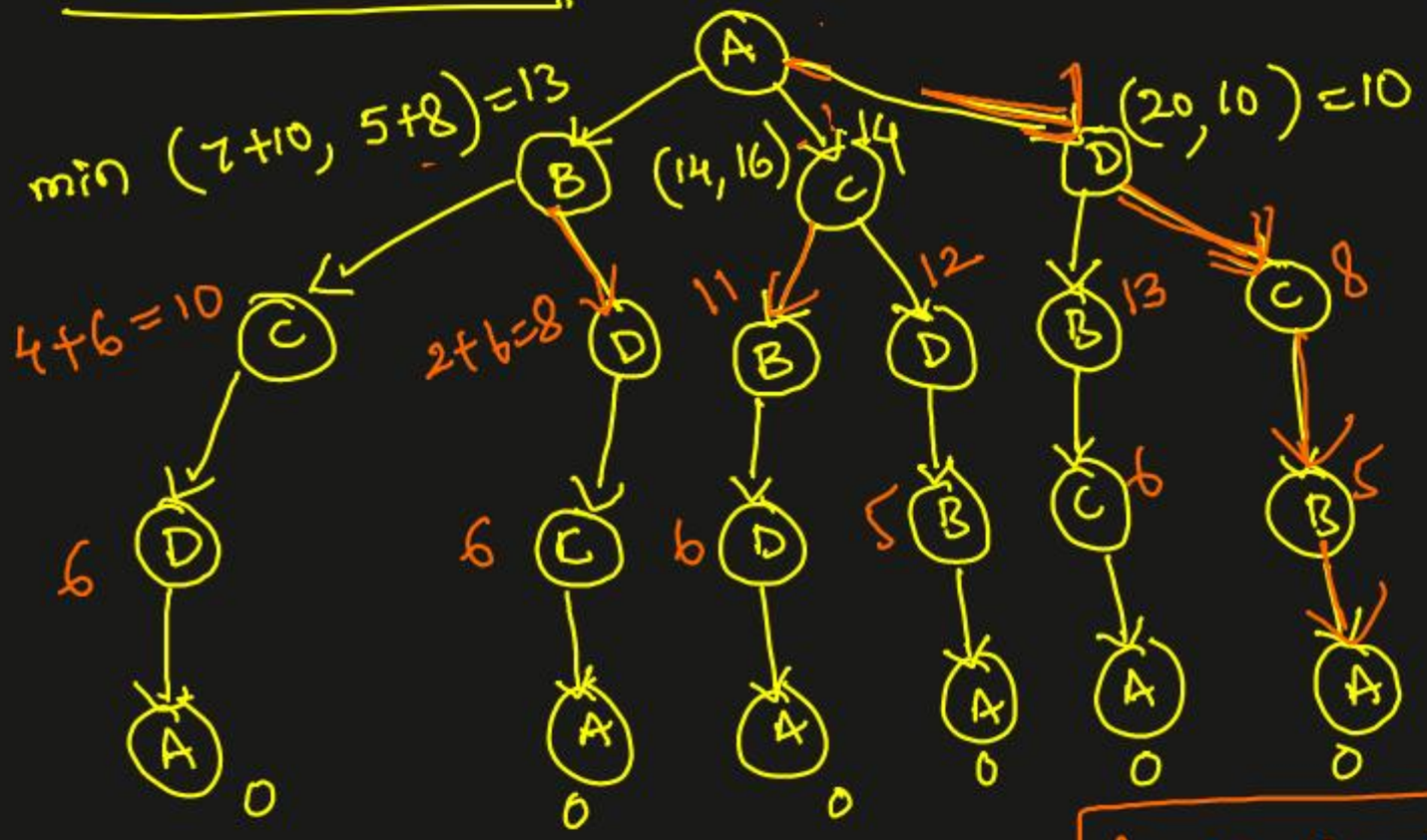
Greedy solution $H_1 \rightarrow H_2 \rightarrow H_3 \rightarrow H_4 \rightarrow H_1 = 2+7+2+1 = 12$

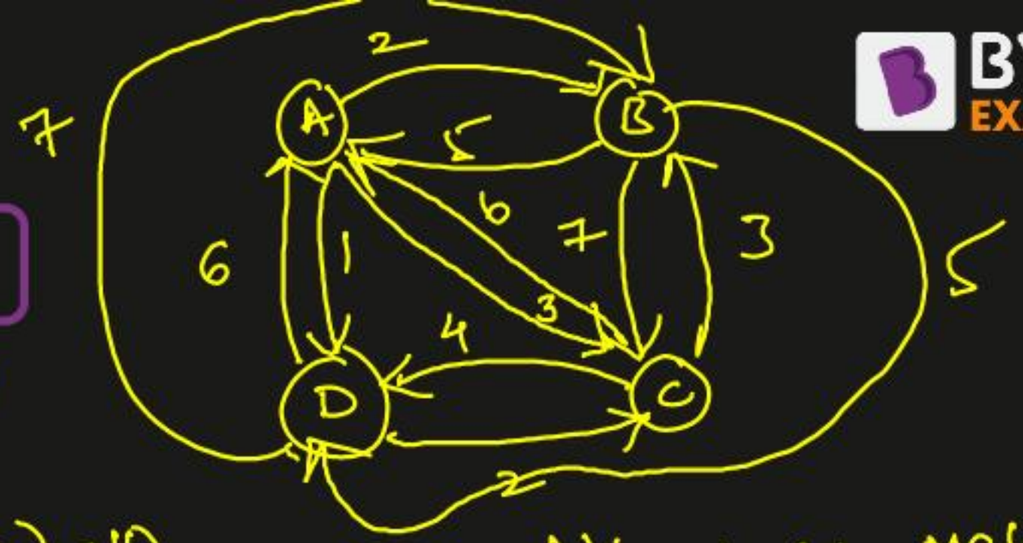
Optimal solution $\Rightarrow$ $H_1 \rightarrow H_3 \rightarrow H_4 \rightarrow H_2 \rightarrow H_1 = 3+2+4+1 = 10$

Travelling Salesman Problem

Travelling Salesman Problem using DP

Brute-Force approach : Let Source vertex = A





Adjacency Matrix

	A	B	C	D
A	0	2	3	1
B	5	0	7	5
C	6	3	0	4
D	6	7	2	0

$\boxed{A - D - C - B - A}$

BYJU'S EXAM PREP

Travelling Salesman Problem using DP

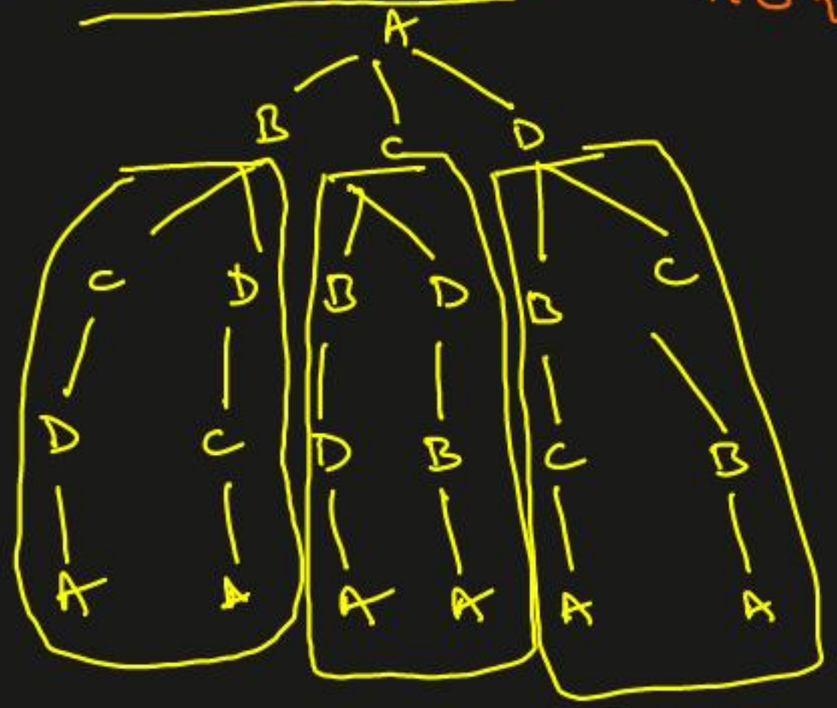
$$\text{Cost}(i, V) = \min_{k \in V} \left\{ \text{Cost}\ c_{ik} + \text{Cost}(k, V-\{k\}) \right\}$$

$i = A,\ V = \{B, C, D\}$

$$\min \begin{cases} k=C & c_{BC} + \{C, D\} \\ k=D & c_{BD} + \{B, C\} \end{cases}$$

$$\rightarrow \text{Cost}(A, \{B, C, D\}) = \min_{k \in \{B, C, D\}} \begin{cases} k=B & c_{AB} + \text{Cost}(\overset{i}{B}, \overset{V}{\{C, D\}}) \\ k=C & c_{AC} + \text{Cost}(C, \{B, D\}) \\ k=D & c_{AD} + \text{Cost}(D, \{B, C\}) \end{cases}$$

Cost(C, C)
Cost(B, D)
Cost(C, D)



$$\min \begin{cases} 2+13 & k=B \\ 3+14 & k=C \\ 1+10 & k=D \end{cases}$$

$\min = \{15, 17, 11\} = 11$

Cost(A, B)
Cost(A, C) — B, D
Cost(A, D) — B, C
Cost(

$$\boxed{\text{Time complexity of TSP} = n^2 \cdot 2^n}$$

n is No. of Nodes to be visited.

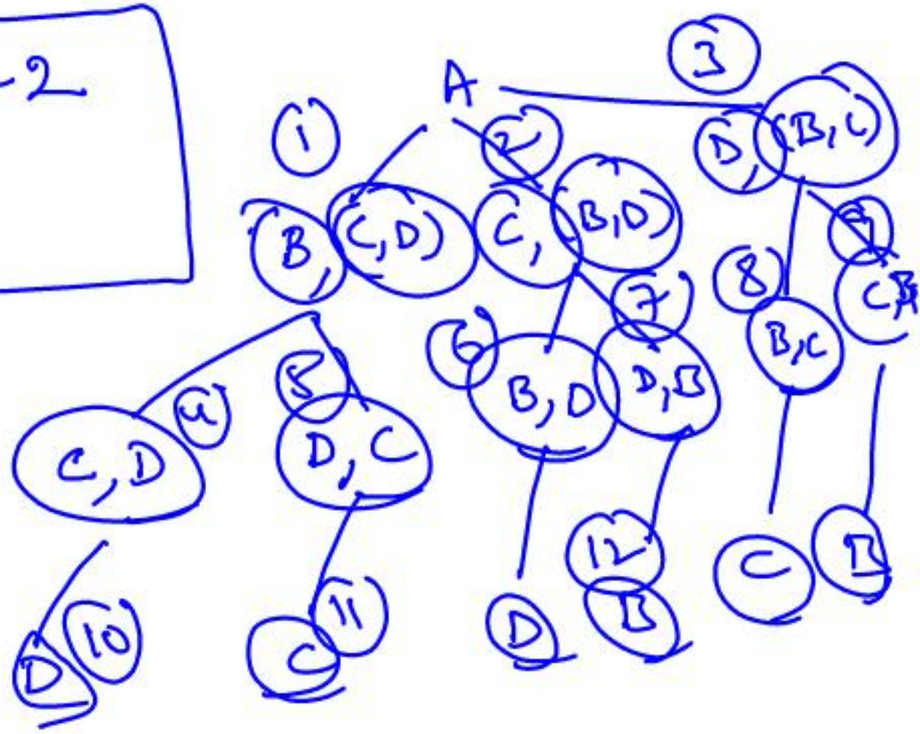
$$\boxed{\text{No. of unique sub problems} = (n-1)\, 2^{n-2}}$$

$n = 4$

$\Rightarrow 3 \cdot 2^{4-2}$

$\Rightarrow 3 \cdot 2^{2}$

$\Rightarrow \boxed{12}$

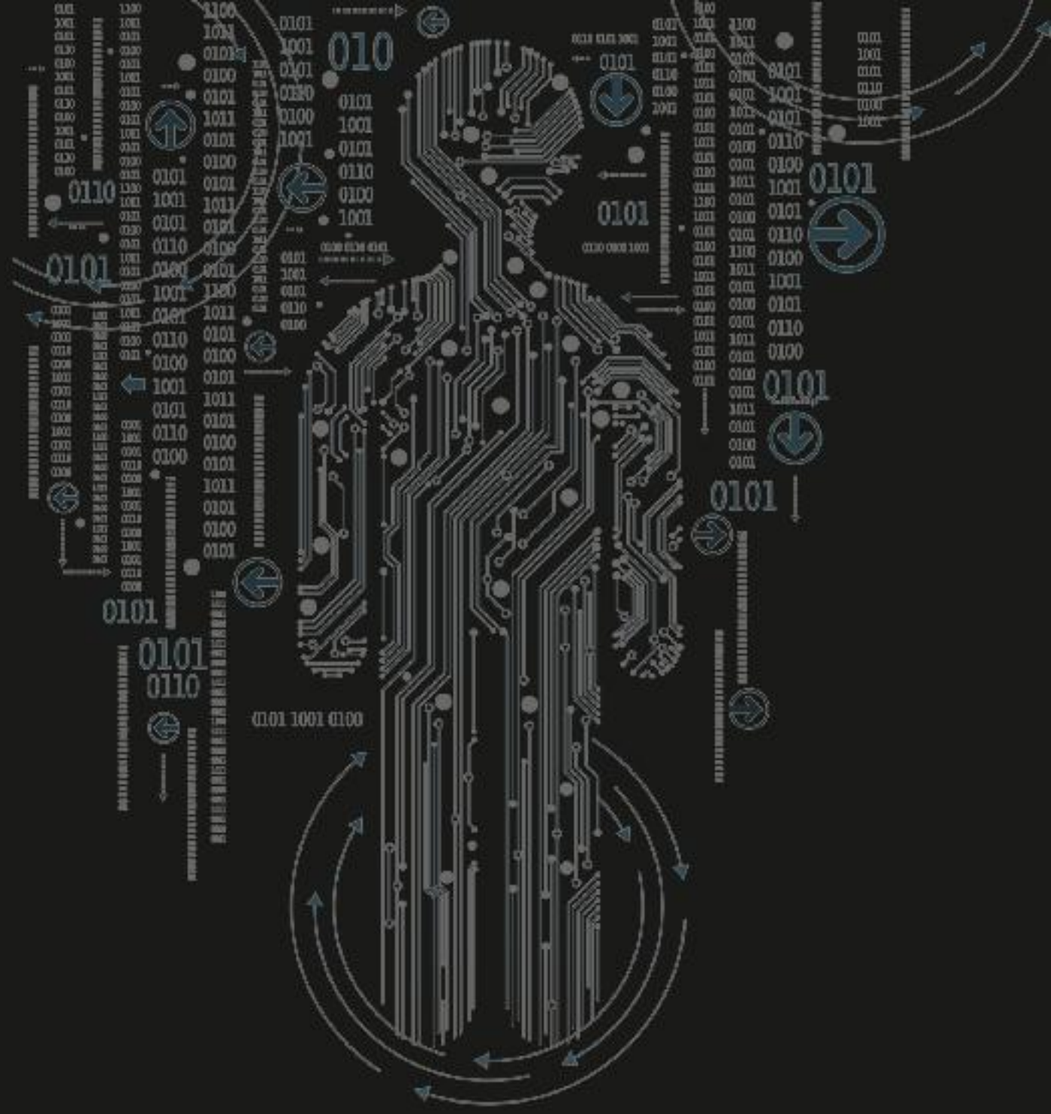


BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Bellman-Ford Algorithm



Guidelines to Attend Live Class

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Travelling Salesman Problem (minimization)

- Greedy
- DP
- Branch & Bound

DP Solution: $C[i, \{V\}] = \min_{k \in \{V\}} \left[c_{ik} + C[k, \{V\} - k] \right]$

Time complexity : $O(n^2 \cdot 2^n)$

No. of unique Sub Problems : $(n-1)\, 2^{n-2}$

BYJU'S EXAM PREP

Single Source Shortest Path Problem

- Single-source means that we need to find the shortest path from the given source to all the nodes.
- There are two algorithms to find single source shortest path:
  - Dijkstra's Algorithm [Greedy-based algo]
  - Bellman-Ford Algorithm [DP-based algo]

BYJU'S EXAM PREP

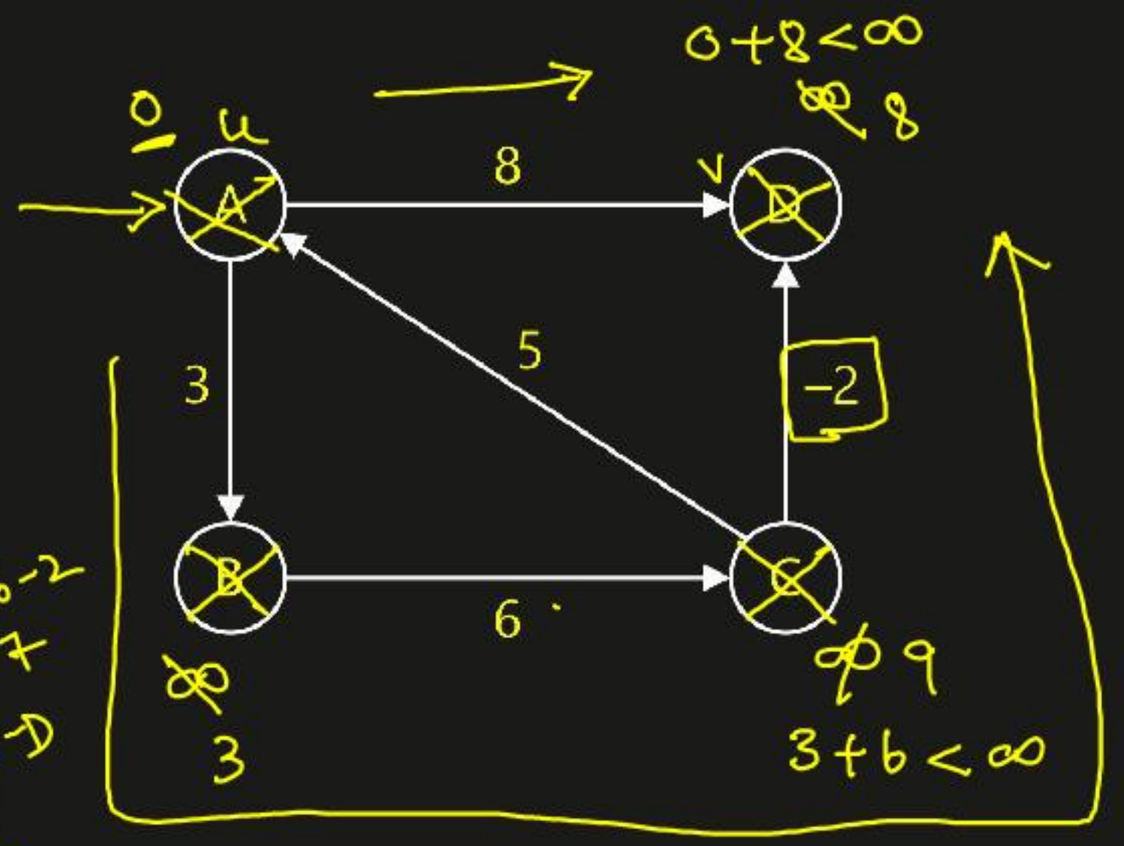
Drawback of Dijkstra's Algorithm

if(c[u] + d{u, v} < c[v])

c[v] = c[u] + d{u, v}

Dijkstra's Algorithm may not find optimum or correct solution for graphs with negative weight.

Example:



Vertex selection

A, B, D, C

A – D shortest cost = 3 + 6 − 2 = 7

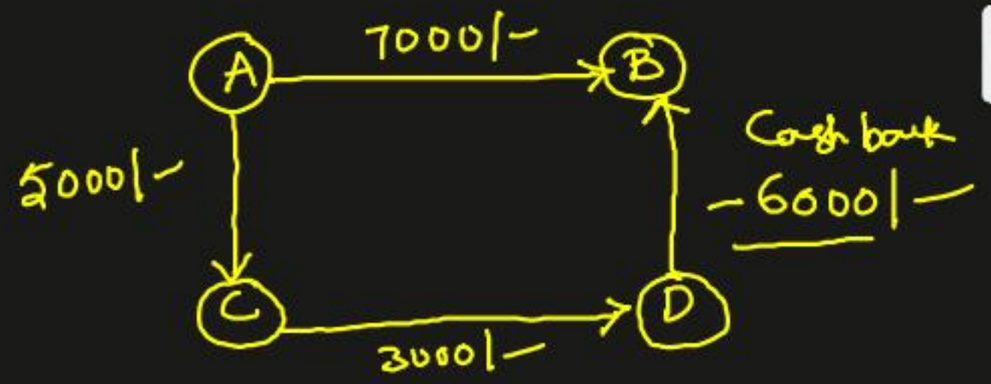
But Dijkstra gives A–D = 8

A

A	0
B	3
C	9
D	8

Bellman-Ford Algorithm

Why Negative Weight ?



- Negative weight edges might seem useless at first but they can explain a lot of phenomena like cashflow, the heat released/absorbed in a chemical reaction, etc.
- For instance, if there are different ways to reach from one chemical A to another chemical B, each method will have sub-reactions involving both heat dissipation and absorption.
- If we want to find the set of reactions where minimum energy is required, then we will need to be able to factor in the heat absorption as negative weights and heat dissipation as positive weights.

BYJU'S EXAM PREP

Bellman – Ford Algorithm

- Bellman Ford algorithm helps us find the shortest path from a vertex to all other vertices of a weighted graph.
- It is similar to Dijkstra's algorithm, but it can work with graphs in which edges can have negative weights.

