Saathi

Date __/__/____

Design And Analysis Of Algorithm

Part 1 Chapter 1 :- Introduction

Algorithm :- Step by step procedure of solving a computation problem is called algorithm.

Program :- Step by step procedure of solving a problem.

Difference between Algorithm & Program :

	Algorithm	Program
(i)	Algos are written at design time	(i) Programs are written at implementation time.
(ii)	A person who writes algo should have domain knowledge	(ii) The programmer can also have domain knowledge.
(iii)	Any language can be used to write algorithm ie. English language can be used [use of mathematical notations]	(iii) Only programming language is used to write a program ie. C / C++, python, JAVA
(iv)	Algo should be hardware or operating system independent	(iv) The program is dependent on hardware & operating system dependent
(v)	Analyse the algo.	(v) Testing of program

Page No.

Saathi

Date ___ / ___ / _____

Priori Analysis & Posteriori Testing

Priori Analysis	Posteriori testing
→ It is done on algorithm	→ It is done on program.
→ Independent of language	→ Language dependent
→ Hardware Independent	→ Hardware dependent
→ Time & Space function	→ Watch time & bytes.

Characteristics / Properties of Algo:-

(i) Input:- Algo must take ~~done~~ input algorithm can take 0 or more input

(ii) Output:- Algo must generate at least one output

(iii) Definiteness:- Every statement should have single & exact meaning. We can not write a statement which cannot be understood or cannot be solved.
Eg:- We cannot use $\sqrt{-1}$ (imaginary no.)

(iv) Finiteness:- Algo must terminate i.e. iteration algo must be finite.

(v) Effectiveness:- Unnecessary statements are prohibited in Algo. If we write a statement in Algo it must have some purpose or some procedure.

How to write an Algorithm:-

(1) Swap two nos.

```
Algo swap (a,b)
Begin
    temp := a;
    a := b;
    b := temp;
End
```

Saathi

Date __/__/____

How to analyze an algorithm

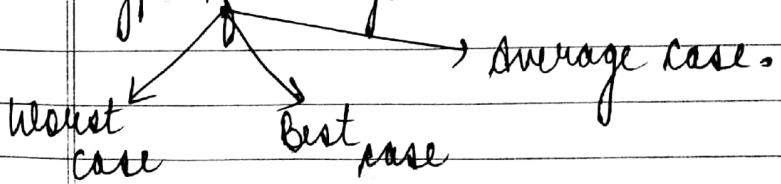
→ Time :- Time taken by algo to execute
→ Space :- Memory space consumed by Algo
→ Network Consumption :- Data transfered (amount)
→ Power :- Power consumed by Algo.
→ CPU registers consumed :- CPU registers consumed by algo.

Goal of the analysis of algorithm :-
The goal of the analysis of algorithms is to compare algorithms mainly in terms of running time but also in terms of other factors (eg → memory, developer, effort. etc).

How to compare algorithms ? :-
→ Execution time :- Not a good measure as execution time are specific to a particular problem.
→ No. of statements executed :- varies
→ Ideal sol^n.

Rate at which running time increases as a function of input is called rate of growth.

Types of analysis.



Lower bound <= Avg time <= Upper bound.

Page No.

Saathi

Date ___ / ___ / ____

Rate of Growth fun^{n} :-

Time complexity	Name
1	constant
$\log n$	Logarithmic
n	Linear
$n \log n$	Linear Logarithmic
n^2	quadratic
n^3	Cubic
a^n	Exponential

Worst case :- Defines the input for which algo takes a long time (slowest time to complete)

Best case :- Defines the input for which algo takes the least time (fastest time to complete)

Average case :- Provides a prediction about the running time of the algo.

Asymptotic Notations :-
Having the expression for the best, average and worst cases, for all three cases having lower bounds & upper bounds.

Saathi

Date __/__/____

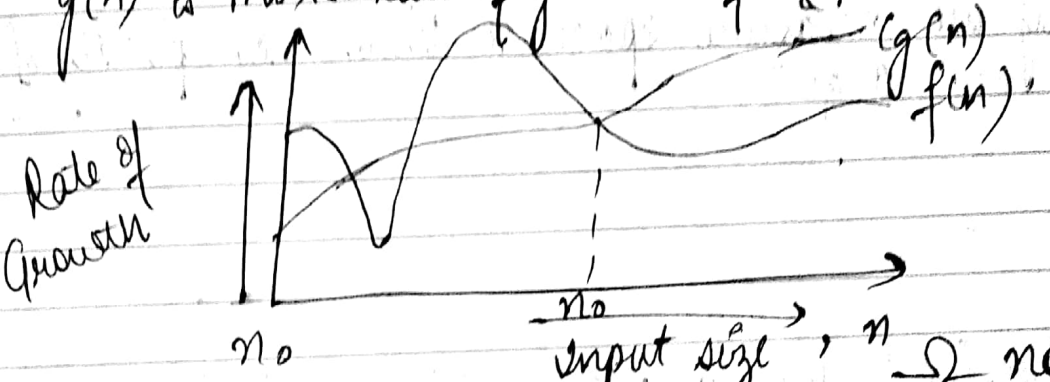
Big-O notation [Upper Bounding fnc?]

Generally it is represented as $f(n) = O(g(n))$ that means, at larger value of n, upper bound of $f(n)$ is $g(n)$.

$f(n) = n + 100n^2 + 10n + 50$

then n^2 is $g(n)$.

$g(n)$ is max. rate of growth for $f(n)$.



$f(n) = an + b$

$f(n) \leq c.g(n)$

After some $n \geq n_0$

$c > 0, n_0 \geq 1$

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

$f(n) = 3n + 2$

$f(n) \leq g(n)$

$3n + 2 \leq 4n$

$c = 4, n_0 = 2$

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

$= O(n)$

Ω notation

Best case

$f(n) = 3n + 2$

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

$3n + 2 \geq c g(n)$

$3n + 2 \geq c(n_0)$

$3n + 2 \geq 1(1)$

$c = 1, n = 1$

Omega Ω Notation

Page No.

Saathi

Continued...

Unit: Asymptotic Notations

(i) Big Oh Notation :-

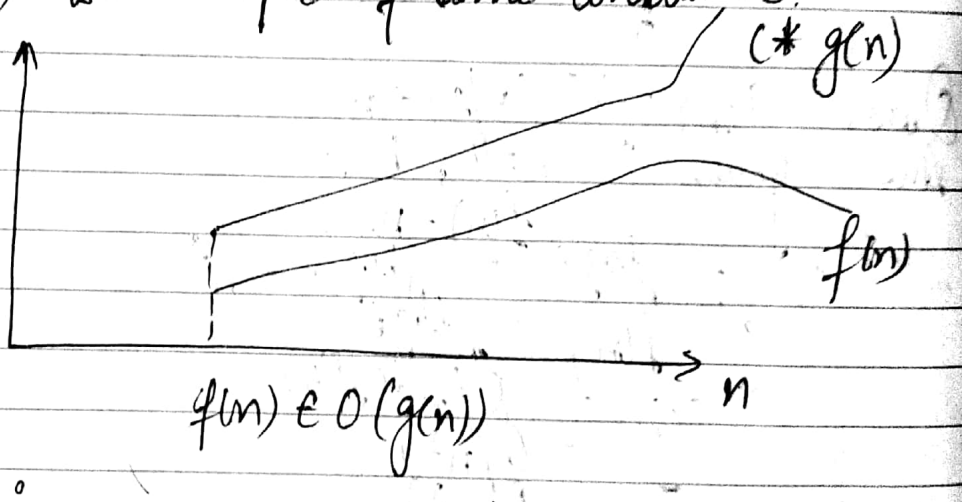
The Big Oh notation is denoted by 'O'. It is method of representing the upper bound of algorithm's running time.

Definition :-

Let $f(n)$ and $g(n)$ be two non-negative functions. Let n_0 and constant c are two integers such that n_0 denotes some value of input & $n > n_0$. Similarly, c is some constant such that $c > 0$. We can write

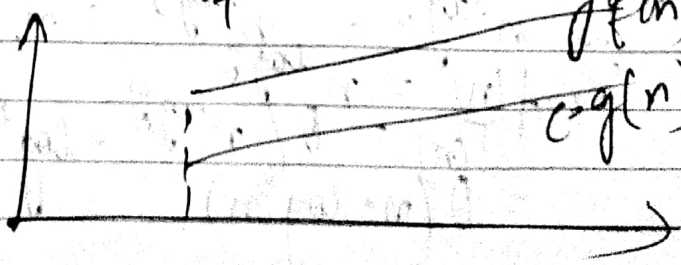
$$f(n) \leq c * g(n)$$

then $f(n)$ is big oh of $g(n)$. It is also denoted as $f(n) \in O(g(n))$ i.e. in other words $f(n)$ is less than $g(n)$ if $g(n)$ is multiple of some constant c.



Omega Notation :-

Omega notation is denoted by 'Ω'. This notation is used to represent the lower bound of algorithm's running time. Using omega notation we can denote shortest amount of time taken by algorithm.



Page No.

Saathi

Date __/__/____

Θ Notation :-

The theta notation is denoted by Θ. By this method the running time is between upper bound and lower bound

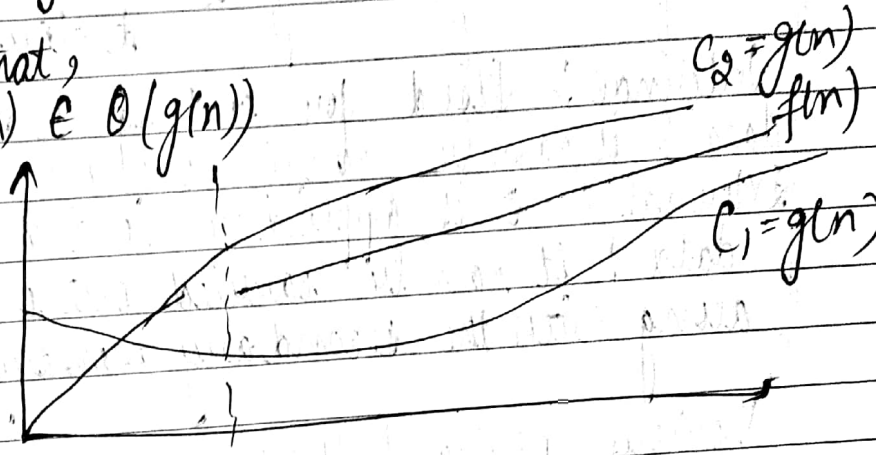
Definition :-

Let $f(n)$ and $g(n)$ be two non negative functions. There are two positive constants.

$$C_1 * g(n) \leq f(n) \leq C_2 * g(n)$$

Then we say that,

$$f(n) \in \Theta(g(n))$$



Recurrence Equation :-

The recurrence equation is an equation that defines a sequence recursively. It is normally in following form -

$$T(n) = T(n-1) + n \quad \text{for } n > 0 \quad -(1)$$

$$T(0) = 0 \quad -(2)$$

Here equation (1) is called recurrence relation & equation (2) is called initial condition. The recurrence equation can have infinite no. of sequences.

Page No.

Saathi

Date ___ / ___ / _____

Space complexity :- The amount of space or memory taken by an algorithm to run as a fun". of the length of input.

Time complexity :- The amount of time taken by a algorithm to run as a fun". of length of the input.

Ques:- Why we measure space & time complexity of an algo?

Ans:-

Recursion :- Any function which calls itself is called recursive. A recursive method solves a problem by calling a copy of itself to work on a smaller problem.

Greedy Method :- An algo based on heuristic that follows local optimal choice at each steps with the hope of finding global optimal sol".

Characteristics of Algo :-

→ precision
→ Uniqueness
→ Finiteness
→ Input
→ Output

Date ___/___/_____

Saathi

Def:- Master's theorem for Divide & conquer:-

$T(n) = aT(n/b) + O(n^k \log^p n)$

$a >= 1$, $b > 1$, $k >= 0$ & p is real no.

1.) if $a > b^k$, then $T(n) = O(n^{\log_b a})$

2.) if $a = b^k$
- if $p > -1$, then $T(n) = O(n^{\log_b a} \log^{p+1} n)$
- if $p = -1$, then $T(n) = O(n^{\log_b a} \log \log n)$
- if $p < -1$, then $T(n) = O(n^{\log_b a})$

3.) if $a < b^k$
- if $p >= 0$ then $T(n) = O(n^k \cdot \log^p n)$
- if $p < 0$ then $T(n) = O(n^k)$

Ques:- $T(n) = 4T(n/2) + n$

Ans:- given :- $a = 4$, $b = 2$

$f(n) = n$

$\therefore n^{\log_b a} = n^{\log_2 4} = n^2$

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

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

$= O(n^2)$

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

Ques:- $T(n) = 2T(n/2) + n$

given :- $a = 2$, $b = 2$

$f(n) = n$

$\therefore n^{\log_b a} = n^{\log_2 2} = n$

$\therefore f(n) = n^{\log_b a}$

$T(n) = O(n^{\log_b a} \cdot \log^{p+1} n)$

$= O(n \cdot \log^1 n)$

Page No.

Saathi

Date __/__/____

Chapter 2:- Basic Algorithm Design techniques

Sorting Algorithm :-

Sorting means arranging the elements in increasing order or decreasing order. The sorting can be of numbers, characters (alphabets), strings or employees record.

Classification of sorting
- Internal sorting
- External sorting

Internal :- Used for sorting reasonable amount of data & it can be carried out on main memory.

External :- It is applied to sort a huge amount of data & it can be carried out on the main memory along with the secondary memory.

Various sorting Methods :-

→ Bubble sort
→ Selection sort
→ Insertion sort
→ Quick sort
→ Merge sort
→ Heap sort
→ Radix sort.

~~Bubble sort~~

Date

complexity → $\Omega(n \log n)$ (time)

Merge sort :-

A	B	C
i ← 2	j ← 5	
8	9	
15	12	(m+n) Elements
18	17	

m → [no. of Elements] ← n.

→ Compare a[i] with b[j]
→ if a[i] < b[j]
then copy it to C[k]
→ Element which is copied there will move it to the next index & index of C will also be moved.
→ Compare a[i] with b[j]
→ Move to next element after copying
This process is Merging

If two lists are already sorted & then the result is sorted

Algo

```
Algo merge (A, B, m, n)        // m, n → size of both lists; A, B → two lists
{
    i=1, j=1, k=1;
    while (i<=m && j<=n)
    {
        if (A[i] < B[j])
        {
            C[k++] = A[i++];
        }
        else
        {
            C[k++] = B[j++];
        }
    }
```

Page No.

Saathi

Date ___/___/_____

```
for ( ; i <= m; i++)
    C[K++] = A[i];
for ( ; j < n; j++)
    C[k++] = B[i];
}
```

→ for Remaining Elements in lists:

Quick Sort :- [follows divide & conquer]

	0	1	2	3	4	5	6	7	8	9
A	10	16	8	12	15	6	3	9	5	∞
	l									max no. [End of list]

pivot = 10 [First Element Initial]

→ will search for element > pivot
will search for element < pivot

Partitioning Procedure :

→ Increment i until you find Element greater then pivot
→ Decrement j until you find Element smaller then pivot

10	16	8	12	15	6	3	9	5	
	i							j	
10	5	8	12	15	6	3	9	16	
			i				j		
10	5	8	9	15	6	3	12	16	
				i		j			
10	5	8	9	3	6	15	12	16	
					j	i			
6	5	8	9	3	10	15	12	16	
					j				

perform Quick Sort recursively

j → partitioning position

Page No.

Date ___ / ___ / _____

Saathi

Complexity => O(nlogn)

Algo :-

```
partition(l, h)
{
    pivot = A[l];
    i = l, j = h;
while (i < j)
{
  do
  {
    i++;
  } while (A[i] <= pivot);
  do
  {
    j--;
  } while (A[j] > pivot);
  if (i < j)
  {
    swap (A[i], A[j]);
  }
  swap (A[l], A[j]);
  return j;
}
Quick Sort (l, h)
{
  if (l < h)
  {
    j = partition (l, h);
    Quick Sort (l, j);
    Quick Sort (j+1, h);
  }
}
```

n time taken

Date __/__/____ → complexity ⇒ Ω(n) Saathi

Insertion sort :-

```
for i = 1 to n
    key ← A[i]
    j ← i - 1
    while j >= 0 and A[j] > key
        A[j+1] ← A[j]
        j ← j - 1
    End while
    A[j+1] ← key
End for
```

Analysis :-

1	2	3	4	5	6	7	8	9	10
2	7	9	4	1	0	5	6	8	3

j = 4

key = A[4] = 4

i = 4 - 1 = 3

while 3 >= 0 and 9 > 4

A[4] = A[8]

A[4] = 9

2	7	9	9	1	0	5	6	8	3

i = 3 - 1 = 2

2 > 0 and 7 > 4

A[3] = 7

2	4	7	9	1	0	5	6	8	3

Worst case

5	4	3	2	1

1 → 1 + 1 = 2

2 → 2 + 2 ≤ 4

3 → 3 + 3 ≤ 6

2(1 + 2 + 3 ... + n)

Page No.

Saathi

Date __/__/____

$\frac{n(n+1)}{2} = O(n^2+n)$

$= O(n^2)$

Best case:- $T_n = C_1 n + C_2(n-1) + C_3(n-1) + C_4(n-1) + 0 + 0 + C_7(n-1)$

$(C_1+C_2+C_3+C_4+C_5)n - (C_2+C_3+C_4+C_4)$

$= an - b$

$= \Omega(n)$

Ques:- When does the worst case of quick sort occur?

Ans:- Worst case occurs when the picked pivot is always an extreme (smallest or largest) element. It happens when input array is sorted or reverse sorted and either first or last element is picked as pivot.

Advantages of Merge sort:-

→ More efficient and works faster than quick sort in case of largest array size or dataset.

→ Merge sort is stable as two elements with equal value appear in the same order in sorted output as they were in the input unsorted array.

→ Merge sort is preferred for link list

→ It has a consistent speed on any size of data.

Disadvantages of Merge sort:-

→ The worst case time complexity could be quadratic

→ Requires additional memory to sort the elements

Page No.

Saathi

Date __/__/____

Merge sort Vs Quick Sort :-

Merge sort	Quick sort
→ The array is parted into just 2 halves (i.e. n/2)	→ The array is parted into any ratio. No compulsion of dividing the array of element into equal parts.
→ Avg & Worst case complexity are same i.e. O(nlogn)	→ Worst case complexity of quick sort is O(n^2)
→ More efficient & faster	→ Less efficient & faster
→ More stable	→ Less stable
→ Preferred for ~~arrays~~ linked list	→ Preferred for arrays.
→	

Ques:- Why Quick sort is preferred for arrays & merge sort for linked list?

Ans:- The merge operations of merge sort can be implemented without extra space for linked lists.
In arrays, we can do random access as elements are continuous in memory.
Therefore the overhead increases for quick sort. Merge sort accesses data sequentially & the need of random access is low.

Ques:- Does merge sort has same best and worst case? If yes. Why?

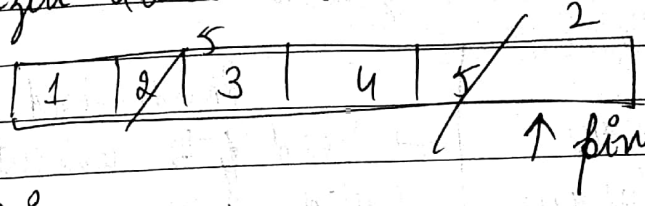
Ans:- Merge sort is a stable sort which means that the same element in an array maintain their original positions with respect to each other.
Overall time complexity of Merge sort is O(nlogn). It is more efficient as it is in worst case also the runtime is O(nlogn).

Page No.

Saathi

Date __/__/____

Randomized Quick sort :-



i j

Algo :-

```
Randomized Quick sort (A, p, r)
    if (p < r)
        i ← random(p, r);
        swap (A[p], A[r]);
        q ← partition (A, p, r);
        Randomized quick sort (A, p, q-1);
        Randomized quick sort (A, q+1, r);
```

Time Complexity = $O(n(\log n))$

Need of Randomized Quick sort :-

→ Used in many applications in algorithms, data structures and computer science!
→ In some cases only known algorithms are randomized or randomness is provably necessary.
→ Often randomized algos are simpler and efficient
→ Several deep connections to mathematics, physics etc.
→ Lots of fun!

Saathi

Date __/__/____

Quick sort V/s Randomized Quick sort

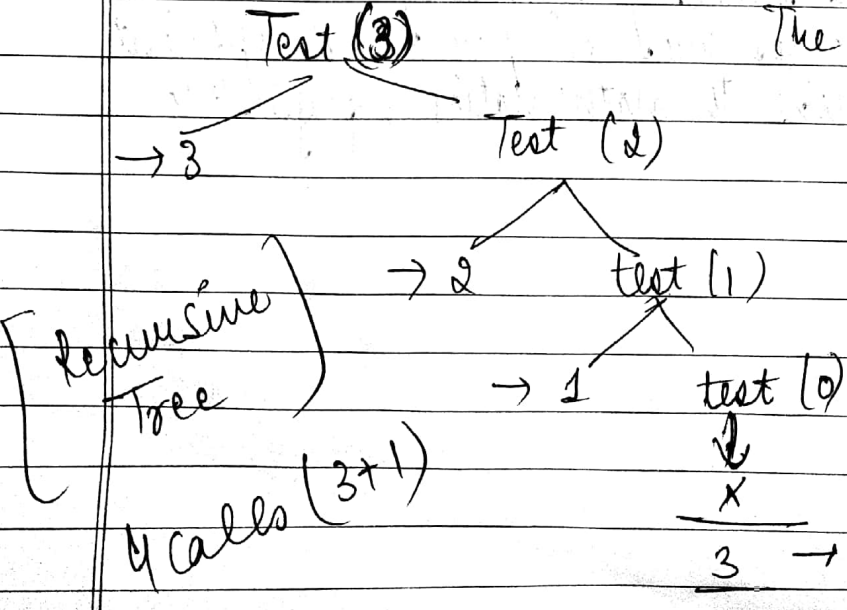
Quick sort	Randomized Quick sort
→ Used for sorting the unsorted array	→ Used to unsort the sorted array.
→ More efficient	→ less efficient
→ More stable	→ less stable
→ Worst case = $O(n^2)$	→ Worst case = $O(n \log n)$

Recurrence Relation :-
It is an equation or inequality that describes a function in terms of its values on smaller inputs.

```
void test (int n)
{
    if (n>0)
    {
        cout << n;
        test (n-1);
    }
}
```

→ 1 unit of time in each call

The amount of work done is printed a



Saathi

Date __/__/____

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

(unit of time) 1 + T(n-1) } (Total Time)

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

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

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

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

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

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

Substitute $T(n-1)$

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

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

Substitute Again

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

$= T(n-3) + 3$

; continue for k times

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

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

Assume $n - k = 0$

$\therefore n = k$

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

$T(n) = T(0) + n$

$T(n) = 1 + n$

$O(n)$

$O(n)$ → Linear loop

Page No.

Saathi

Date __/__/____

Design And Analysis Of Algorithms.

Part II

Chapter 1 :- Searching & Sorting.

Linear Search in Ordered Array :-

Sequential search is made over all items one by one.

O(n)

4	5	10	12	16	24	33	42
0	1	2	3	4	5	6	7

Algo :-

1) i = 1 → max. size of list
2) if i > n, go to step 7
3) if A[i] = x, go to step 6
4) i = i + 1
5) go to step 2
6) Print Element i found at i
7) Print Element not found
8) Exit

i = 0, x = 16
A[i] = A[0] = 4
A[i] = x or whether 4 = 16 [not]
i = 0 + 1 = 1
A[1] = 16 X
i = 2, A[2] = 16, X
i = 4, A[4] = 16 ✓
Element 16 found at 4.
exit

Date __/__/____

Binary Search in Ordered Array :- → O(log n) saathi

→ Fast search algo with time complexity of O(log n)
→ Based on divide & conquer
→ Data item should be sorted order

Eg :- Find location of value 44

10	14	19	26	31	42	44
0	1	2	3	4	5	6

→ V = 44

Algo :-

```
while low <= high
    do mid ← (low + high)/2
    if V = A[mid],
        return mid
    if V > A[mid]
        low ← Mid + 1
    else
        high ← mid - 1
return Nil
```

loop ↑ ↓

V = 44
44 > 26
low = mid + 1
= 3 + 1 = 4
mid = 4 + 6 / 2 = 5
low = mid + 1 = 6
high = mid - 1 = 4
location = 3

Hashing :-
Used for Searching

Key = 8, 3, 13, 6, 4, 10

10 ⇒ Value of element as the index

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

A :

lot of space is wasted in this method

Key = 10

key = 12

Page No.

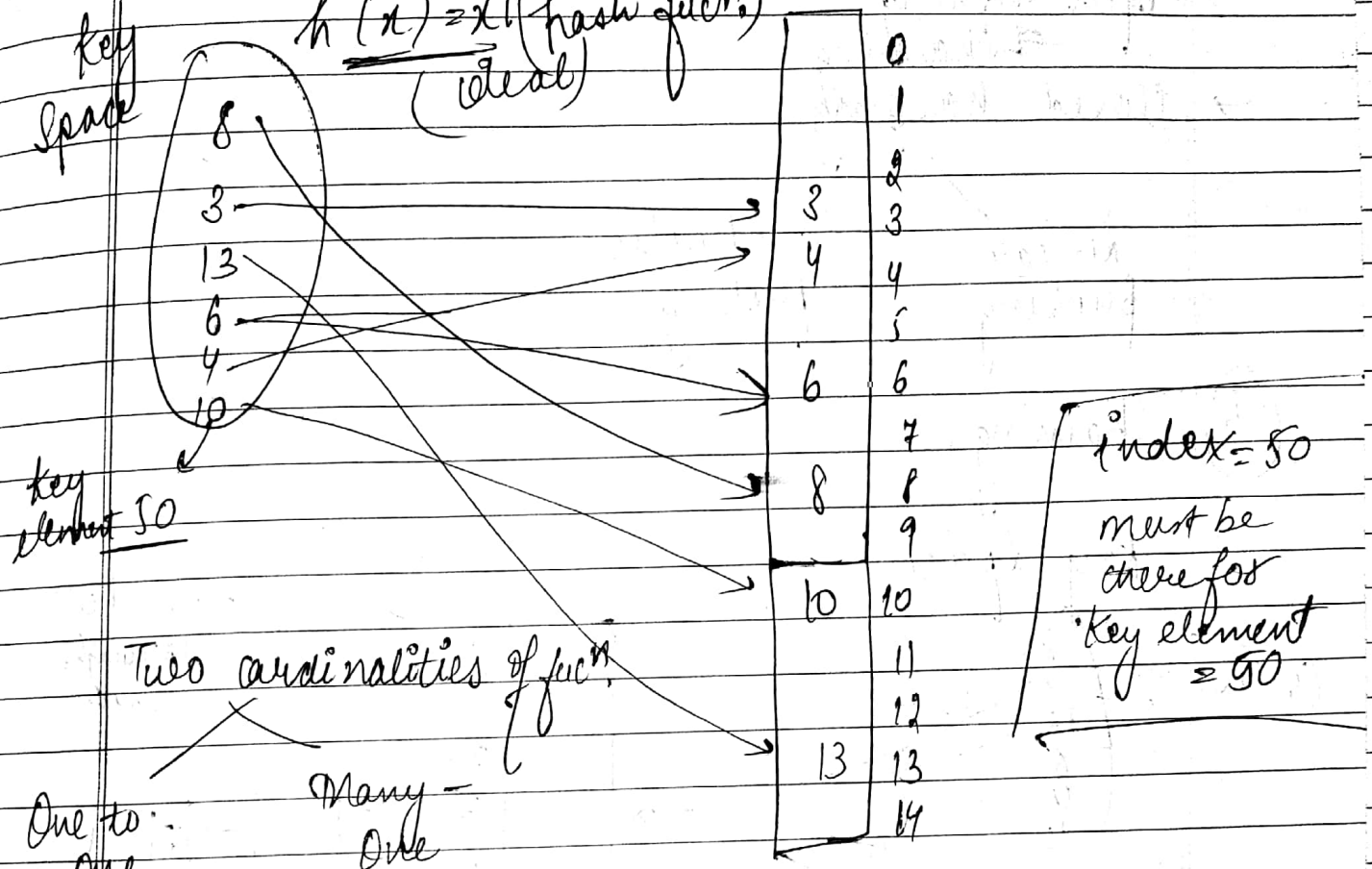
Saathi

Date __/__/____

For few elements we need very large size array where most of the spaces are vacant.