BYJU'S EXAM PREP

Bellman – Ford Algorithm

- Bellman Ford algorithm works by overestimating the length of the path from the starting vertex to all other vertices.
- Then it iteratively relaxes those estimates by finding new paths that are shorter than the previously overestimated paths.
- By doing this repeatedly for all vertices, we can guarantee that the result is optimized.

Bellman-Ford Algorithm

Bellman – Ford Algorithm

Relax all Edges repetitively by (n-1) times for n vertices graph.

Step 1: Start with the weighted graph

$n = 5$

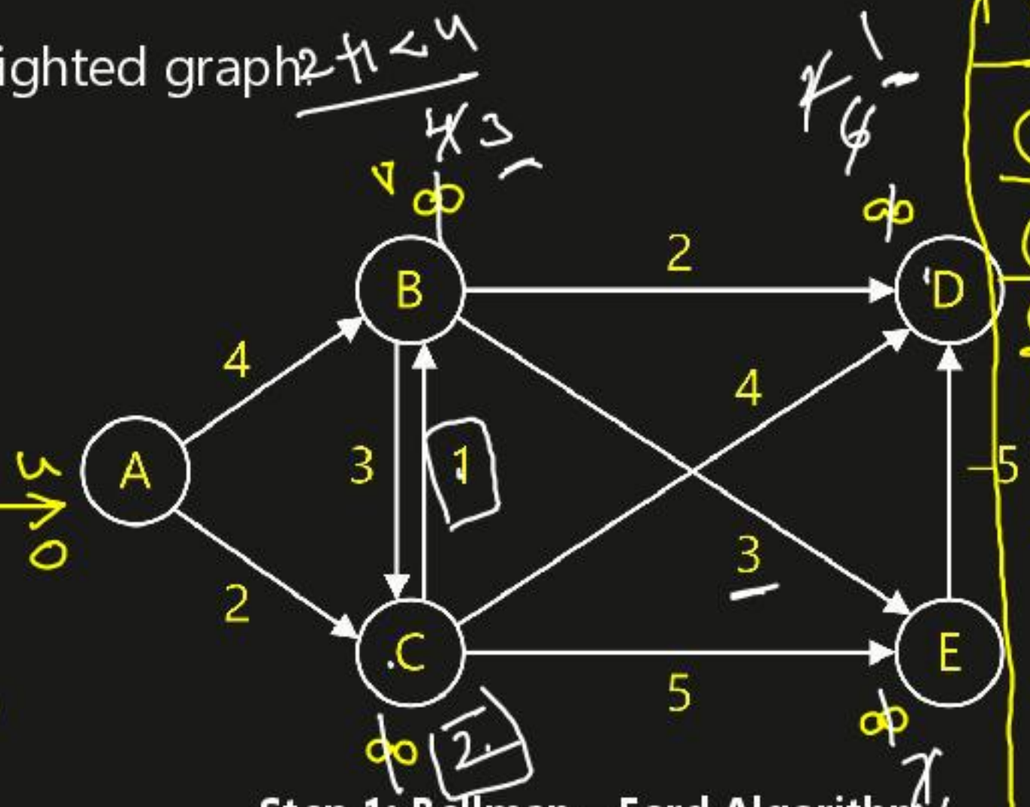
$\Rightarrow$ Relax all Edges

$(n-1) = 4$ times

if

$c[u] + d(u,v) < c[v]$

$c[v] = c[u] + d[u,v]$

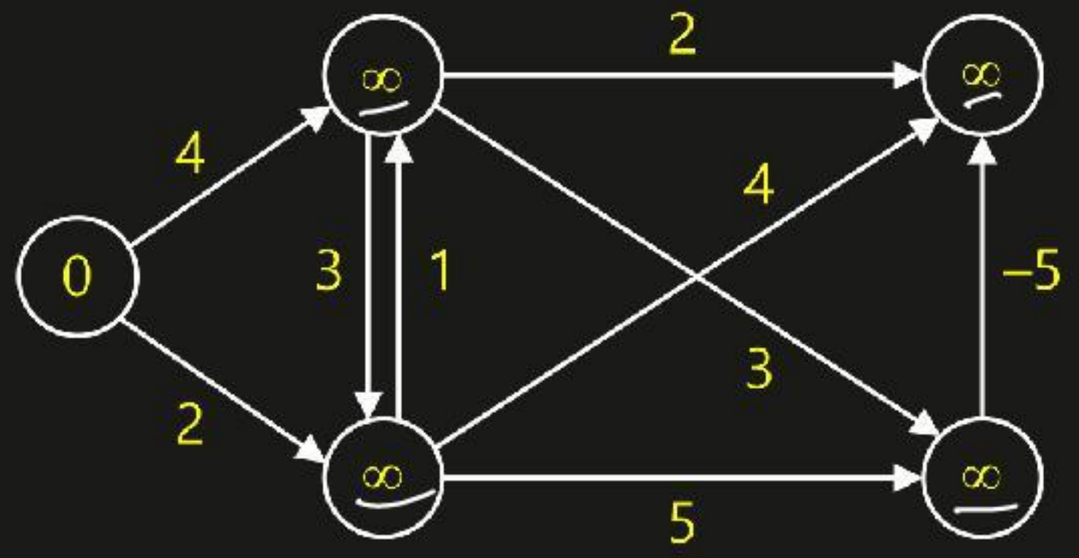


Step 1: Bellman – Ford Algorithm

Edge	1st	2nd	3rd	4th
(A, B)	4	–	–	–
(A, C)	2	–	–	–
(B, D)	6	–	–	–
(B, E)	7	~~7~~ 6	–	–
(B, C)	2	–	–	–
(C, B)	~~4~~ 3	–	–	–
(C, D)	–	–	–	–
(C, E)	–	–	–	–
(E, D)	~~6~~ 2	~~2~~ 1	–	–

Bellman – Ford Algorithm

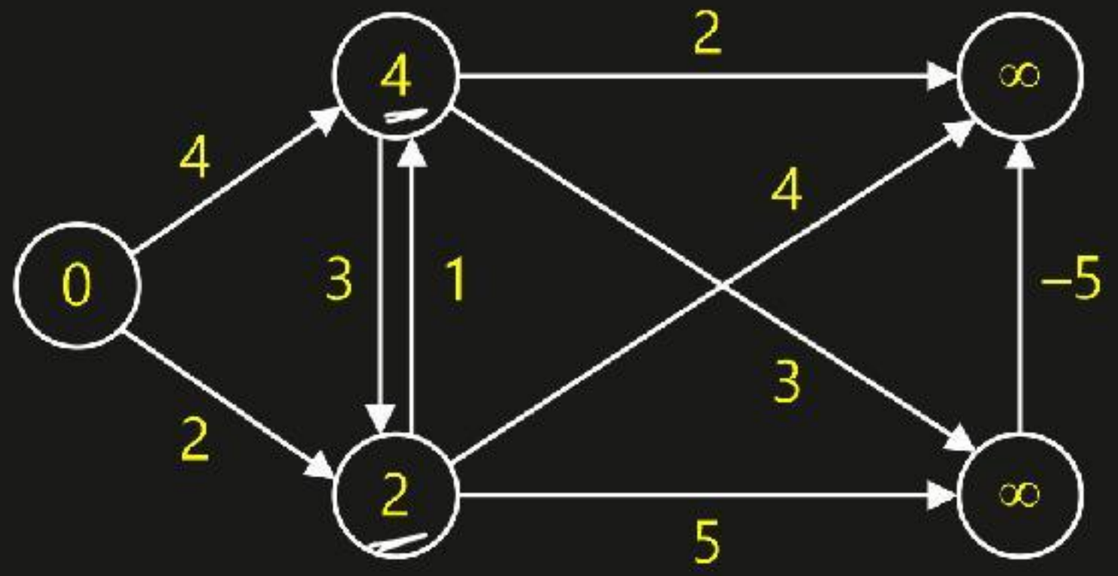
Step 2: Choose a starting vertex and assign infinity path values of all other vertices.



Step 2: Bellman – Ford Algorithm

Bellman – Ford Algorithm

Step 3: Visit each edge and relax the path distances if they are inaccurate.



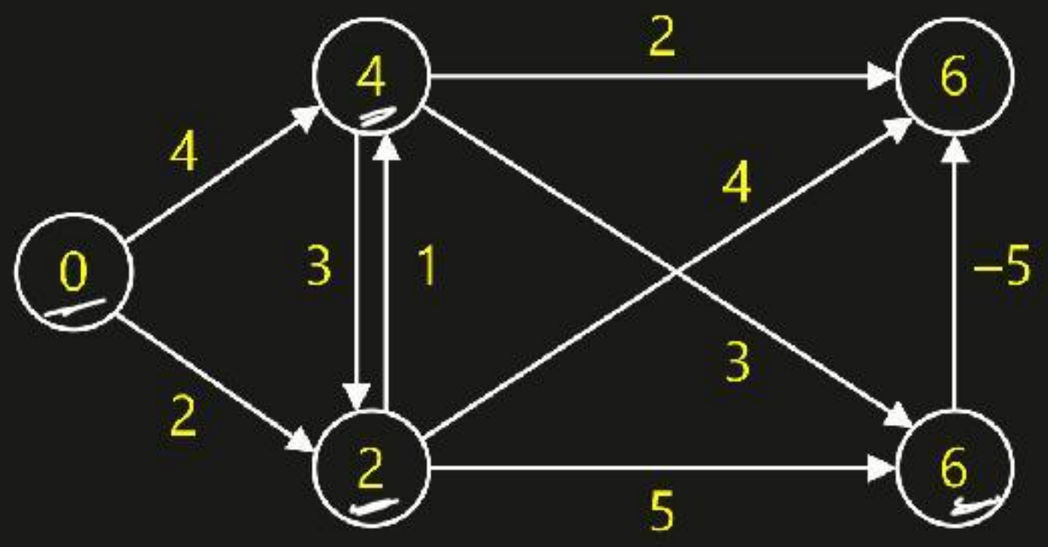
Step 3: Bellman – Ford Algorithm

Bellman – Ford Algorithm

(N-1) times

Step 4: We need to do this ~~V~~ times because in the worst case, a vertex's path length might need to be readjusted ~~V~~ times.

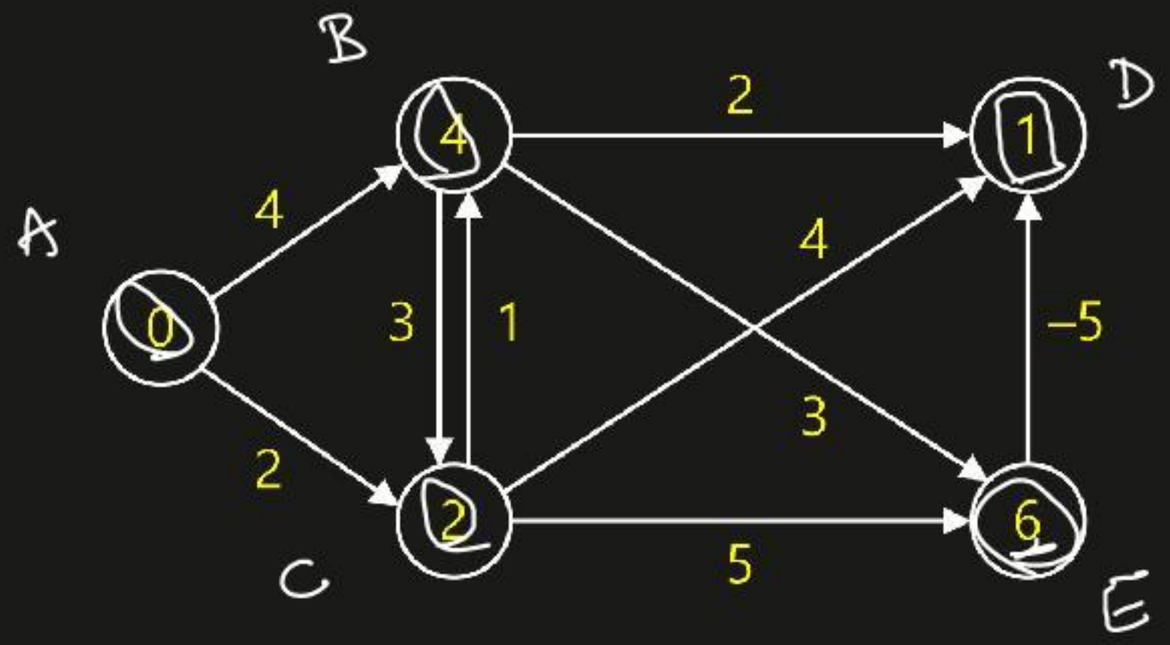
(V-1)



Step 4: Bellman – Ford Algorithm

Bellman – Ford Algorithm

Step 5: Notice how the vertex at the top right corner had its path length adjusted.



Step 5: Bellman – Ford Algorithm

BYJU'S EXAM PREP

Bellman – Ford Algorithm

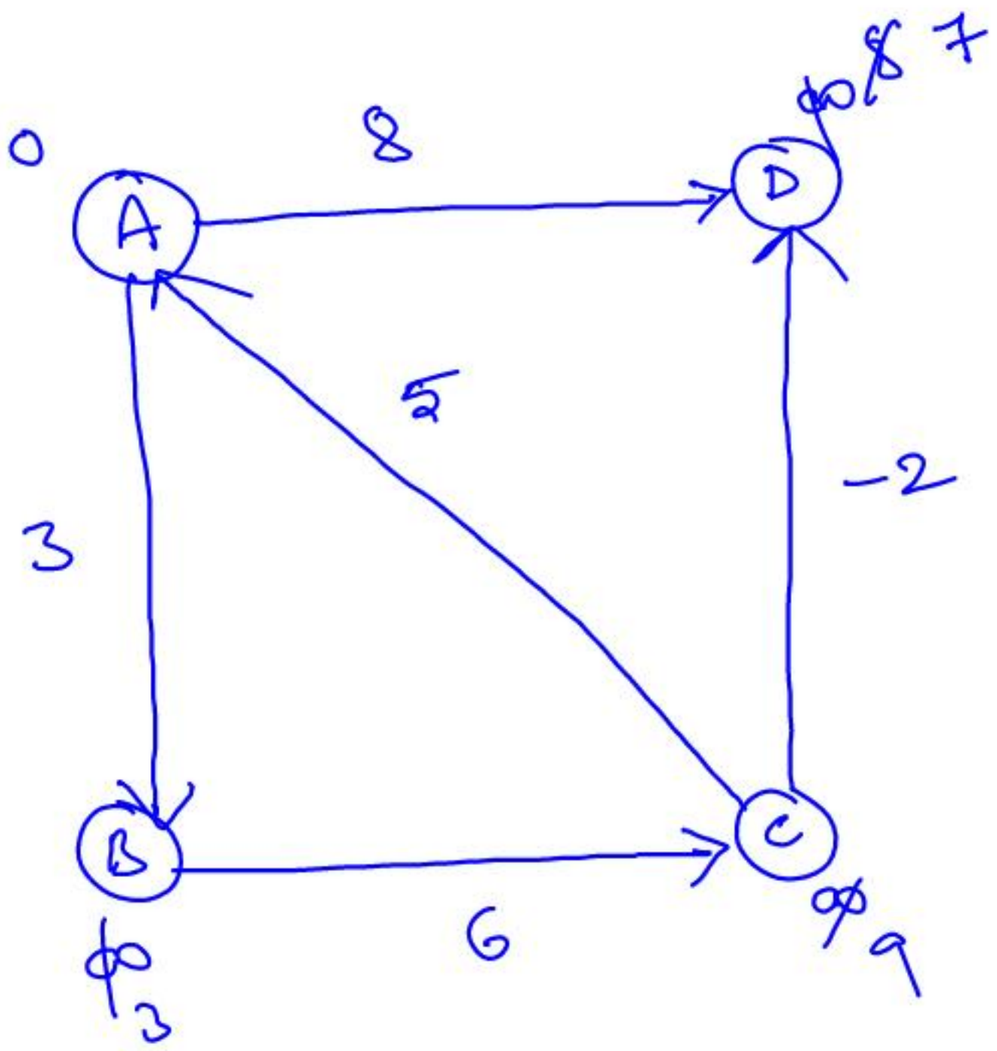
Step 6: After all the vertices have their path lengths, we check if a negative cycle is present.

A	B	C	D	E
0	∞	∞	∞	∞
0	4	2	∞	∞
0	3	2	6	6
0	3	2	1	6
0	3	2	1	6

Shortest Paths.

Step 6: Bellman – Ford Algorithm

BYJU'S EXAM PREP



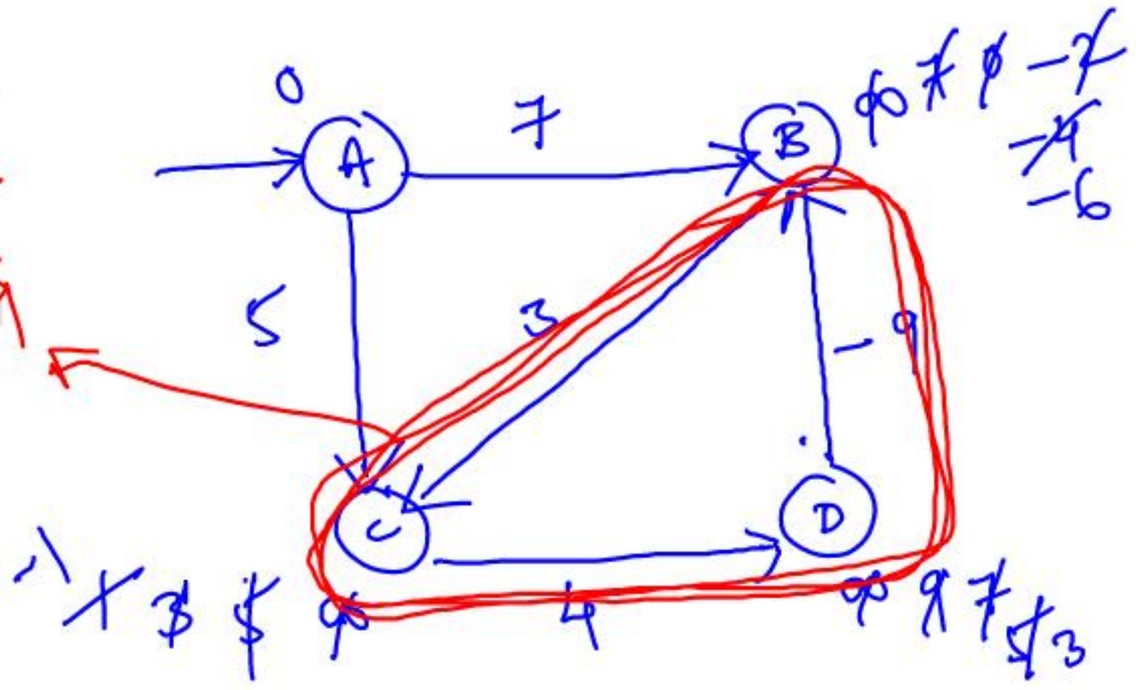
$n = 4$

relax all edges $(n-1) = 3$ times

Edge	1st	2nd	3rd
(A, B)	~~∞~~ 3	—	—
(A, D)	~~∞~~ 8	—	—
(B, C)	~~∞~~ 9	—	—
(C, D)	~~8~~ 7	—	—
(C, A)	—	—	—

Bellman-Ford algorithm gives optimal solution for graph with Edges of negative weight.

Drawback of Bellman-Ford algo



Cost of cycle: $3+4-9 = -2$

4 vertices ⇒ relax all Edges 3 times

Edge	1st	2nd	3rd	4th	5th
(A, B)	~~∞~~ 7	—	—	—	—
(A, C)	~~∞~~ 5	—	—	—	—
(B, C)	—	~~7~~ 3	~~7~~ 1	~~1~~ -1	~~-1~~ -3 ---
(C, D)	~~∞~~ 9	~~9~~ 7	~~7~~ 5	~~5~~ 3	~~3~~ 1 --
(D, B)	~~7~~ 0	~~0~~ -2	~~-2~~ -4	~~-4~~ -6	~~-6~~ -8 --

A – C – D – B – C – D – B – C – D – B

When There is cycle with negative weight → Bellman Ford algo cannot work. ⇒ NO solution.

Bellman-Ford Algorithm

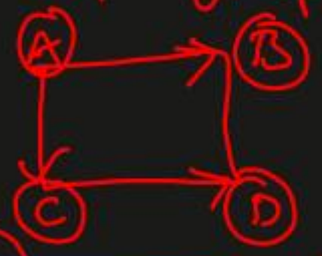
Bellman – Ford Pseudo Code

```
function bellmanFord(G, S)
  for each vertex V in G
    distance[V] <- infinite
      previous[V] <- NULL
       distance[S] <- 0

  for each vertex V in G
    for each edge (U,V) in G
      tempDistance <- distance[U] + edge_weight(U, V)
       if tempDistance < distance[V]
        distance[V] <- tempDistance
        previous[V] <- U

  for each edge (U,V) in G
    If distance[U] + edge_weight(U, V) < distance[V}
      Error: Negative Cycle Exists

  return distance[ ], previous[ ]
```

Time complexity: $E * (V-1)$ times

$\Rightarrow O(E \cdot V)$

A simple graph with 'n' vertices, 'n' edges



$\Rightarrow V = n, E = n$

$T.C = O(n \cdot n) = O(n^2)$

Relaxing all edges $(V-1)$ times

if further relaxation happens that indicates, cycle with negative weight.

For Complete graph : Every vertex is directly connected to Every other Vertex.

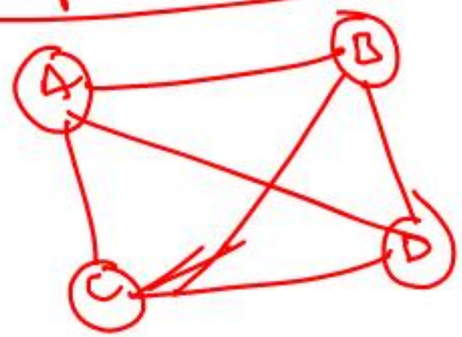
No. of vertices, $V = n$, No. of edges, $E = {}^{n}C_{2} = \frac{n(n-1)}{2}$

$\Rightarrow$ Time Complexity $= O\left(n \cdot \frac{n(n-1)}{2}\right)$

$= O\left(n \cdot \left(\frac{n^2 - n}{2}\right)\right)$

$= O\left(\frac{n^3}{2}\right) - O\left(\frac{n^2}{2}\right) \Rightarrow O\left(\frac{n^3}{2}\right) = O(n^3)$

Complete graph

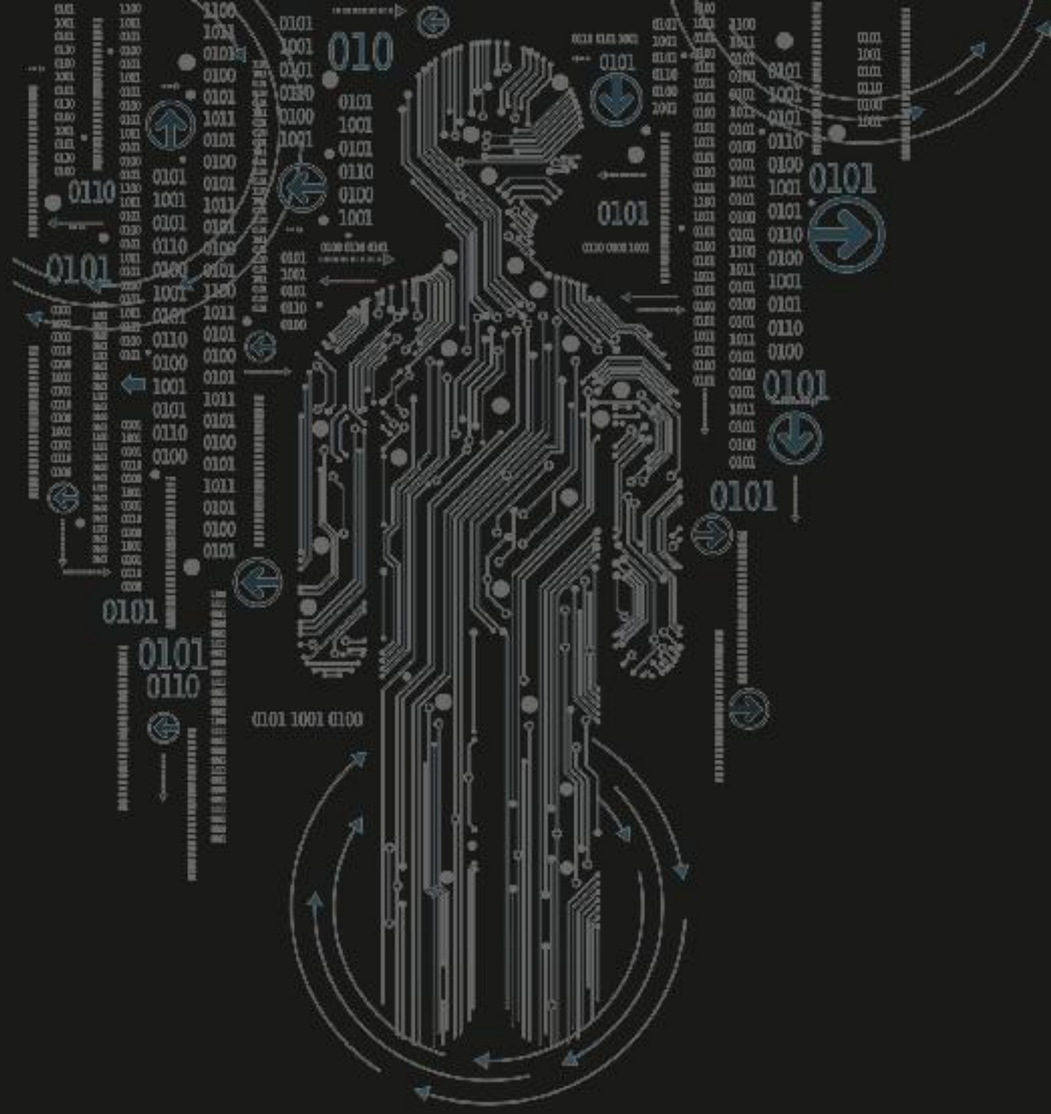


BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Knapsack Problem (0/1, Fractional)



Guidelines to Attend Live Class

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Bellman – Ford Algorithm

- Single source shortest Path Problem
  - Dijkstra
  - Bellman – Ford

Dijkstra algorithm – Greedy algorithm, may not find optimal solution for a graph with Negative weight Edges.

Bellman – Ford algorithm – works for a graph with Negative weight Edges but It cannot find solution for a graph with Negative weight Cycle.

→ relax all Edges $(n-1)$ times with 'n' vertices

Knapsack Problem (0/1, Fractional)

Knapsack Problem

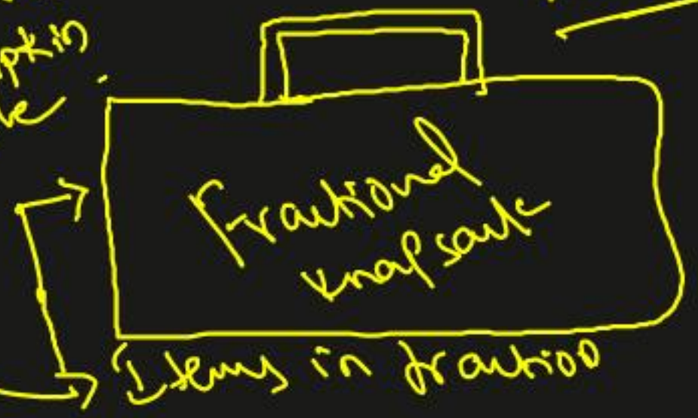
Refrigerator
T.V
A.C
Washing Machine → Select as Atomic or none
(1) (0)

0/1 Knapsack Problem.

- Here knapsack is like a container or a bag. Suppose we have given some items which have some weights or profits. We have to put some items in the knapsack in such a way total value produces a maximum profit.
- For example, the weight of the container is 20 kg. We have to select the items in such a way that the sum of the weight of items should be either smaller than or equal to the weight of the container, and the profit should be maximum.
- There are two types of knapsack problems:
  - 0/1 knapsack problem
  - Fractional knapsack problem

1 unit = 1 kg

- Lady finger
- Pumpkin
- Bottle
- Tomato
- Potato
- Onion
- Brinjal



Bag.
10 kg.

BYJU'S EXAM PREP

0/1 Knapsack Problem

- The 0/1 knapsack problem means that the items are either completely or no items are filled in a knapsack.
- For example, we have two items having weights 2kg and 3kg, respectively. If we pick the 2kg item then we cannot pick 1kg item from the 2kg item (item is not divisible); we have to pick the 2kg item completely.
- This is a 0/1 knapsack problem in which either we pick the item completely or we will pick that item.
- The 0/1 knapsack problem is solved by the dynamic programming.

Knapsack Problem (0/1, Fractional)

0/1 Knapsack Problem

⇒ Maximization Problem

Consider the problem having weights and profits are:

- Weights: {3, 4, 6, 5}
- Profits: {2, 3, 1, 4}
- The weight of the knapsack is 8 kg
- The number of items is 4

if 'n' items : 2^n combinations

⇒ Brute-Force T.C = $O(2^n)$

Without DP (or) Brute-Force approach

Items

I_1	I_2	I_3	I_4
1	1	1	1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
		⋮	

$2^4 = 16$

16 different Possibilities

Knapsack Problem (0/1, Fractional)

0/1 Knapsack Problem

$I_1\ I_2\ I_3\ I_4$

Weights: {3, 4, 6, 5}

$I_2\ I_1\ I_3\ I_4$

Profits: {2, 3, 1, 4}

1, 2, 3, 4

Profit Matrix → [n+1, n+1]

Knapsack Capacity

- First, we create a matrix shown as below:

No items

	~~Profit~~ / Weight	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0	0
I_3	1	0								
I_1	2	0								
I_2	3	0								
I_4	4	0								

- The first row and the first column would be 0.

column → Items in increasing order of Profits.

Knapsack Problem (0/1, Fractional)

W=8 { $I_1, I_2 = 2+3=5$; $I_1, I_4 = 2+4=6$ }

0/1 Knapsack Problem

Weights: {3, 4, 6, 5} Profits: {2, 3, 1, 4}

W=7 { I_1, I_2 } or I_3 or I_4

W=5 { $I_1 = 2$, $I_2 = 3$, $I_4 = 4$ }

- When i=1, W=1
- w1 = 3; Since we have only one item in the set having weight 3, but the capacity of the knapsack is 1. We cannot fill the item of 3kg in the knapsack of capacity 1 kg so add 0 at M[1][1] shown as below:

Knapsack weight

Profit / Weight	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
I_3 1, 6	0	0	0	0	0	0	1	1	1
I_1 2, 3	0	0	0	2 I_1	2	2	2	2	2
I_2 3, 4	0	0	0	2	3 I_2	3	3	5 I_1, I_2	5 I_1, I_2
I_4 4, 5	0	0	0	2	3	4 I_4	4	5 I_1, I_2	6 I_1, I_4

Knapsack Problem (0/1, Fractional)

After I_4 selection, remaining $W = 8 - 5 = 3$

I_1	I_2	I_3	I_4
1	0	0	1

0/1 Knapsack Problem

Weights: {3, 4, 6, 5} Profits: {2, 3, 1, 4}

$W = 5 \begin{cases} I_1 \Rightarrow 2 \\ I_2 \Rightarrow 3 \\ I_4 \Rightarrow 4 \end{cases}$ $W = 7 \begin{cases} I_1, I_2 = 5 \\ I_4 = 4 \end{cases}$ $W = 8 \begin{cases} I_1, I_2 = 5 \\ I_1, I_4 = 6 \end{cases}$

- When i = 1, W = 2
- w1 = 3; Since we have only one item in the set having weight 3, but the capacity of the knapsack is 2. We cannot fill the item of 3kg in the knapsack of capacity 2 kg so add 0 at M[1][2] shown as below:

Items in the order of weight (Follow)

Profit	Profit / Weight	0	1	2	3	4	5	6	7	8
	0	0	0	0	0	0	0	0	0	0
2	$I_1 = 3$	0	0	0	2	2	2	2	2	2
3	$I_2 = 4$	0	0	0	2	3	3	3	5	5
4	$I_4 = 5$	0	0	0	2	3	4	4	5	6
1	$I_3 = 6$	0	0	0	2	3	4	4	5	6

BYJU'S EXAM PREP

0/1 Knapsack Problem

Weights: {3, 4, 6, 5} Profits: {2, 3, 1, 4}

- When i=1, W=3
- w1 = 3; Since we have only one item in the set having weight equal to 3, and weight of the knapsack is also 3; therefore, we can fill the knapsack with an item of weight equal to 3. We put profit corresponding to the weight 3, i.e., 2 at M[1][3] shown as below:

Profit / Weight	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	2					
2	0								
3	0								
4	0								

BYJU'S EXAM PREP

0/1 Knapsack Problem

Weights: {3, 4, 6, 5} Profits: {2, 3, 1, 4}

- Repeat this procedure for all matrix:

Profit / Weight	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
1	0	0	0	2					
2	0								
3	0								
4	0								

Knapsack Problem (0/1, Fractional)

0/1 Knapsack Problem

Weights: {3, 4, 6, 5}

Profits: {2, 3, 1, 4}

W-Wi

5-5=0

- Final table would be as follows:

W

Profit / Weight	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
I 1	0	0	0	2	2	2	2	2	2
I 2	0	0	0	2	3	3	3	5	5
I 3 = 5	0	0	0	2	3	4	4	5	6
I 4	0	0	0	2	3	4	4	5	6

w_i

v_i

4

$w_i \le W$

$4 + v[1,0] = 4+0 > v[1,5]$, $4 > 3$ True

BYJU'S EXAM PREP

0/1 Knapsack Algorithm

KNAPSACK (n, W)

1. for w = 0, W
2. do V [0,w] ← 0
3. for i=0, n
4. do V [i, 0] ← 0
5. for w = 0, W
6. do if (wi≤ W & vi + V [i-1, W - wi]> V [i -1,W])
7. then V [i, w] ← vi + V [i - 1, W - wi]
8. else V [i, w] ← V [i - 1, w]

Time Complexity = $O(n \cdot W)$

BYJU'S EXAM PREP

Fractional Knapsack Problem

- The fractional knapsack problem means that we can divide the item.
- For example, we have an item of 3 kg then we can pick the item of 2 kg and leave the item of 1 kg.
- The fractional knapsack problem is solved by the Greedy approach.

BYJU'S EXAM PREP

Fractional Knapsack Problem

According to the problem statement,

- There are n items in the store
- Weight of ith item $w_i > 0$
- Profit for i^{th} item $p_i > 0$ and Capacity of the Knapsack is W
- In this version of Knapsack problem, items can be broken into smaller pieces. So, $0 \leqslant x_i \leqslant 1$
- The ith item contributes the weight $x_i.w_i$ to the total weight in the knapsack and profit $x_i.p_i$ to the total profit.
- Hence, the objective of this algorithm is to maximize $\sum (x_i.p_i)$ subject to constraint, $\sum_n (x_i.w_i) \leqslant W$

BYJU'S EXAM PREP

Fractional Knapsack Problem - Example

Let us consider that the capacity of the knapsack W = 60 and the list of provided items are shown in the following table –

Item	A	B	C	D
Profit	280	100	120	120
Weight	40	10	20	24
P/W	7	10	6	5

$W = 60$.

Select items in descending order of P/W value

	A	B	C	D
Profit:	280	100	120	120
Weight:	40	10	20	24
P/W:	7	10	6	5

	Profit gained	Remaining Knapsack weight
Select B:	100	$60 - 10 = 50$
Select A:	$100 + 280 = 380$	$50 - 40 = 10$
Select $\frac{1}{2}$ C:	$380 + \frac{1}{2}(120)$ $= 440$	$10 - \frac{1}{2}(20) = 0$

Maximum Profit

BYJU'S EXAM PREP

Fractional Knapsack Algorithm

Algorithm:

```
Greedy-Fractional-Knapsack (w[1..n], p[1..n], W)
for i = 1 to n
do x[i] = 0
weight = 0
for i = 1 to n
if weight + w[i] ≤ W then
x[i] = 1
weight = weight + w[i]
else
x[i] = (W - weight) / w[i]
weight = W
break
return x
```

$O(n \cdot \log n)$

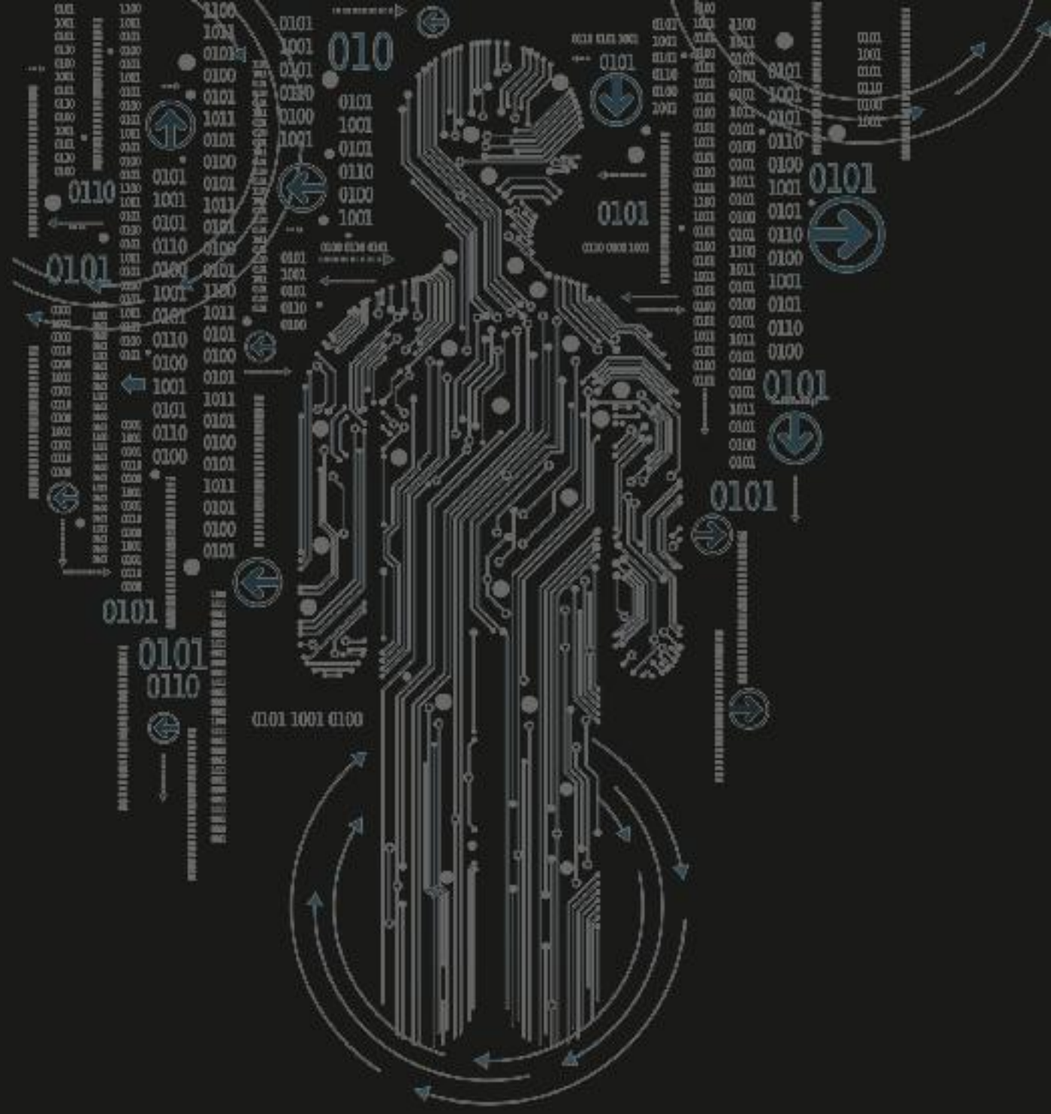
n = no. of items

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

All Pair Shortest Path Problem



Guidelines to Attend Live Class

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

Knapsack Problem : Given knapsack of weight, W and items $I_1, I_2, \ldots I_n$ with weights $W_1, W_2, \ldots W_n$ and Profits $P_1, P_2, \ldots P_n$.

Select items
- as a whole – 0/1-DP
- In fraction also – fractional Knapsack – Greedy based

Item quantity should not exceed 1.

Max Profit,

$\sum w_i <= W$

All Pair Shortest Path Problem

Problem Statement

- Given a graph G with set of vertices V and edges E.
- The Objective is to find shortest path or cost between every pair of vertices. Hence it is said to be All Pair Shortest Path Problem.
- There are two popular algorithms for this:
  - Floyd – Warshall Algorithm
  - Johnson's Algorithm

BYJU'S EXAM PREP

Floyd – Warshall Algorithm

- Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph.
- Floyd-Warshall algorithm is also called as Floyd's algorithm, Roy-Floyd algorithm, Roy-Warshall algorithm.
- This algorithm follows the dynamic programming approach to find the shortest paths.
- This algorithm works for both the directed and undirected weighted graphs. But it does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

BYJU'S EXAM PREP

Floyd – Warshall Algorithm

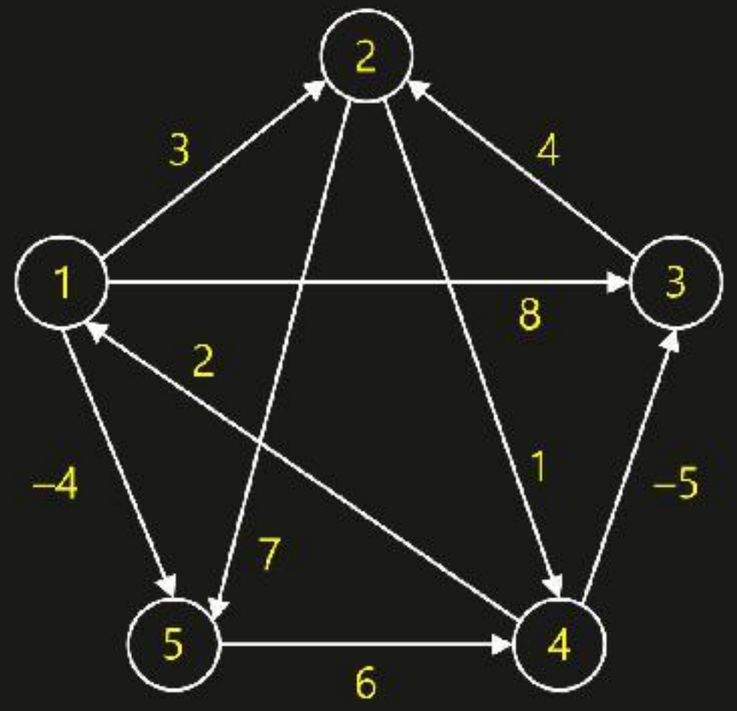
- A recursive definition is given by

Base Case

$$d_{ij}^{(k)} = \begin{cases} W_{ij} & \text{if } k = 0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 \end{cases}$$

All Pair Shortest Path Problem

- Apply Floyd-Warshall algorithm for constructing the shortest path. Show that matrices D(k) and π(k) computed by the Floyd-Warshall algorithm for the graph.