$h(x) = x$ (hash fucn.)

(ideal)



Two cardinalities of fucn.

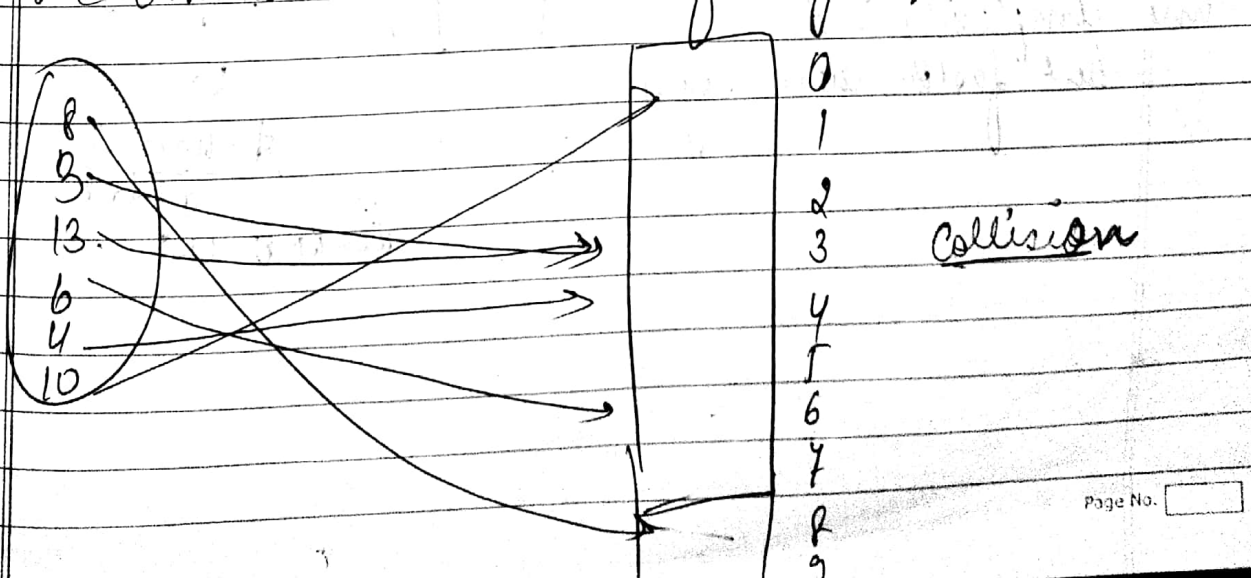
One to one

Many - One

$h(x) = x$ → Responsible for Maping elements.

$h(x) = x \% 10$ → {size of hash table}

We can take hash table of any size ie $h(x) = x \%$ size



Page No.

Date ___/___/___

Saathi

Collision resolving Methods

→ Open hashing / Chaining

→ Closed hashing

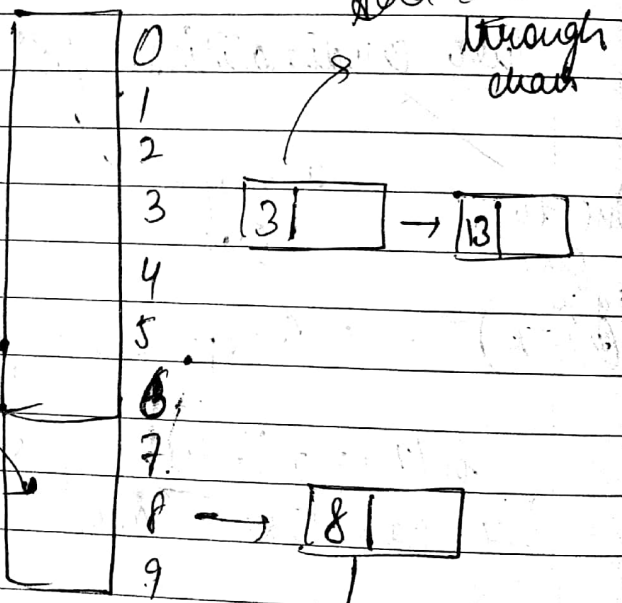
- Linear probing
- Quadratic probing

1. Chaining.

key space

8
3
13
6
4
10

Collision



key = 13

13 mod 10 = 3.

Time complexity > 1
but faster than log n

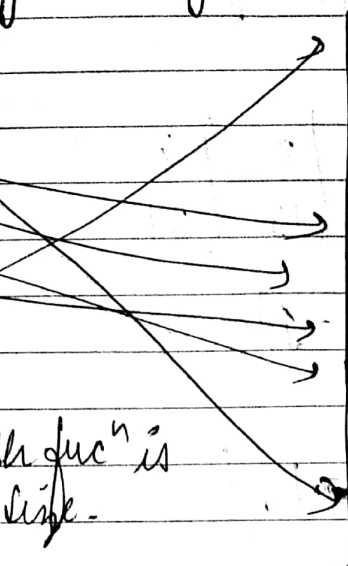
A chain is added
Linked list

Date ___/___/_____ saathi

Linear Probing

key = 4 ✓ found
key = 94 ✗ not found

key space: 8, 3, 13, 6, 4, 10



find next free space & store the element in next free space.

The actual hash fun^n is
$h(x) = x \% \text{size}$.

A new hash fun^n is $= [h(x) + f(i)] \% \text{size}$

$f(i) = i$

$i = 0, 1, 2, \ldots$

$h'(13) = [h(13) + f(0)] \% 10$

$= 3.$

$h'(13) = [h(13) + f(1)] \% 10$

$= 4$

Chapter 2:- Sorting Algorithms

Heap sort :-

Height	1	2	3	4
Nodes	3	7	15	31

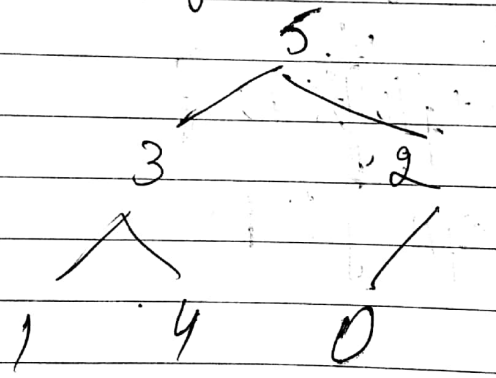
$$\text{Nodes } (n) = (2^{h+1} - 1)$$

$$h = \lceil \log n \rceil$$

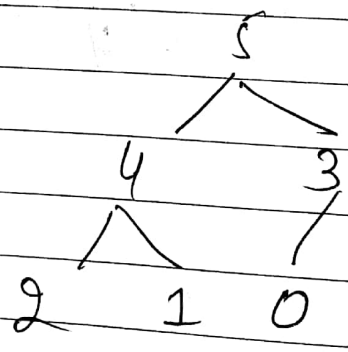
5	3	2	1	4	0

Left child $= 2i$

Right child $= 2i + 1$



5	4	3	2	1	0

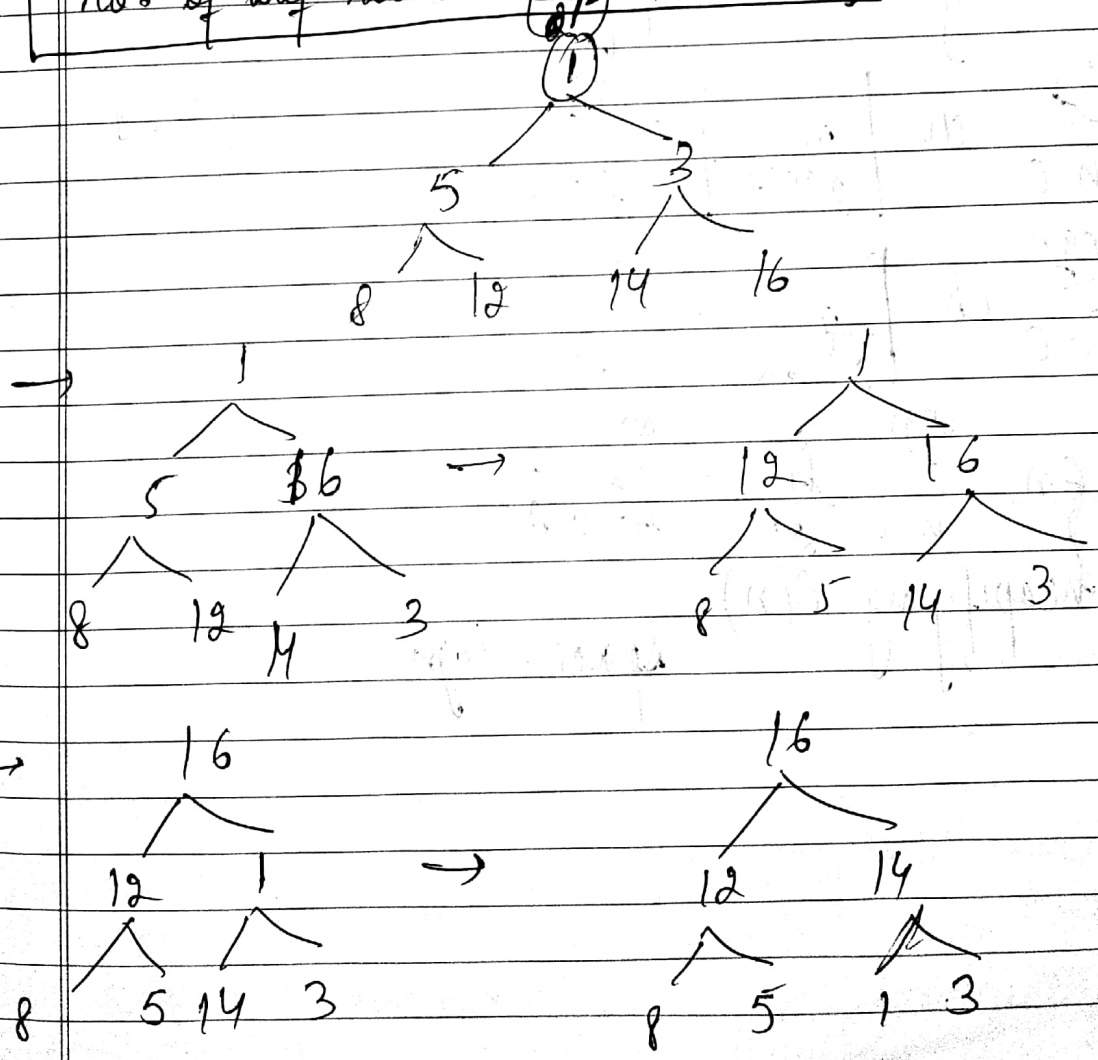


Saathi

Date: 1/1/

```
max heapify (A, i)
{
  l = 2i;
  r = 2i + 1;
  if (l ≤ A.heapsize and A[l] > A[i])
      largest = l;
  else largest = i;
  if (r ≤ A.heapsize and A[r] > A[largest])
      largest = r;
  if (largest ≠ i)
    exchange A[i] with A[largest]
    max_heapify (A, largest);
}
```

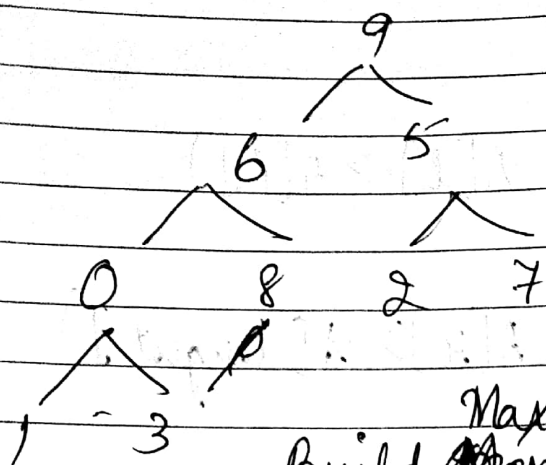
No. of leaf nodes = (n/2) + 1



Page No.

Date ___/___/_____

9	6	5	0	8	2	7	1	3



Max
Build ~~Max~~ heap (A)
{
A heap size = A length
for (i = LA. length /2 down to 1)
max_heapify (A, i)
}

$$\sum_{h=0}^{\log n} ch \left| \frac{n}{2^{h+1}} \right| < \sum_{h=0}^{\infty}$$

$2^n, 2^1$

$$\sum_{h=0}^{\log n} ch \left| \frac{n}{2^h \cdot 2} \right| < \sum_{h=0}^{\infty}$$

$$\frac{cn}{2} \sum_{h=0}^{\log n} \frac{h}{2^n} < \sum_{h=0}^{\infty} \frac{h}{2^h}$$

heapifying $O(n)$

Space = log n

Date __/__/____

Saathi

~~Counting Sort~~:- Features of heap sort :-

→ The Time complexity heap sort is $O(n \log n)$

→ For random input it works slower than quick sort.

→ Heap sort is not a stable sorting method.

→ The space complexity of heap sort is $O(1)$. as it does not require any extra storage to sort.

Bucket Sort :-

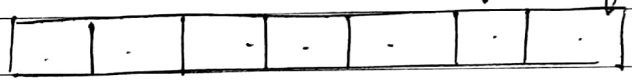
~~Algo~~ Steps :-

1. Set up an array of initially empty buckets.
2. Put each element in corresponding bucket.
3. Sort each non empty bucket
4. Visit the bucket in order & put all the elements into a sequence & print them.

Technique

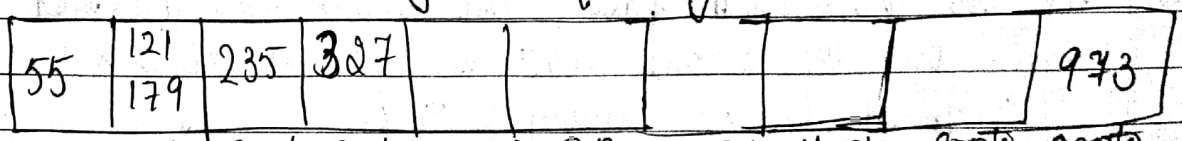
→ Sort the following elements in ascending order.
121, 235, 55, 973, 327, 179

1. We will set up an array. as follows



0 to 100	100 - 200	200 - 300	300 - 400	400 - 500	500 - 600	600 - 700

2. Fill each bucket by corresponding element in list.



0 to 100	100 to 200	200 to 300	300 to 400	400 to 500	500 to 600	600 to 700	700 to 800	800 to 900	900 to 1000

3. Now visit the bucket & sort it individually.

Page No.

Date ___/___/_____

Saathi

55, 124, 179, 235, 387, 973.

Algorithm:-

```
void bucketsort (int a[], int n, int max)
{
    int i, j=0;
    int * buckets = calloc (max+1, sizeof(int));
    for (int i=0; i<n; i++)
        buckets[a[i]] ++;
    for (i=0; i<max; i++)
        while (buckets[i]--)
            b[j++] = i;
```

Drawbacks :-

1. For bucket sort the maximum value of the element must be known.
2. We must have to create enough buckets in the memory for every element to place in the array.

Radix Sort

→ Radix sort works by sorting on the least significant digits first.

→ On the first pass, all the numbers are sorted on the least significant digit and combined in an array.

→ Then on the second pass, the entire no. are sorted again on the second least significant digits & combined in an array & so on.

Page No.

Saathi

Date __/__/____

Technique:-
Digit by digit sorting

1. Sort the fol^n. 45, 37, 05, 09, 06, 11, 18, 97.

Now sort element according to last digit

Last digit	0	1	2	3	4	5	6	7	8	9	
Element		11				45 05	06	37 97	18	09	

2. Now sort array with the help of second last digit.

Second last	Element
0	05, 06, 09
1	11, 18.
2	97
3	37
4	45
5	
6	
7	
8	
9	

Elements are of two digits only so we will stop comparing. Now whatever list we have got is of sorted elements.

05, 06, 09, 11, 18, 97, 37, 45.

Insertion Sort Algo:-

```
Insertion (arr)
{ for (j = 2 to length[Arr])
  { key = arr[j];
    i = j - 1;
    while(i>0 and arr[i] > key)
    { Arr[j+1] = Arr[i];
      i - = 1;
    }
Arr[i+1] = key; } }
```

Best case complexity of insertion sort is $\Omega(n)$

Average case complexity of Insertion sort is $O(n^2)$

Worst case complexity of insertion sort is $O(n^2)$

Page No.

Date ___/___/___

Saathi

Counting Sort :-

It is not a comparison sort algo.
It gives O(n) complexity for sorting

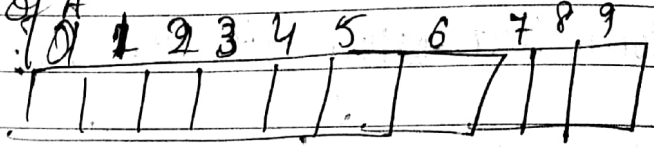
Analysis & Algo :-

Array of 15 values.

A → | 3 | 9 | 4 | 5 | 3 | 6 | 8 | 5 | 4 | 8 | 7 | 4 | 6 | 9 | 7 |
0 1 14

B → same length as that of A

C → k length array.



1. Initialise array C.

```
for (i=0; i<k; i++)
    C[i] = 0;
```

2. count occurance of every value. and after that the count of occurance will be filled in array C.

| | 1 | 1 | 1 | 3 | 3 | 2 | 2 | 2 | 2 |
0 1 2 3 4 5 6 7 8 9

3.
```
for (i=1; i<k; i++)
    C[i] = C[i] + 1;
```

4.
```
for (i=1; i<k; i++)
    C[i] = C[i] + C[i-1];
```
→ [values of i or less than i]

| | | | 1 | 4 | 7 | 9 | 11 | 13 | 15 |
0 1 2 3 4 5 6 7 8 9

Saathi

Date ___ / ___ / _____

5. B =

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

```
for (j = N-1; j >= 0; j--){
    B[C[A[j]] - 1] = A[j];
    C[A[j]] = C[A[j]] - 1;
```

Saathi

Date ___/___/_____

Chapter 3 :- Dynamic Programming

Longest common Subsequence (LCS) :-

Consider two strings

String 1 : a b c d e f g h i j
String 2 : c d g i

abcdefghi
cdgi

We need to find out set of characters common in sequence.

[lines should not intersect with each other]

cdgi, dgi, gi, cd, dg

Algo :-

```
int LCS (char=X, char=Y, int m, int n)
{
   if (m==0 || n==0)
      return 0;
   if (X[m-1] == Y[n-1])
      return 1 + LCS(X, Y, m-1, n-1)
   else
      return max( LCS(X, Y, m, n-1)
                  LCS(X, Y, m-1, n))
int max (int a, int b)
{
   when (a>b)? a:b

int main ()
{
   char x[] = "AGGTAB";
```

Saathi

Date __/__/____

```
char X[] = "GXTXAYB";
    int m = strlen(X);
    int n = strlen(Y);
    cout << "length of LCS", LCS(X, Y, m, n));
    return 0;
```

Method / Technique :-

(1) Str 1 :- STONE [Rows]
Str 2 :- LONGEST [Columns]

		L	O	N	G	E	S	T
	0	1	2	3	4	5	6	7
S 0	0	0	0	0	0	0	0	0
T 1	0	0	0	0	0	0	1	1
O 2	0	0	0	0	0	0	1	2
N 3	0	0	1	1	1	1	1	2
E 4	0	0	1	2	2	2	2	2
	0	0	1	2	2	3	3	3

O N E → 3

For match add 1 to diagonal.
For different take max diagonal wise

(2) Str 1 :- b d. [Rows]
Str 2 :- a b c d [Columns]

		a	b	c	d
	0	1	2	3	4
a 0	0	0	0	0	0
b 1	0	0	1	1	1
d 2	0	0	1	1	2
		a	b		d → 2

Page No.

Date __/__/____

Saathi

~~Matrix Chain Multi~~ → Principal of Optimal

Dynamic Programming :- It is an algo paradigm that solves a given complex program by breaking it into sub problems and storing the results of sub problems to avoid computing the same results again & again.

Conditions :- optimal sub structure
- overlapping subproblem.

~~Overlapping~~ &
Matrix Chain Multiplication :-

Matrix multiplication

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}_{2\times 3} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}_{3\times 2}$$

condition :-
→ No. of columns of first matrix must be equal of rows of second matrix.
→ The dimension of product will be no. of rows of X no. of columns of second
→ Multiply rows of first with columns of Second.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$C = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21} & \end{bmatrix}_{2\times 2}$$

Page No.

Date ___/___/_____

Saathi

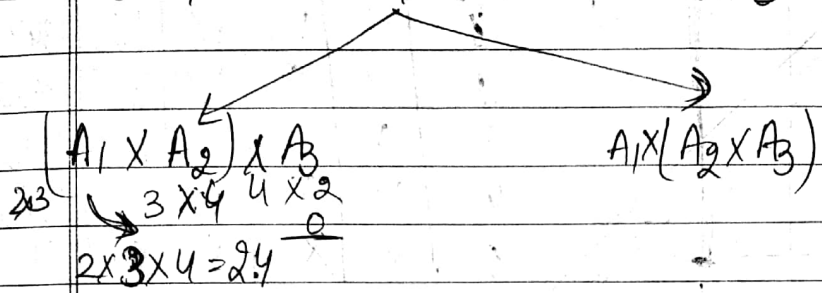
The no. of multiplications reqd → 2×3×2 = 12

$A_1 \times A_2 \times A_3$

$2\times3 \quad 3\times4 \quad 4\times2$

$d_0\ d_1 \quad d_1\ d_2 \quad d_2\ d_3$

Possible Methods



$2\times3\times4 = 24$

$$M[i,j] = \begin{cases} 0 & \text{if } i=j \end{cases}$$

Define goal :- Find optimal way to multiply these matrices to perform fewest multiplication

Bottom Up Approach

1.) check if the problem have optimal substructure & overlapping subproblem.

2.) Define recursive Formula $\quad i \le k < j$

Formula

$$M[i,j] = \begin{cases} 0 & \text{if } i=j \\ M[i,k] + M[k+1,j] + p_{i-1} p_j p_k \end{cases}$$

Page No.

Function chain Matrix Algorithm

Step 1:- Initialization loop for (i → 1 to n)
Set mmul [i , j] ← 0 // i = j

Step 2. loop, length of subsequence .
for length ← 2 to n
{
for i ← 1 to n - length + 1
Set j ← i + length - 1
Set mmul [i , j] ← ∞
for k ← i to j - 1
{ Set q ← mmul [i , k] + mmul [k+1 , j]
P[i-1] x P[k] x P[j] }

if [q < mmul[i,j] then
set mmul[i,j] ← q
set s[i,j] ← k
mmul[i,j] ← s[i,j]

Step 3 Date Return

1. A1 A2 A3 A4 A5
4x10 10x3 3x12 12x20 20x7
p_0 p_1 p_1 p_2 p_2 p_3 p_3 p_4 p_4 p_5

i/j	1	2	3	4	5
1	0	120	264	1080	1344
2	X	0	360	1320	1350
3	X	X	0	720	1140
4	X	X	X	0	1680
5	X	X	X	X	0

(i) k=1
(a) M[1,2] = M[1,1] + M[2,2] + $p_0 p_1 p_2$
= 0+0+4x10x3
= 120

(ii) k=2
M[2,3] = M[2,2] + M[3,3] + $p_1 p_2 p_3$
= 0+0+10x3x12
= 360

(iii) k=3
M[3,4] = M[3,3] + M[4,4] + $p_2 p_3 p_4$
= 0+0+3x12x20
= 720

(iv) k=1
M[1,3] = M[1,1] + M[2,3] + $p_0 p_1 p_3$
= 0+360+4x10x12
= 360+480
= 840

Date __/__/____

Saathi

$k=2$

$M[1,3] = M[1,2] + M[3,3] + p_0 p_2 p_3$

$= 120 + 0 + 4 \times 3 \times 12$

$= 120 + 0 + 144$

$= 264$

(v) $K=2$

$M[2,4] = M[2,2] + M[3,4] + p_1 p_2 p_4$

$= 0 + 720 + 10 \times 3 \times 20$

$= 720 + 600$

$= 1320$

$K=3$

$M[2,4] = M[2,3] + M[4,4] + p_1 p_3 p_4$

$= 360 + 0 + 10 \times 20 \times 12$

$= 360 + 2400$

$= 2760$

(vi) $K=4$

$M[4,5] = M[4,4] + M[5,5] + p_3 p_5 p_4$

$= 0 + 0 + 20 \times 12 \times 7$

$= 1680$

(vii) $K=3$

$M[3,5] = M[3,3] + M[4,5] + p_2 p_3 p_5$

$= 0 + 1680 + 3 \times 20 \times 12 \times 7$

$= 1680 + 720$

$= 2400$ ~~0~~ 19320

$K=4$

$M[3,5] = M[3,4] + M[5,5] + p_3 p_4 p_5$

$= 720 + 0 + 3 \times 7 \times 20$

$= 720 + 420 = 1140$

Page No.

Date __/__/____

Saathi

(viii) $k = 1$

$M[1,4] = M[1,1] + M[2,4] + p_0 p_1 p_4$

$= 0 + 1320 + 4 \times 10 \times 3$

$= 1320 + 120$

$= 1440$

$k = 2$

$M[1,4] = M[1,2] + M[3,4] + p_0 p_2 p_4$

$= 120 + 720 + 4 \times 3 \times 20$

$= 120 + 720 + 240$

$= 1080$

$k = 3$

$M[1,4] = M[1,3] + M[4,4] + p_0 p_3 p_4$

$= 264 + 0 + 4 \times 18 \times 20$

$= 264 + 0 + 960$

$= 1224$

(ix) $k = 2$

$M[2,5] = M[2,2] + M[3,5] + p_1 p_2 p_5$

$= 0 + 1140 + 10 \times 3 \times 7$

$= 1140 + 210$

$= 1350$

$k = 3$

$M[2,5] = M[2,3] + M[4,5] + p_1 p_3 p_5$

$= 360 + 1680 + 10 \times 12 \times 7$

$= 360 + 1680 + 840$

$= 2880$

Page No.

Date __/__/____

Saathi

$k=4$

$M[2,5] = M[2,4] + M[5,5] + p_1 p_4 p_5$

$= 1320 + 0 + 10 \times 20 \times 7$

$= 1320 + 0 + 1400$

$= 2720$

(X) $k=1$

$M[1,5] = M[1,1] + M[2,5] + p_0 p_1 p_5$

$= 0 + 1350 + 4 \times 10 \times 7$

$= 1350 + 280$

$= 1630$

$k=2$

$M[1,5] = M[1,2] + M[3,5] + p_0 p_2 p_5$

$= 120 + 1140 + 4 \times 3 \times 7$

$= 120 + 1140 + 84$

$= 1344$

$k=3$

$M[1,5] = M[1,3] + M[4,5] + p_0 p_3 p_5$

$= 264 + 1680 + 4 \times 12 \times 7$

$= 264 + 1680 + 336$

$= 2280$

$k=4$

$M[1,5] = M[1,4] + M[5,5] + p_0 p_4 p_5$

$= 1080 + 0 + 4 \times 20 \times 7$

$= 1080 + 560$

$= 1640$

Saathi

Date __/__/____

K table:

K	1	2	3	4	5
1		2	2	2	2
2			2	2	2
3				3	4
4					4
5					

Parenthesization

$((A_1 \quad A_2) \quad (A_3 \quad A_4)) \quad (A_5)$

(a) $A_1 \times A_2 \times A_3 \times A_4$

3 2 2 4 4 2 2 5

$p_0 \; p_1 \;\; p_1 \; p_2 \;\; p_2 \; p_3 \; p_3 \; p_4$

(i) $k = 1$

$M[1,2] = M[1,1] + M[2,2] + p_0 p_1 p_2$

$= 0 + 0 + 3 \times 2 \times 4$

$= 24$

(ii) $k = 2$

$M[2,3] = M[2,2] + M[3,3] + p_1 p_2 p_3$

$= 0 + 0 + 2 \times 4 \times 2$

$= 16$

(iii) $k = 3$

$M[3,4] = M[3,3] + M[4,4] + p_2 p_3 p_4$

$= 0 + 0 + 4 \times 2 \times 5$

$= 40$

(iv) $k = 1$

$M[1,3] = M[1,1] + M[2,3] + p_0 p_1 p_3$

$= 0 + 16 + 3 \times 2 \times 2$

$= 16 + 12 = 28$

i/j

i \ j	1	2	3	4
1	0	24	28	58
2	X	0	16	36
3	X	X	0	40
4	X	X	X	0

k

k	1	2	3	4
1		1	1	3
2			2	2
3				3
4				

Page No.