Graph

BYJU'S EXAM PREP

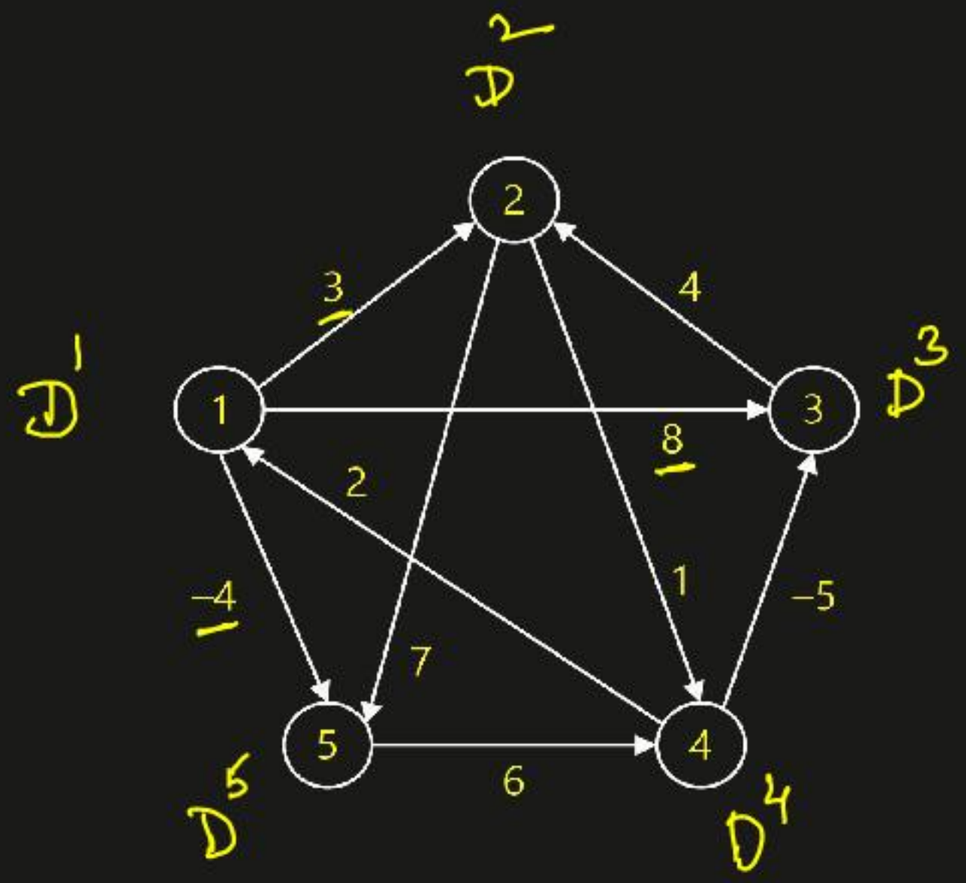
All Pair Shortest Path Problem

$$d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$$

- Step (i) When k = 0

Initial Matrix

$$D^{(0)} = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 8 & \infty & -4 \\ 2 & \infty & 0 & \infty & 1 & 7 \\ 3 & \infty & 4 & 0 & -5 & \infty \\ 4 & 2 & \infty & \infty & 0 & \infty \\ 5 & \infty & \infty & \infty & 6 & 0 \end{array}$$

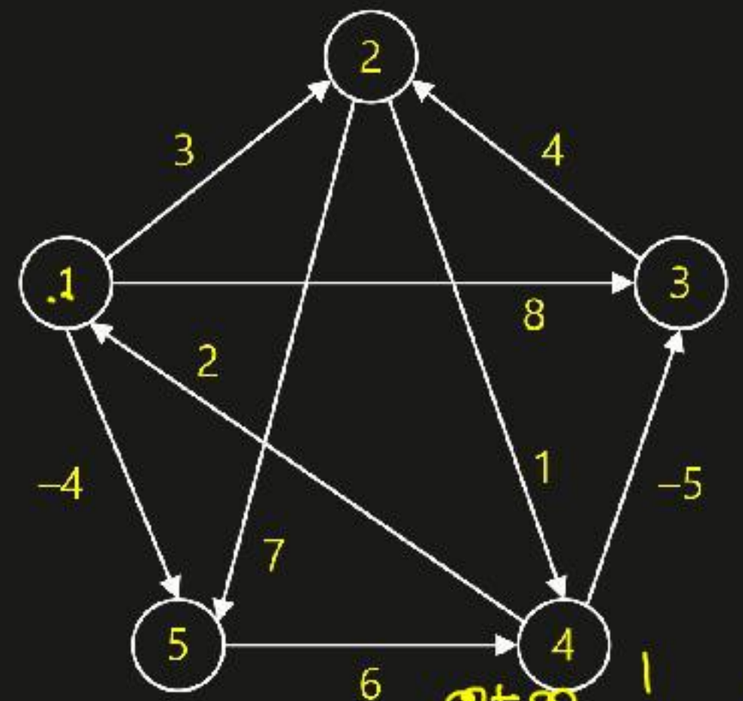


BYJU'S EXAM PREP

d_{14}^{0}

All Pair Shortest Path Problem

- Step (ii) When k = 1 D^1



d_{24}^{1} $\min(d_{21}+d_{14}, d_{24}^{0})$ $\infty+\infty$ 1

$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$$d_{14}^{(1)} = \min\left(d_{14}^{(0)}, d_{11}^{(0)} + d_{14}^{(0)}\right)$$

$$d_{14}^{(1)} = \min(\infty, 0+\infty) = \infty$$

$$d_{15}^{(1)} = \min\left(d_{15}^{(0)}, d_{11}^{(0)} + d_{15}^{(0)}\right)$$

$$d_{15}^{(1)} = \min(-4, 0-4) = -4$$

$$d_{21}^{(1)} = \min\left(d_{21}^{(0)}, d_{21}^{(0)} + d_{11}^{(0)}\right)$$

$$d_{21}^{(1)} = \min(\infty, \infty+0) = \infty$$

$$d_{23}^{(1)} = \min\left(d_{23}^{(0)}, d_{21}^{(0)} + d_{13}^{(0)}\right)$$

$$d_{23}^{(1)} = \min((\infty, \infty+8) = \infty$$

$i \quad j$

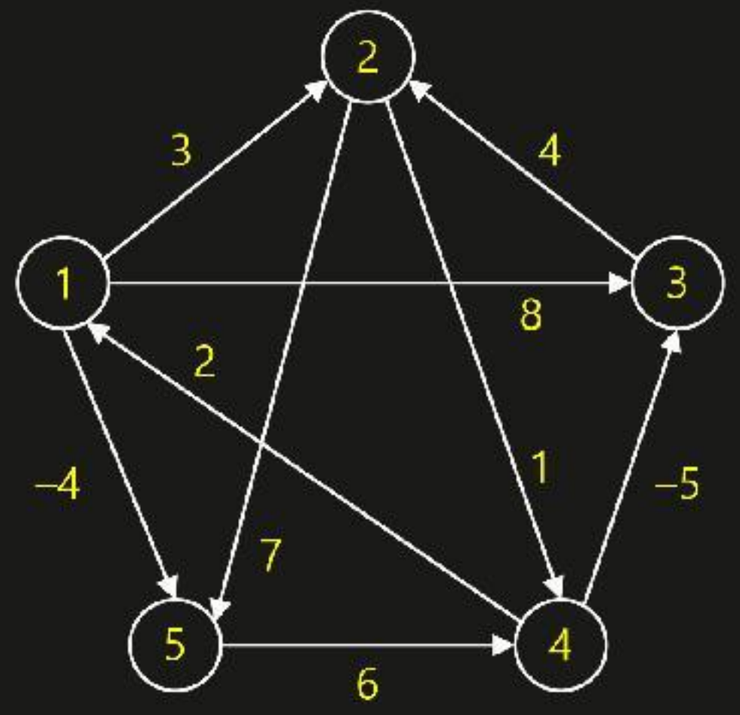
$$d_{31}^{(1)} = \min\left(d_{31}^{(0)}, d_{31}^{(0)} + d_{11}^{(0)}\right)$$

$$d_{31}^{(1)} = \min(\infty, \infty+0) = \infty$$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	
3	∞		0		
4	2			0	
5	∞				0

All Pair Shortest Path Problem

- Step (ii) When k = 1



$$d_{35}^{(1)} = \min(d_{35}^{(0)}, d_{31}^{(0)} + d_{15}^{(0)})$$

$$d_{35}^{(1)} = \min(\infty, \infty + (-4)) = \infty$$

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

$$d_{42}^{(1)} = \min(\infty, 2 + 3) = 5$$

$$d_{43}^{(1)} = \min(d_{43}^{(0)}, d_{41}^{(0)} + d_{13}^{(0)})$$

$$d_{43}^{(1)} = \min(\infty, 2 + 8) = 10$$

$$d_{45}^{(1)} = \min(d_{45}^{(0)}, d_{41}^{(0)} + d_{15}^{(0)})$$

$$d_{45}^{(1)} = \min(\infty, 2 + (-4)) = -2$$

$$d_{51}^{(1)} = \min(d_{51}^{(0)}, d_{51}^{(0)} + d_{11}^{(0)})$$

$$d_{51}^{(1)} = \min(\infty, \infty + 0) = \infty$$

BYJU'S EXAM PREP

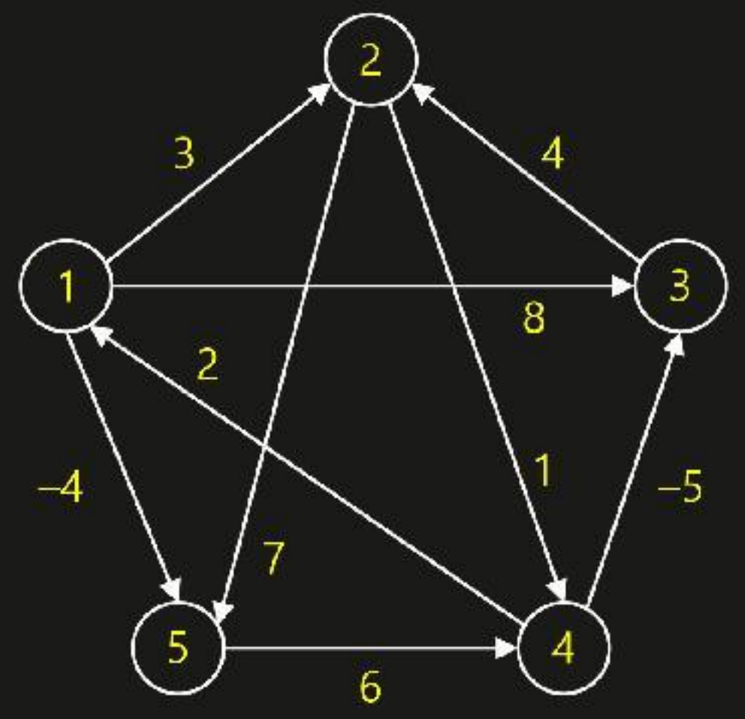
All Pair Shortest Path Problem

- Step (ii) When k = 1

$$D_{ij}^{(1)} = \begin{matrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & \infty \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{matrix}$$

All Pair Shortest Path Problem

- Step (iii) When k = 2



$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$$d_{14}^{(2)} = \min\left(d_{14}^{(1)}, d_{12}^{(1)} + d_{24}^{(1)}\right)$$

$$d_{14}^{(2)} = \min(\infty, 3+1) = 4$$

$$d_{21}^{(2)} = \min\left(d_{21}^{(1)}, d_{22}^{(1)} + d_{21}^{(1)}\right)$$

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

$$d_{34}^{(2)} = \min\left(d_{34}^{(1)}, d_{32}^{(1)} + d_{24}^{(1)}\right)$$

$$d_{34}^{(2)} = \min(-5, 4+1) = -5$$

$$d_{35}^{(2)} = \min\left(d_{35}^{(1)}, d_{32}^{(1)} + d_{25}^{(1)}\right)$$

$$d_{35}^{(2)} = \min(\infty, 4+7) = 11$$

$$d_{43}^{(2)} = \min\left(d_{43}^{(1)}, d_{42}^{(1)} + d_{23}^{(1)}\right)$$

$$d_{43}^{(2)} = \min(10, 5+\infty) = 10$$

BYJU'S EXAM PREP

All Pair Shortest Path Problem

- Step (iii) When k = 2

$$D_{ij}^{(2)} = \begin{matrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & 11 \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{matrix}$$

BYJU'S EXAM PREP

All Pair Shortest Path Problem

- Step (iv) When k = 3

$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

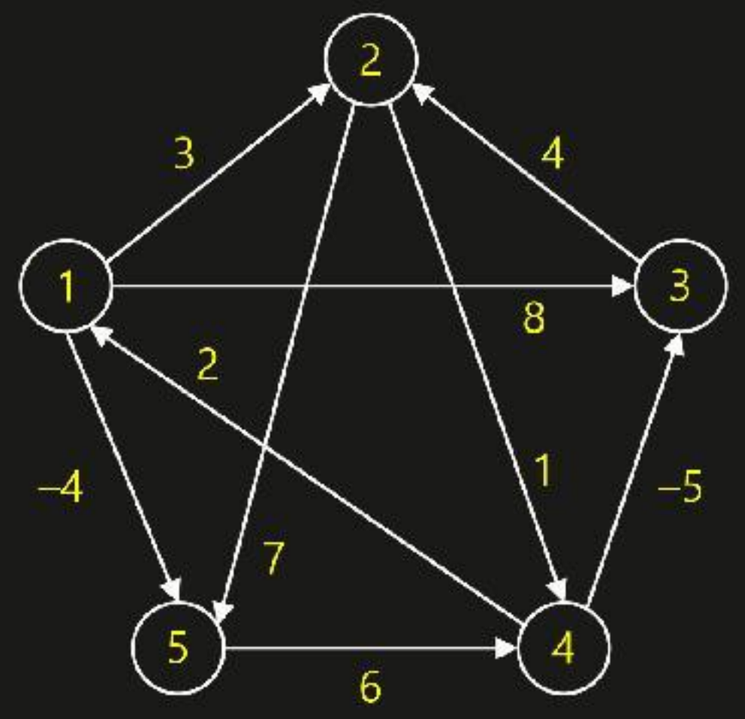
$$d_{14}^{(3)} = \min\left(d_{14}^{(2)}, d_{13}^{(2)} + d_{34}^{(2)}\right)$$

$$d_{14}^{(3)} = \min\left(4, 8 + (-5)\right) = 3$$

$$D_{ij}^{(3)} = \begin{matrix} 0 & 3 & 8 & 3 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & -5 & 11 \\ 2 & 5 & 10 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{matrix}$$

All Pair Shortest Path Problem

- Step (v) When k = 4



$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$$d_{21}^{(4)} = \min\left(d_{21}^{(3)}, d_{24}^{(3)} + d_{41}^{(3)}\right)$$

$$d_{21}^{(4)} = \min(\infty, 1+2) = 3$$

$$d_{23}^{(4)} = \min\left(d_{23}^{(3)}, d_{24}^{(3)} + d_{43}^{(3)}\right)$$

$$d_{23}^{(4)} = \min(\infty, 1+10) = 11$$

$$d_{25}^{(4)} = \min\left(d_{25}^{(3)}, d_{24}^{(3)} + d_{45}^{(3)}\right)$$

$$d_{25}^{(4)} = \min(7, 1+(-2)) = -1$$

$$d_{31}^{(4)} = \min\left(d_{31}^{(3)}, d_{34}^{(3)} + d_{41}^{(3)}\right)$$

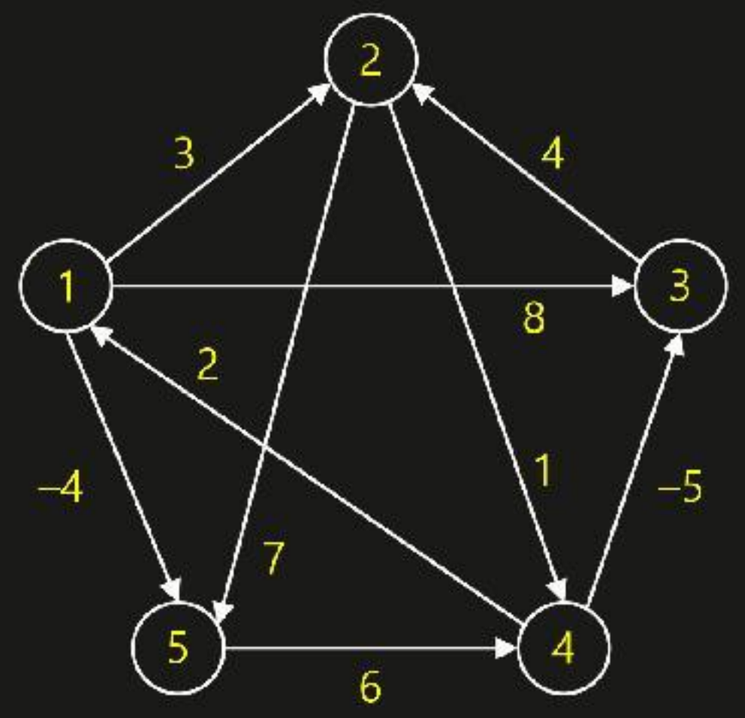
$$d_{31}^{(4)} = \min(\infty, -5+2) = -3$$

$$d_{32}^{(4)} = \min\left(d_{32}^{(3)}, d_{34}^{(3)} + d_{42}^{(3)}\right)$$

$$d_{32}^{(4)} = \min(4, -5+5) = 0$$

All Pair Shortest Path Problem

- Step (v) When k = 4



$$d_{51}^{(4)} = \min(d_{51}^{(3)}, d_{54}^{(3)} + d_{41}^{(3)})$$

$$d_{51}^{(4)} = \min(\infty, 6+2) = 8$$

$$d_{52}^{(4)} = \min(d_{52}^{(3)}, d_{54}^{(3)} + d_{42}^{(3)})$$

$$d_{52}^{(4)} = \min(\infty, 6+5) = 11$$

$$d_{53}^{(4)} = \min(d_{53}^{(3)}, d_{54}^{(3)} + d_{43}^{(3)})$$

$$d_{53}^{(4)} = \min(\infty, 6+10) = 16$$

BYJU'S EXAM PREP

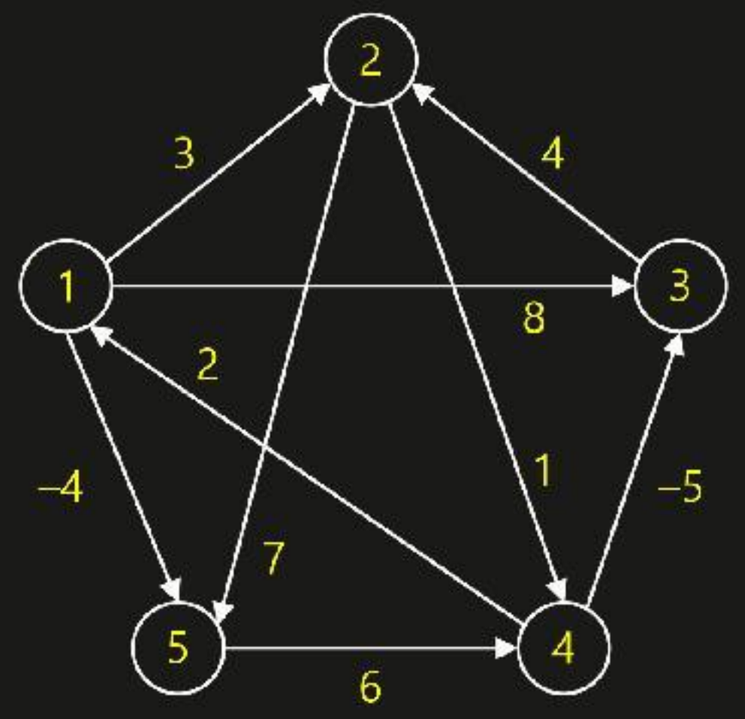
All Pair Shortest Path Problem

- Step (v) When k = 4

$$D_{ij}^{(4)} = \begin{matrix} 0 & 3 & 8 & 3 & -4 \\ 3 & 0 & 11 & 1 & -1 \\ -3 & 0 & 0 & -5 & -7 \\ 2 & 5 & 10 & 0 & -2 \\ 8 & 11 & 16 & 6 & 0 \end{matrix}$$

All Pair Shortest Path Problem

- Step (vi) When k = 5



$$d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$$

$$d_{25}^{(5)} = \min\left(d_{25}^{(4)}, d_{25}^{(4)} + d_{55}^{(3)}\right)$$

$$d_{25}^{(5)} = \min\left(-1, -1 + 0\right) = -1$$

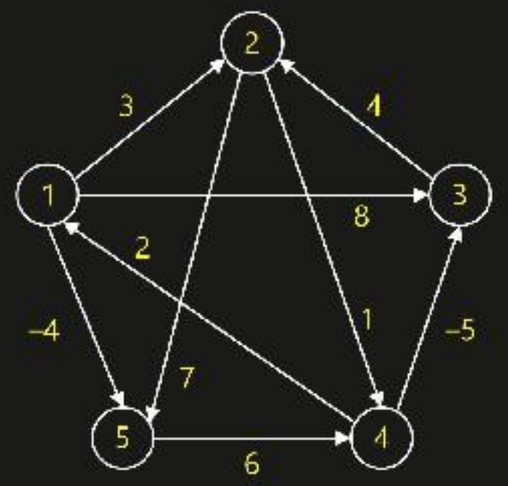
$$d_{23}^{(5)} = \min\left(d_{23}^{(4)}, d_{25}^{(4)} + d_{53}^{(3)}\right)$$

$$d_{23}^{(5)} = \min\left(11, -1 + 16\right) = 11$$

$$d_{35}^{(5)} = \min\left(d_{35}^{(4)}, d_{35}^{(4)} + d_{55}^{(3)}\right)$$

$$d_{35}^{(5)} = \min\left(-7, -7 + 0\right) = -7$$

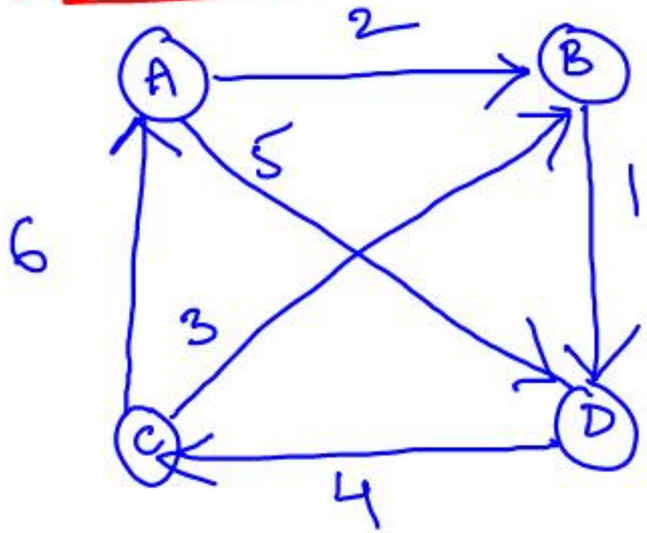
All Pair Shortest Path Problem



- Step (vi) When k = 5

$$D_{ij}^{(5)} = \begin{matrix} 0 & 3 & 8 & 3 & -4 \\ 3 & 0 & 11 & 1 & -1 \\ -3 & 0 & 0 & -5 & -7 \\ 2 & 5 & 10 & 0 & -2 \\ 8 & 11 & 16 & 6 & 0 \end{matrix}$$

Example-2



Initial Matrix

$$D^0 = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 2 & \infty & 5 \\ B & \infty & 0 & \infty & 1 \\ C & 6 & 3 & 0 & \infty \\ D & \infty & \infty & 4 & 0 \end{array}$$

$$D^A = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 2 & \infty & 5 \\ B & \infty & 0 & \infty & 1 \\ C & 6 & 3 & 0 & 11 \\ D & \infty & \infty & 4 & 0 \end{array}$$

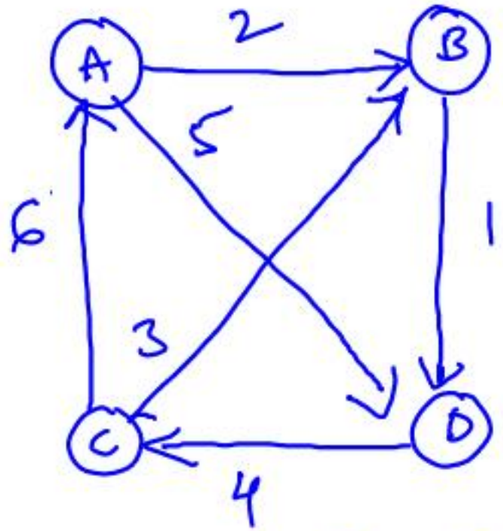
$D^A_{DC} = D \to A + A \to C$
$= \infty + \infty = \infty$

$D^A_{DB} = D - A + A - B$
$= \infty + 2 = \infty$ $\quad \min(\infty, \infty) = \infty$

$D^A_{BC} = (\infty, \infty + \infty)$
$B \to A \to C$ $\quad \min(1, \infty + 5)$,

D^A_{CB} $(C - A, A - B) = 6 + 2 = 8$
$\min(3, 8)$

$D^A_{CD} = C - A + A - D$
$6 + 5 = 11$
$(11, \infty)$



$$D^A = \begin{matrix} 0 & 2 & \infty & 5 \\ \infty & 0 & \infty & 1 \\ 6 & 3 & 0 & 11 \\ \infty & \infty & 4 & 0 \end{matrix}$$

$$D^B = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 2 & \infty & 3 \\ \infty & 0 & \infty & 1 \\ 6 & 3 & 0 & 4 \\ \infty & \infty & 4 & 0 \end{bmatrix} \end{array}$$

$D^B_{DA} = D-B+B-A = \infty+\infty=\infty$

$D^B_{Dc} = D-B+B-C = \infty+\infty=\infty$

$\min(4,\infty)=4$

$D^B_{Ac} = A\to B + B\to C = 2+\infty = \infty$

$\min(\infty,\infty)=\infty$

$D^B_{AD} = A\to B + B\to D = 2+1 = 3$

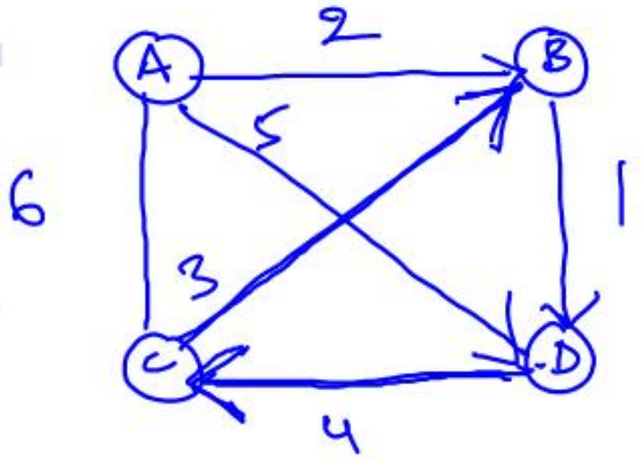
$\min(3,5) = \underline{\underline{3}}$

$D^B_{CA} = C\to B + B\to A = 3+\infty = \infty$

$\min(6,\infty) = 6$

$D^B_{CD} = C\to B + B\to D = 3+1 = 4$

$\min(4,11) = 4$



$$D^{B} = \begin{matrix} \infty & 2 & \infty & 3 \\ \infty & 0 & \infty & 1 \\ 6 & 3 & 0 & \underline{\underline{4}} \\ \infty & \infty & 4 & 0 \end{matrix}$$

$$D^{C} = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 2 & \infty & 3 \\ B & \infty & 0 & \infty & 1 \\ C & 6 & 3 & 0 & 4 \\ D & 10 & 7 & 4 & 0 \end{array}$$

$D^{C}_{AB} = A \to C + C \to B = \infty + 3 = \infty$

$\min(2, \infty) = 2$

$D^{C}_{AD} = A \to C + C \to D = \infty + 4 = \infty$

$\min(3, \infty) = 3$

$D^{C}_{BA} = B \to C + C \to A = \infty + 6 = \infty$

$\min(\infty, \infty) = \infty$

$D^{C}_{BD} = B \to C + C \to D = \infty + 4 = \infty$

$\min(1, \infty) = 1$

$D^{C}_{DA} = D \to C + C \to A = 4 + 6 = 10$

$\min(\infty, 10) = 10$

$D^{C}_{DB} = D \to C + C \to B = 4 + 3 = 7 \quad \min(\infty, 7) = 7$

$$D^{D} = \begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 2 & 7 & 3 \\ B & 11 & 0 & 5 & 1 \\ C & 6 & 3 & 0 & 4 \\ D & 10 & 7 & 4 & 0 \end{array}$$

All Pair shortest cost Matrix

$$D^{C} = \begin{matrix} \infty & 2 & \infty & 3 \\ \infty & 0 & \infty & 1 \\ 6 & 3 & 0 & 4 \\ 10 & 7 & 4 & 0 \end{matrix}$$

$D^{D}_{AB} = A \to D + D \to B = 3+7 = 10$ $\quad \min(2,10) = 2$

$D^{D}_{AC} = A \to D + D \to C = 3+4 = 7$ $\quad \min(7,\infty) = 7$

$D^{D}_{BA} = B \to D + D \to A = 1+10 = 11$ $\quad \min(11,\infty) = 11$

$D^{D}_{BC} = B \to D + D \to C = 1+4 = 5$ $\quad \min(5,\infty) = 5$

$D^{D}_{CA} = C \to D + D \to A$
$= 4+10 = 14$
$\min(6,14) = 6$

$D^{D}_{CB} = C \to D + D \to B$
$= 4+7 = 11$
$\min(3,11) = 3$

All Pair Shortest Path Problem

Floyd – Warshall Algorithm

FLOYD - WARSHALL (W)

1. n ← rows [W].
2. D0 ← W
3. for k ← 1 to n
4. do for i ← 1 to n
5. do for j ← 1 to n
6. do $d_{ij}(k) \leftarrow \min(d_{ij}(k-1), d_{ik}(k-1) + d_{kj}(k-1))$
7. return D(n)

$n *$
$n *$
$n *$

Time complexity.

Construct Matrix for Each Vertex.

n

$O(n^2)$

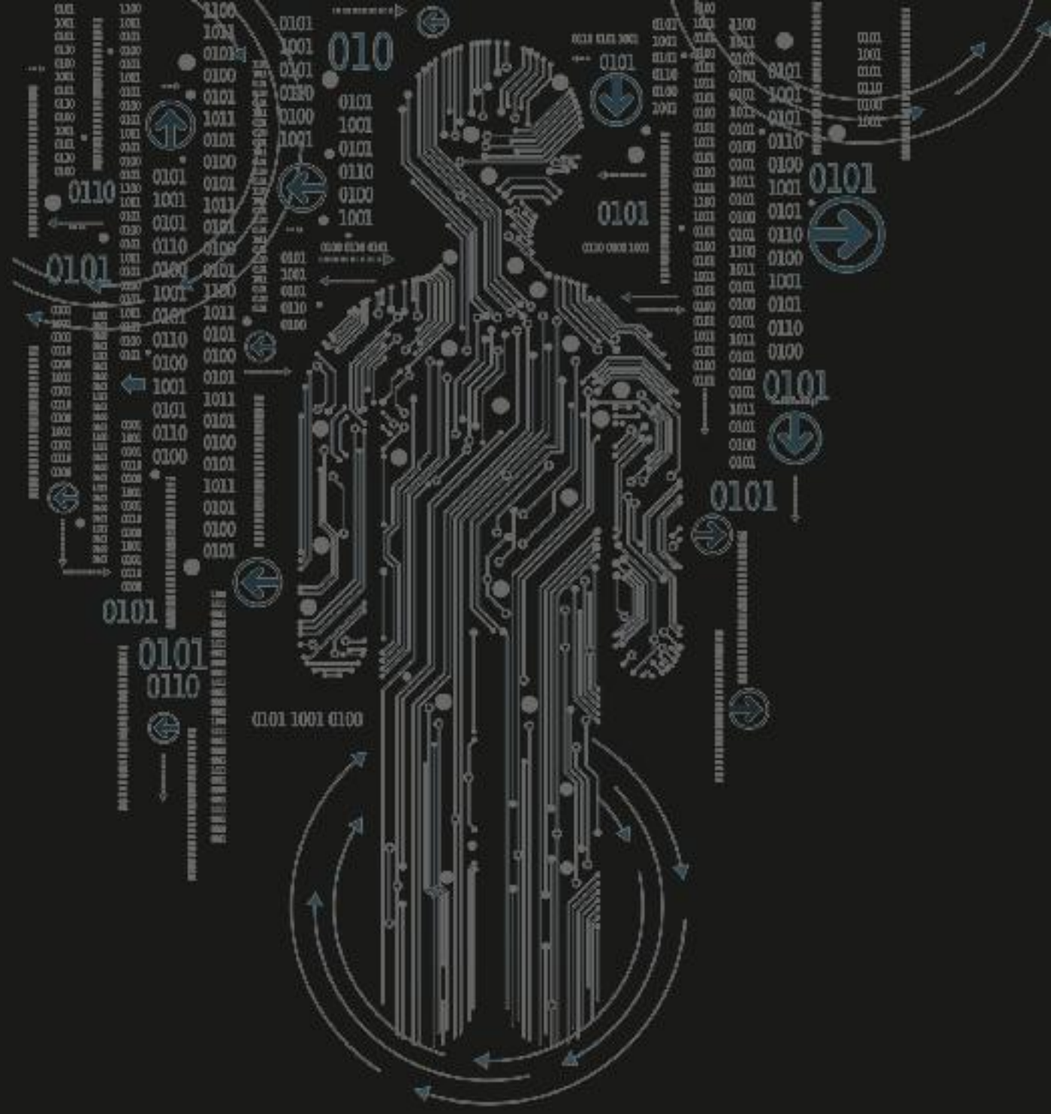
$\Rightarrow O(n^2 * n) = O(n^3)$

BYJU'S
EXAM PREP
4.5
Google Play Review
GET IT ON
Google Play

BYJU'S EXAM PREP

ALGORITHMS

Graphs – DFS, BFS



Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

BYJU'S EXAM PREP

Last Class – Quick Revision

All Pair shortest Path Problem (minimization Problem)

- From Each Node to Each other Node, shortest Cost can be found.
- 2 algorithms
  - Floyd-Warshall*
  - Johnson's

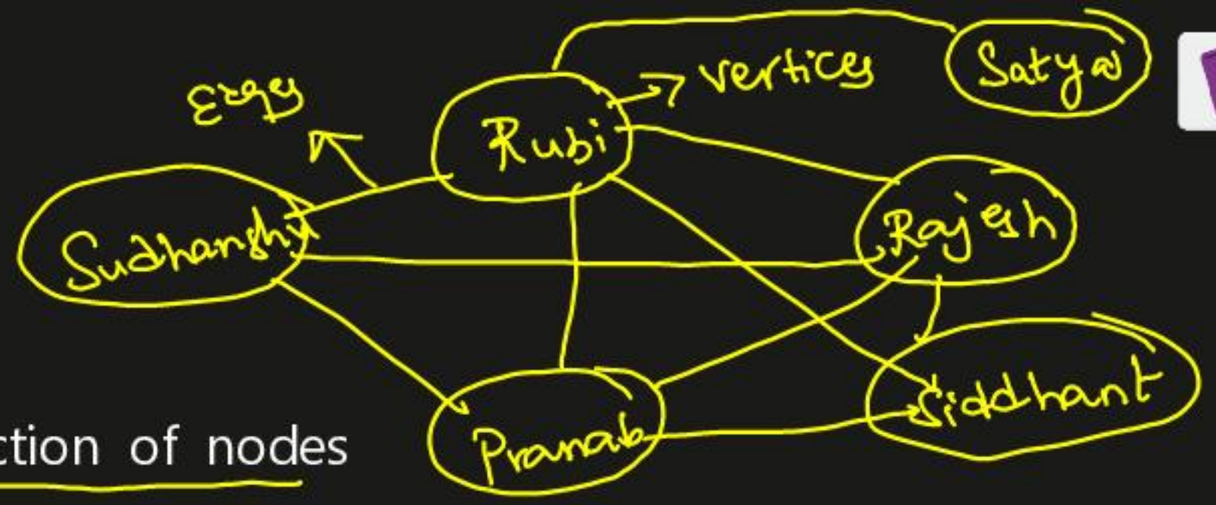
Given a graph, 'G' with 'V' vertices, Then Initial Matrix : C^0

$C^1, C^2, C^3, C^4 \ldots \boxed{C^V} \rightarrow$ All Pair shortest Cost.

$$\boxed{C_{ij}^{k} = \min\left\{ C_{ij}^{k-1},\ C_{ik} + C_{kj} \right\}}$$

$T.C = O(n^3)$

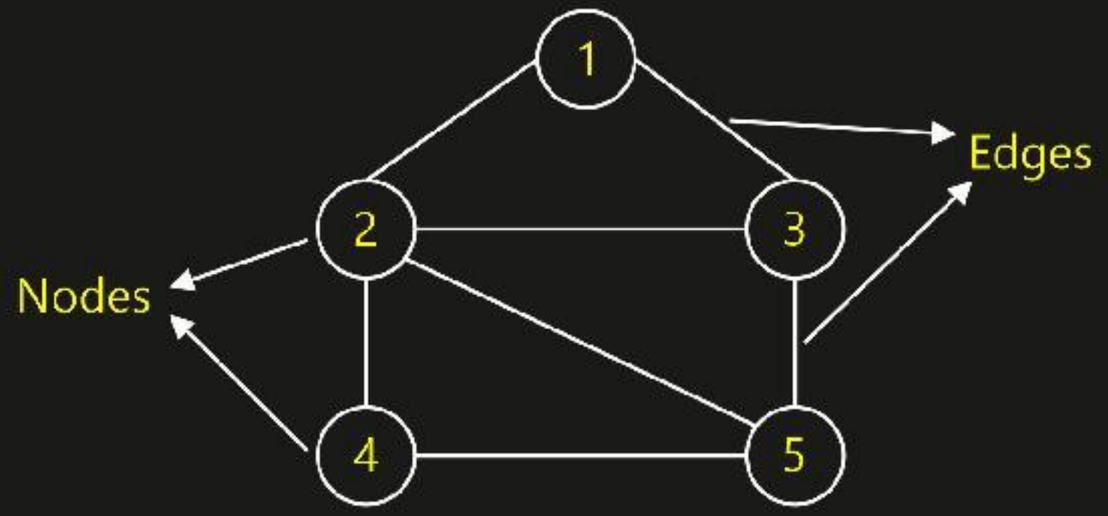
BYJU'S EXAM PREP



Graph

Non-linear

- A graph data structure is a collection of nodes that have data and are connected to other nodes.
- Graphs in data structures are used to address real-world problems in which it represents the problem area as a network like telephone networks, circuit networks, and social networks. For example, it can represent a single user as nodes or vertices in a telephone network, while the link between them via telephone represents edges.



Graph

Graph - Example

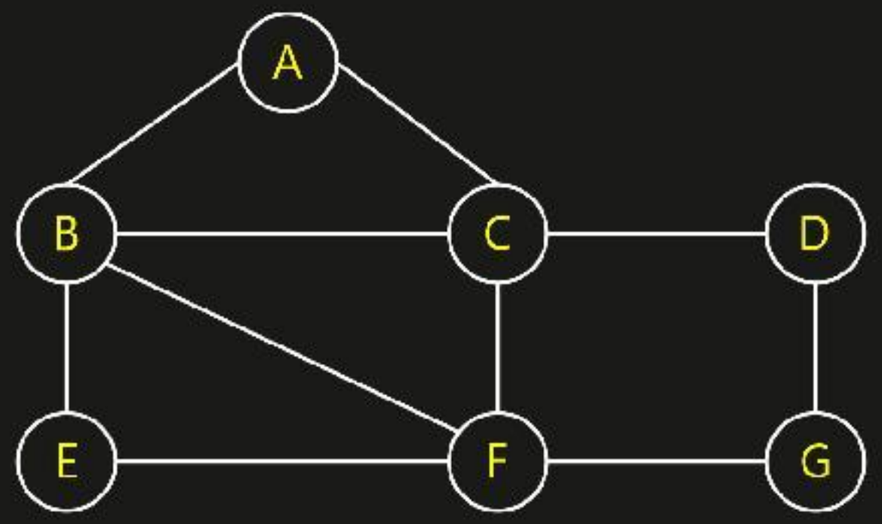
- On facebook, everything is a node. That includes User, Photo, Album, Event, Group, Page, Comment, Story, Video, Link, Note...anything that has data is a node.
- Every relationship is an edge from one node to another. Whether you post a photo, join a group, like a page, etc., a new edge is created for that relationship.

Graph - Example

BYJU'S EXAM PREP

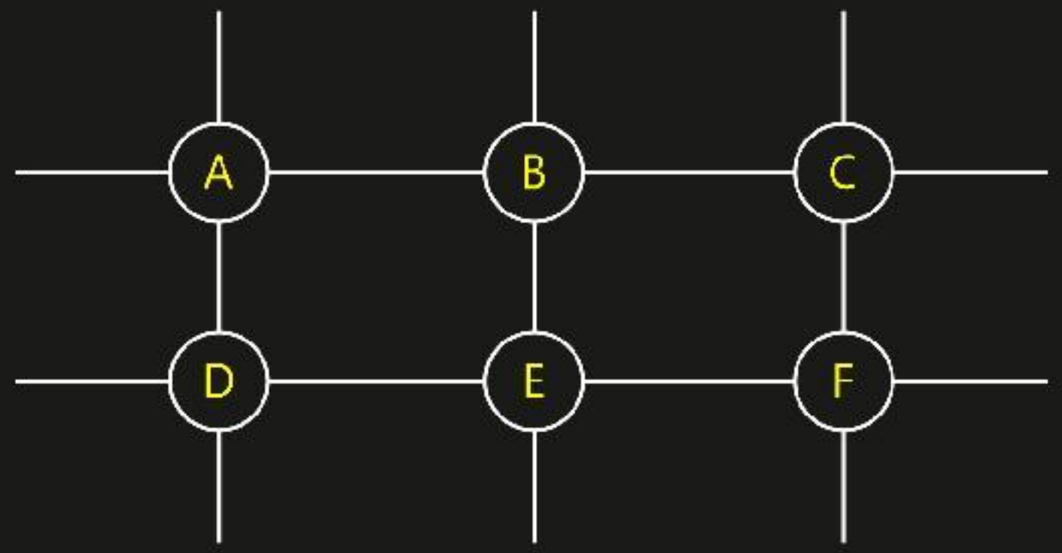
Types Of Graphs

- Finite Graph: The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is limited in number.



Finite Graph

- Infinite Graph: The graph G=(V, E) is called a finite graph if the number of vertices and edges in the graph is interminable.