Date __/__/____

Saath

K=2

$M[1,3] = M[1,2] + M[3,3] + P_0 P_2 P_3$

$= 24 + 0 + 3 \times 4 \times 2$

$= 24 + 0 + 24$

$= 48$

(v) K=2

$M[2,4] = M[2,2] + M[3,4] + P_1 P_2 P_4$

$= 0 + 40 + 2 \times 4 \times 5$

$= 40 + 40$

$= 80$

K=3

$M[2,4] = M[2,3] + M[4,4] + P_1 P_3 P_4$

$= 16 + 0 + 2 \times 2 \times 5$

$= 16 + 20$

$= 36$

(vi) K=1

$M[1,4] = M[1,1] + M[2,4] + P_0 P_1 P_4$

$= 0 + 36 + 3 \times 2 \times 5$

$= 36 + 30 = 66$

K=2

$M[1,4] = M[1,2] + M[3,4] + P_0 P_2 P_4$

$= 24 + 40 + 3 \times 4 \times 5$

$= 24 + 40 + 60$

$= 124$

K=3

$M[1,4] = M[1,3] + M[4,4] + P_0 P_3 P_4$

$= 28 + 0 + 3 \times 2 \times 5$

$= 28 + 30 = 58$

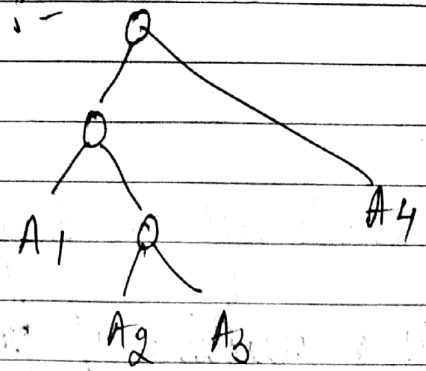
Date __/__/____

Saathi

Parenthesization

$((A_1)(A_2 A_3))(A_4)$

fig:-



0/1 knapsack problem:-

W → Maximum capacity

Total wt = 7

wt	val
1	1
3	4
4	5
5	7

Some of values is max but some of wt. should <= Total wt.

Total columns => Total wt +1

Val	wt	0	1	2	3	4	5	6	7
(1)	1	0	1	1	1	1	1	1	1
(2)	3	0	1	1	4	5	5	5	5
(3)	4	0	1	1	4	5	6	6	9
(4)	5	0	1	1	4	5	7	8	9

No. of values = Total items

max (4 + T0)] [0]

3-3

Top columns

Sort item by value/wt. in a non increasing order

max [4 + T(1) [0]] = 4

max(1, 4)

Page No.

Saathi

Date __/__/____

Design And Analysis Of Algorithms

Part III

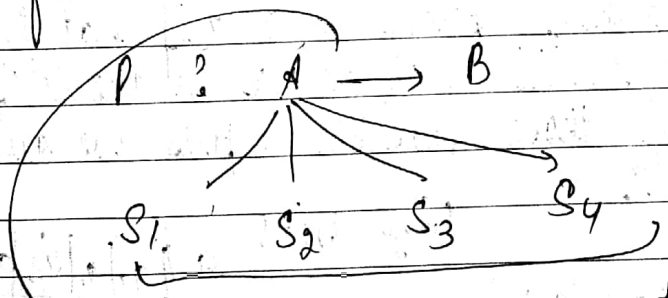
Chapter :- Greedy Strategy

Greedy method :- Approach to solve optimization problems

The problems which demands / requires either minimum results or maximum results.

P : A ⟶ B

Suppose there be problem P which says, it has to travel from location A to location B.



Multiple sol^n to reach from A to B.

But there may be a condition of covering the journey within 12 hrs.

The sol^n satisfying condition of problem are called feasible sol^n

Saathi

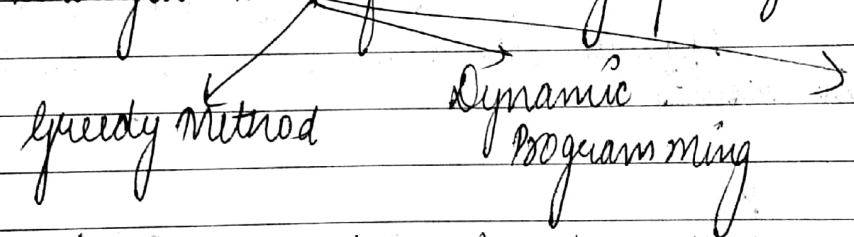
Date ___/___/_____

If the problem demands that result should be minimum it called minimization problem.
ie minimum time reqd. to reach the destination ie from A to B.

A sol^n which is already feasible and gives best results is called optimal sol^n. For any problem there can be only 1 optimal sol^n.

If a problem requires either minimum or maximum results, that type of problem is known as optimization problem.

Strategies used for solving optimization problems.



Greedy Method :- It says that a problem should be solved in stages. In each state we will consider one input from a given problem and if that input is feasible then we will include it in the sol^n. By including all those feasible inputs we will get a optimal sol^n.

(Size) $n = 5$

a

a_1	a_2	a_3	a_4	a_5
1	2	3	4	5

Algo :- Algorithm Greedy (a, n)

```
{
  for i = 1 to n do
  {
    x = Select(a);
    if Feasible(x) then
      sol^n = sol^n + x;
  }
}
```

Page No.

Date ___ / ___ / _____

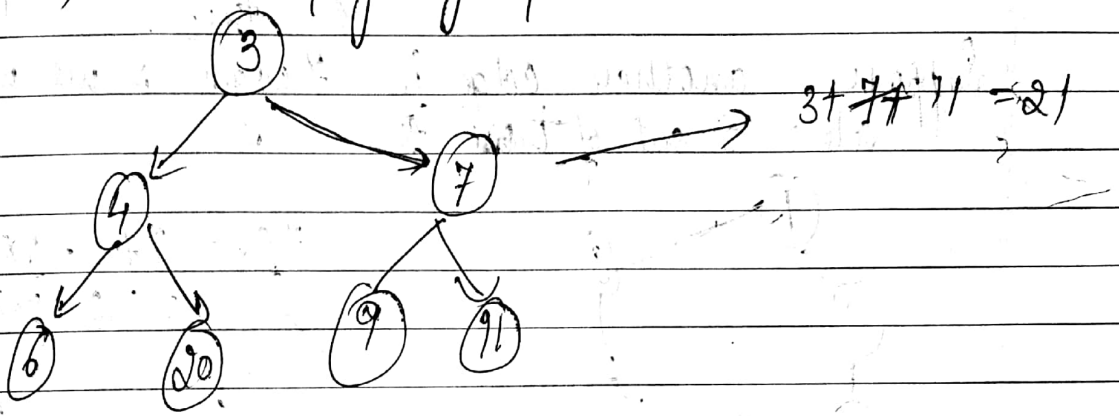
Saathi

~~Minimum Cost Spanning tree~~

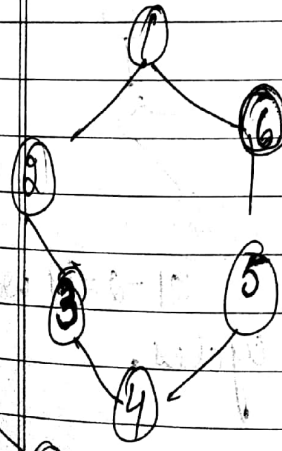
Greedy algo :- It is an algorithmetic paradigm that follows problem solving approach of making local optimal choice at each stage with hope of finding a global optimal sol^n.

pros :- 1) Simple and easy to implement.
2) Runs fast

Cons :- 1) Not sure of getting optimal sol^n.

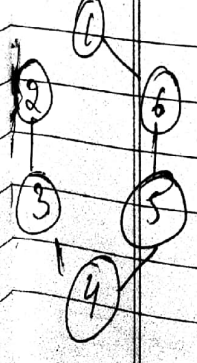


Minimum Cost Spanning tree

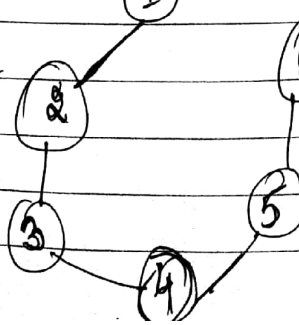


$G = (V, E)$
$V = \{1, 2, 3, 4, 5, 6\}$
$E = \{(1,2), (2,3), (3,4) \ldots\}$

$|V| = n = 6$
$n-1 = 5$ (edges)



(take any)

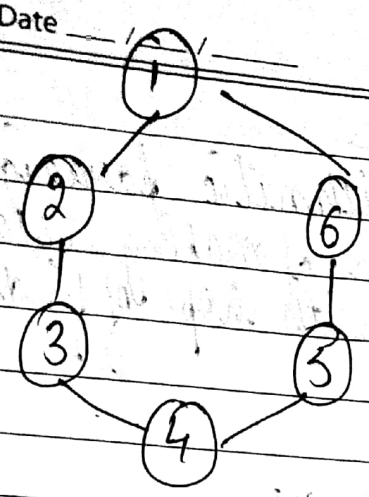


→ Spanning tree

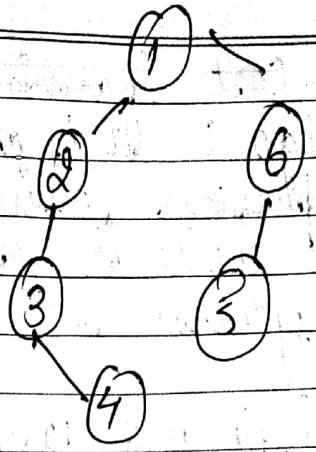
$S \subseteq G$
$S = \{(V', E') \mid V' = V, E = |V| - 1\}$

Date

Saathi



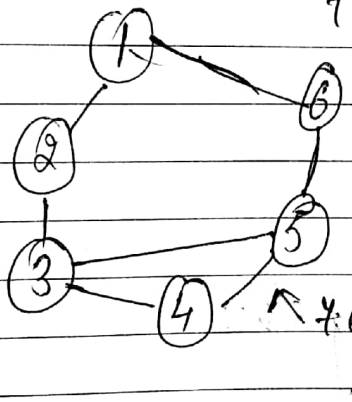
$|E| = 6$



${}^6C_5 = \frac{6!}{5! \times 1!} = \frac{6 \times 5!}{5!} = 6$

6 ways to make a spanning tree.

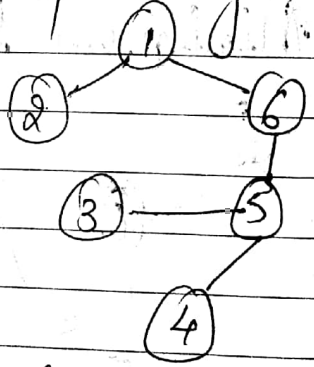
Suppose another edge is 7 edges & we want to select 5 out of those 7.



↖ 7 edges

${}^7C_5 = \frac{7!}{5! \times 2!} = \frac{7 \times \overset{3}{6} \times 5!}{5! \times 2 \times 1} = 21$ ways

Spanning tree



${}^7C_5 - 2 = 21 - 2 = 19$ ways

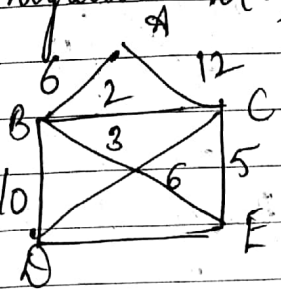
$|E|C_{|V|-1}$ - no. of cycles.

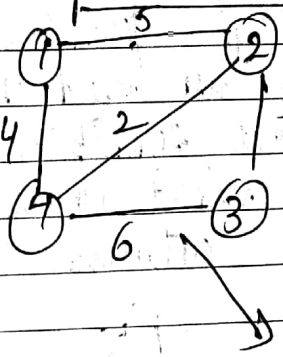
Saathi

Date ___ / ___ / _____

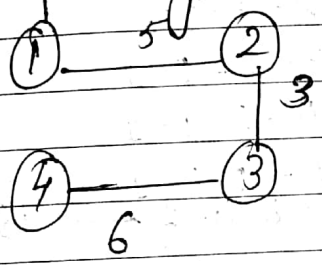
Weighted Graph / Labelled Graph :- A graph G is called labelled graph if its edges or vertices are assigned data of one kind or another.
In other particular, if each edge e of G assigned a non-negative $l(e)$ is called **positive Number**.

Eg :-

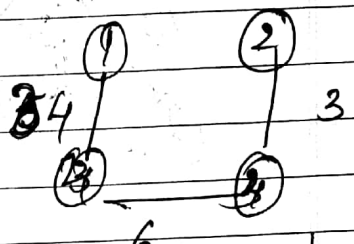




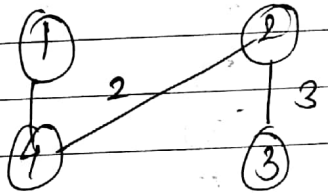
Spanning Tree.



cost = 5+3+6
= 14



cost = 4+6+3
= 13



cost = 4+2+3
= 9.

Methods of finding Minimum cost Spanning tree :-

→ Try all spanning trees and find the minimum one.
→ Greedy Methods
- Prim's Algorithm
- Kruskal's algorithm.
- 0/1 Knapsack & ~~Fractional Knapsack~~
- Fractional Knapsack

Date ___/___/_____

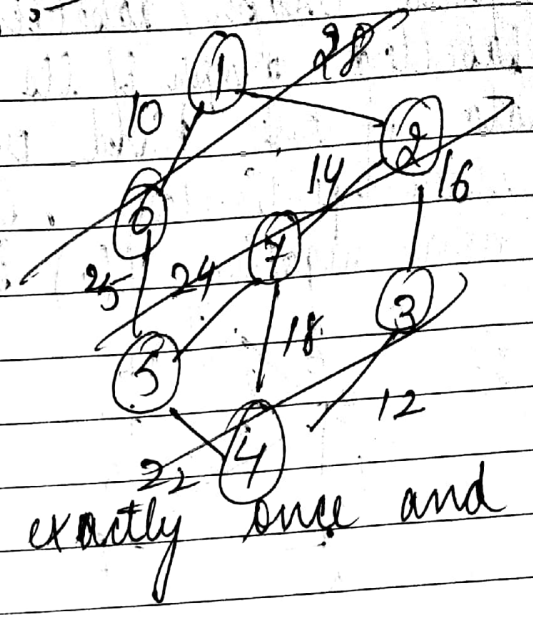
Saathi

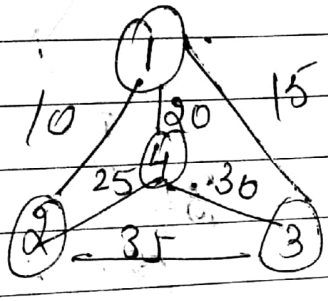
Kim's Algorithm :-

~~Kim's~~

Travelling Salesman problem :-

When we need to find shortest possible route that visit each city exactly once and return





95 1-2-3-4-1

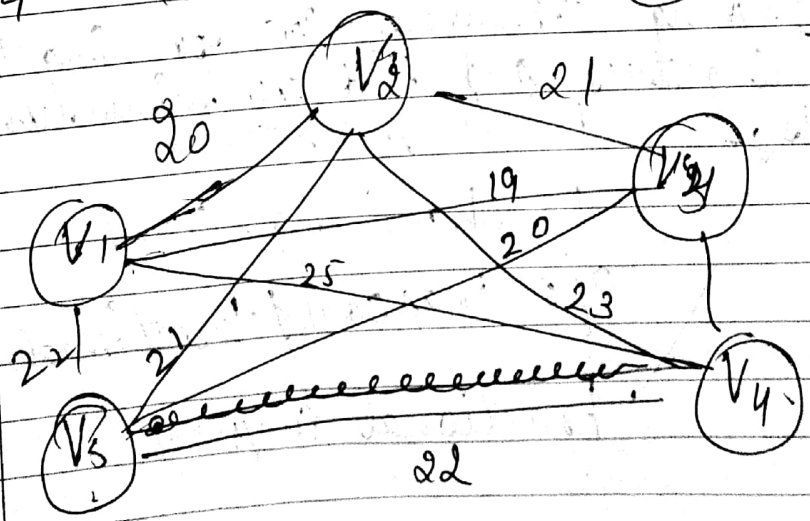
80 1-2-4-3-1

80 1-3-4-2-1

95 1-4-3-2-1

93 1-4-2-3-1
1-3-2-4-1 — 95

	1	2	3	4
1	0	10	15	20
2	10	0	35	25
3	15	25	0	30
4	20	25	30	0



Time taken → 21

Scanned by CamScanner

Date ___/___/_____

Saathi

	1	2	3	4	5
1	0	20	(19)	25	22
2	20	0	21	(23)	21
3	19	21	0	21	(20)
4	(25)	23	21	0	22
5	22	(21)	20	22	0

$V_1 - V_5 - V_3 - V_2 - V_4 - V_1$

$V_1 - V_4 - V_3 - V_5 - V_2 - V_1$

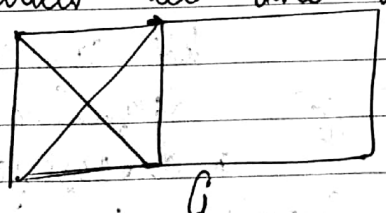
25
21
20
21
20
107

22
20
21
23
25
109

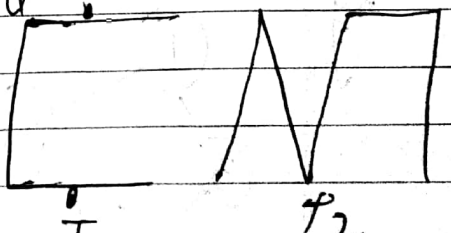
~~Fractional Knapsack~~
$W = 15$

Spanning Tree :- A sub graph T of a connected graph G is called spanning tree of G if T is a tree and includes all the vertices of G.

Eg :-



G



T T_2

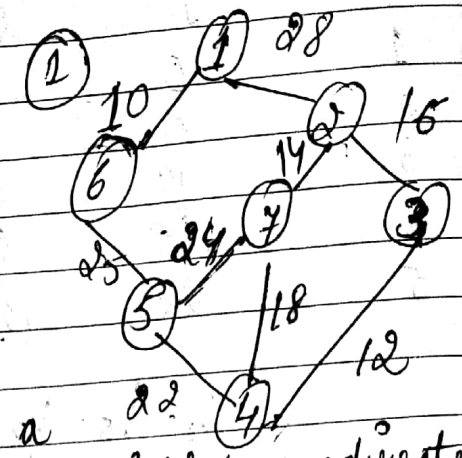
Minimum Spanning tree :- The spanning tree of the graph whose sum of weights of edges is minimum

Saathi

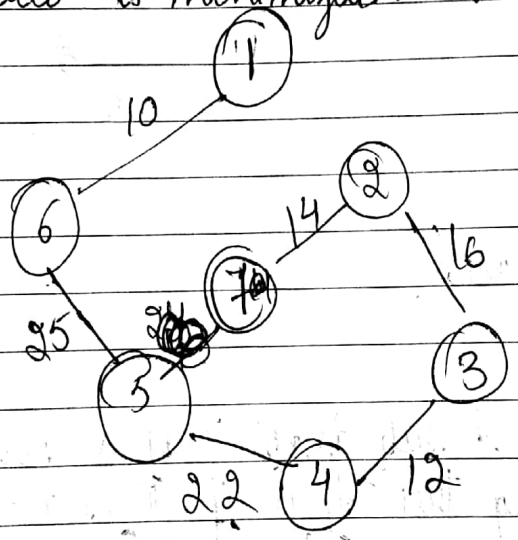
Date __/__/____

Prim's algorithm.

For a prim's algorithm
Select a minimum cost edge
from a graph.
→ Always maintain a tree.

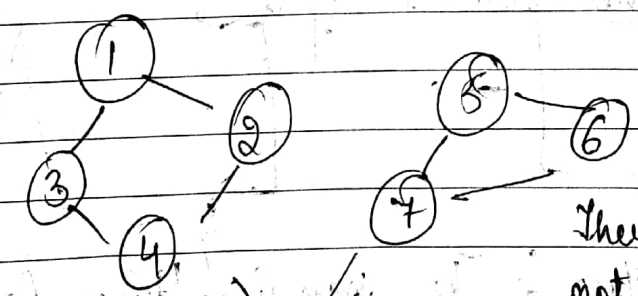


A greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means that it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.



∴ Cost = 10+14+25+16+12
+22
Cost = 99

Eg:-



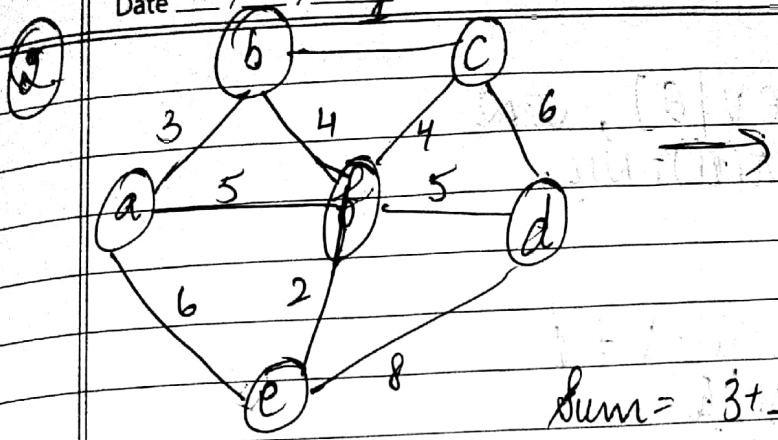
We cannot find a spanning tree of a disconnected graph. Therefore Prim's algo will not work on that graph.

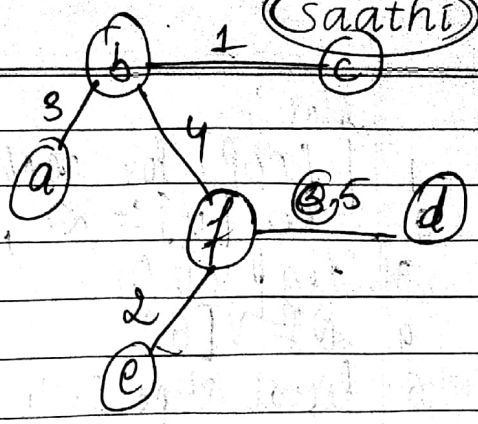
(Prim's algorithm) It can find Spanning tree of only one component not other.

Page No.

Saathi

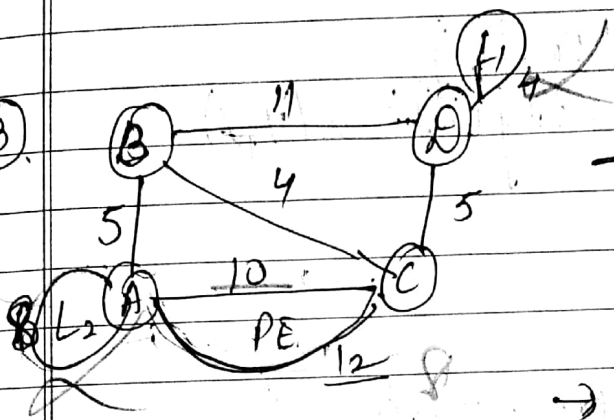
Date / / 1





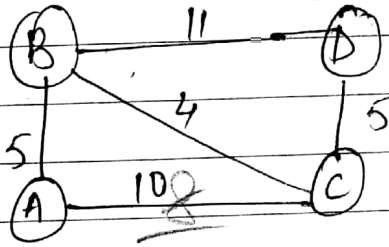
Sum = 3+1+4+5+2 = 15

(3)

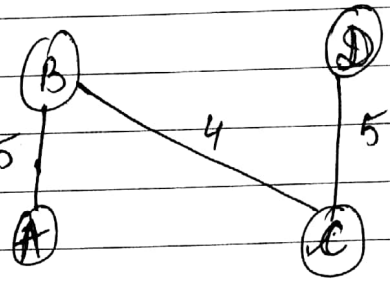


Processes to be followed.
→ Remove loops
→ Remove ||el edge.

→ For removing ||el edge, remove that edge which is having more weight.
→ Remove all loops



MST:-



Sum = 5+4+5 = 14

→ Prim's algo has the property that the edges in the set A always form a single tree.
→ The tree starts from an arbitrary vertex r and grows until the tree spans all the vertices in V.

Saathi

Algo:-

1. [LOOP] for each $v \in V[G]$, steps2
2. key[u] := ∞ , $\pi(u) = NIL$,
3. key[r] = 0
4. $Q = V[G]$
5. Repeat steps 6, to 8, while $Q \neq \phi$
6. u := EXTRACT-MIN(Q)
7. For each $v \in Adj[u]$
8. If $v \in Q$ and $w(u,v) < key[v]$
   then: $\pi[v] := u$, key[v] := w(u,v)
9. Exit

Or

~~Kruskal's Technique~~

1. for each $v \in G$ V
2. ~~u key = ∞~~
3. ~~u. π = v~~

Or

1. for each $u \in G$ V.
2. u key = ∞
3. u.π = Nil
4. s.key = 0
5. Q = Q.V
6. While $Q \neq \phi$
7. u = Extract_Min (Q)
8. for each $x \in G$ Adj[u]
9. if $v \in Q$ and $w(u,v) < v.key$
10. v.π = u
11. v.key = w

Date ___/___/___

Time taken by Kruskal to find minimum cost edge = $O(\log n)$

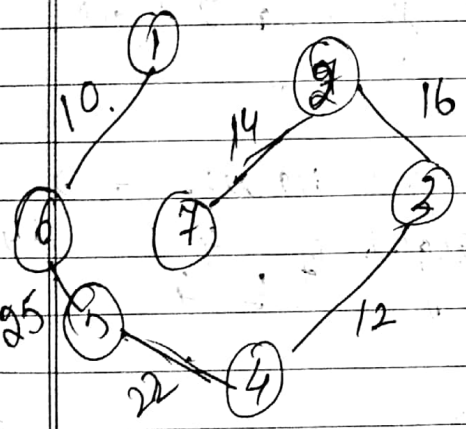
Kruskal's technique

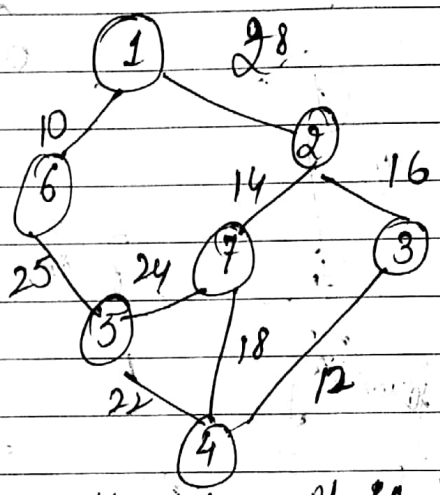
Algorithm

1. Sort all the edges in non decreasing order of the weight
2. Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.
3. Repeat step 2 until there are $(V-1)$ edges in spanning tree.

Kruskal's Technique

It says that always select the minimum cost edge.





Kruskal method say if the minimum edge forms a cycle don't include it in the solution, ie discard that edge.

If Min heap is used it will always give minimum value

cost = 10 + 14 + 16 + 12 + 22 + 25

= 99

$|E|$ edges.