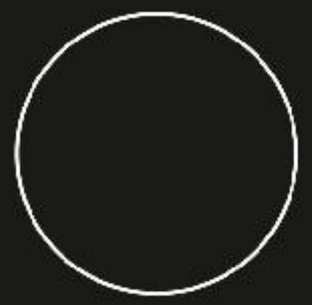
Infinite Graph

BYJU'S EXAM PREP

Graphs → PDS : DFS, BFS Traversals
→ Algo : Prim's, Kruskal's, Dijkstra's, B-F, F-W algo

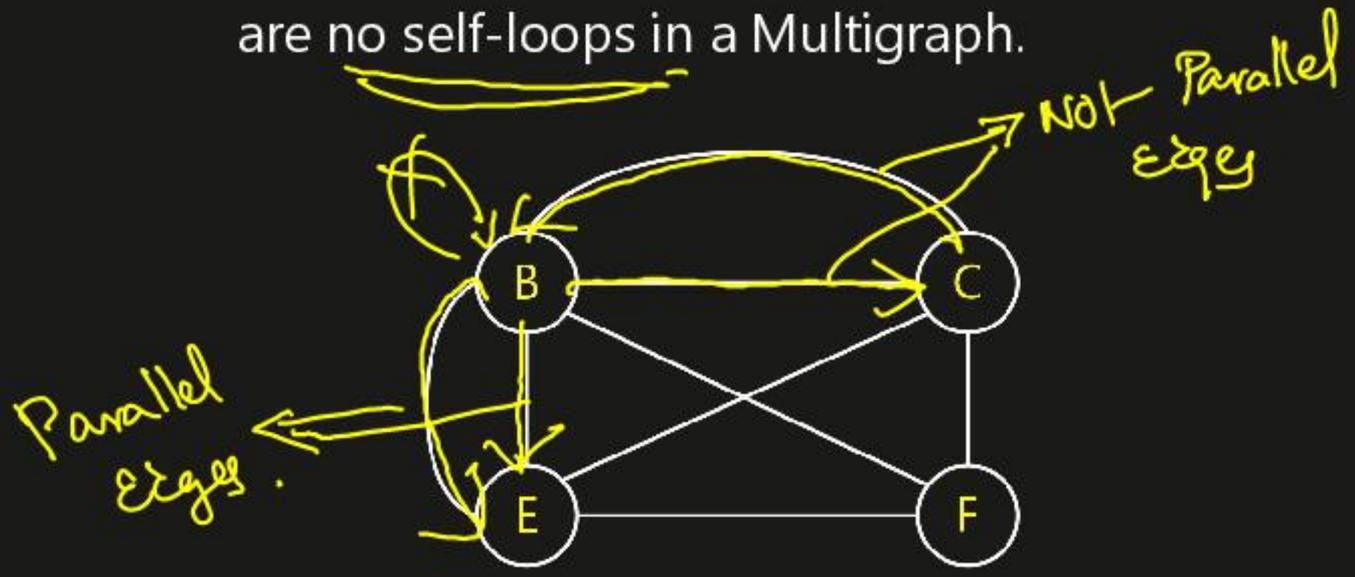
Types Of Graphs

- Trivial Graph: A graph G= (V, E) is trivial if it contains only a single vertex and no edges.



Trivial Graph

- Multi Graph: If there are numerous edges between a pair of vertices in a graph G= (V, E), the graph is referred to as a multigraph. There are no self-loops in a Multigraph.



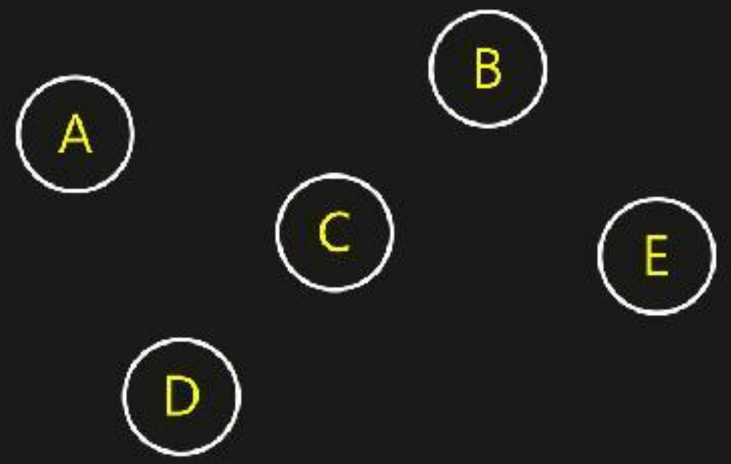
Multi Graph

→ out Edge

← In Edges

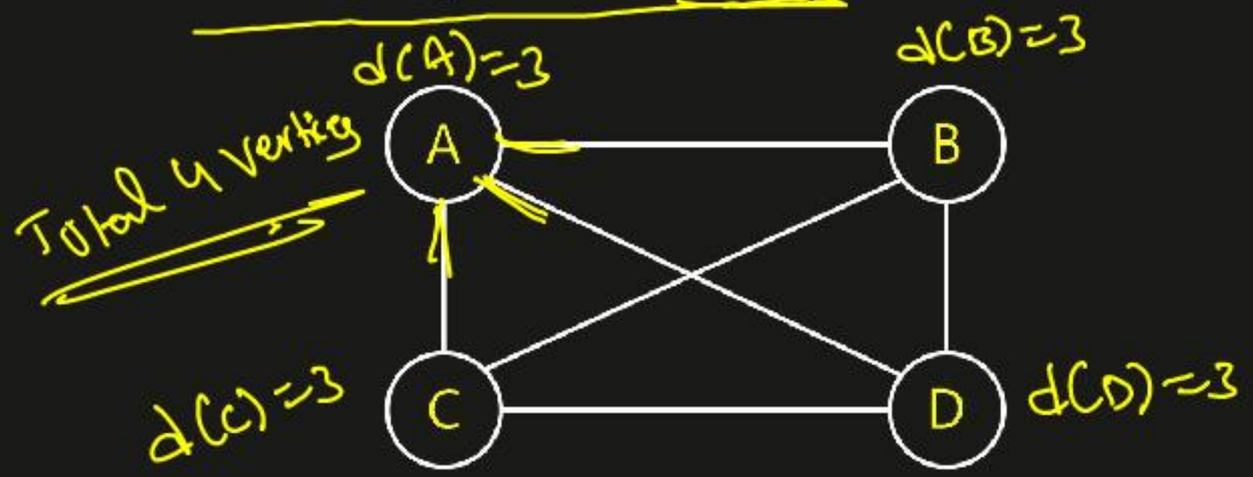
Types Of Graphs

- Null Graph: It's a reworked version of a trivial graph. If several vertices but no edges connect them, a graph G= (V, E) is a null graph.



Null Graph

- Complete Graph: If a graph G= (V, E) is also a simple graph, it is complete. Using the edges, with n number of vertices must be connected. It's also known as a full graph because each vertex's degree must be n-1.

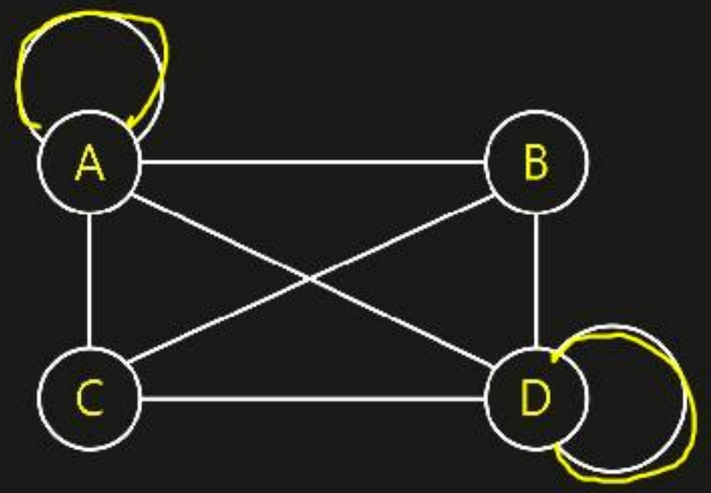


Complete Graph

BYJU'S EXAM PREP

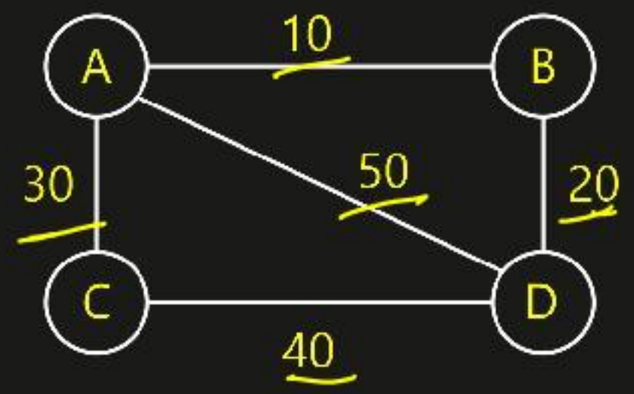
Types Of Graphs

- Pseudo Graph: If a graph G= (V, E) contains a self-loop besides other edges, it is a pseudograph.



Pseudo Graph

- Weighted Graph: A graph G= (V, E) is called a labeled or weighted graph because each edge has a value or weight representing the cost of traversing that edge.

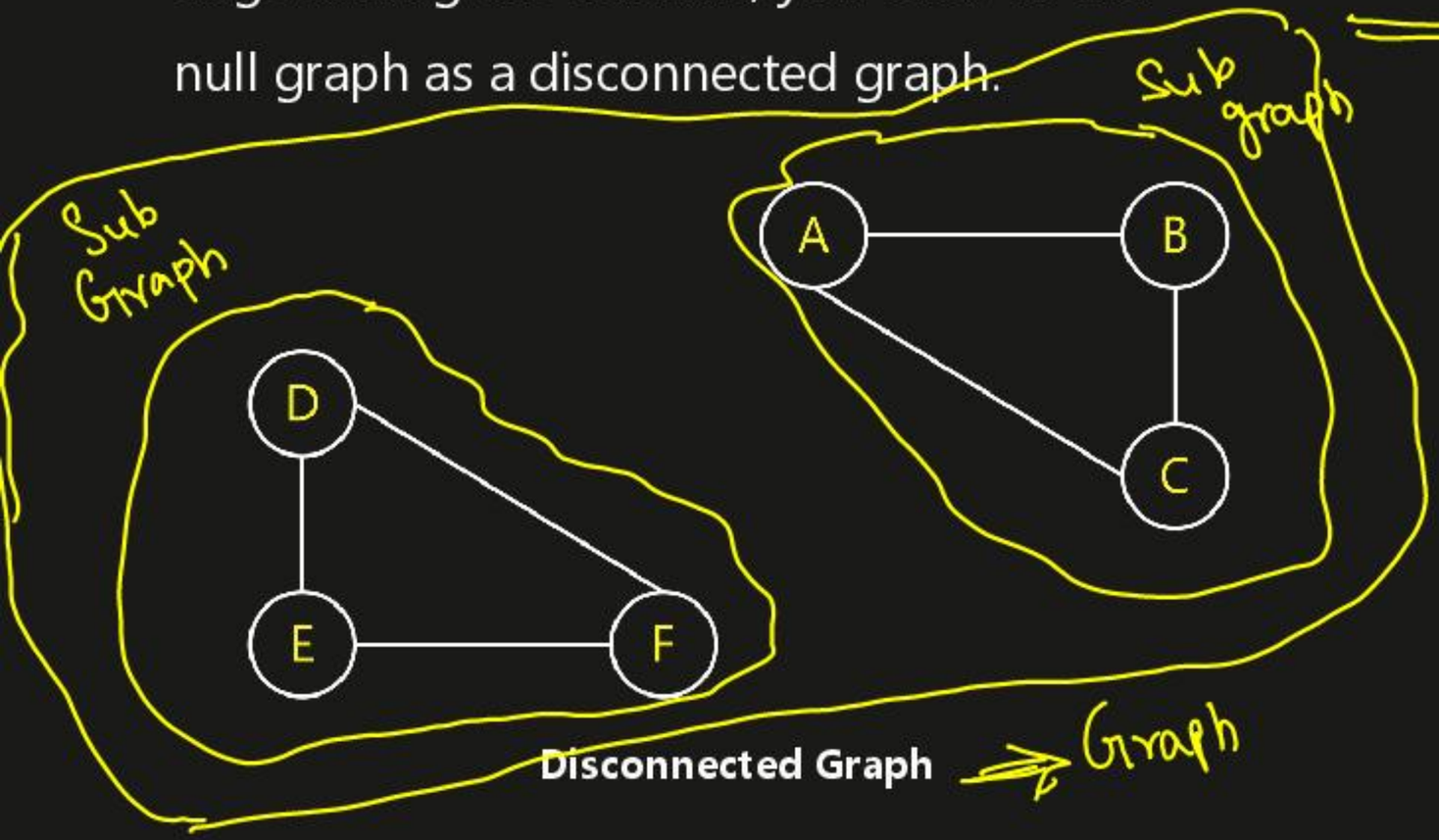


Weighted Graph

BYJU'S EXAM PREP

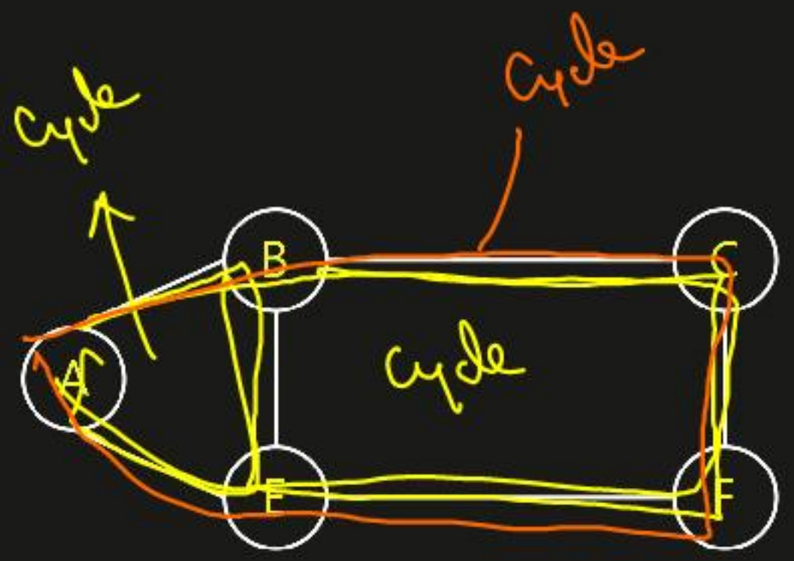
Types Of Graphs

- Disconnected Graph: When there is no edge linking the vertices, you refer to the null graph as a disconnected graph.



Disconnected Graph

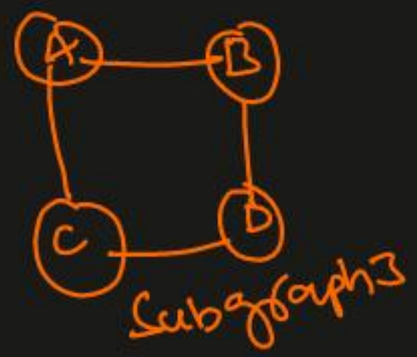
- Cyclic Graph: If a graph contains at least one graph cycle, it is considered to be cyclic.

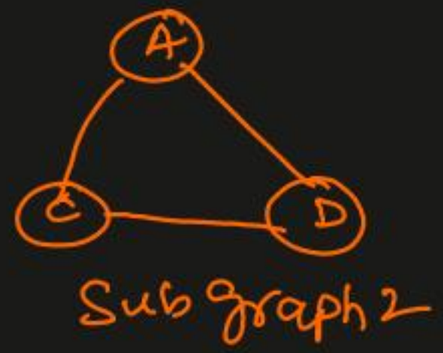


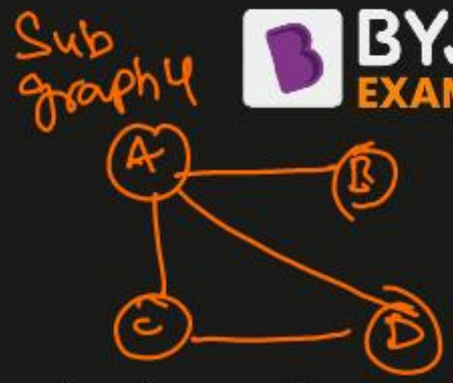
Cyclic Graph

BYJU'S EXAM PREP

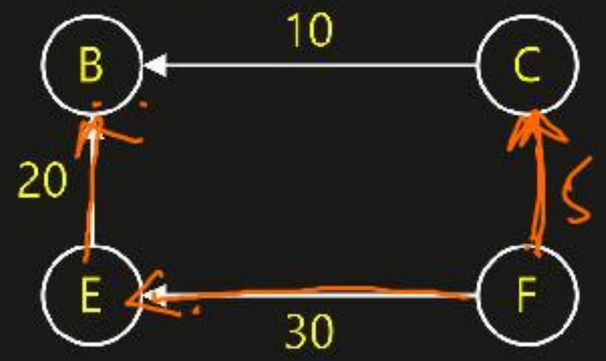
Types Of Graphs





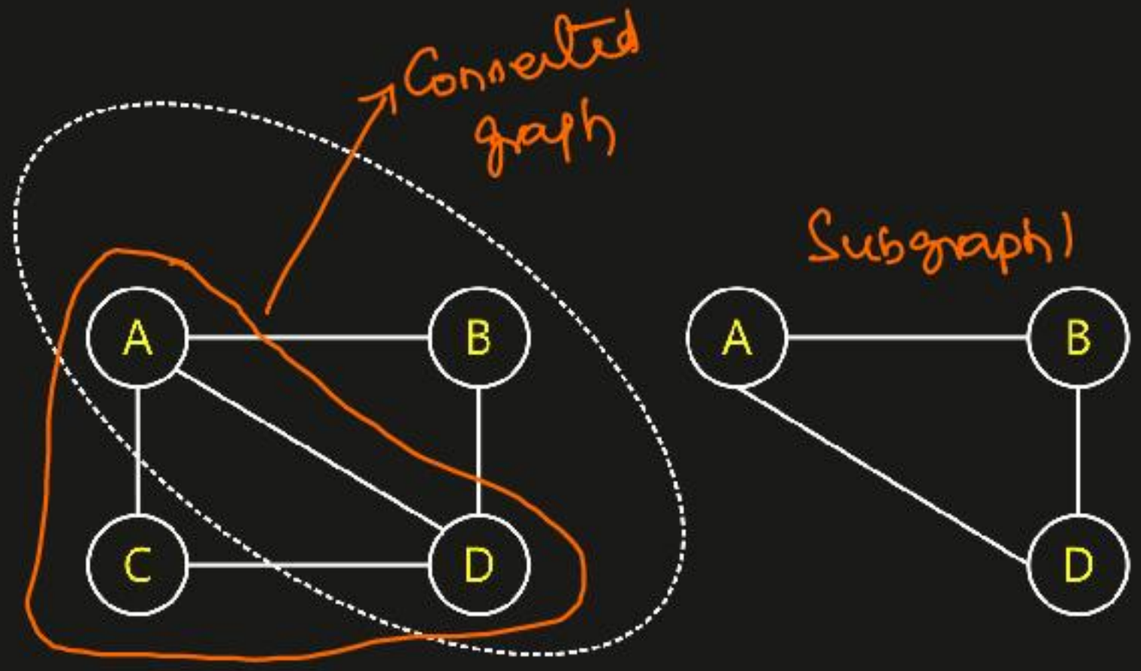


- Directed Acyclic Graph: It's also known as a directed acyclic graph (DAG), and it's a graph with directed edges but no cycle. It represents the edges using an ordered pair of vertices since it directs the vertices and stores some data.



Directed Acyclic Graph

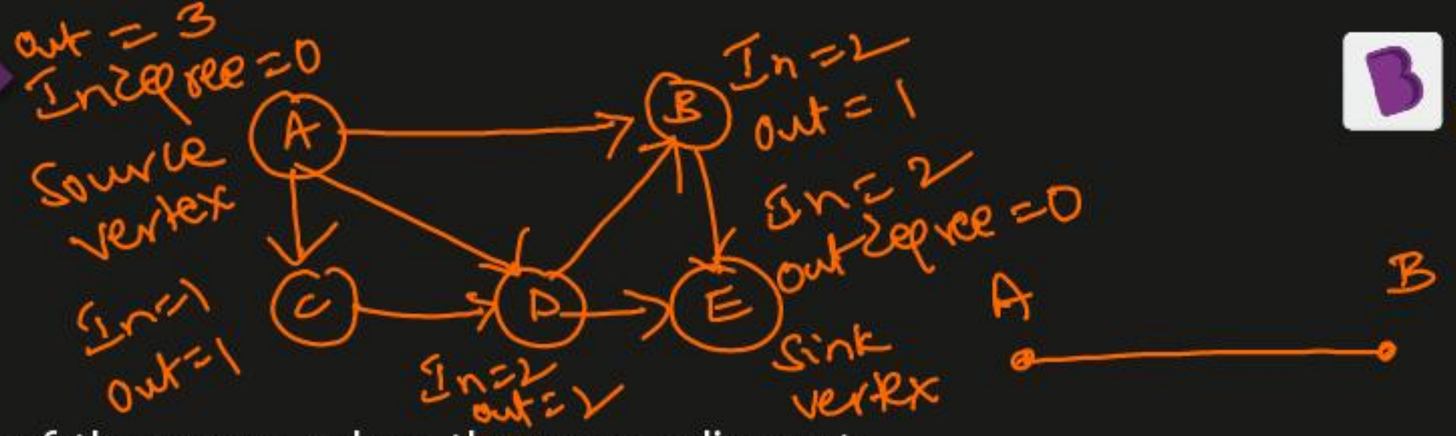
- Subgraph: The vertices and edges of a graph that are subsets of another graph are known as a subgraph.



Subgraph

BYJU'S EXAM PREP

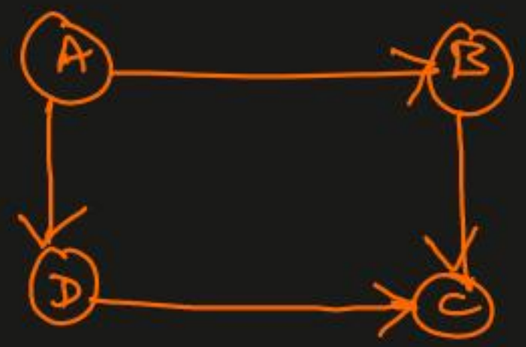
Graph - Terminology



- If two vertices are endpoints of the same edge, they are adjacent.
- A vertex's outgoing edges are directed edges that point to the origin.
- A vertex's incoming edges are directed edges that point to the vertex's destination.
- The total number of edges occurring to a vertex in a graph is its degree.
- The out-degree of a vertex in a directed graph is the total number of outgoing edges, whereas the in-degree is the total number of incoming edges.
- A vertex with an in-degree of zero is referred to as a source vertex, while one with an out-degree of zero is known as sink vertex.

BYJU'S EXAM PREP

Graph - Terminology



A→B Edge

A→B→C Path

- An isolated vertex is a zero-degree vertex that is not an edge's endpoint.
- A path is a set of alternating vertices and edges, with each vertex connected by an edge.
- The path that starts and finishes at the same vertex is known as a cycle.
- A path with unique vertices is called a simple path.
- For each pair of vertices x, y, a graph is strongly connected if it contains a directed path from x to y and a directed path from y to x.
- A directed graph is weakly connected if all of its directed edges are replaced with undirected edges, resulting in a connected graph. A weakly linked graph's vertices have at least one out-degree or in-degree.

BYJU'S EXAM PREP

Graph - Terminology

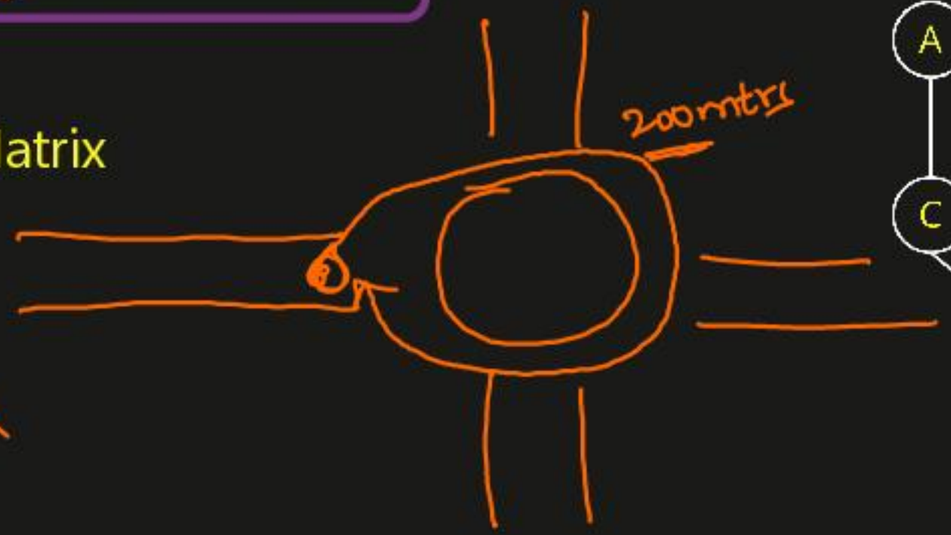
- A tree is a connected forest. The primary form of the tree is called a rooted tree, which is a free tree.
- A spanning subgraph that is also a tree is known as a spanning tree.
- A connected component is the unconnected graph's most connected subgraph.
- A bridge, which is an edge of removal, would sever the graph.
- Forest is a graph without a cycle.

DM.

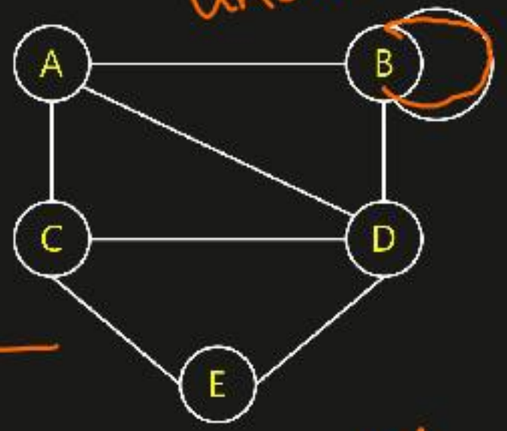
BYJU'S EXAM PREP

Graph - Representation

- Adjacency Matrix
- Adjacency list

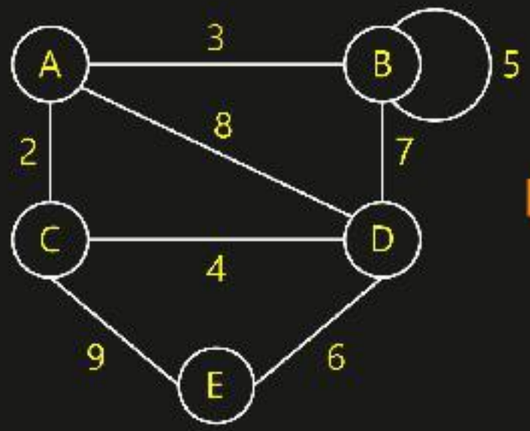


unweighted
undirected



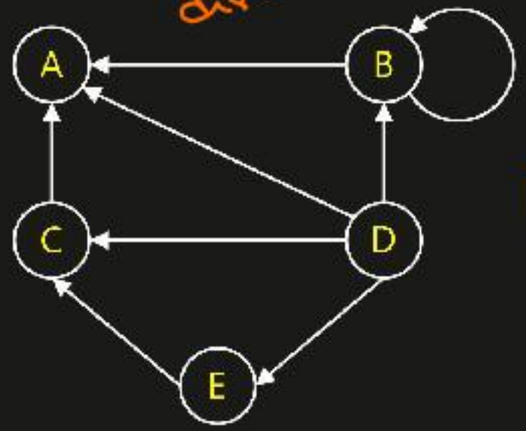
$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 1 & 1 & 1 & 0 \\ B & 1 & \boxed{1} & 0 & 1 & 0 \\ C & 1 & 0 & 0 & 1 & 1 \\ D & 1 & 1 & 1 & 0 & 1 \\ E & 0 & 0 & 1 & 1 & 0 \end{array}$$

Weighted



$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 3 & 2 & 8 & 0 \\ B & 3 & 5 & 0 & 7 & 0 \\ C & 2 & 0 & 0 & 4 & 9 \\ D & 8 & 7 & 4 & 0 & 6 \\ E & 0 & 0 & 9 & 6 & 0 \end{array}$$

unweighted directed graph.

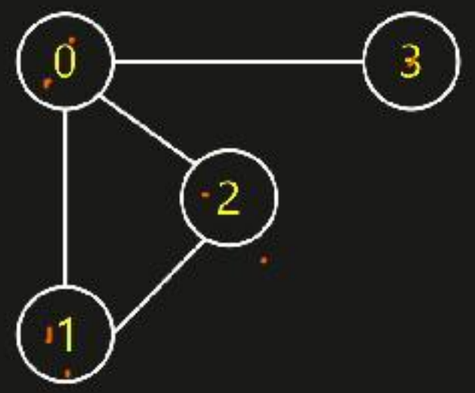


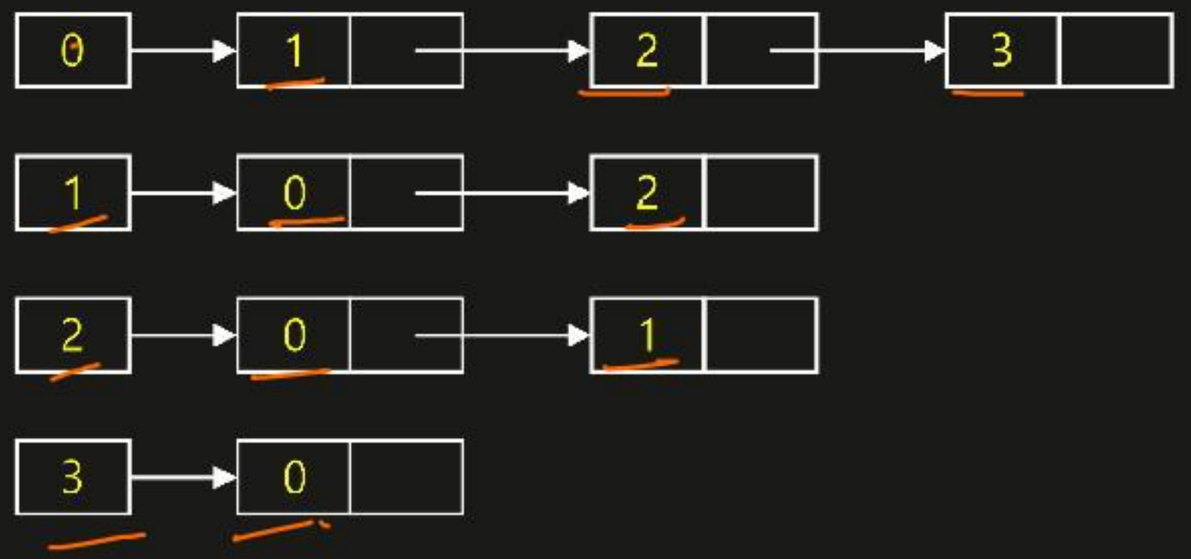
$$\begin{array}{c|ccccc} & A & B & C & D & E \\ \hline A & 0 & 0 & 0 & 0 & 0 \\ B & 1 & 1 & 0 & 0 & 0 \\ C & 1 & 0 & 0 & 0 & 0 \\ D & 1 & 1 & 1 & 0 & 1 \\ E & 0 & 0 & 1 & 0 & 0 \end{array}$$

BYJU'S EXAM PREP

Graph - Representation

- Adjacency List





Adjacency List

BYJU'S EXAM PREP

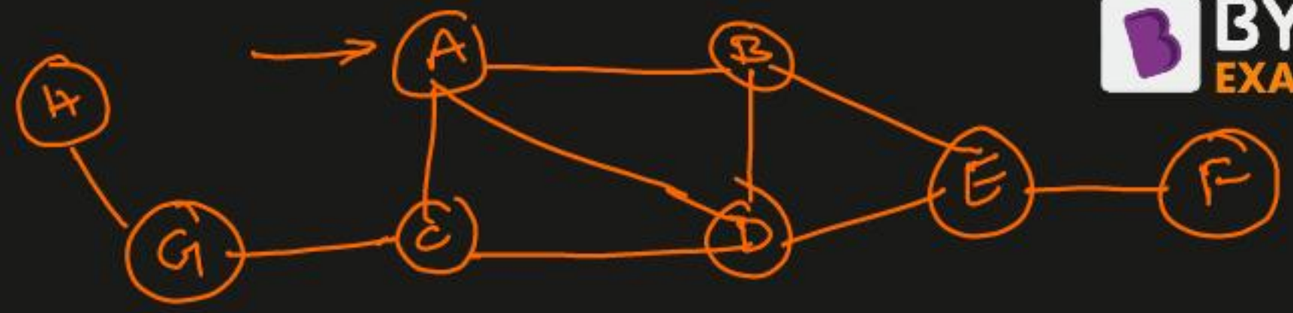
Graph Traversals

[checking]

- The process of visiting or updating each vertex in a graph is known as graph traversal. The sequence in which they visit the vertices is used to classify such traversals. Graph traversal is a subset of tree traversal.
- There are two techniques to implement a graph traversal algorithm:
  - Breadth-first search : FIFO : Queue
  - Depth-first search : LIFO : Stack.

BYJU'S EXAM PREP

Breadth – First Search



- BFS is a search technique for finding a node in a graph data structure that meets a set of criteria.
- It begins at the root of the graph [Source] and investigates all nodes at the current depth level before moving on to nodes at the next depth level.
- To maintain track of the child nodes that have been encountered but not yet inspected, more memory, generally you require a queue.

G

~~A~~,	C	D,	B	E	F, G, H

Invalid BFS

BYJU'S EXAM PREP

BFS - Example

4, 5, 2, 6, 1, 3 valid

Find Invalid BFS sequence

a) 1, 2, 4, 3, 5, 6 ✗

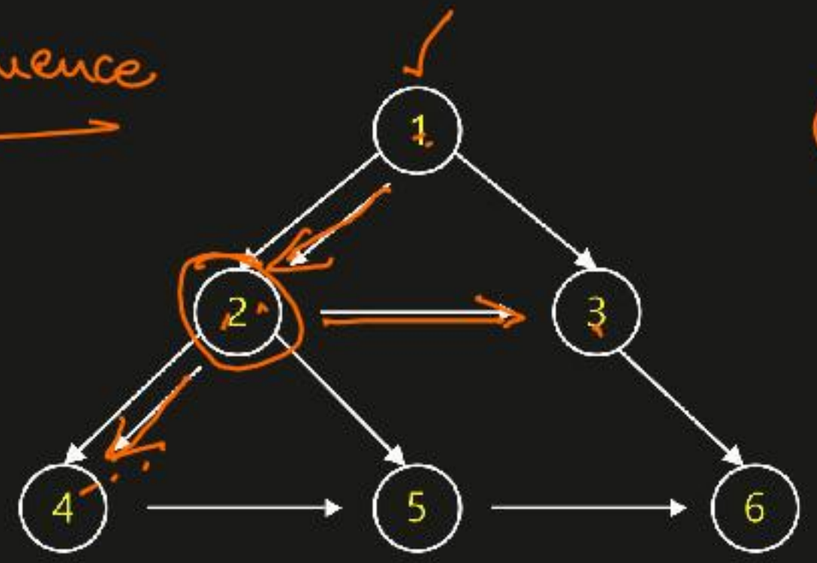
b) 1, 3, 2, 6, 5, 4 ✓

c) 1, 3, 2, 6, 4, 5 ✓

d) 1, 2, 4, 3, 6, 5 ✗

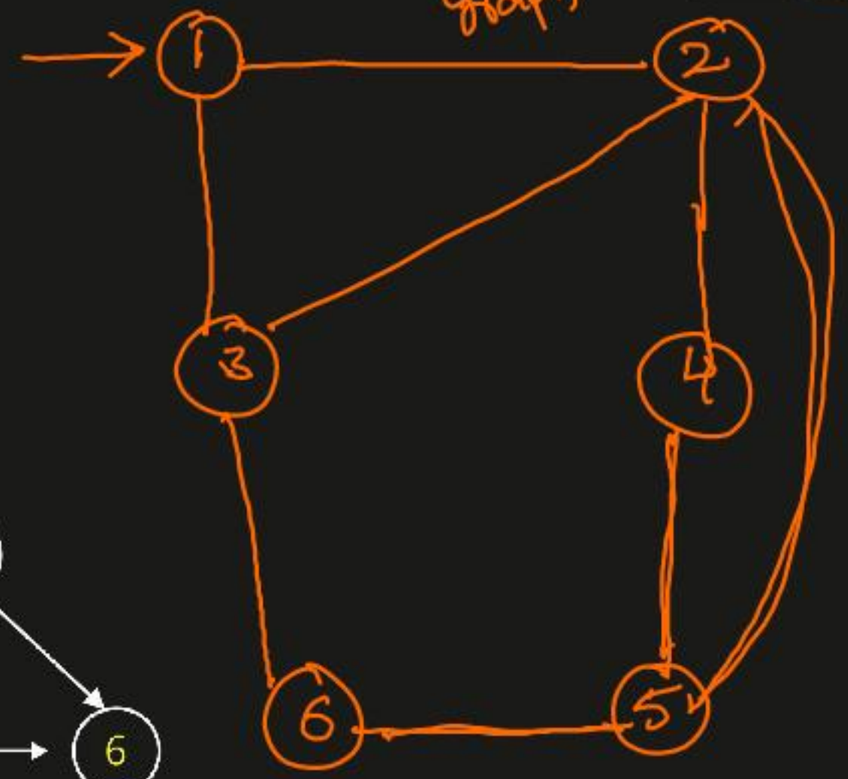
e) 1, 2, 3, 6, 4, 5 ✗

f) 1, 3, 6, 2, 5, 4 ✗



BFS	1	2	3	4	5	6

BFS - Example



3, 1, 2, 6, 4, 5 valid

5, 6, 4, 2, 1, 3 Invalid.

BYJU'S EXAM PREP

BFS - Algorithm

- Step 1: Consider the graph you want to navigate.
- Step 2: Select any vertex in your graph, say v1, from which you want to traverse the graph.
- Step 3: Examine any two data structures for traversing the graph.
- Visited array (size of the graph)
- Queue data structure
- Step 4: Starting from the vertex, you will add to the visited array, and afterward, you will v1's adjacent vertices to the queue data structure.
- Step 5: Now, using the FIFO concept, you must remove the element from the queue, put it into the visited array, and then return to the queue to add the adjacent vertices of the removed element.
- Step 6: Repeat step 5 until the queue is not empty and no vertex is left to be visited.

BYJU'S EXAM PREP

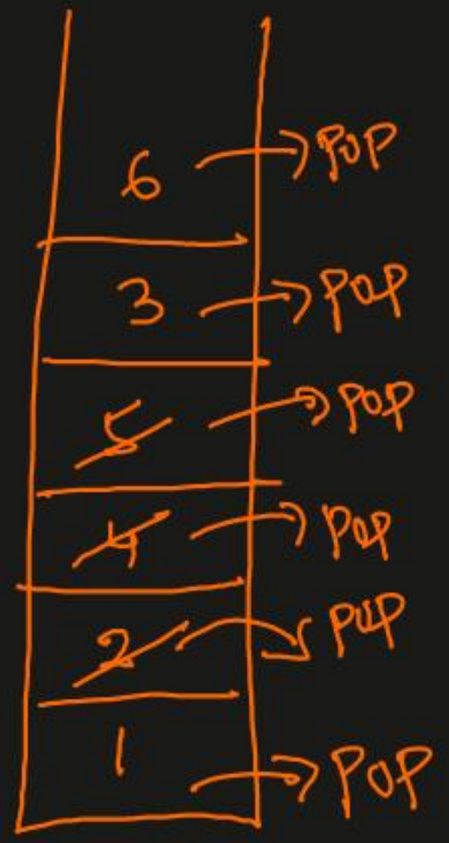
Depth – First Search

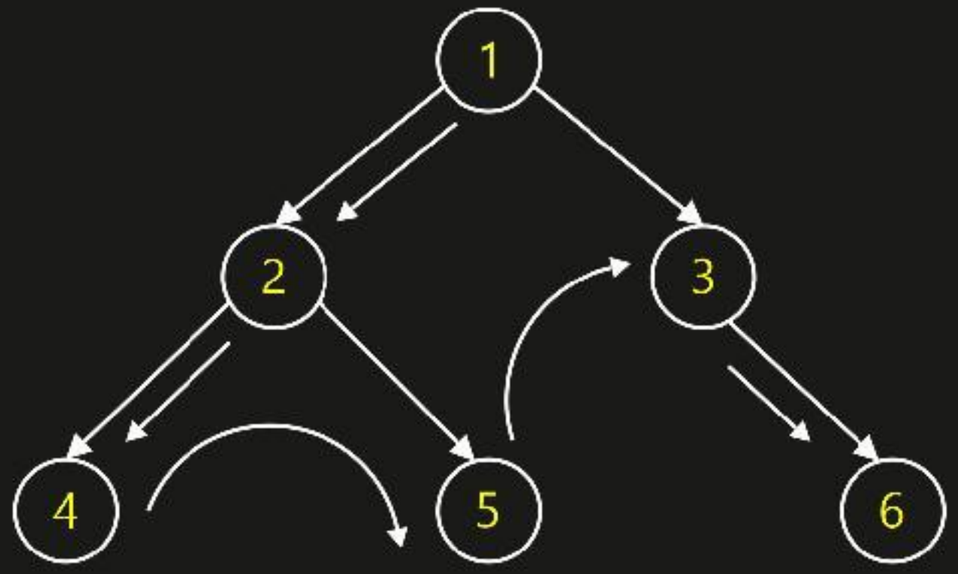
- DFS is a search technique for finding a node in a graph data structure that meets a set of criteria.
- The depth-first search (DFS) algorithm traverses or explores data structures such as trees and graphs. The DFS algorithm begins at the root node and examines each branch as far as feasible before backtracking.

  Source
- To maintain track of the child nodes that have been encountered but not yet inspected, more memory, generally a stack, is required.