It will be selecting $|V|-1$ edges

Total time = $O(|V|\,|E|) = O(ne)$

(no. of vertices; no. of edges)

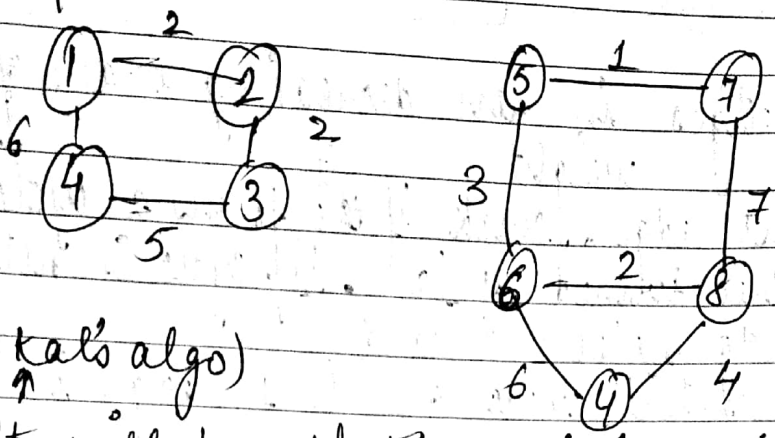
$O(n^2)$ (If both are n)

Page No. ____

Date ___/___/_____

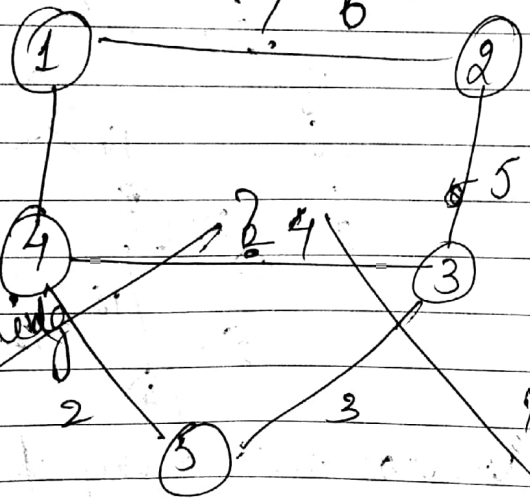
Saathi

Spanning tree of Non-connected graph is not possible.



(Kruskal's algo)

It will be selecting minimum edges from both graphs' components.



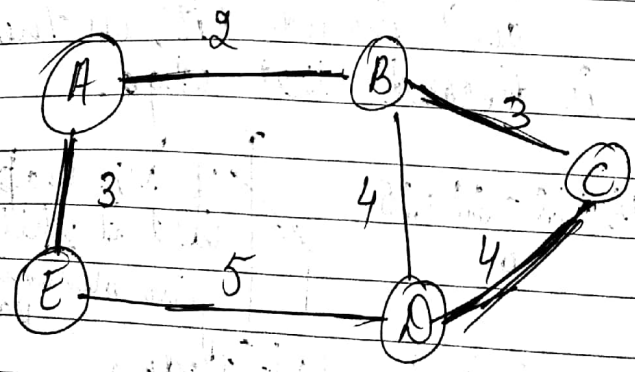
Minimum val. of this could be 6

As 3-4 will not be considered in a spanning tree coz it is forming a cycle.

It is forming a cycle

Minimum values → weight of 4-3 could be 4

#



Cost = ~~5+4+3+2=14~~
2+3+3+4 = 12

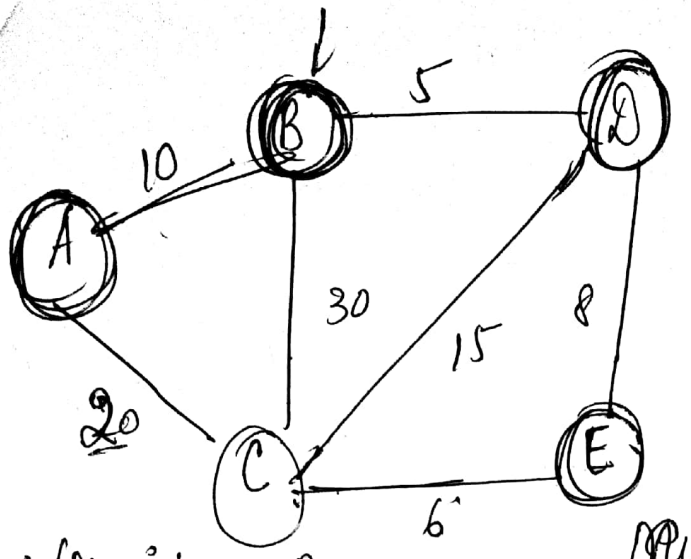
Page No.

Prim's Algorithm

1. Create an array parent[ ] of size V and Initialize it with NIL
2. Create a Min heap (or priority Queue) of size V Let the min heap be H.
3. Insert all the vertices into H such that key value of starting vertex is 0 and key value of other vertices is infinite
4. While H is not empty
   a) u = extractMin(H)
   b) for every adjacent v of u
   c) if v is in H.
      (i) Update key value of v in H if weight of edge u-v is < current key value of v.
      (ii) parent[v] = u

} } }

included in MST



Vertex	Key	Parent
A X	0	NIL
B X	~~∞~~ 10	NIL A
C	~~∞~~ ~~20~~ 15 6	NIL A D E
D X	~~∞~~ 5	NIL B
E X	~~∞~~ 8	NIL D

→ Consider B.

i) Compare key value with edge AB, $10 < \infty$ update ∞ as 10 & update parent as A
Adjacent should be in heap → condition.

(ii) Compare current key value of D with edge BD

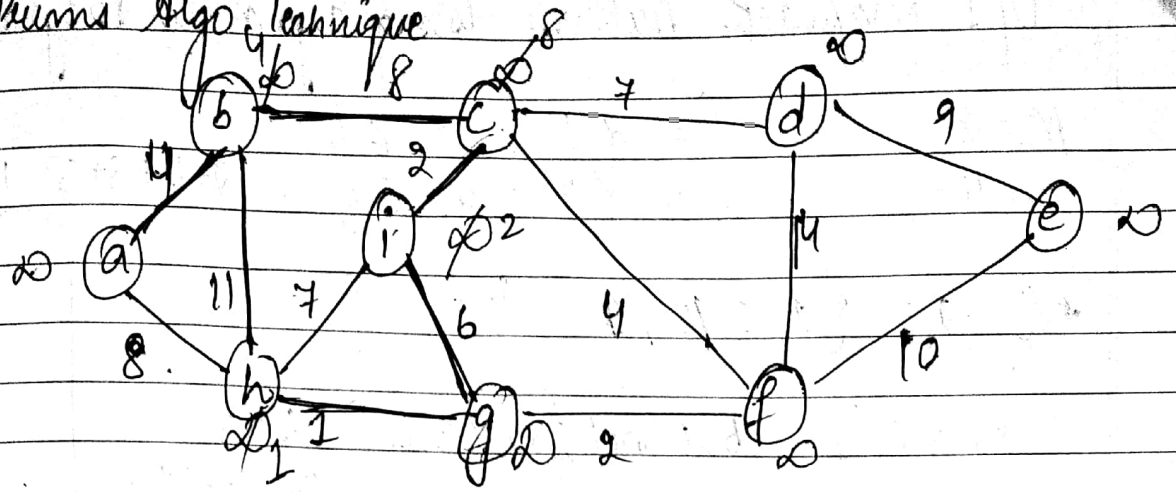
Minimum Cost = $10 + 5 + 8 + 6 = 29$

Always choose the adjacent edges for comparison & it is executed for every adjacent of all the Vertices

Date ___/___/_____

Prim's Algo Technique

Eg-2



Vertices	Key	Parent
a	0	Nil
b	~~∞~~ 4	~~Nil~~ a
c	~~∞~~ 8	~~Nil~~ b
d	~~∞~~ 7	~~Nil~~ e
e	~~∞~~ 9	~~Nil~~ d
f	~~∞~~ 4	~~Nil~~ c
g	~~∞~~ 6	~~Nil~~ i
h	~~∞~~ 81	~~Nil~~ g
i	~~∞~~ 2	~~Nil~~ c

d → c, ~~f~~, e
g → i, ~~h~~, f
h → b, a, ~~g~~, i

Priority Queues:- Priority queues is an abstract data-type which is like a regular queue or stack data structures but where additionally each element has a "priority" associated with it.

Use of Priority Queue:- 1) It is used to keep track of unexplored routes, the one for which a lower bound on the total path length is smallest is given highest priority.

Page No.

Date ___/___/____

Saathi

0/1 knapsack Problem using dynamic Programming

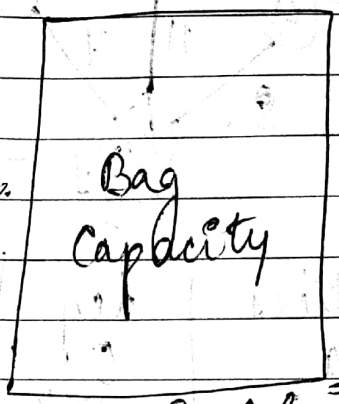
$M = 8$ $P = \{1, 2, 5, 6\}$

$n = 4$ $W = \{2, 3, 4, 5\} = 2+4+3+5 = 14$

$\{g1-x\} = \{1, 0, 0, 1\}$

$x_i = 0/1$ but not fraction.

Fraction of objects cannot be taken.



$m = 8$ (given)

$\max \sum p_i x_i$

$\sum w_i x_i \leq m$ (Capacity of bag)

→ Dynamic Programing is useful for solving optimization problem.

→ Dynamic Programming says that a problem should be solved in a sequence of decisions

→ Dynamic programming says that all possible solutions must be tried and the best one should be choosed/picked.

For n objects sol^n. $= 2^n$ Time Taken → 2^n

Time Complexity = $O(2^n)$

0/1 Knapsack - Greedy Method

1) Find Optimal Solution for knapsack problem -

$n = 7,\ m = 16$

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 15, 12, 4, 6, 16, 8)$

$(W_1, W_2, W_3, W_4, W_5, W_6, W_7) = (2, 4, 5, 4, 2, 3, 3)$

Sol^n :-

Objects	1	2	3	4	5	6	7
P_i	10	15	12	4	6	16	8
W_i	2	4	5	4	2	3	3 — cannot be inserted
P_i/W_i	5	3.75	2.4	1	3	5.33	2.66

Select Object with Maximum profit - Method 2

Object	Profit	weight	Rem. weight
		—	16
6	16	3	16-3 = 13
2	15	4	13-4 = 9
3	12	5	9-5 = 4
1	10	2	4-2 = 2
7	2 × 2.66 = 5.32	2	2-2 = 0

Total profit = 16 + 15 + 12 + 10 + 5.32

= 58.32

Object with minimum Weight

Method 3

Object	Profit	Wt.	Remaining Wt.
—	—	—	16
1	10	2	16-2=14
5	6	2	14-2=12
6	1.6	3	12-3=9
7	8	3	9-3=6
2	15	4	6-4=2
4	1X2=2	2	2-2=0

Total profit = 57

Date __/__/__ V.

P_i	W_i	V	W: 0	1	2	3	4	5	6	7
			0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3
5	4	3	0	0	1	2	5	5	6	7
6	5	4	0	0	1	2	5	6	6	7

While considering second object, 1st object must be included! and similarly for every object and so on.

Formula $V[i, w] = \max\{V[i-1, w], V[i-1, w-w[i]] + P[i]\}$

formula:

$V[i, w] = \max\{V[i-1, w], V[i-1, w-w[i]] + P[i]\}$

$V[4, 1] = \max\{V[3, 1], V[3, -4] + 6\}$

$= 0$

$V[4, 5] = \max\{V[3, 5], V[3, 5-5] + 6\}$

$= 5, 0+6 \rightarrow 6$

$V[4, 6] = \max\{V[3, 6], V[3, 6-5] + 6\}$

$= \max\{3, 0+6\}$

$= \max(3, 6)$

$V[4, 7] = \max\{V[3, 7], V[3, 2] + 6\}$

$= \max\{7, 7\} = 7$

$V[4, 8] = \max\{V[3, 8], V[3, 3] + 6\}$

$= \max\{7, 8\}$

$= 8.$

Date

Saathi

$x_1\ x_2\ x_3\ x_4$ [Sequence]

Maximum profit = 8

Object x_1 x_2 x_3 x_4

{0 1 0 1} 8 − 6 = 2; (2 is in 2nd & 3rd row both)

2 − 2 = 0

{0 1 0 1}

$[x_2 \& x_4]$ Max Profit = 8.

~~Solve by using Set Method~~

Fractional knapsack problem.

W = 15

(1.)

Wi	Vi
2	10
3	5
5	15
7	7
1	6
4	18
1	3

Soln:-

Wi	Vi	Vi/Wi (Profit)
2	10	5
3	5	1.67
5	15	3
7	7	1
1	6	6
4	18	4.5
1	3	3

In case of same profit use FCFS

Page No.

Date ___ / ___ / _____

Saa

w_i	v_i	v_i/w_i
1	6	6
2	10	5
4	18	4.5
5	15	3
1	3	3
3	5	1.67
7	7	1

→ FCFS use

Date ___/___/_____

Chapter :- Back Tracking

It is ~~called~~ uses Brute force approach.
Brute force Approach says that for any given problem you should try all possible solutions and pick up desired solution.
It is not for optimal solutions. Back Tracking for Multiple sol^n. of a problem.

Eg :- B_1 B_2 G_1

Arrange them in 3 chairs

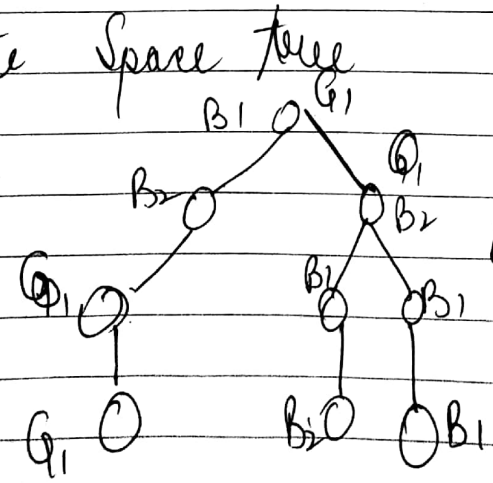
2 boys, 1 Girl.

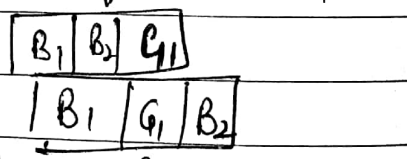
3 chairs

No. of ways = $3! = 6$ ways

For finding all possible arrangements, we can solve the problem and sol^n. is represented in the form of tree called solution tree / State Space tree.

State Space tree

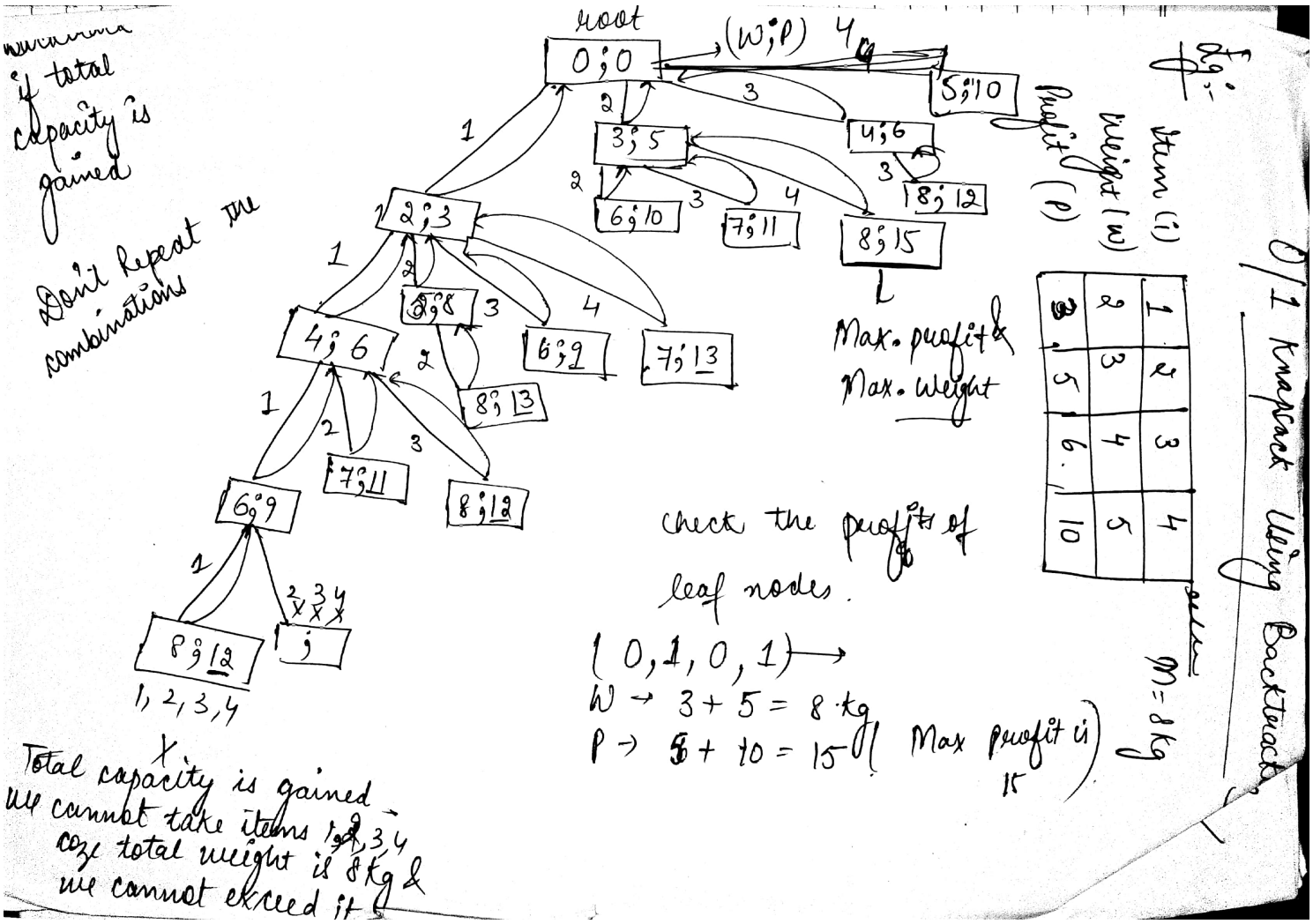




$^nC_n = {}^3C_3 = \frac{3!}{3 \times 1} = 1$

$3! = 3 \times 2 \times 1 = 6$ ways.

0/1 Knapsack Using Backtracking



Item (i)	1	2	3	4
Weight (w)	2	3	4	5
Profit (p)	3	4	5	10

Eg:-

Scanned by CamScanner

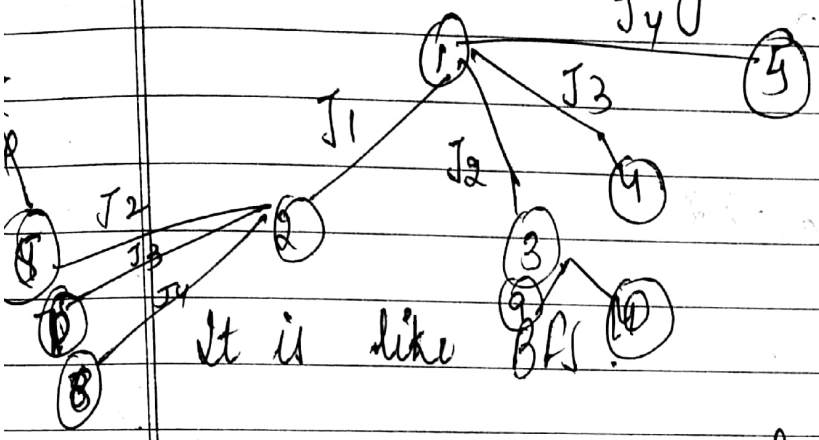
0/1 Knapsack Problem using branch & bound.

1. LC- Branch & Bound.

Profit	10	10	12	18
Weight	2	4	6	9

$m = 15,\ n = 4$

Branch & Bound.
→ Problem solving strategy
→ Similar to Backtracking



It is like BFS.

Job Sequencing
Jobs = {J1, J2, J3, J4}
P = {10, 5, 8, 3}
d = {1, 2, 1, 2}
SI = {J1, J4}

Backtracking was DFS & Branch & Bound is BFS

0/1 Knapsack Problem

Profit	10	10	12	18
Weight	2	4	6	9

$m = 15,\ n = 4$

You either take complete object or don't include the project.

Without fraction: Upper Bound $= \sum_{i=1}^{n} p_i x_i \le m$

With fraction: Cost $= \sum_{i=1}^{n} p_i x_i$

$S = \{x_1, x_3\}$
$S = \{1, 0, 1, 0\}$

Maximization can be converted into Minimization

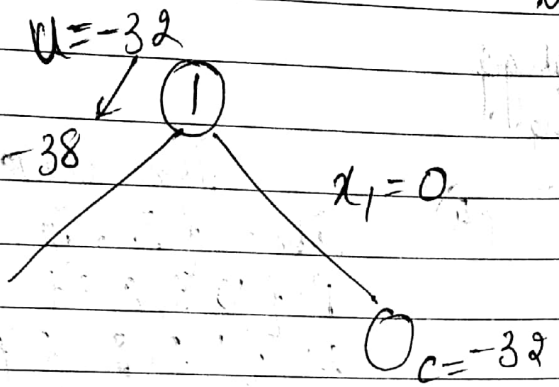
LC-BB.

Saathi

Cost = Profit + Weight (If every object which is to be included)

$= 10 + 10 + 12 + \frac{18}{9} \times 3$

$= 32 + 6 = -38$ → As we have converted the maximization problem to minimization so we must take the cost as -ve.



N Queens Problem.

4 Queens.

(Without attacks)

	1	2	3	4
1	Q_1			
2		Q_2		
3			Q_3	
4				Q_4

Avoid Same → Row
→ Column
→ Diagonal

Ways = $^{16}C_4 = \frac{16!}{4! \times 12!} = \frac{16 \times 15 \times 14 \times 13 \times 12!}{4 \times 3 \times 2 \times 1 \times 12!}$

$= 14 \times 13 \times 10$

$= 1820$.

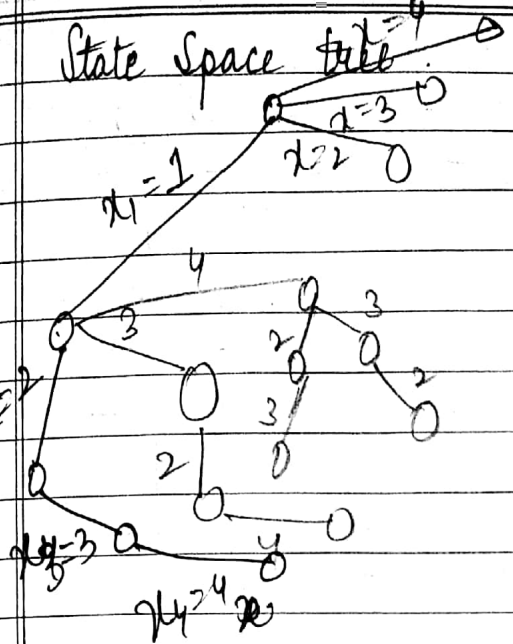
14
13
32
16x
16x

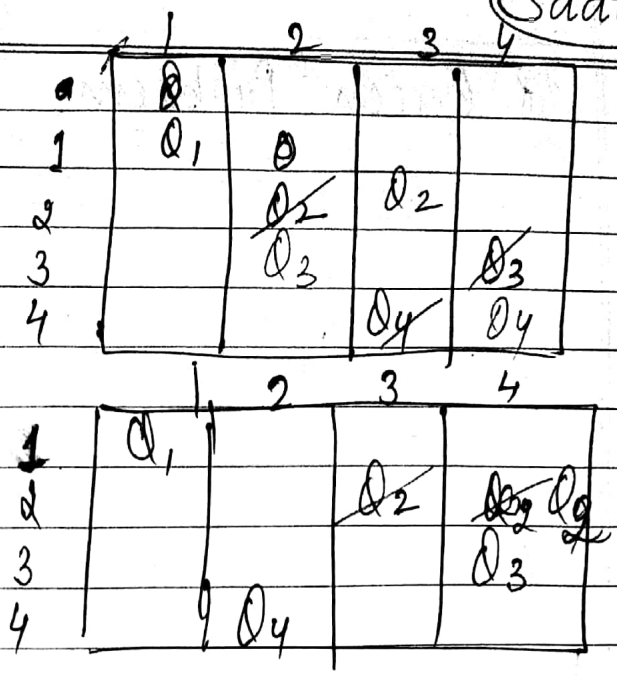
Page No.

Date ___/___/_____

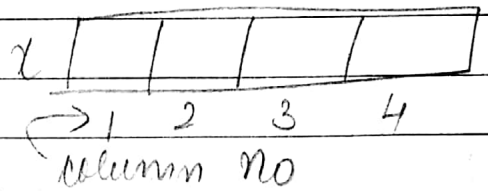
Saathi

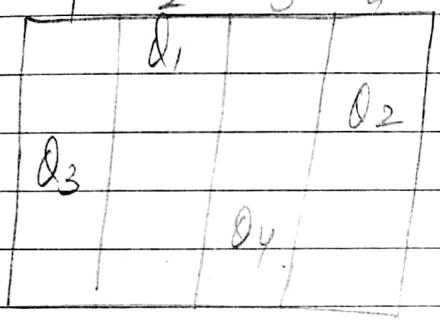
State Space tree



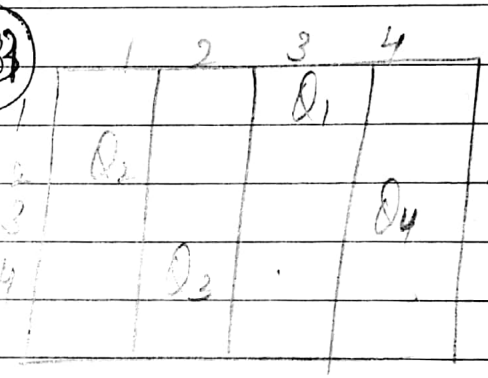


1+4+

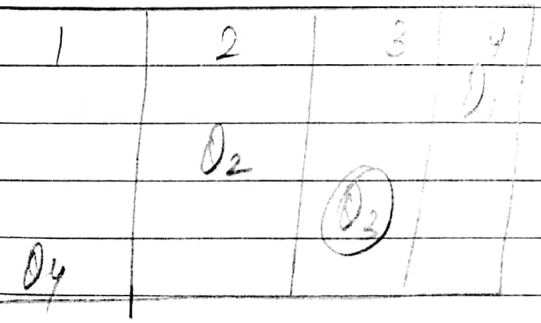




(2413)



→ (3124)



Solutions
2 4 1 3 } Mirror
3 1 2 4 } images.

The N Queen problem is placing N chess queens on an N X N chessboard so that no two queens attack each other.

For

Date ___/___/_____

Saathi

8 Queen Problem:

	1	2	3	4	5	6	7	8
1	Q_1	–	–	–	–	–	–	–
2	–	–	Q_2	–	–	–	–	–
3	–	–	–	–	Q_3	–	–	–
4	–	Q_4	–	–	–	–	–	–
5	–	–	–	Q_5	–	–	–	–
6	–	–	–	–	–	–	–	–
7	–	–	–	–	–	–	–	–
8	–	–	–	–	–	–	–	–

	1	2	3	4	5	6	7	8
1	–	Q_1	–	–	–	–	–	–
2	–	–	–	Q_2	–	–	–	–
3	Q_3	–	–	–	–	–	–	–
4	–	–	Q_4	–	–	–	–	–
5	–	–	–	–	Q_5	–	–	–
6	–	–	–	–	–	–	–	–
7	–	–	–	–	–	–	–	–
8	–	–	–	–	–	–	–	–

	1	2	3	4	5	6	7	8
1	–	–	Q_1	–	–	–	–	–
2	Q_2	–	–	–	–	–	–	–
3	–	–	–	Q_3	–	–	–	–
4	–	Q_4	–	–	–	–	–	–
5	–	–	–	–	Q_5	–	–	–
6	–	–	–	–	–	–	–	–
7	–	–	–	–	–	–	–	–
8	–	–	–	–	–	–	–	–

Date ___/___/_____

Saathi

	1	2	3	4	5	6	7	8
1	–	–	Q_1	–	–	–	–	–
2	~~Q_8~~	–	–	–	Q_2	–	–	–
3	–	~~Q_5~~	–	–	–	–	Q_3	–
4	Q_4	–	–	–	–	–	–	–
5	–	–	–	~~Q_5~~	–	–	–	–
6	–	Q_6	–	–	–	–	–	–
7	–	–	–	–	–	–	–	~~Q_7~~
8	–	–	–	–	–	Q_8	–	–

Q_7

X

	1	2	3	4	5	6	7	8
1	–	–	Q_1	–	–	–	–	–
2	–	–	–	–	–	Q_2	–	–
3	–	Q_3	–	–	–	–	–	–
4	–	–	–	–	–	–	Q_4	–
5	Q_5	–	–	–	–	–	–	–
6	–	–	–	Q_6	–	–	–	–
7	–	–	–	–	–	–	–	Q_7
8	–	–	–	–	Q_8	–	–	–

Soln.

36127145

Saathi

Date

Graph Coloring Problem:-

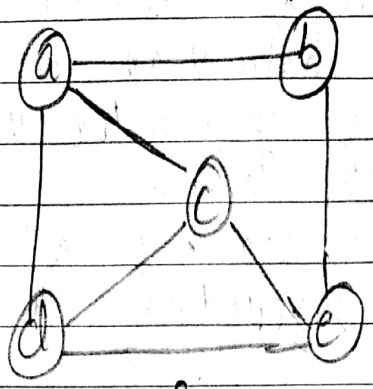
Graph:- A graph $G=(V,E)$ is composed of:

V: set of vertices

E: set of edges connecting the vertices in V.

An edge $e=(u,v)$ is a pair of vertices.

Eg:-



$V = \{a, b, c, d, e\}$

$E = \{(a,b), (a,c), (a,d), (b,c), (c,c), (c,d), (d,e)\}$

Applications of Graphs:-

- Used to represent flow of computation.
- Google maps uses graph for building transportation systems, where intersection of two roads are considered to be a vertex and the road connecting two vertices is considered to be an edge.
- Facebook's friend suggestion algorithm uses graph theory.
- Electronic circuits
- Networks (roads, flights, communications)

Terminology:-

- **Adjacent:** If (V_0, V_1) is an edge in an undirected graph,
  - (i) V_0 & V_1 are adjacent
  - (ii) The edge (V_0, V_1) is incident on vertices V_0 & V_1
- If $\langle V_0, V_1 \rangle$ is an edge in a directed graph.
  - (i) V_0 is adjacent to V_1 & V_1 is adjacent from V_0
  - (ii) The edge $\langle V_0, V_1 \rangle$ is incident on V_0 & V_1

Page No.

Date __/__/____

Saathi

Terminologies of Vertex

→ The degree of a vertex is the number of edges incident to that vertex.

→ For directed Graph :-

(i) The in degree of vertex v is the no. of edges that have v as the head

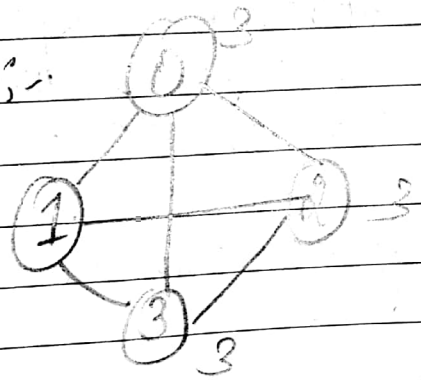
(ii) The out degree of a vertex v is the no. of edges that have v as the tail.

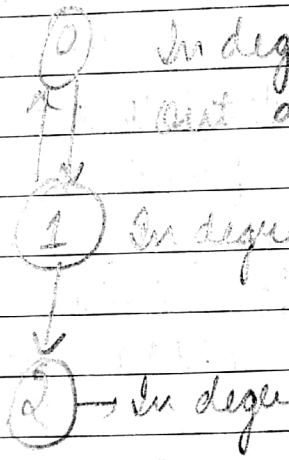
(iii) If d_i is the degree of vertex i in a graph G with n vertices & e edges, the no. of edges is

$$e = \left(\sum_{0}^{n-1} d_i\right)/2$$

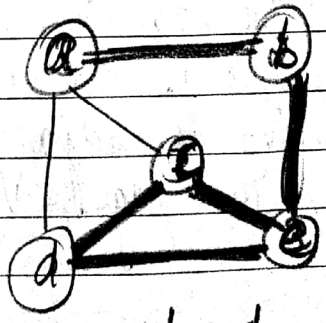
Adjacent vertices each count the adjoining edge it will be counted twice

Eg :-





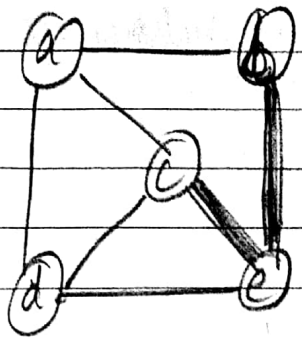
Path :- The sequence of vertices $V_1, V_2, V_3 \ldots V_K$ such that consecutive vertices V_i and V_{i+1} are adjacent



Date ___/___/_____

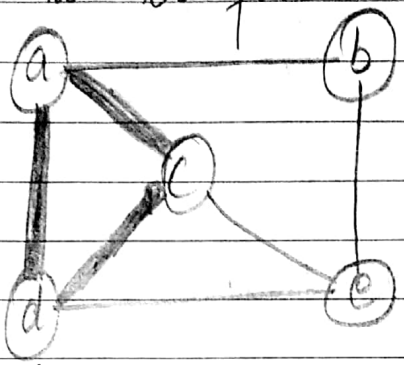
Saathi

Simple Path :- No repeated Vertices



path = bec

Cycle :- Simple Path, except that the last vertex is the same as the first vertex

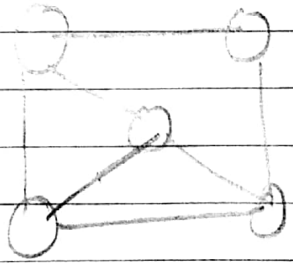


acdba

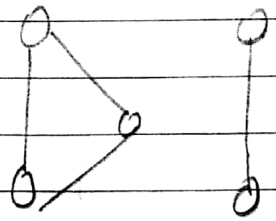
Connected graph :- any two vertices are connected by some path.

Or

A graph is said to be connected if there is a path between any two of its vertices.



connected



not connected

A graph is disconnected if there is no path between any two of its vertices

Sub graph :- Subset of Vertices & Edges formating a Graph

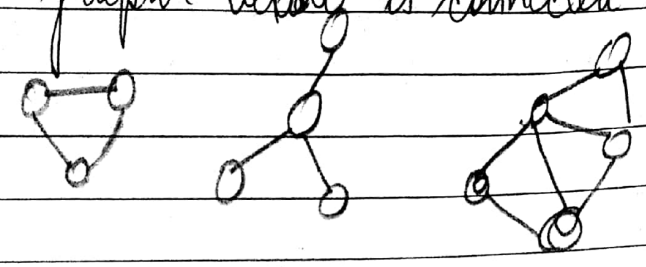
Page No. [ ]

Date ___/___/_____

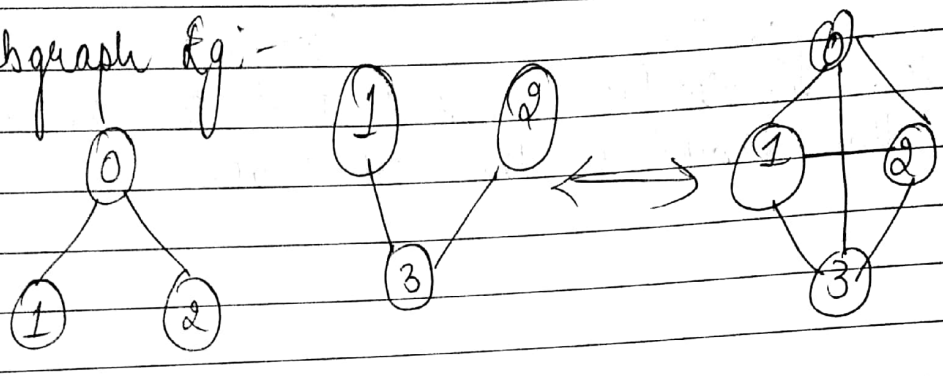
Saathi

Connected components :- Maximum connected subgraph.

Eg :- graph below is connected with 3 components

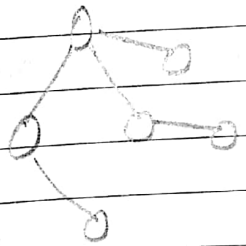


Subgraph Eg :-

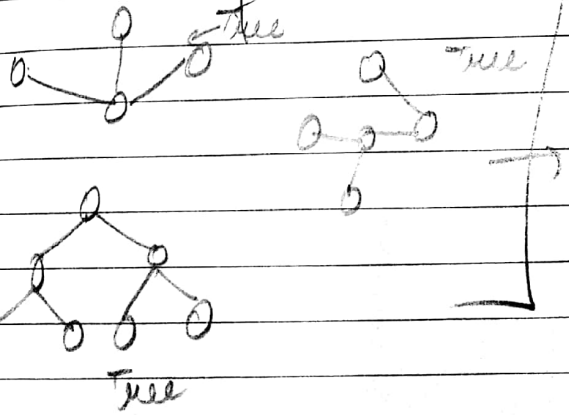


Tree :- Connected Graph without cycles

Eg :-



Forest :- Collection of trees



Connectivity :-

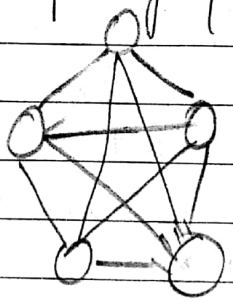
Let n = # Vertices, m = # edges

A complete graph :- One in which all pairs of vertices are adjacent

Page No.

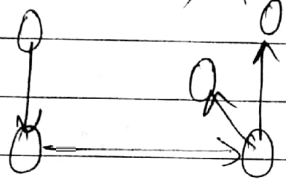
Date ___/___/_____ Saathi

Each of the n vertices is incident to n-1 edges, however, we would have counted each edge twice! Therefore intuitively, $m = n(n-1)/2$.
Therefore, if a graph is not complete, $m < n(n-1)/2$.



$n = 5$
$m = (5*4)/2 = 10$

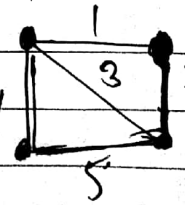
Oriented (Directed) Graph :- A graph where edges are directed



Directed graph :- A graph in which each edge is a directed pair of vertices, $\langle V_0, V_1 \rangle \neq \langle V_1, V_0 \rangle$
tail → head

Undirected graph :- A graph in which each edge is a directed pair of vertices, $\langle V_0, V_1 \rangle = \langle V_1, V_0 \rangle$

Minimum Spanning tree :- A minimum spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycle and with minimum possible total edge weight.



Cost 1+2+5=11

Page No.

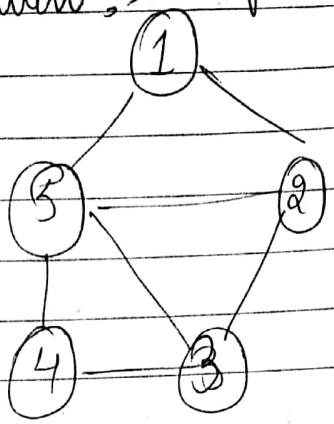
Saathi

Date ___ / ___ / _____

Graph Coloring Using backtracking

The problem is solved by using backtracking.

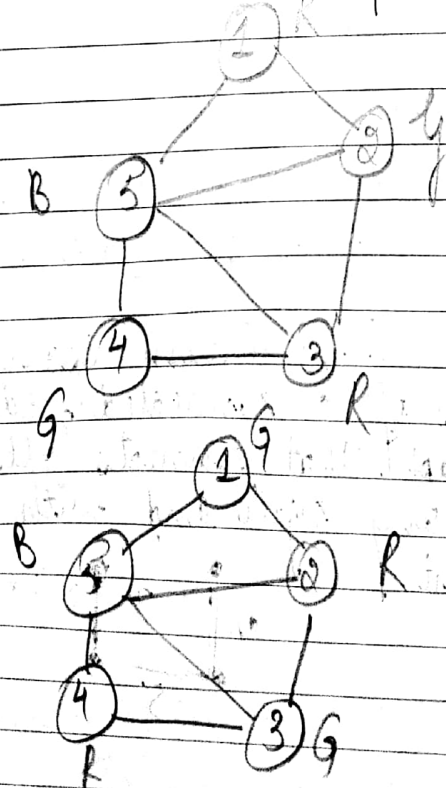
A graph is given:-



Some colors are given
Let us assume 3 colours are given
Red, Green, Blue.
Red → R , Blue → B
Green → G

The problem is we need to color the vertices of graph such that no two neighbouring vertices or adjacent vertices will have same color.

Let us see the possible outcomes



1	2	3	4	5
R	G	R	G	B

This not the only solution for coloring graph.

1	2	3	4	5
G	R	G	R	B

It means same vertices can be coloured in different ways.

Date ___ / ___ / ____

Saathi

The graph cannot be coloured with two colours. But it can be colored with four or many more.

Let us assume a graph and the colors are given If we just want to know whether the graph can be coloured by using these colours or not. So this type of problem is known as m colouring decision problem.

If a graph is given and we need to find out minimum colours required to fill the graph is called m colouring optimization problem.

m colouring decision problem :- The graph colouring problem is to discover whether the nodes of the graph G can be covered in such a way, that no two adjacent nodes have the same color yet only m colors are used

m colouring optimization problem :- The optimization problem deals with the smallest integer m for which the graph G can be colored.
The integer is known as chromatic number of graph

Three types of problems :-

→ Graph and colors are given and calculate the ways in which vertices can be coloured.

→ The colors are given & we need to know the graph can be colored with those vertices or not.

→ Minimum colors reqd. to color the graph.

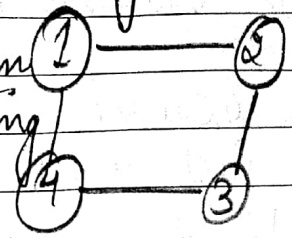
Page No.

Date ___/___/_____

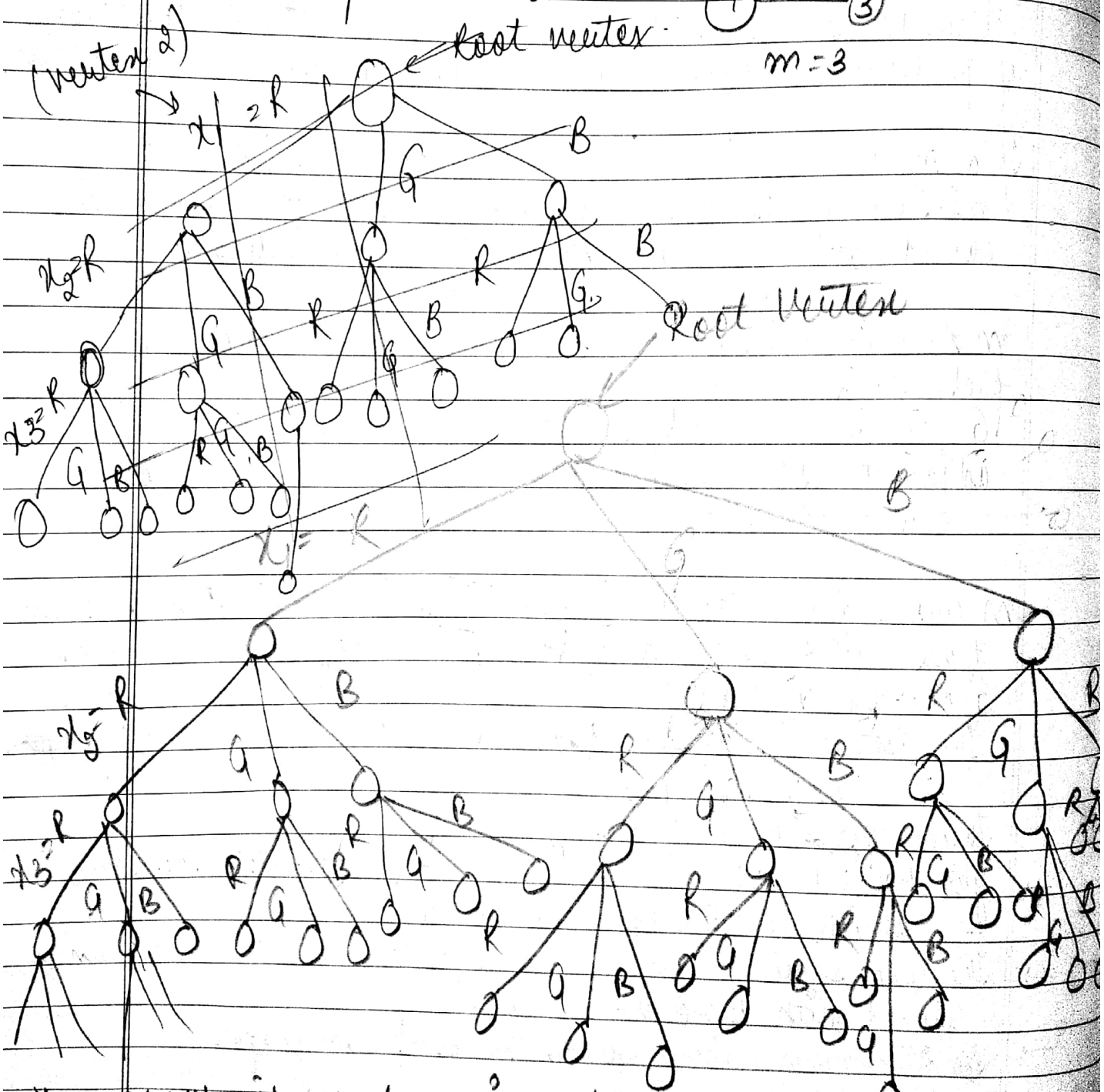
Saathi

Problem solved using backtracking

The method of solving a problem in backtracking is by generating a state space tree.



{R, G, B}



The graph is made without any conditions. But in Backtracking we import conditions and then solve the problem.

Page No.

Date ___ / ___ / _____

Saathi

Total Nodes = Level 1 = 1

Level 2 = 3

Level 2 = 3x3 = 9

Level 3 = 3x3x3 = 27

Level 4 = 3x3x3x3 = 81

$1+3+6+27+81 = 128$

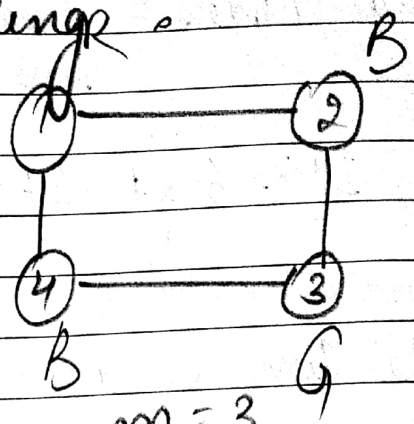
$$\frac{3^{4+1}-1}{3-1} = \frac{3^{4+1}-1}{2}$$

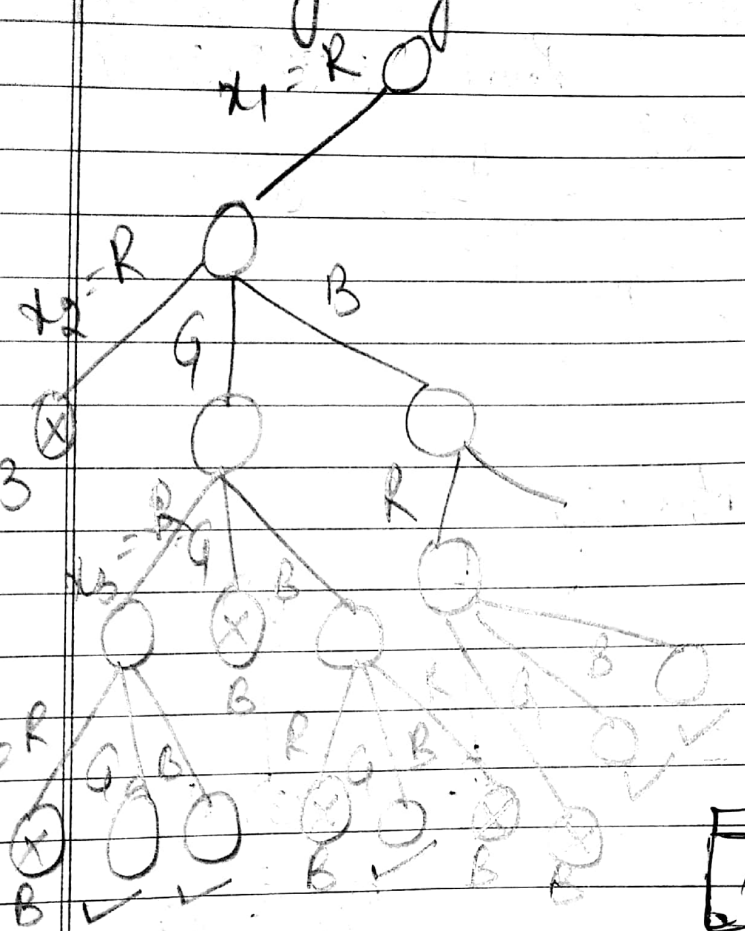
Total time Spend = $3^{n+1} = C^{n+1}$

Date ___/___/_____

Saathi

Solving By Backtracking :-





$m = 3$

{R, G, B}

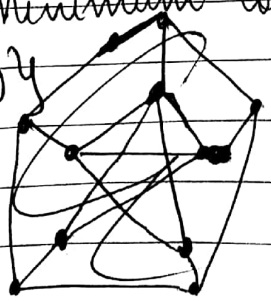
1) R, G, R, G
2) R, G, R, B
3) R, G, B, G
4) R, B, B, G
5) R, B, R, B.

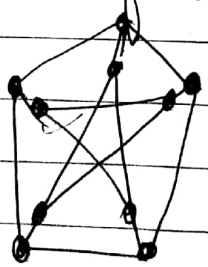
Time Complexity $= 3^n$

Use of Graph colouring problem

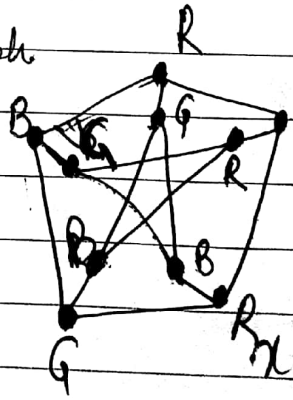
Calculate Minimum Colours to fill graph

{R, G, B, Y, W}

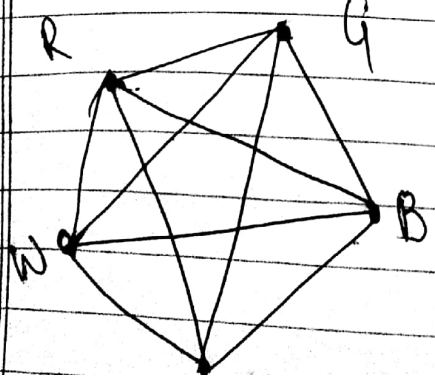




→



$\chi(G) = 3$



$\chi(G) = 5$

Date __/__/____

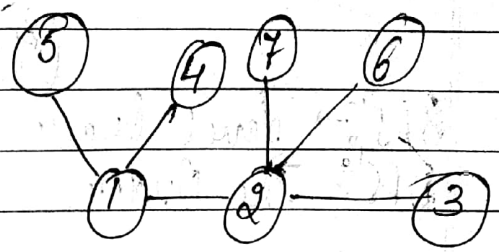
Saathi

Chapter :- Graph Algorithms : Graph Traversals.

Breadth-first search (BFS) :-

This is graph Traversing Methods

→ Visiting a vertex (only once)
→ Exploring of Vertex



BFS : 1, 2, 4, 5, 7, 3, 6 (Any order)
DFS :- 1, 2, 3, 6, 7, 4, 5.

[We need to explore only that vertex which we have visited]

After visiting 2 start exploring it, there is no other node left now again start with go back to 2 & start exploring 2, go to 6 explore it & come back to 2. explore 2 again & so on.

→ Traversals are different i.e. Results are different.
In BFS, we first explore the vertex and then go on next vertex for exploration.
→ In DFS, once we have visited a new vertex we will suspend previous vertex & start exploring the visited vertex.

Standard graph-Searching Algo.

BFS (Breadth first Search)

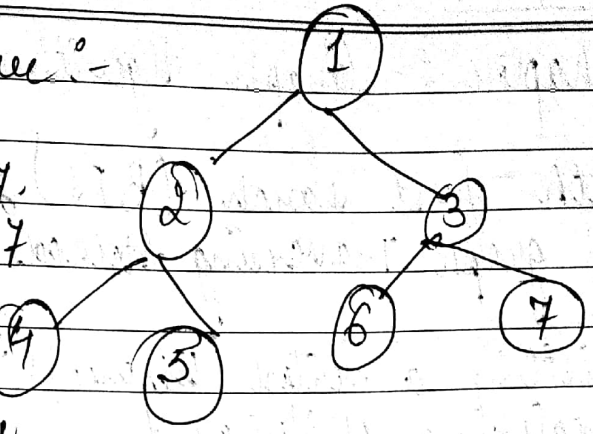
DFS (Depth first Search)

Page No.

Date ___/___/_____

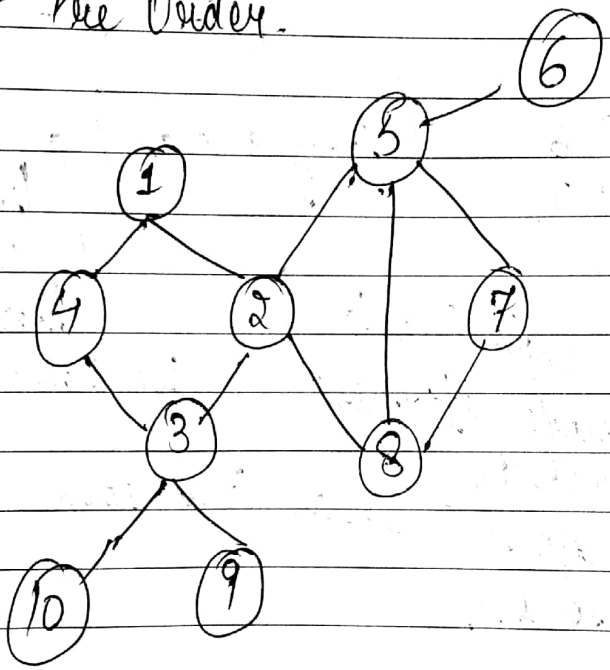
Saathi

Binary Tree :-



BFS :- 1, 2, 3, 4, 5, 6, 7.
DFS :- 1, 2, 4, 5, 3, 6, 7.

BFS :→ Level Order
DFS :- Tree Order.



→

BFS

→ For performing BFS consider a data Structure Q

Q | 1 | 4 | 2 | 3 | 5 | 8 | 7 | 10 | 9 | 6 |

Initial Step :- Start Exploration from any of the vertex

→ Select Vertex 1.