BYJU'S EXAM PREP

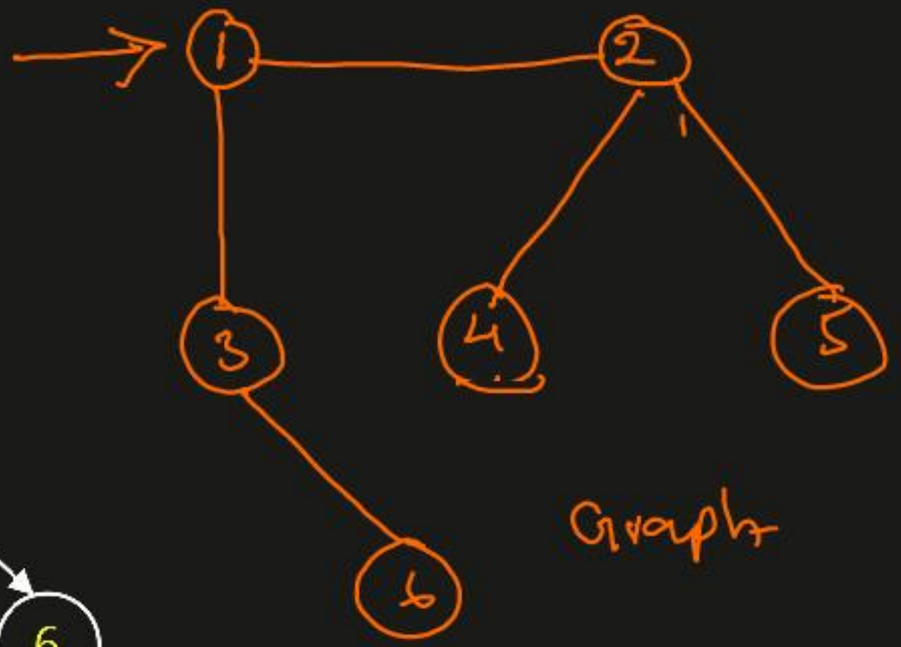
DFS - Example





DFS	1	2	4	5	3	6

DFS - Example

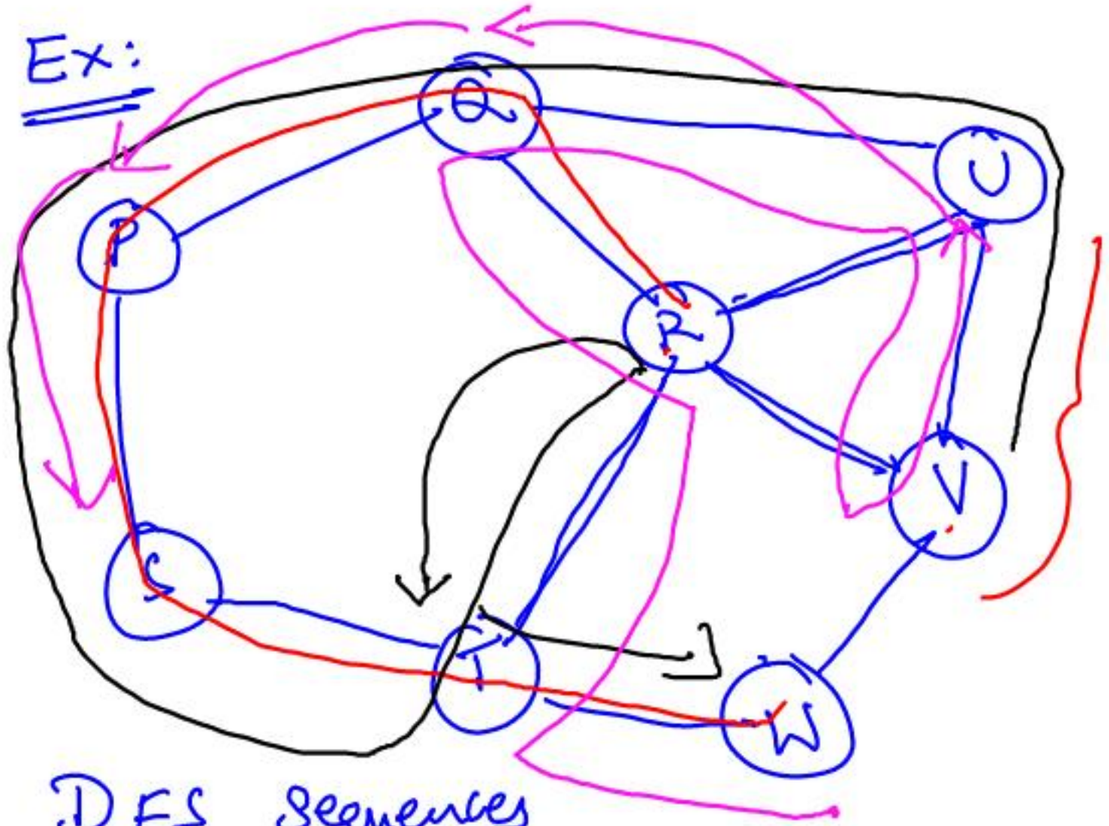


BYJU'S EXAM PREP

DFS - Algorithm

- Step 1: Consider the graph you want to navigate.
- Step 2: Select any vertex in our graph, say v1, from which you want to begin traversing the graph.
- Step 3: Examine any two data structures for traversing the graph.
- Visited array (size of the graph)
- Stack data structure
- Step 4: Insert v1 into the array's first block and push all the adjacent nodes or vertices of vertex v1 into the stack.
- Step 5: Now, using the LIFO principle, pop the topmost element and put it into the visited array, pushing all of the popped element's nearby nodes into it.
- Step 6: If the topmost element of the stack is already present in the array, discard it instead of inserting it into the visited array.
- Step 7: Repeat step 6 until the stack data structure isn't empty.

Ex:



DFS Sequences

① P Q R S T U V W Invalid.

② R Q P S T W U V Invalid

③ Q U R V T W S P Invalid

④ V U Q P S T R W Valid

⑤ W T R Q U V P S Valid

BFS Sequences

① Q R U V S T P W Invalid.

② P S Q T R V W U Invalid

③ V R U W Q T S P Invalid.

- @ 4pm

How to Balance College and GATE Preparation

Link is shared in TG

BYJU'S EXAM PREP

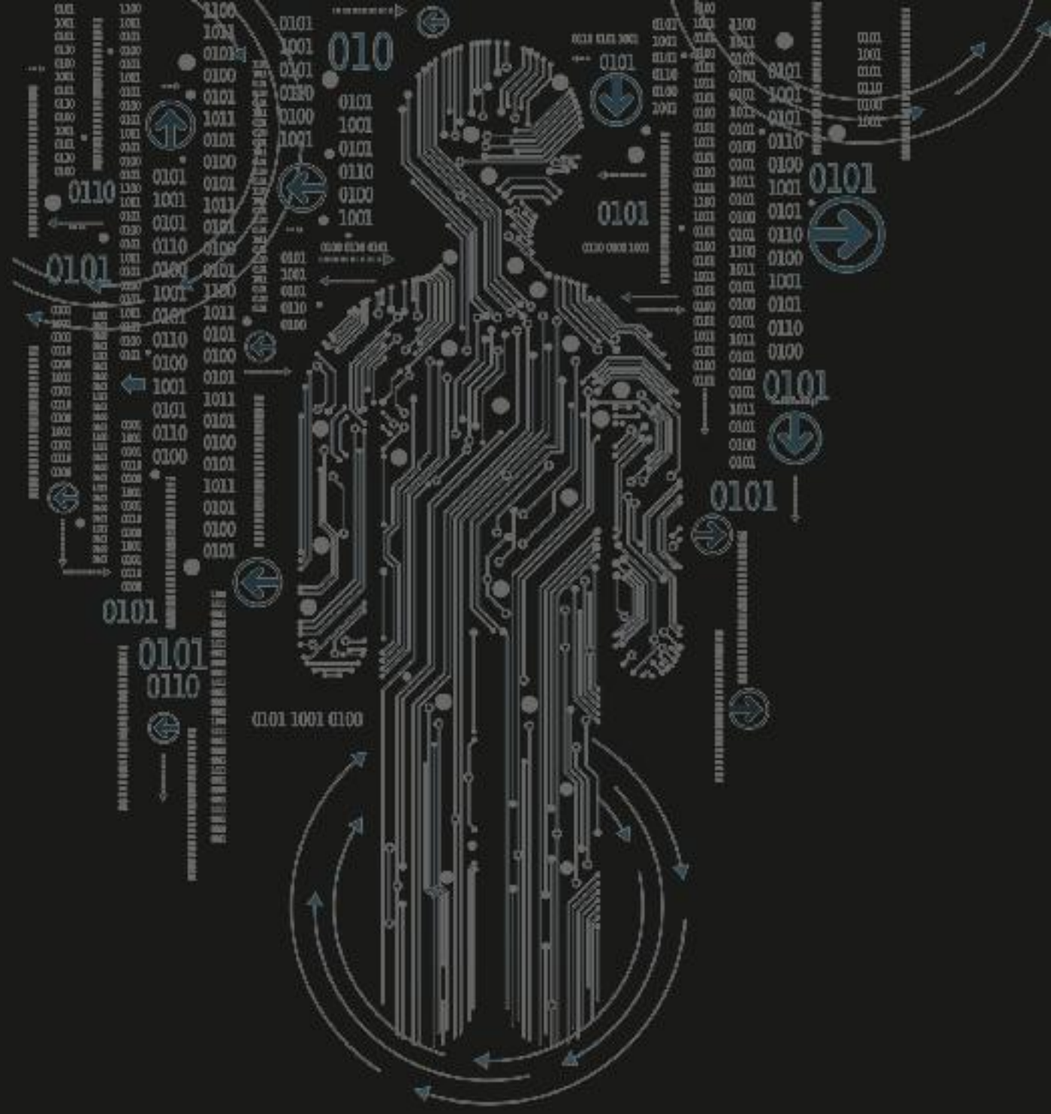
4.5 Google Play Review

GET IT ON Google Play

BYJU'S EXAM PREP

ALGORITHMS

Interactive Quiz - 2



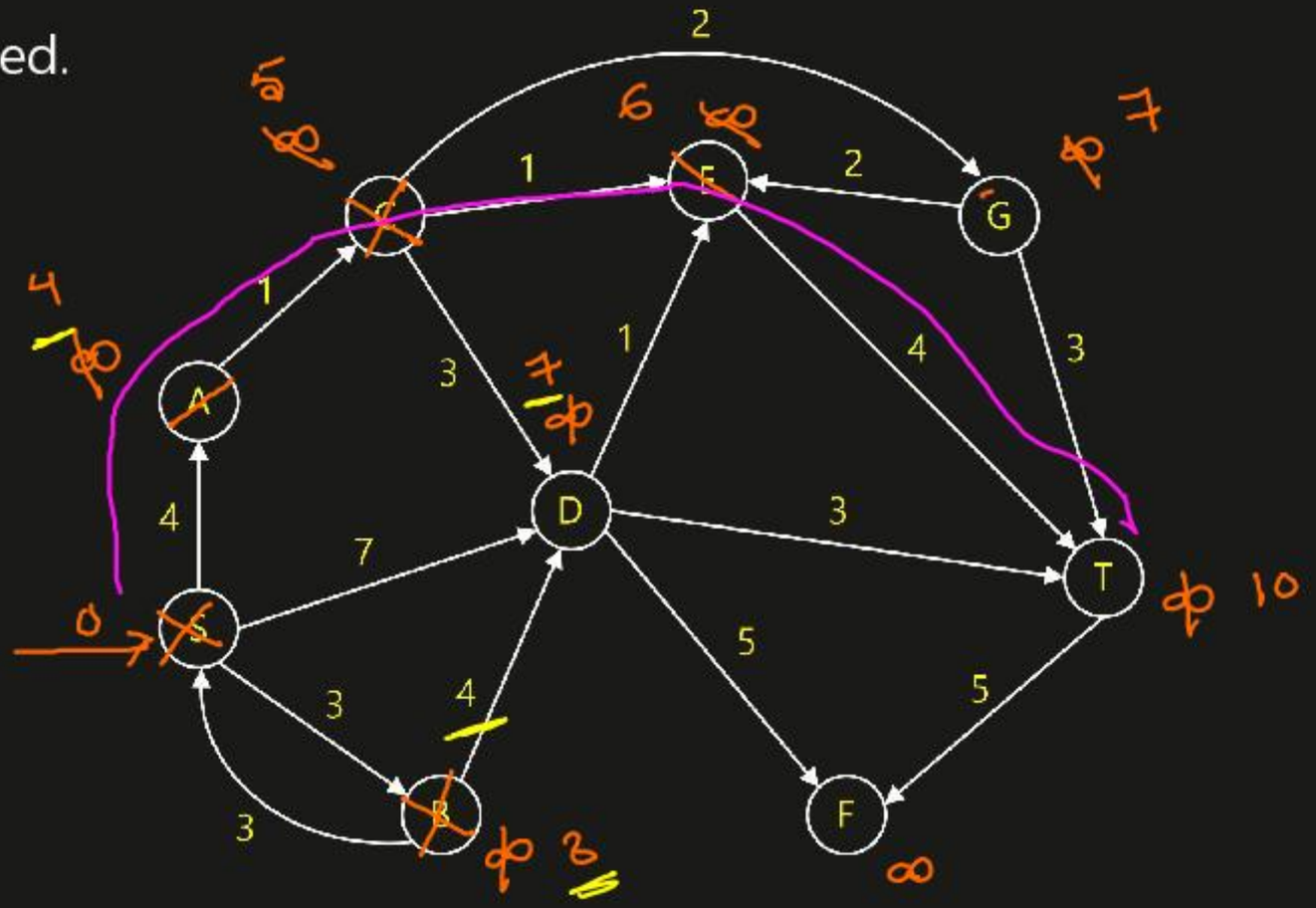
Guidelines to Attend Live Class

BYJU'S EXAM PREP

- ✓ Pen, Notebook and Calculator are must while attending the online class.
- ✓ Regularity and Punctuality is necessary.
- ✓ Hold chat while attending the class. We will allow you to ask and put your questions in the comment box.
- ✓ Attempt Quizzes Daily as per the schedule.
- ✓ Strictly follow the day-wise study plan.

Interactive Quiz - 2

Q1. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Which one will be reported by Dijkstra's shortest path algorithm? Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



A. SDT

B. SBDT

C. SACDT

D. SACET

Interactive Quiz - 2

BYJU'S EXAM PREP

Knapsack weight = 11 kg

0/1 Knapsack

Q2. Consider the weights and values of items listed below. Note that there is only one unit of each item. The task is to pick a subset of these items such that their total weight is no more than 11 Kgs and their total value is maximized. Moreover, no item may be split. The total value of items picked by an optimal algorithm is denoted by V_{opt}. A greedy algorithm sorts the items by their value-to-weight ratios in descending order and packs them greedily, starting from the first item in the ordered list. The total value of items picked by the greedy algorithm is denoted by V_{greedy}.

The value of $V_{opt} - V_{greedy}$ is _____ .

Item number	Weight (in Kgs)	Value (in Rupees)	V/W
1	10	60	6
2	7	28	4
3	4	20	5
4	2	24	12

A. 14

B. 16

C. 20

D. 24

$W = 11$ kg

	W	P	P/W
1	10	60	6
2	7	28	4
3	4	20	5
4	2	24	12

Greedy approach

Pick I_4 Profit = 24

$W = 11 - 2 = 9$

Pick I_1 Not Possible, Because $w_1 = 10 > W$

Pick I_3 Profit = $24 + 20 = 44$

$W = 9 - 4 = 5$

Pick I_2 Not Possible, Because $w_2 > W$

Total Profit using Greedy approach, $V_{Greedy} = 44$

$\Rightarrow$ Optimal algorithm $\Rightarrow$ Pick I_1 $\Rightarrow$ Profit $= 60 = V_{opt}$ $W = 11 - 10 = 1$

$V_{opt} - V_{greedy} = 60 - 44$
$= \underline{16}$

Interactive Quiz - 2

BYJU'S EXAM PREP

Q3. Given below are some algorithms, and some algorithm design paradigms.

	List-I		List-II
A.	Dijkstra's Shortest Path	1.	Divide and Conquer
B.	Floyd-Warshall algorithm to compute all pairs shortest path	2.	Dynamic Programming
C.	Binary search on a sorted array	3.	Greedy design
D.	Backtracking search on a graph	4.	Depth-first search

Match the above algorithms on the left to the corresponding design paradigm they follow.

Codes: A B C D

A 3 2 1 4

B 1 2 3 4

C 3 1 2 4

D 4 2 1 3

Interactive Quiz - 2

BYJU'S EXAM PREP

Ans: A, D

Q4. Consider 4 matrices P, Q, R, and S with orders 2*3, 3*4, 4*3, and 3*2. The First product that can happen in order to obtain the result in the optimum number of multiplications is _____.

[MSQ]

A. P*Q

B. Q*R

C. P*S

D. R*S

64

① (PQ)(RS)

PQ = 2x3x4 = 24

RS = 4x3x2 = 24

(PQ)(RS) = 2x4x2 = 16

60

② ((PQ) R) S

PQ = 24 [2x4]

(PQ) R = 2x4x3 = 24

((PQ) R) S = 2x3x2 = 12

66

③ (P (QR)) S

QR = 3x4x3 = 36

P(QR) = 2x3x3 = 18

Result = 2x3x2 = 12

66

④ P ((QR) S)

QR = 36

(QR) S = 3x3x2 = 18

Result = 2x3x2 = 12

60

⑤ P (Q (RS))

RS = 24

Q(RS) = 3x4x2 = 24

Result = 2x3x2 = 12

80

⑥ (RS) (PQ)

RS = 24 4x2

PQ = 24 2x4

Result = 4x2x4 = 32

Interactive Quiz - 2

BYJU'S EXAM PREP

$n^2 \log n = O(n^2)$

Q5. There are n unsorted arrays: A1, A2, ...,An. Assume that n is odd. Each of A1, A2, ..., An contains n distinct elements. There are no common elements between any two arrays. The worst-case time complexity of computing the median of the medians of A1, A2, ...,An is _____.

1, 5, 10 2, 3, 8

$A_1 = 1, 10, 5$ (n log n)

$A_2 = 8, 3, 2$ (n log n)

$A_3 = 4, 11, 9$ (n log n)

Merge

A. O(n log n)

B. $O(n^2)$

C. O(n)

D. $\Omega(n^2 \log n)$

① Compute Median of Each array → $n(n \log n)$ time $= n^2 \log n$

may not be required. → ② Merge all arrays into single array → n log n

③ Computer Median of whole array → Constant

$n^2 \log n + n \log n$

Interactive Quiz - 2

n^2 $n\log n$ n^2

Q6. The worst-case running times of Insertion sort, Merge sort and Quick sort, respectively, are:

A. $O(n \log n)$, $O(n \log n)$ and $O(n^2)$

B. $O(n^2)$, $O(n^2)$ and $O(n \text{ Log } n)$

C. $O(n^2)$, $O(n \log n)$ and $O(n \log n)$

D. $O(n^2)$, $O(n \log n)$ and $O(n^2)$

Interactive Quiz - 2

BYJU'S EXAM PREP

Q7. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is?

A. Θ(nlogn)

B. Θ(n)

C. Θ(logn)

D. Θ(1)

First 3 elements need to be compared.

13 10 40 ---

2 comparisons

(13, 10) 13 > 10 is not minimum

(13, 40) 13 < 40 is not maximum

Interactive Quiz - 2

BYJU'S EXAM PREP

Ans: 16

Q8. Consider a double hashing scheme in which the primary hash function is h1(k) = k mod 21, and the secondary hash function is h2(k) = 1+(k mod 19). Assume that the table size is 21. Then the address returned by probe 1 in the probe sequence (assume that the probe sequence begins at probe 0) for key value k = 97 is _____.

Double Hashing Hash Function : $h(k, i) = [h_1(k) + i\, h_2(k)] \% m$

$$h(97, 1) = [h_1(97) + 1 \cdot h_2(97)] \% 21$$

$$= [(97 \% 21) + [1 + 97 \% 19]] \% 21$$

$$= (13 + [1 + 2]) \% 21 = (13 + 3) \% 21$$

$$= 16 \% 21 = 16$$

Interactive Quiz - 2

qpqr ✓ qprr ✓ qqrr ✗
pqrr ✓

Longest = 4 Sub sequences
Common Sub sequences = 3

Q9. Consider two strings A = "qpqrr" and B = "pqprqrp". Let x be the length of the longest common subsequence (not necessarily contiguous) between A and B and let y be the number of such longest common subsequences between A and B. Then x + 10y = 34.

x = 4
y = 3

(A) 33

(B) 23

(C) 43

(D) 34 ✓

String: ArshPreet

Pre, Pr, Arsh, A, Ar, Ars, Preet, eet, et

Sub strings

ArshPreet

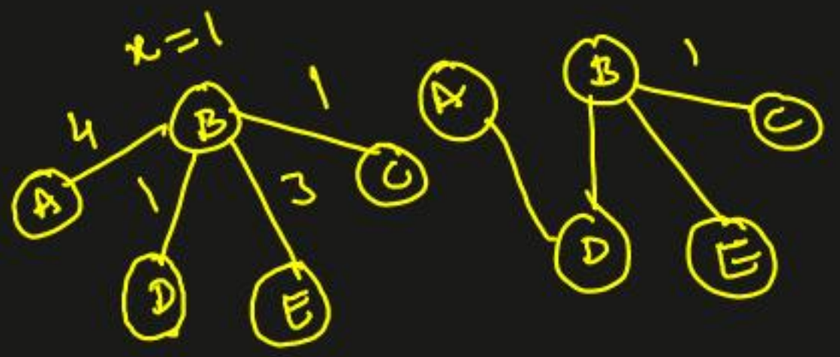
Arshp, Areet, reet, rpreet, hpret, Arspe, rshp, Arhpe ⇒ Sub sequence

Interactive Quiz - 2

Ans: 2

Q10. Consider the following undirected graph G:

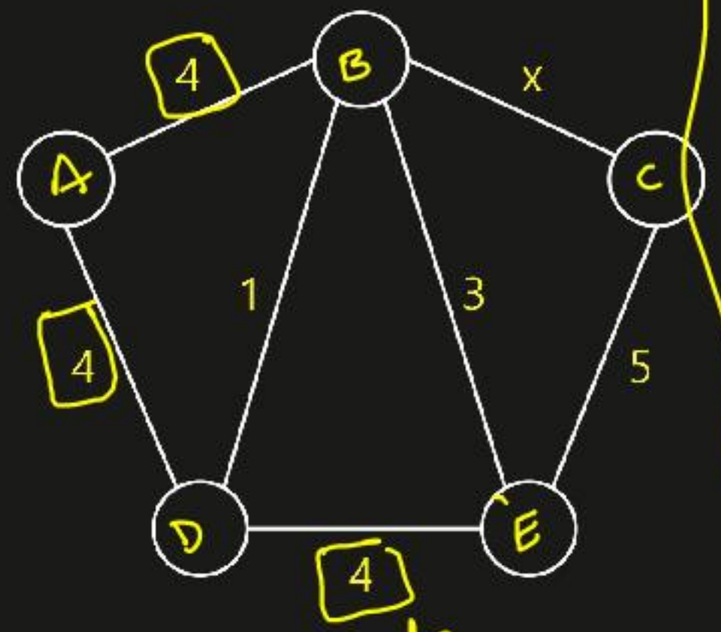
Choose a value for x that will maximize the number of minimum weight spanning trees (MWSTs) of G. The number of MWSTs of G for this value of x is _____.



2 MWSTS

x=2 Same as above

2 MWSTS



cycle

⇒ x=5 4 MWSTS

x=3 Same as above

x=4

2 MWSTS

ALL THE VERY BEST

(A) Θ(nlogn)
(B) Θ(n)
(C) Θ(logn)
(D) Θ(1)

BYJU'S EXAM PREP
4.5 Google Play Review
GET IT ON Google Play