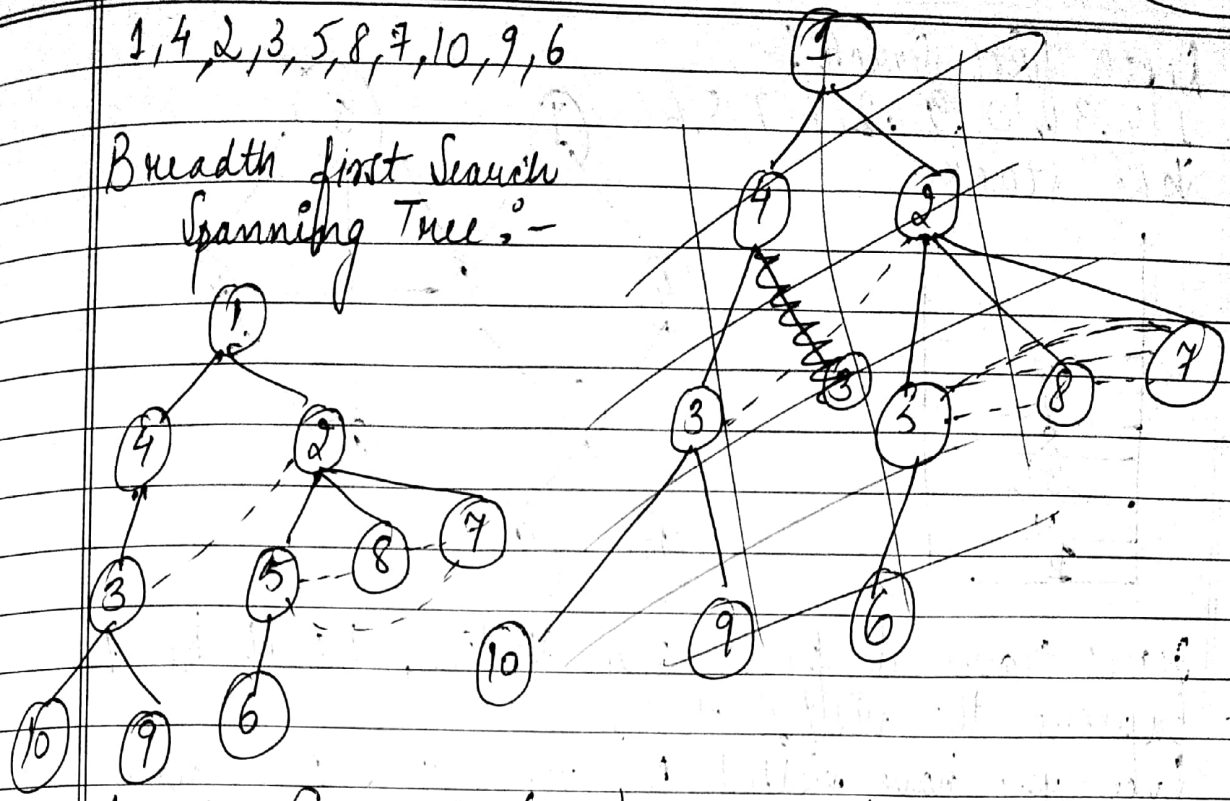
→ Now perform repeating steps

Date __/__/____

Saathi

1,4,2,3,5,8,7,10,9,6

Breadth first Search Spanning Tree :-



Input :- Graph $G = (V, E)$, either directed or undirected, and source vertex $s \in V$.

Output :- $d[v]$ = distance to v, for all $v \in V$. $d[v] = \infty$ if v is not reachable from s.

Algorithm :-

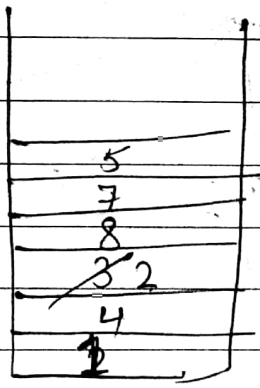
```
for each vertex u in V[G] - {s}
    do color[u] ← white
       d[u] ← ∞
       π[u] ← nil
color[s] ← gray
d[s] ← 0
π[s] ← nil
Q ← ϕ
enqueue(Q, s)
while Q ≠ ϕ
    do u ← dequeue(Q)
for each v in Adj[u]
    do if color[v] = white
       then color[v] ← gray
            d[v] ← d[u] + 1
            π[v] ← u
            enqueue(Q, v)
color[u] ← black
```

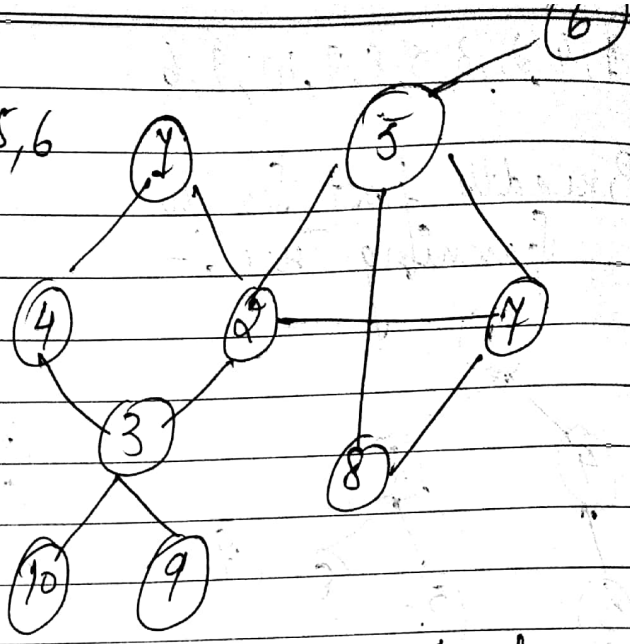
Page No.

Depth first Search:-

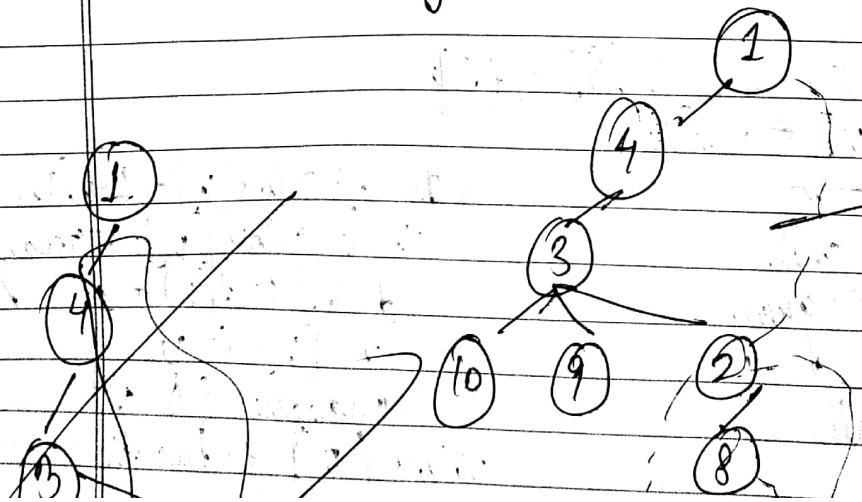
DFS:- 1, 4, 3, 10, 9, 2, 8, 7, 5, 6

Take stack





1. Start traversal from any Vertex, let it be 1
2. Explore the visited Vertex
   Once you have visited the vertex 4, (2 is also remaining leave it, Start exploring 4 and visit 3. Suspend Vertex 1 & Keep it on stack for exploring it later
3. Now go back to 3 after exploring 10 as the node ends. We have 9.
4. The process goes on.



Pre Order →
1, 4, 3, 10, 9, 2, 8, 7, 5, 6

Date __/__/____

Saathi

Algo

1. for each vertex $v \in V[G]$
   do color[v] ← white
   $\pi[v] \leftarrow NIL$
   time ← 0
2. for each vertex $v \in V[G]$
   do if color[v] = White
   then DFS-visit(v)

DFS Visit(v)

1. color[v] ← Grey
2. time ← time + 1
   d[v] ← time
   for each $v \in Adj[v]$
   do if color[v] = White
   then $\pi[v] \leftarrow u$
   DFS - Visit(v)
   color[v] ← Black
   f[v] ← time ← time + 1

Applications of BFS :-

1. Peer to Peer Networks
2. Social Networking Websites
3. GPS navigation
4. Broadcasting in Networks
5. Path finding

Applications of DFS :-

1. Path finding
2. Topological Sorting
3. Finding Strongly connected components of a graph
4. Solving puzzles with only one sol^n.

BFS	DFS
1. Breadth First Search	Depth First Search
2. Uses Queue data Structure	Uses stack data Structure
3. More suitable for searching vertices	More suitable for searching Solutions
4. Not suitable for decision making trees used in games & puzzles	Suitable for that kind of stuff
5. Time complexity → O(V+E)	Time Complexity → O(V+E)

Page No.

Date ___/___/_____

Saathi

Topological Sorting :-

→ It is a sorting process of items over which partial ordering is defined.

→ Topological sort of a directed acyclic graph (DAG) G is an ordering of the vertices of G such that for every edge (e_i, e_j) of G.

We have

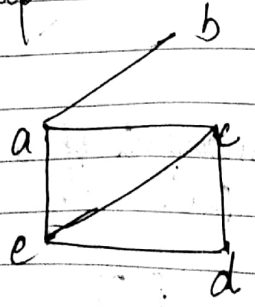
$i < j$ { e_i appears before e_j in the Graph }

Traverses Vertices in increasing order.

Example Using Adjacency Matrix-

Adjacency Matrix & Adjacency list.

Adjacency list :- It is a way to represent a graph without multiple edges. It specifies all the vertices that are adjacent to each vertex of a graph



Vertex	Adjacent Vertex
a	b, c, e
b	a
c	a, e, d
d	c, e
e	a, c, d

A:

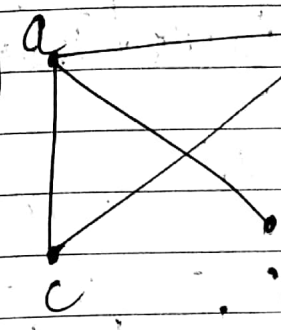
Date ___ / ___ / _____

Saathi

Adjacency Matrices :- $A(Ag)$ is a $n \times n$ zero-one matrix with 1 as its $(i,j)^{th}$ entry when i and j are adjacent & 0 otherwise.

$$A = [a_{ij}] = \begin{cases} 1 & \text{if } (i,j) \text{ is edge of } G \\ 0 & \text{otherwise} \end{cases}$$

Eg (1)



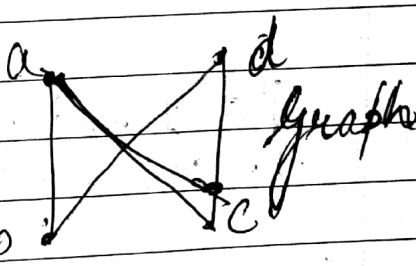
Matrix = $n \times n$

i \ j	a	b	c	d
a	0	0	0	1
b	1	0	1	0
c	1	1	0	0
d	1	0	0	0

If i & j will have edge in between fill 1 otherwise 0

(2) Draw a graph using Adjacency Matrix

	a	b	c	d
a	0	1	1	0
b	1	0	0	1
c	1	0	0	1
d	0	1	1	0



Graph

Graph

Date ___/___/_____

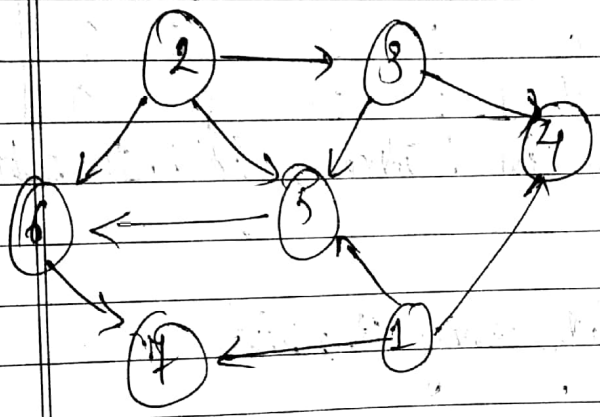
Saathi

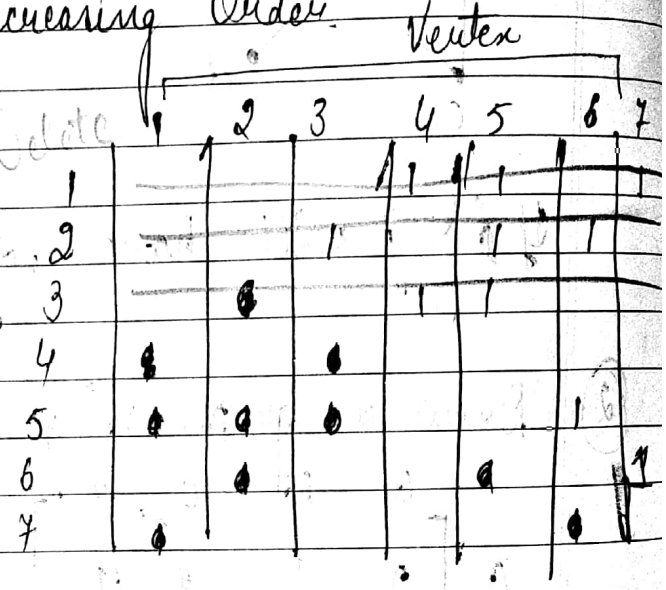
Topological Sorting :-

→ Is a sorting process of items over which partial ordering is defined.

→ Topological sort of a directed acyclic graph (DAG) G is an ordering of the vertices of G such that for every edge (ei, ej) of G, we have i < j
(ei appears before ej in the Graph G)

→ Traverse Vertices in Increasing Order





Put 1 ~~whi~~ under the no. which is derived by the selected no.

To write topological sort :-

1) Select the vertex which does not have any incoming node
ie 1, 2 → Select any 1 node out of it.
Let us select 1 in this case
2) Now delete the row of selected node
3) Now after omitting again do same procedure & delete 2
4) As 3 was having 1 incoming but after omitting out 2, it does not have any incoming

Page No.

Date ___ / ___ / _____

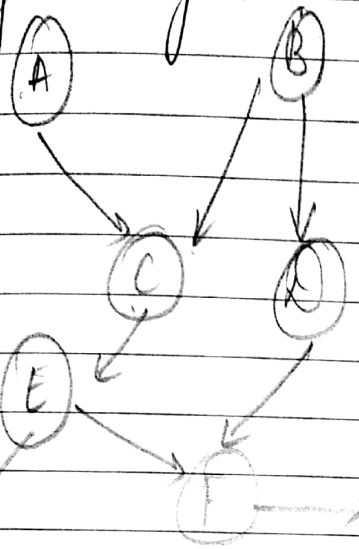
Saathi

5) So Select 3

6) After omitting out 3, We can select nodes from 4 and 5. And the procedure will go on till end.

[1, 2, 3, 4, 5, 6, 7] → Topological Sort.

Eg:- Using DFS (Depth First Search)

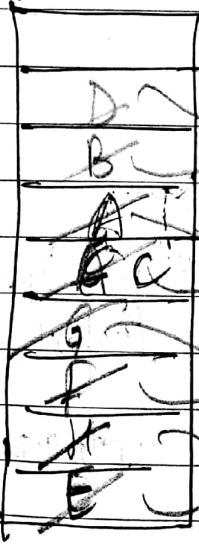


Take 2 stacks of
↳ Visited Set
→ Sorted Set

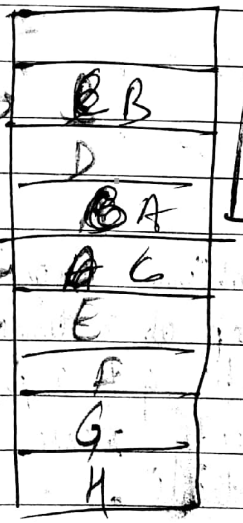
1. Select any node for starting
Suppose selected node = C

We cannot derive any node from last node

Visited Set



Sorted Set



BDACEFGH

2. Now Traverse all nodes which can be reached from E. i.e H.

3 We cannot reach any node from H. So pop H and put it in Sorted Set

Page No.

Saathi

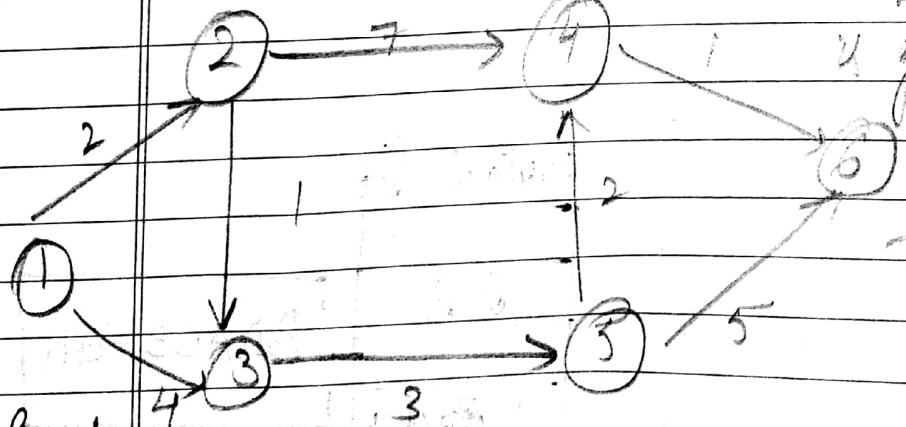
Date ___ / ___ / _____

4. Once E is done we need to select another node. Supposing it to be A
5. After completion of A come to new node i.e. node B. Visit node B and Explore that node

Single Source Shortest paths in Graph

(i) Dijkstra Algorithm

Variants of Shortest path
→ Single Source Shortest path
→ Single destination path
→ Single pair Shortest path
→ All pair path Shortest path



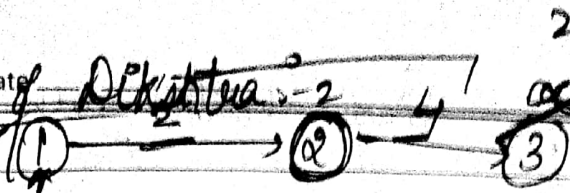
If a weighted graph is given then the path can be find from one starting Vertex to the other Vertices.

Greedy Method:
- The problem should be solved in stages considering one step at a time & considering one input at a time to get a Optimal Sol".
- The greedy have pre defined procedures to get Optimal Solutions.

It can work on both directed & non directed graphs

Page No.

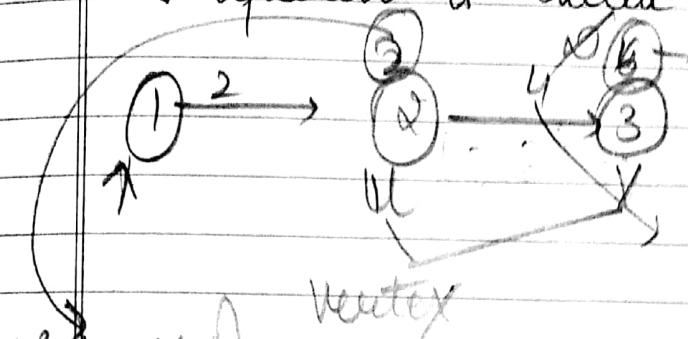
Approach of Dikshtra :-



Consider only directed path. But to follow Dikshtra algo, select the shortest path

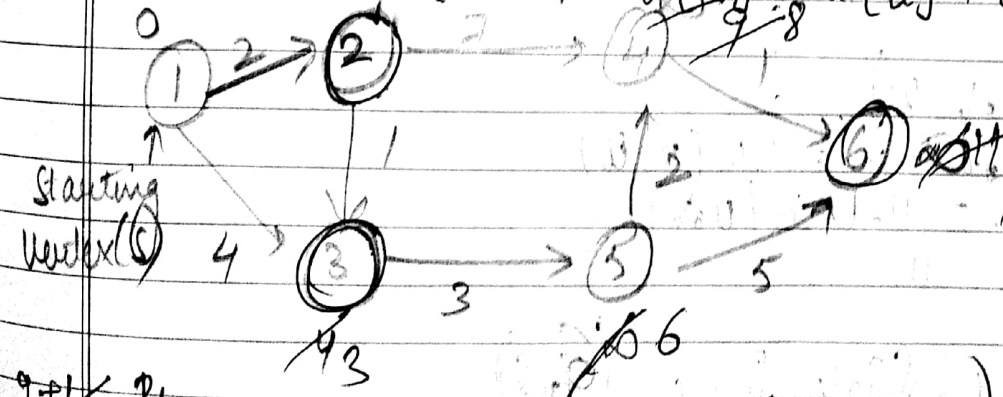
* Dikshtra says if we have selected one shortest path to any vertex then see whether we have any other shortest path to other vertices or not.

This Updation is called Relaxation



Relaxation :- if (d[u] + c(u,v) < d[v])
d[v] = d[u] + c(u,v)

distance
cost of edge



Repeating Steps

Saathi

Date __/__/____

Single Shortest path Algo

Shortest path from 1

V	d(V)
2	2
3	3
4	8
5	6
6	9

Dikstra Bellman Ford

$n = |V|$

Worst Case Time $O(|V|^2) = O(n^2)$

(i)

Algorithm :-

Dikstra (G, w, s)
1. Initialize – Single – Source (G, s)
2. $S = \phi$
3. $Q = G.V$
4. while $Q = \phi$
5. u = Extract – Min (Q)
6. $S = S \cup \{u\}$
7. for each Vertex $v \in Q$ Adj [u]
   Relax (u, v, w)

Relax (u, v, w)
1. if $v.d > u.d + w(v, u)$
2. $v.d = u.d + w(u, v)$
3. $v.\pi = u$

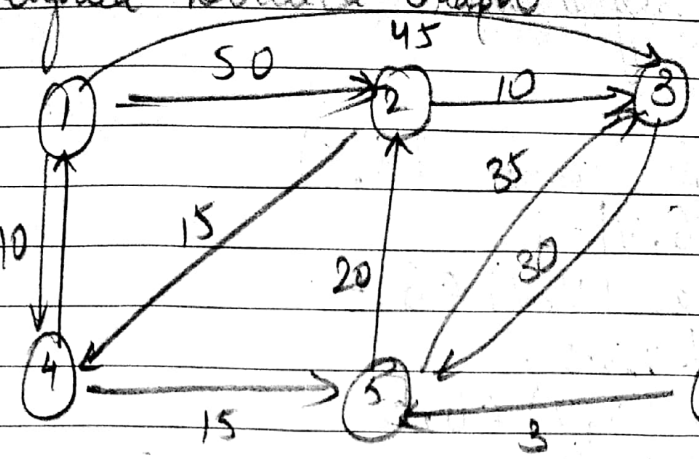
Initialize – Single Source (G, s)
1. for each Vertex $v \in G, V$
2. $v.d = \infty$
3. $v.\pi$ = Nil
4. $s.d = 0$

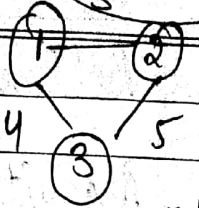
Page No.

Date ___/___/_____

saathi

(2) Weighted Directed Graph





It can work on Non directed Graph as well

no incomming at 6, so, it is ∞

Starting Vertex 1

Selected Vertex	2	3	4	5	6
	~~∞~~	~~∞~~			
4	50	45	10	∞	∞
5	50	45	10	25	∞
2	45	45	10	25	∞
3	45	45	10	25	∞
6	45	45	10	25	∞

10+15=25

Select the smallest vertex

Select next smallest vertex

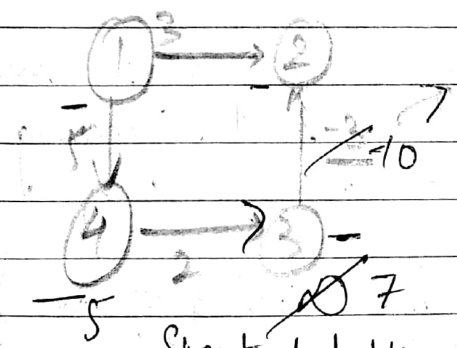
[no distance]

25+20

45+10 ~~+15~~

0 3

(3)



Distance cannot be negative

but Dikstra Algo has not considered.

Drawback of Dikstra Algo

Shortest path = 2

Select the smallest only

Variants of Shortest Path Algorithms

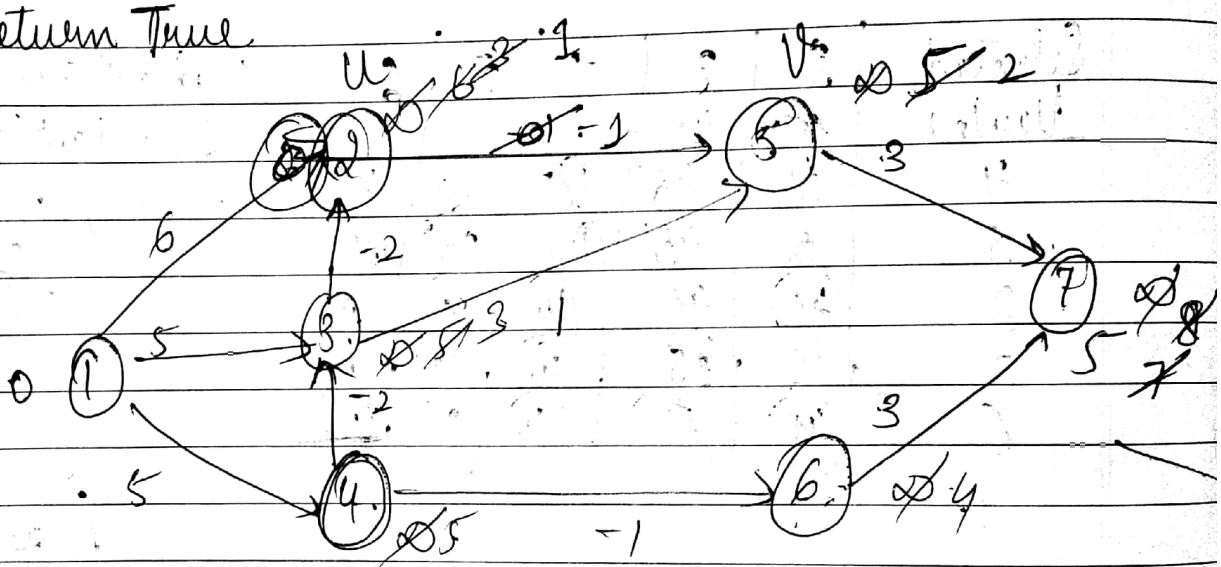
Page No.

Date __/__/____

Bellman Ford Algorithm

Algorithm:-

1. Initialize_Single_Source (G, s)
2. for i = 1 to |G, V| - 1
3. for each edge (u, v) ∈ G, E
4. Relax (u, v, w)
5. for each edge (u, v) ∈ G,
6. if v.d > u.d + w(u, v)
7. return False
8. return True



The problem is in a directed weight graph one of the source vertex is to be selected and finding the shortest path to all other vertices.

Let us select Vertex 1 as 1 is Starting Vertex & then find the shortest path to all other vertices.

This Algo follows dynamic programming strategies which tells that try all possible solutions and pick up the best solution

Relax all edges for |V| n-1 times
Vertices

Saathi

Date __/__/____

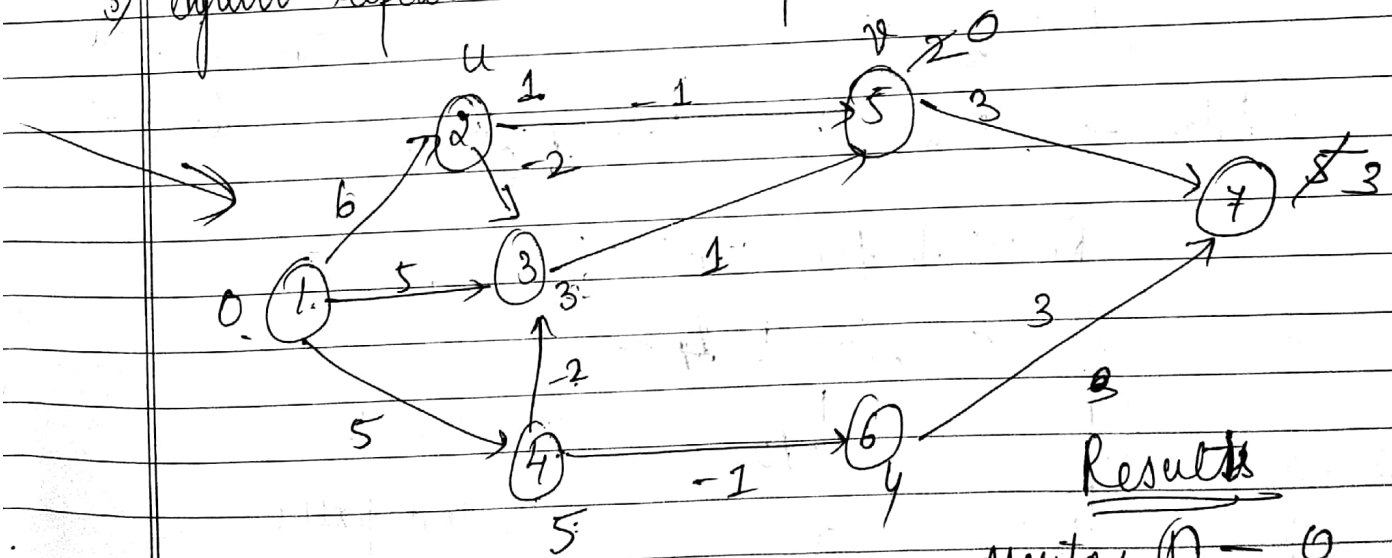
Relaxations :- ~~Between~~ if $(d[u] + c[u,v] < d[u])$ then $d[v] = d[u] + c[u,v]$

→ Prepare list of all edges & select each edge vertex by vertex.

→ Make sure all edges must be selected.

Edgelist → (1,2), (1,3), (1,4), (2,5), (3,2), (3,5), (4,3), (4,6), (5,7), (6,7)

1) Changes 1st time
2) Then again try the changes, start the algo again with same vertices & edges and find minimum.
3) Again repeat the same procedure



4) Again repeat same steps
5) Now this has stopped changing.
6) Do the algo until it stops changing.

Results

vertex (1) — 0
vertex (2) — 1
(3) — 3
(4) — 5
(5) — 0
(6) — 4
(7) — 3

Page No.

Date ___/___/_____

Saathi

Time complexity → $O(|E|\,|V|-1)$

→ $O(|E|\,|V|)$

$= O(n^2)$ → $n \times n$ (min time)

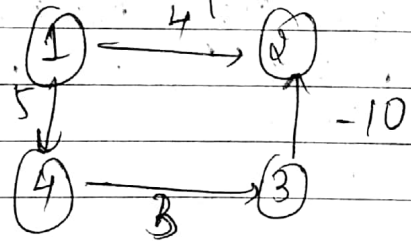
Total edges = $\frac{n(n-1)}{2}$

Time = $\frac{n(n-1)}{2} \times n-1 = \frac{(n^3-n)}{2}(n-1)$ $O(n^3)$

$= n^4 - n$

at most time in case of complete Graph.

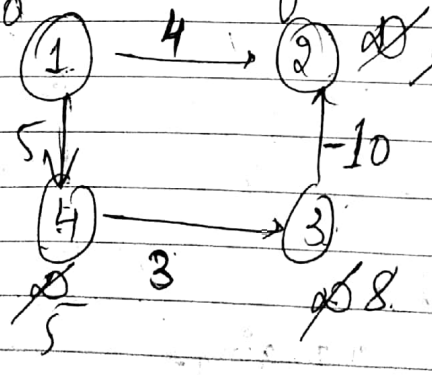
~~Drawback of Bellman Ford~~



Vertices in Reverse

Edges → (3,2), (4,3), (1,4), (1,2)

Relax it for (4-1) = 3 times.



$n = 4-1 = 3$

Shortest paths

Vertex 1 - 0

" 2 - (-2)

" 3 - 8

" 4 - 5

1st ✓

2nd ✓

3rd ✓

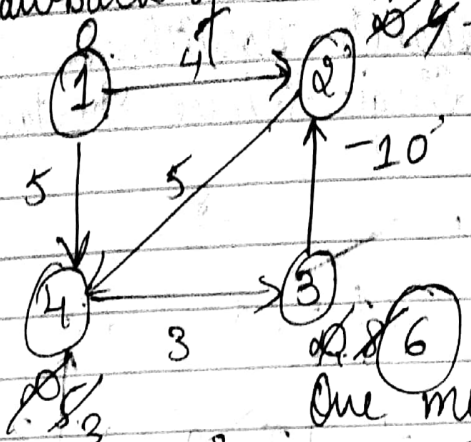
Bellman ford will get this for even -ve edges as well.

Page No.

Saathi

Date / /

Drawback of Bellman Ford



Edges → (3,2), (4,3), (1,4), (1,2), (2,4)

1st → ✓ 3rd → ✓

2nd → ✓

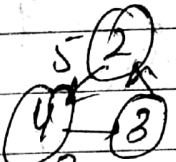
n = 1 = 4 - 1 = 3

One more vertex is relaxed.

We have tried n-1 times, let us try one more time i.e 4th time

For next time even any other Vertex will be changed

Reason:- There is a cycle of edges is



Total weight of cycle is -ve & thus the path will be reduced every time. So, there is no end for this one & if there is -ve weight cycle, the graph cannot be solved. The Bellman Ford Fails to solve the graph for -ve weight cycle. But it can detect if there is -ve weight cycle or not.

$5 + 3 - 10 = -2$

Graph Traversals:- It is a technique used for searching vertex in a graph. It is also used to decide the order of vertex in the search process. A graph traversal finds the edges to be used in search process

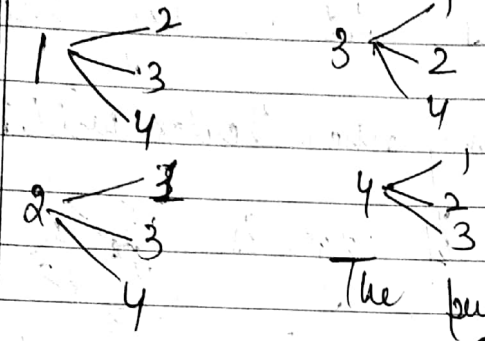
Page No.

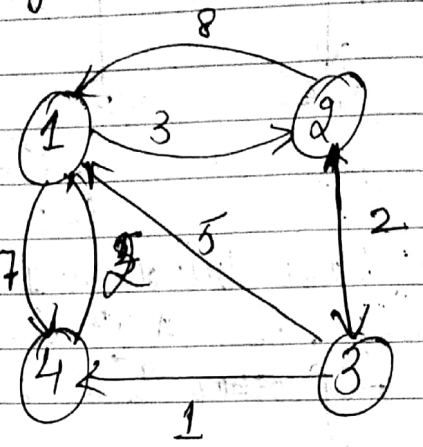
Saathi

Date __/__/____

All pair shortest path in Graphs :-
Floyd Warshall's Algorithm.
(Single Source Shortest path)

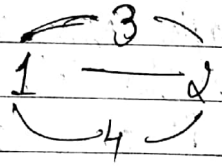
The problem is to find the shortest path between every pair of vertices





The problem can be solved using Greedy Method also.

Shortest path between Vertex 1 to 2



Decide whether the shortest path is going ~~but~~ through other vertex or not.

This can be done preparing Matrices.

→ Loops are not possible

→ If there is no edge from starting to ending vertex so, it is kept as infinity

A^0 =

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

For absence of edge it is kept as infinity & for self loop it is kept as 0!

Page No.

Saathi

Date __/__/____

Solving it by preparing Matrices

$A^0 =$

	1	2	3	4
1	0	3	2	7
2	8	0	2	15
3	5	8	0	1
4	2	5	7	0

Let us prepare a matrix for A^1.

Let us consider the intermediate vertex as vertex 1.

So, all the paths belonging to vertex 1 will remain unchanged

There are no self loops, so diagonals will be taken as 0.

1) $A^0[2,3] = 2 < A^0[2,1] + A^0[1,3]$ (8 + ∞)

2) $A^0[2,4] = ∞ > A^0[2,1] + A^0[1,4]$ (8 + 7)

3) $A^0[3,2] = ∞ > A^0[3,1] + A^0[1,2]$ (5 + 3)

4) $A^0[3,4] = 1 < A^0[3,1] + A^0[1,4]$ (5 + 7)

5) $A^0[4,2] = ∞ > A^0[4,1] + A^0[1,2]$ (2 + 3)

6) $A^0[4,3] = ∞ < A^0[4,1] + A^0[1,3]$ (2 + ∞)

Include Vertex 1 as intermediate vertex in between

Generate Matrix going via Vertex 2

$A^2 =$

	1	2	3	4
1	0	3	5	7
2	8	0	2	15
3	5	8	0	1
4	2	5	7	0

So 2nd row & 2nd column will remain same

1) $A^1[1,3] = ∞ > A^1[1,2] + A^1[2,3]$ (3 + 2)

Page No.

Saathi

Date ___/___/___

2) $A^1[1,4] \quad 7 \quad < \quad A^1[1,2] + A^1[2,4] \quad 3+15$

3) $A^1[3,1] \quad 5 \quad < \quad A^1[3,2] + A^1[2,1] \quad 8+8$

4) $A^1[3,4] \quad 1 \quad < \quad A^1[3,2] + A^1[2,4] \quad 8+15$

5) $A^1[4,1] \quad 2 \quad < \quad A^1[4,2] + A^1[2,1] \quad 5+8$

6) $A^1[4,3] \quad \infty \quad > \quad A^1[4,2] + A^1[2,3] \quad 5+2$

$A^3 =$

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

1) $A^2[1,2] \quad 3 \quad < \quad A^2[1,3] + A^2[3,2] \quad 5+8$

2) $A^2[1,4] \quad 7 \quad > \quad A^2[1,3] + A^2[3,4] \quad 5+1$

3) $A^2[2,1] \quad 8 \quad < \quad A^2[2,3] + A^2[3,1] \quad 2+5$

4) $A^2[2,4] \quad 15 \quad > \quad A^2[2,3] + A^2[3,4] \quad 2+1$

5) $A^2[4,1] \quad 2 \quad < \quad A^2[4,3] + A^2[3,1] \quad 7+5$

6) $A^2[4,2] \quad 5 \quad < \quad A^2[4,3] + A^2[3,2] \quad 7+8$

Saathi

Date __/__/____

A^4 =

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

1) $A^2[1,2] = A^3[1,4] + A^3[4,2]$
   $3 < 6 + 5$

2) $A^3[1,3]$ $A^3[1,4] + A^3[4,3]$
   $5 < 6+7$

3) $A^3[2,1]$ $A^3[2,4] + A^3[4,1]$
   $7 > 3+2$

4) $A^3[2,3] \cdot A^3[2,4] + A^3[4,3]$
   $2 < 3+7$

4) $A^3[3,1]$ $A^3[3,4] + A^3[4,1]$
   $5 > 1+2$

5) $A^3[3,2]$ $A^3[3,4] + A^3[4,2]$
   $8 > 1+5$

Formula

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

where k is the matrix.

Algo :-

```
for (k=1; k<=n; k++)
{
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
        {
            A[i,j] = min(A[i,j], A[i,k] + A[k,j]);
        }
    }
}
```

Time complexity → $O(n^3)$

Saath

Date __/__/____

Distance Matrix

	1	2	3	4
1	0	3	∞	7
2	8	0	2	∞
3	5	∞	0	1
4	2	∞	∞	0

Path Matrix.

	1	2	3	4
1				
2				
3				
4				

Applications

→ Widest Path
→ Clenels Algorithm
→ Transitive Closure.

Decision problem ~~Optimization problem~~

Decision problem :- There are many problem for which the answers is yes or no, whether given graph is colored or not.

Date __/__/____

Saathi

Design And Analysis Of Algorithm

Part IV

Chapter :- Pattern Matching algorithms

String Matching Algo :-

Naive String Matcher

String Matching

Text (String) (T) | a | b | a | b | - - - - - |

Length = n

Pattern (P) | a | b | - - - → Matching this pattern to that String

Length = m

'P' is subset of 'T' if P is found in T.

Characters of T and P are derived from a finite alphabet, Let us say Σ.

Values of Σ are different. Generally these are in terms of (0,1) or $\Sigma = (a, b \ldots z)$

Saathi

Date ___ / ___ / ____

Shift :-

Text = | a | b | c | a | b | a | a | b | c - - - |

Pattern = | a | b | a | a |

Start Matching pattern from shift 0

→ S = 0

↳ S = 1 ie we must consider the first option

| b | c | a | b | a | a | b | c | - | - | - n

| a | b | a | a | m

length

S = 2

| c | a | b | a | a | b | c | - | —

| a | b | a | a |

S = 3

| a | b | a | a | b | c | —

| a | b | a | a |

(p)

Pattern occurs with shift S = 3 in Text T

$0 \leq S \leq n-m$

Date __/__/____

Saathi

Naive String Matching Algorithm

It is the simplest method for pattern matching. It is also known as Brute Force Algorithm

→ Performs checking at all positions in the text, whether an occurrence of the pattern starts there or not

→ After each attempt, the algorithm shifts the pattern by exactly one position to the right.

$$T[S+1 \ldots S+m] = P[1 \ldots m]$$

$$T[S+j] = P[j], \quad 1 \le j \le m$$

Algorithm:

```
n = T.length    [Text length]
m = P.length    [Pattern length]
for S = 0 to n - m        [Shift S]
    if P[1..m] == T[S+1 .. S+m]
        print "Pattern found with Shift" S
```

(1) T = acaabc, P = aab

T = | a | c | a | a | b | c |

S = 0

P = | a | a | b |

(a ✓, c ✗)

T = | a | c | a | a | b | c |

S = 1

P = | a | a | b |

(a ✗)

T = | a | c | a | a | b | c |

S = 2

P = | a | a | b |

(a ✓, a ✓, b ✓)

P is found in T at S = 2

Date ___/___/_____

(2) T = ABCABABCDD

P = BAB.

T = | A | B | C | A | B | A | B | C | D | D |

l = 0

P = | B | A | B |

T = | A | B | C | A | B | A | B | C | D | D |

l = 1

P = | B | A | B |

l = 2 T = | A | B | C | A | B | A | B | C | D | D |

P = | B | A | B |

T = | A | B | C | A | B | A | B | C | D | D |

l = 3

P = | B | A | B |

Patterns Matched at S = 4

S = 4 T = | A | B | C | A | B | A | B | C | D | D |

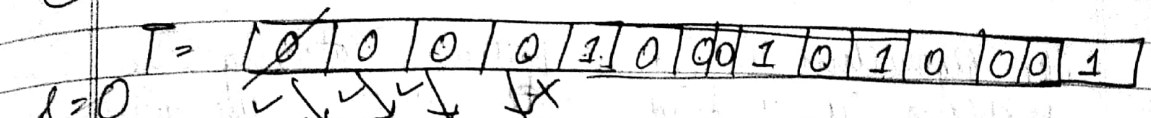
P = | B | A | B |

S = 4

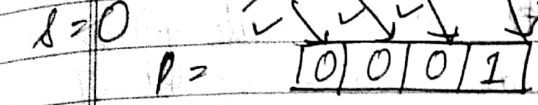
Saathi

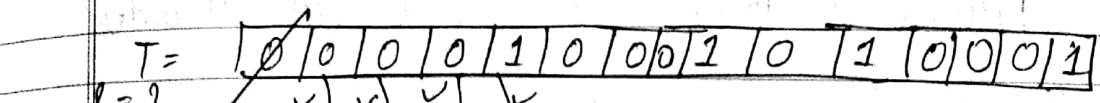
Date ___/___/___

(3) T = 0000 1000 1 0 1 0 0 0 1, P = 0001

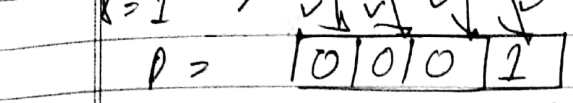


s = 0

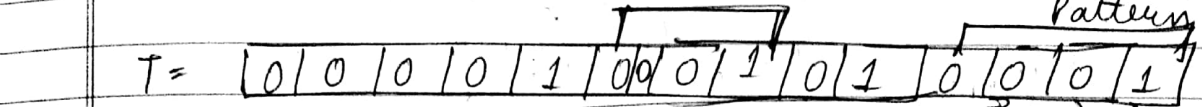


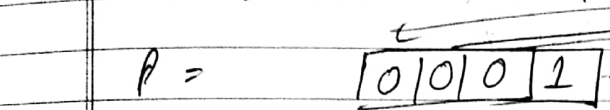


s = 1



Pattern Pattern of S=1 Pattern





S = 1 2, S = 5

Solve it till at end.

Drawbacks of Naive Algorithm

→ Waste of time as in case of mismatch we need to start again with an increasing index

→ ~~Repe~~ Backtracking needs to be avoided.

Worst case.

(n) String :- a a a a a a a b

(m) pattern a a a b

Again start from beginning

S = 5.

The algo is shifting just one step in case of mismatch.

Time Complexity = O(m n)

Page No.

Date __/__/__

Saathi

KMP (~~Knuth~~ Morris Pratt Algorithm)

- The algorithm is used for pattern Matching. The problem is to find out whether the given pattern is existing inside string or not. If pattern is there then find out its index.

The terminology used in KMP algo.
Consider a pattern given here.

Pattern

a	b	c	d	a	b	c
1	2	3	4	5	6	7

prefix a, ab, abc, abcd,

Sufix c, bc, abc, dabc,

Eg:- In KMP algo for a pattern, A pie table is generated

π or lps

P_1 → a b c d a b e a b f
0 0 0 0 1 2 0 1 2 0

P_2 = a b c d e a b f a b c
0 0 0 0 0 1 2 0 0 1 2 3

P_3 = a a b c a d a a b c

0	1	2	3	4	5	6	7	8	9
0	1	0	0	1	0	1	2	3	0

Date __/__/____

Saathi

Ex: a a a b b a a c d

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

Tracing of j algo

(n) String

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

(m) pattern

0	1	2	3	4
a	b	a	b	d
0	0	1	2	0

$O(n+m)$ ← Total Time

Search + Table

→ If i & j matches move j & i ... initial point of
→ Again compare follow same.
→ If i & j does not matches, only increase j by 1

When we got match don't backtrack i.

Saathi

Date __/__/____

Longest proper Suffix (LPS)

(1.) text[] = a b c x a b c d a b x a b c d a b c d a b c y

pattern[] = a b c d a b c y

0 1 2 3 4 5 6 7

LPS[] = | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 0 |

(2.) Text[] = A B X A B → pattern[]

i i i i i

| A | B | X | A | B |

0 1 2 3 4

Match → i, j increase

not Match → j increases

LPS | 0 | 0 | 0 | 1 | 2 |

Date __/__/____

Saathi

Robin Karp Algorithm

It is pattern matching or string matching algorithm.

The problem is if a string is given and a pattern is given, we need to find out whether this pattern is present in the given text or not

	1	2	3	4	5	6
Text :-	a	a	a	a	a	b

m (size) = 6

	1	2	3
Pattern :-	a	a	b

n (size) = 3

Convert the alphabets into Numeric Values.

a - 1
b - 2
c - 3
d - 4
e - 5
f → 6
g - 7
h - 8
i - 9
j = 10

Using these codes we will find the single values for the pattern.

1	2	3
a	a	b

1 + 1 + 2 = 4 → hashcode → pattern

hash fun^n. (P)

1	2	3	4	5	6
a	a	a	a	a	b

1 + 1 + 1 = 3

As hash function values are not matching, so there is no need of checking the characters.

The hash function is used to find the pattern inside the text

1	2	3
a	a	b

1 + 1 + 2 = 4 → no. of letters

Take letters same as size of the second pattern

Saathi

Date ___/___/_____

(2.) Consider a plain Text

	1	2	3	4	5	6	7	8
Text(n) =	a	b	c	d	a	b	c	e

n = 8

1 + 2 + 3 = 6

pattern(m) : b^2 c^3 e^5

m = 3

2 + 3 + 5 = 10

a = 1
b = 2
c = 3
d = 4
e = 5
f = 6

→ Find hash value for pattern

The hash value for pattern (m) is 5

6 − 1 + 4 = 9

And similarly, substract the code of letter we are skipping and add the code of letter we are adding and thus a new pattern will be recieved. So, therefore when we will recieve 10, it will be the answer.

	1	2	3	4	5	6	7	8
n =	a	b	c	d	a	b	c	e

n = 8

2 + 3 + 4 = 9 ✗

Increment i again

pattern (m) b c e

2 + 3 + 5 = 10

3 + 4 + 1 = 8 ✗

Increment → 4 + 5 + 2 = 11 ✗

Increment → n[i] = = n[i+3]

⇒ 1 + 2 + 3 = 6 ✗

Increment → 2 + 3 + 5 → ✓

pattern found at position 7.

a b c d a b c e → pattern

Date

Drawback of Robin Karp Algo :-

The main string is gone through only once & pattern is also checked once.

Time Complexity (Avg time) - $O(n-m+1)$

Worst Case.

		1	2	3	4	5	6	7	8	9	10	11
n=11	Text	c	c	a	c	c	a	a	e	d	b	a
		3+3+1=7										
	pattern	d	b	a								
	m=3	4+2+1=7 (Hash Code)										

a	1
b	2
c	3
d	4
e	5

Though we got a match in hash code values but the string is different.

5+4+2=11

1+3+3=7 | 3+1+1=5 | 1+5+4=10

c c a c c a a e d b a

3+1+3=7

3+3+1=7

5+1+1=7

Again hash code is same but pattern is different

4+2+1=7

If the hash function will be so simple, then there is a possibility of collision with other substrings which are having the same code, though they are not the patterns.

Drawback is there are many hash codes those are matching but these are not hash functions. These are called as Spurious Hits. In these situations, max. time of Algo is $O(mn)$. may be

~~Expectation~~

Page No.

Saathi

Date ___/___/_____

If there are no spurious hits, the maximum time taken by algo is $O(n-m+1)$

To avoid the spurious hits, hash function should be strong.

Hash

Actual function suggested by Rabin Karp

Text
$n=1$

Pattern d b a

$m=3 \rightarrow 4\times10^2+2\times10^1+1\times10^0$

$p[1]\times10^{m-1} + p[2]\times10^{m-2} + p[3]\times10^{m-3}$

Base Value [no. of alphabets using]

[Any value can be taken]

a – 1
b – 2
c – 3
d – 4
e – 5
f – 6
g – 7
h – 8
i – 9
j – 10

$\rightarrow 400+20+1 = 421$

$1\times10^2+3\times10^1+3\times10^0$
$100+30+3$
133

Text → c c d c c a a e d b a
$n=11$

$3\times10^2+3\times10^1+1\times10^0$
$300+30+1$
$= 331$

$[3\times10^2+3\times10^1\times1\times10^0] = 3\times10^2\times10+3\times10^0$

$[300\times10]+0+3$

313

$3\times10^2+1\times10^1+3\times10^0$
$300+10+3$
313

Page No.

Date __/__/____ Saathi

KMP (Knuth Morris ~~Pratt~~ Pratt Algorithm)

Step by Step Soln

In the pattern matching algo like naive method or Boyer-Moore, we often compare the pattern characters that do not match in the text, and on occurence of mismatch we simply throw away the information and restart the comparison for another set of characters from the text

Consider a text → a b a d a b

Step by step implementation

→ Initially we will put 0 in 0th location of prefix array

a	1	2	3	4	5
a	b	a	d	a	b
0					

→ Consider a string ab, No match of prefix and sufix, Hence put 0 in prefix array at position 1

0	1	2	3	4	5
a	b	a	d	a	b
0	0				

Prefix : a
Suffix : b

→ Consider a string aba, The length of matching prefix-suffix is 1. Hence make entry 1 in prefix table

0	1	2	3	4	5
a	b	a	d	a	b
0	0	1			

Prefix :- a, ab
Suffix :- a, ba

Date ___ / ___ / _____

Saathi

→ Consider string abad
The length of matching prefix-suffix is 0, Hence make entry of 0 in prefix table.

0	1	2	3	4	5
a	b	a	d	a	b
0	0	1	0		

Prefix - a, ab, aba
Sufix :- ~~b, ba, bad~~
d, ad, bad.

→ Consider string abada, The length of matching prefix-suffix is 1. Hence make entry 1 in prefix table.

0	1	2	3	4	5
a	b	a	d	a	b
0	0	1	0	1	

Prefix :- a, ab, aba, abad
Suffix :- a, da, ada, bada

→ Consider string abadab. The length of matching prefix-suffix is 2. Hence make entry 2 in prefix table.

0	1	2	3	4	5
a	b	a	d	a	b
0	0	1	0	1	2

Prefix :- a, ab, aba, abad
abada
Suffix :- b, ab, dab,
adab, badab.

Saathi

Date __/__/____

Longest Proper Subsequence (LPS)

Comparing pattern with pattern.

Text → b a d b a b a b a d a a b

Pattern → a b a b a d a

→ Compute prefix table for pattern

0	1	2	3	4	5	6
a	b	a	b	a	d	a
0	0	1	2	3	0	1

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

→ Compare b and a, as it is not matching, Compare Text[1] with Pattern[0]

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

→ Text[1] = Pattern[0], Now compare Text[2] with Pattern[1]

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

i

As Text[2] ≠ Pattern[1].

Backtrack pattern & compare Pattern[0] with Text[3]

Page No.

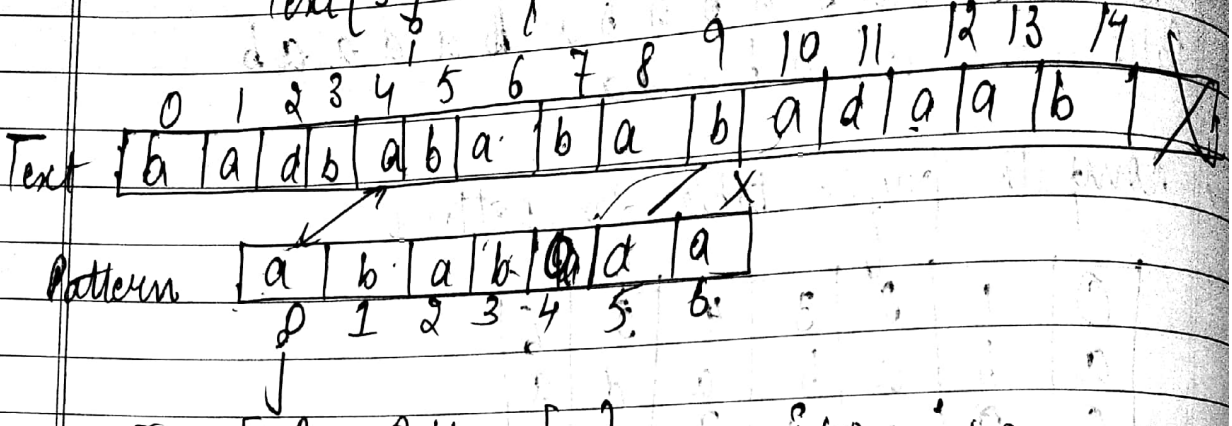
Date __/__/____

↪ Again Text[3] ≠ Pattern[0]

Compare Text[4] with pattern[0], i++, j++

Text[4] = pattern[0]

Text[5] = pattern



Text[4] == Pattern[0] i++, j++

Text[5] == pattern[1] i++, j++

Text[6] == pattern[2] i++, j++

Text[7] == pattern[3] i++, j++

Text[8] == pattern[4] i++, j++

But Text[9] ≠ pattern[5]

Backtract the ~~text~~ pattern

Considering

Text[10] == pattern[4] i++, j++

Text[11] == pattern[5] i++, j++

Text[12] == pattern[6] i++, j++

Thus we have reached to the last character of pattern, at the same time i is positioned at 12th location in Text.

The last character of pattern is also matching with Text[12]

Hence pattern[6] == Text[12]

Page No.

Saathi

Date ___ / ___ / _____

Therefore Required 'Result'

Text :-

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

&

Pattern :-

a	b	a	b	a	d	a
0	1	2	3	4	5	6

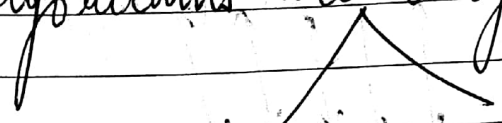
Required pattern is at location 6 in Text. using KMP algo.

Date ___ / ___ / _____

Chapter :- NP and P completeness.

This topic is related to research.

Algorithms are categorised into 2 groups



- → Linear Search - n
- → Binary Search - $\log n$
- → Insertion Sort - n^2
- → Merge Sort - $n \log n$
- → Matrix Multiplication - n^3

- → 0/1 Knapsack - 2^n
- → Travelling SP - 2^n
- → Sum of Subsets - 2^n
- → Graph Coloring - 2^n
- → Hamiltonian Cycle - 2^n

Decision algorithm :- Any problem for which either yes or no is the answer, it is called decision problem. The algorithm of decision problem is called decision algorithm.

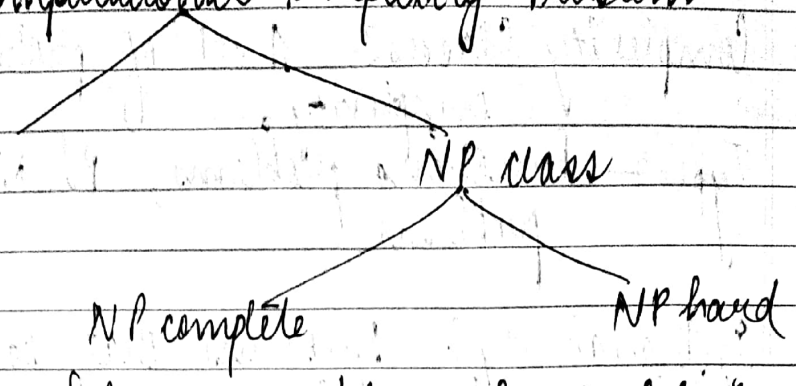
Optimization algorithm :- Any problem that involves the identification of optimal cost (minimum or maximum) is called optimization problem. The algorithm of optimization problem is called optimization algorithm.

Definition of P :- Problems that can be solved in polynomial time, where P stands for "polynomial"
Eg :- Searching of key element, Sorting of elements etc.

Saathi

Date __/__/____

Definition of NP :- It stands for "non-deterministic polynomial time" where NP stands for "non-polynomial"



Eg :- Travelling Salesman problem, Graph Colouring problem, Knapsack problem.

NP complete :-

→ A problem D is called NP complete if -
(i) It belongs to class NP
(ii) Every problem in NP can also be solved in polynomial time

→ If an NP-hard problem can be solved in polynomial time then all the NP complete problems can also be solved in polynomial time.

→ All NP complete problems are NP hard but all the NP hard problems cannot be NP complete

Computational complexity :- The computational problems is an infinite collection of instances with a solution for every instance. The computational problems can be functional problems

Page No.

Saathi

Date ___/___/_____

Functional problems :- The function problem is computational problem where single output is expected for every input.

Complexity Classes :- A set of problems of related complexity.

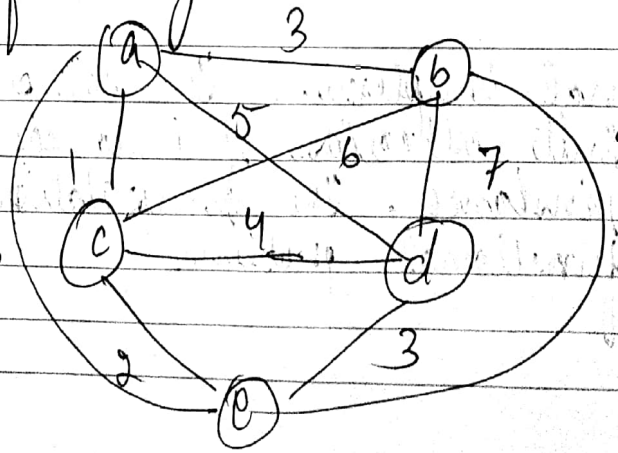
Types — Function problems, P & NP classes, optimization problems.

Intractability :- Problems that can be solved by taking a long time for their solution.

Example of P class Problem
MST using Kruskal's algo.

Example of NP Class Problem :- Travelling Salesman Problem.

Travelling Salesman problem states as "Given a set of cities and cost to travel between each pair of cities, determine whether there is a path that visits every city once and returns to the first city such that the cost travelled is less.



Page No.

Date ___ / ___ / ____

Saathi

Tour path :- a - b - d - e, c - a
Total cost - 3 + 7 + 3 + 2 + 1
= 16

NP hard Problem :- A problem is said to be NP-hard if everything in NP can be transformed in polynomial time and we get no solution by applying certain algos.

NP complete problem :- A problem is said to be NP-complete, if we get a solution by applying certain algos.

The problems solved in polynomial time are known as P class problems.

Example
- Binary Search (List is divided at mid.) (Complexity - O(log n))
- Evaluation of polynomial (Summation of each term of polynomial) (Complexity - O(n))
- Sorting a list (Elements in list are arranged either in asc/desc) (Complexity O(n log n))

Date ___/___/___

Saathi

Satisfiability Problem → This problem is to determine whether a boolean Expression is satisfiable.

~~A 3 SAT Problem :-~~
~~A 3 SAT problem is a problem which takes a Boolean formula S in CNF form~~

(i) CNF sat problem :-

A problem is based on boolean formula
A boolean formula has foll" boolean operations like → OR(+), AND(·) and NOT.

Notations :-

→ means implies
↔ means if and only if

A boolean formula is in Conjuctive Normal form (CNF) if it is formed as collection of subexpression These subexpressions are clauses.

(ii) A 3 SAT Problem :-

3 SAT

A problem is a problem which takes a Boolean formula S in CNF form with each clause having exactly three literals and check whether S is satified or not.

Foll" formula is an instance of 3SAT Problem :-

$(\bar{a} + b + g)(c + \bar{e} + f)(\bar{b} + d + f)(a + e + h)$

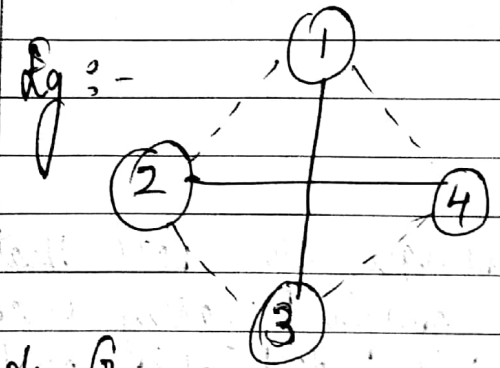
Date ___/___/_____

Saathi

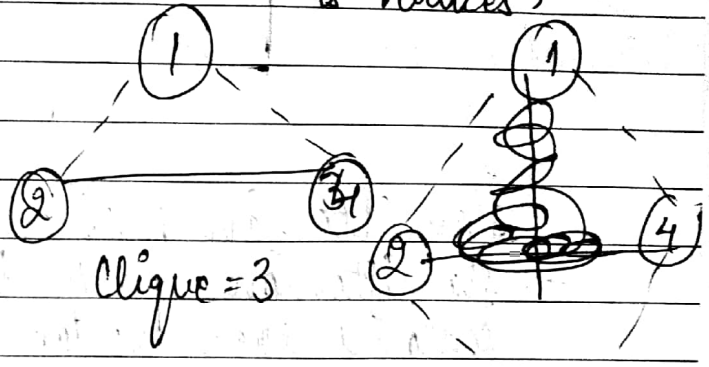
Clique Problem:- Clique is nothing but a maximal complete sub graph of a graph G. The graph G is a set of (V,E) where V is a set of Vertices and E is a set of edges.

Here [Size of Clique = n(V)] → where n is number & V is Vertices.

Eg:-



Graph G.



Clique = 3

Clique = 4

Max Clique Problem:- It is an optimization problem in which the largest size clique is to be obtained. In the decision problem of Clique, we have to determine whether G has a clique of size at least k for given k.

Vertex Cover:- A Vertex cover problem is to find vertex cover of minimum size in a given graph. The word vertex cover means each vertex covers its incident ~~vertex~~ edges.

Example from NOTES of Six.

Page No.

Hamiltonian Problem :-

→ Hamiltonian path is a single open path that contains each vertex in a graph exactly once. And if the source vertex and destination vertex of the Hamiltonian path is the same then it is called Hamiltonian path.

→ Hamiltonian path problem is the problem to determine whether a given graph contains a Hamiltonian path.

→ For a given graph to show the problem to be NP complete we first show that it belong to class NP.

Ans :- 3 SAT is NP complete Problem

Let S be the Boolean formula having 3 literals in each clause for which we can construct a simple non-deterministic algo which can guess an assignment of boolean values of S.
If S is evaluated as 1 so S is satisfied.
Thus 3SAT is in NP complete.

Page No.

Date ___/___/_____ (TSP) Saathi

Travelling Salesman Problem - From Book.

Approximation Algo :-

The approx. algos are algos used to find approximate solutions to optimization problems.

These are often associated with NP-hard problems because it is very difficult to get an efficient polynomial time exact algo for solving NP-hard problems.

Heuristics :- Collection of common sense rules defined from experience.

Approximation algo for Travelling Salesman

The problem is based on the idea of obtaining optimum tour when travelling between many cities.

The decision version of algo ∈ class NP complete problem & optimization ~~problems~~ version ∈ NP hard class of problem.

Two approx. algos for TSP
- Nearest Neighbour algo
- Twice-around the tree algo

→ **Nearest Neighbour Algo :-**

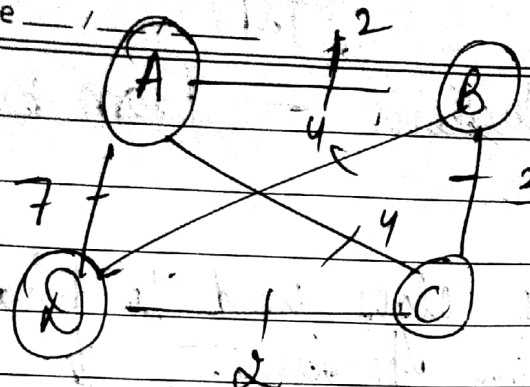
Algo is based on idea of choosing Nearest Neighbour

Algo :-

1. Start at any ℘ city
2. Repeat until all the nodes are visited
3. Return to the starting city

Page No. ____

Date



(S_a) = A - B - C - D - A → 2 + 3 + 2 + 7 = 14

(S^*) = A - B - D - C - A → 12

$$\text{Accuracy Ratio} :- \frac{f(s_a)}{f(s^*)} = \frac{14}{12} = 7:6$$

→ Twice around the tree - algo.
Algo is based on important subset instance called Euclidean.

Triangle inequality

Symmetric $dist[i,j] = dist[j,i]$

$dist[i,j] \leq dist[i,k] + dist[k,j]$

$r(s_a)$ = Accuracy Ratio :- $\frac{f(s_a)}{f(s^*)} \leq \frac{1}{2}\left[(\log_2 n) + 1\right]$

n = total no. of cities

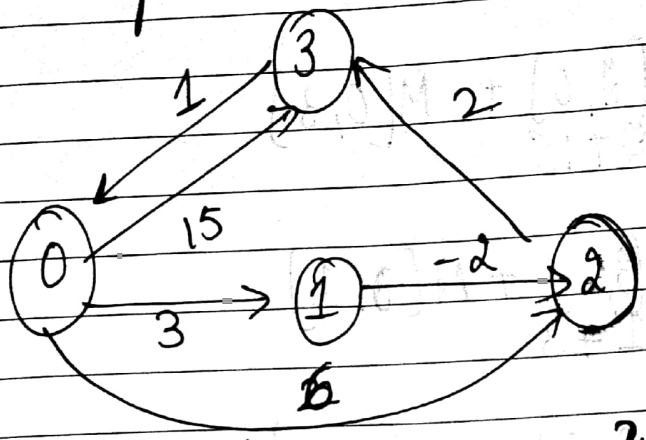
Algo :-
1. Compute Minimum Spanning tree
2. Start any of arbitrary city for Walk
3. Eliminate duplicates from Generated Node list

Page No.

Saathi

Date __/__/____

Floyd Warshall MST-2



$$M = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 6 & 15 \\ \infty & 0 & -2 & \infty \\ \infty & \infty & 0 & 2 \\ 1 & \infty & \infty & 0 \end{bmatrix}$$

Intermediate Vertex → 0

Vertex - 0 , Put Row 0 & column 0 as it is and solve for other rows.

$$M = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & 6 & 15 \\ \infty & 0 & -2 & \infty \\ \infty & \infty & 0 & 2 \\ 1 & 4 & 7 & 0 \end{bmatrix}$$

$M^0[1,2] = M[1,0] + M[0,2]$

$-2 < \infty + 2$

$M^0[1,3] = M[1,0] + M[0,3]$

$\infty < \infty + 15$

Page No.

Date __/__/____

$M^0[2,1] = M[2,0] + M[0,1]$
$\infty \quad < \quad \infty + 3$

$M^0[2,3] = M[2,0] + M[0,3]$
$2 \quad < \quad \infty + 15$

$M^0[3,1] = M[3,0] + M[0,1]$
$\infty \quad > \quad 1 + 3$

$M^0[3,2] = M[3,0] + M[0,2]$
$\infty \quad > \quad 1 + 6$

Vertex 1, Row 1 & Column 1 will be same

$$M' = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 0 & 1 & 2 & 3 \\ 0 & 3 & -2 & 15 \\ \infty & 0 & -2 & \infty \\ \infty & \infty & 0 & 2 \\ 1 & 4 & 2 & 0 \end{bmatrix}$$

Consider Vertex 1 as intermediate Vertex

$M'[0,2] = M^0[0,1] + M^0[1,2]$
$6 \quad > \quad 3 + -2$

$M'[0,3] = M^0[0,1] + M^0[1,3]$
$15 \quad > \quad 3 + \infty$

$M'[2,0] = M^0[2,1] + M^0[1,0]$
$\infty \quad < \quad \infty + \infty$

$M'[2,3] = M^0[2,1] + M^0[1,3]$
$2 \quad < \quad \infty + \infty$

$M'[3,0] = M^0[3,1] + M^0[1,0]$
$1 \quad < \quad 4 + \infty$

Page No.

Saairii

Date ___/___/_____

$M^1[3,2] = M^0[3,1] + M^0[1,2]$
$7 = 4 + (-2)$
$7 > 2$

Vertex 2, Row 2 & Column 2 will be same & Intermediate Vertex will be 2.

$M^2 =$

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

$M^2[0,1] = M^1[0,2] + M^1[2,1]$
$3 \leq 1 + \infty$

$M^2[0,3] = M^1[0,2] + M^1[2,3]$
$15 > 1 + 2$

$M^2[1,0] = M^1[1,2] + M^1[2,0]$
$\infty < -2 + \infty$

$M^2[1,3] = M^1[1,2] + M^1[2,3]$
$\infty > -2 + 2 = 0$

$M^2[3,0] = M^1[3,2] + M^1[2,0]$
$1 < 2 + \infty$

$M^2[3,1] = M^1[3,2] + M^1[2,1]$
$4 < 2 + \infty$

Page No.

Saathi

Date ___/___/_____

Vertex 3 , Row 3 & Column 3 will be same &
Intermediate Vertex will be 3

$M^3 =$

	0	1	2	3
0	0	3	1	3
1	1	0	-2	0
2	3	6	0	2
3	1	4	2	0

$M^3[0,1] = M^2[0,3] + M^2[3,1]$
$3 \leq 3+4$

$M^3[0,2] = M^2[0,3] + M^2[3,2]$
$1 < 3+2$

$M^3[1,0] = M^2[1,3] + M^2[3,0]$
$\infty > 0+1$

$M^3[1,2] = M^2[1,3] + M^2[3,2]$
$-2 < 0+2$

$M^3[2,0] = M^2[2,3] + M^2[3,0]$
$\infty > 2+1$

$M^3[2,1] = M^2[2,3] + M^2[3,1]$
$\infty > 2+4$

Date __/__/____

Question Papers Sol

Saathi

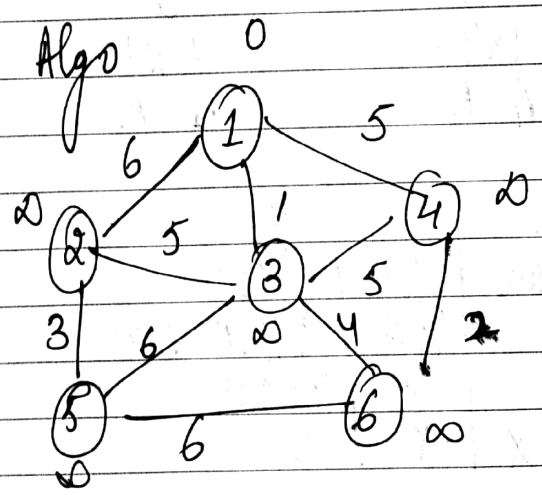
1. LCS

"AGGTAB" & "GXTXAYB"
[Row] [Column]

		G	X	T	X	A	Y	B
	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1	1	1
G	0	1	1	1	1	1	1	1
G	0	1	1	1	1	1	1	1
T	0	1	1	2	2	2	2	2
A	0	1	1	2	2	3	3	3
B	0	1	1	2	2	3	3	4

G T A B

2. Prims Algo



Vertex	Key	Parent
1	0	Nil
2	~~∞~~ 3	~~Nil~~ 5
3	~~∞~~ 1	~~Nil~~ 1
4	~~∞~~ 2	~~Nil~~ 6
5	~~∞~~ 6	~~Nil~~ 3
6	~~∞~~ 5	~~Nil~~ 3

Page No.

Saathi

Date __/__/____

(4) 0/1 Knapsack problem, Dynamic Programming

items = 4, Knapsack Size = 8

profit = {1, 2, 5, 6}

wt = {2, 3, 4, 5}

Pi	Wi		0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

(7.) (30, 1, 40, 10, 25)

A: 30×1, P_0 P_1 | B: 1×40, P_1 P_2 | C: 40×10, P_2 P_3 | D: 10×25, P_3 P_4

i/j	1	2	3	4
1	0	1200	400	1400
2	X	0	400	650
3	X	X	0	10000
4	X	X	X	0

k=1

$M[1,2] = M[1,1] + [2,2] + P_0 P_2 P_1$

$= 0 + 0 + 30 \times 40 \times 1$

$= 1200$

k=2

$M[2,3] = M[2,2] + M[3,3] + P_1 P_3 P_2$

$= 0 + 0 + 1 \times 10 \times 40$

$= 400$

Page No.

Date __/__/____

K=3

M[3,4] = M[3,3] + M[4,4] + $P_2 P_3 P_4$

= 0 + 0 + 40 × 10 × 25

= 10000

K=1

M[1,3] = M[1,1] + M[2,3] + $P_0 P_1 P_3$

= 0 + 400 + 30 × 1 × 10

= 400 + 300 = 700

K=2

M[1,3] = M[1,2] + M[3,3] + $P_0 P_2 P_3$

= 1200 + 0 + 30 × 40 × 10

= 1200 + 12000 = 13200

K=2

M[2,4] = M[2,2] + M[3,4] + $P_1 P_2 P_4$

= 0 + 10000 + 1 × 40 × 25

= 10000 + 1000 = 11000

K=3

M[2,4] = M[2,3] + M[4,4] + $P_1 P_3 P_4$

= 400 + 0 + 1 × 10 × 25

= 400 + 250 = 650

K=1

M[1,4] = M[1,1] + M[2,4] + $P_0 P_1 P_4$

= 0 + 650 + 30 × 1 × 25

= 650 + 750 = 1400

K=2

M[1,4] = M[1,2] + M[3,4] + $P_0 P_2 P_4$

= 1200 + 10000 + 30 × 40 × 25

= 1200 + 10000 + 30000 = 41200

Page No. ____

Saath

Date __/__/____

K=3

M[1,4] = M[1,3] + M[4,4] + $P_0 P_3 P_4$

= 700 + 0 + 30 × 10 × 25

= 700 + 7500 = 8200

K table

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

Parenthesization

(A) (B C D)

Paper 2

(4) Fractional Knapsack

Item	Weight	Value	V_i/W_i
1	5	15	3
2	7	28	4
3	4	20	5
4	6	24	4
5	8	32	4

FCFS

Arrangement

Item	Wt.	Value	V_i/W_i
1	5	15	3
2	7	28	4
4	6	24	4
5	8	32	4
3	4	20	5

Page No.

Saathi

Date / /

Greedy Method W = 15

Item	Value	Wt.	Rem. Wt.
—	—	—	15
3	20	4	15-4 = 11
2	28	7	11-4 = 7
4	24	6	7-4 = 3
5	4x3 = 12	3	3-3 = 0

Total Profit → 20+28+24+12 = 84

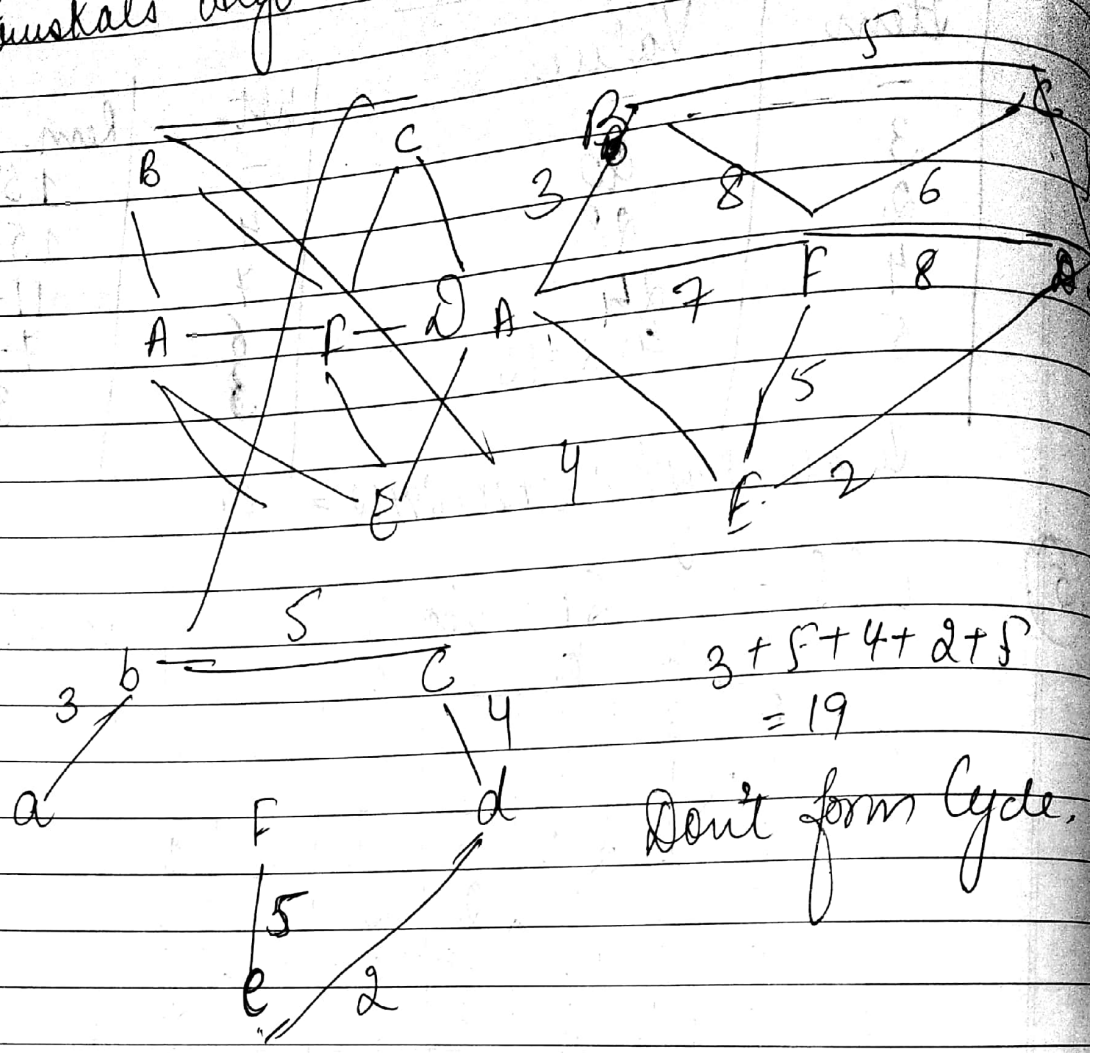
5. LCS → S_1 = abccd S_2 = accadb

	0	a	b	c	a	b	d
0	0	0	0	0	0	0	0
a	0	1	1	1	1	1	1
b	0	1	1	1	1	2	2
c	0	1	2	2	2	2	2
c	0	1	2	3	3	3	3
d	0	1	2	3	3	3	4

a c c d

Date __/__/____

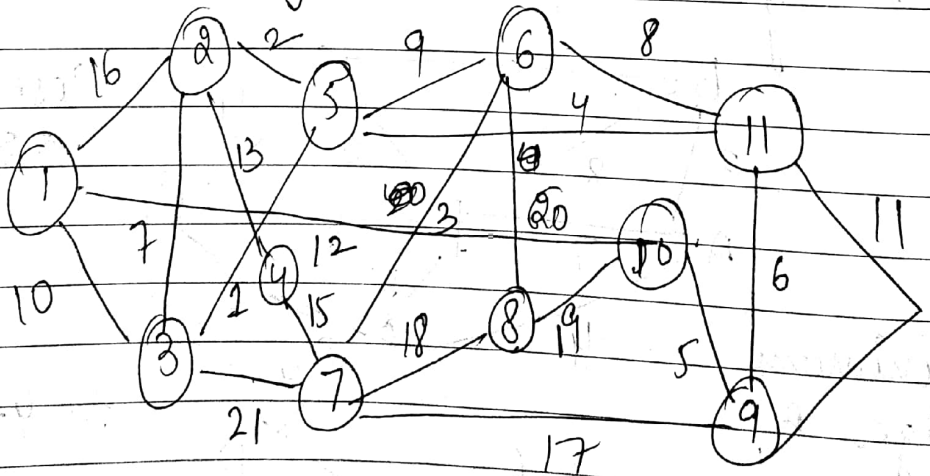
7. Kruskal's Algo



3+5+4+2+5
=19

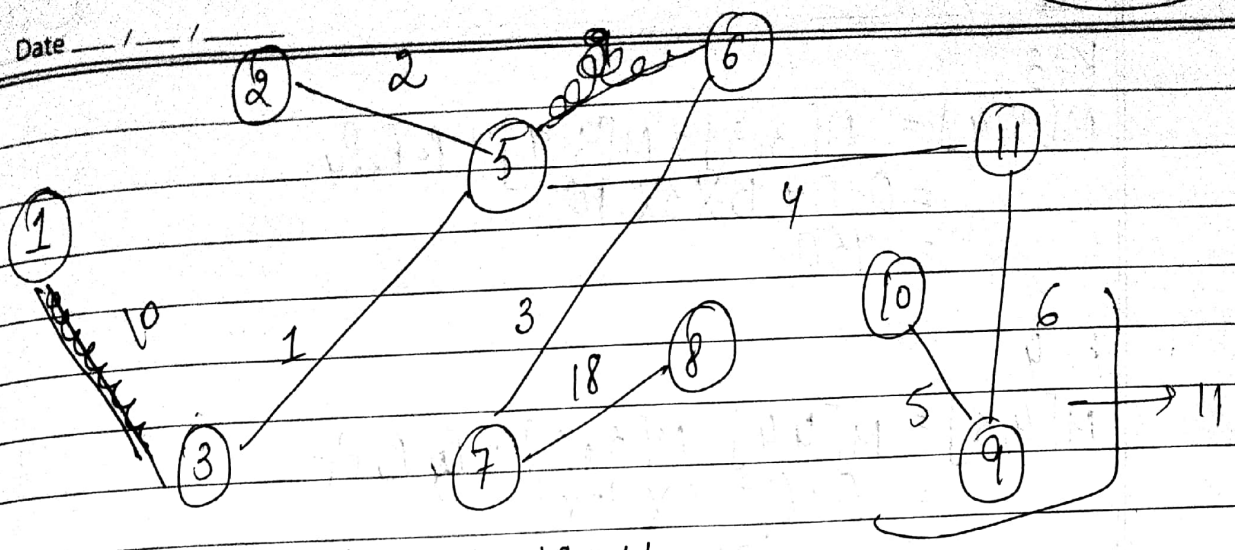
Don't form Cycle.

8. Kruskal's Algo.



Page No.

Date __/__/____



Cost = 2+10+1+3+4+18+11
= 49

(8) CMM

{30, 35, 15, 5, 10, 20, 25}

A	B	C	D	E	F
30×35	35×15	15×5	5×10	10×20	20×25
P_0 P_1	P_2	P_3	P_4	P_5	P_6

i/j	1	2	3	4	5	6
1	0	15750	7875			
2	X	0	2625	4375		
3	X	X	0	750	2500	
4	X	X	X	0	1000	3500
5	X	X	X	X	0	5000
6	X	X	X	X	X	0

→ Same Method

k=1

M[1,2] = M[1,1] + M[2,2] + $P_0 P_1 P_2$
= 0+0+30×35×15
= 15750

k=2

M[2,3] = M[2,2] + M[3,3] + $P_1 P_2 P_3$
= 0+0+35×15×5
= 2625

Page No.

Date ___/___/_____

K=3

$M[3,4] = M[3,3] + M[4,4] + p_2 p_3 p_4$

$= 0 + 0 + 15 \times 5 \times 10$

$= 750$

K=4

$M[4,5] = M[4,4] + M[5,5] + p_3 p_4 p_5$

$= 0 + 0 + 5 \times 10 \times 20$

$= 1000$

K=5

$M[5,6] = M[5,5] + M[6,6] + p_4 p_5 p_6$

$= 0 + 0 + 10 \times 20 \times 25$

$= 5000$

K=1

$M[1,3] = M[1,1] + M[2,3] + p_0 p_1 p_3$

$= 0 + 2625 + 30 \times 35 \times 5$

$= 2625 + 5250 = 7875$

K=2

$M[1,3] = M[1,2] + M[3,3] + p_1 p_2 p_3$

$= 15750 + 0 + 35 \times 15 \times 5$

$= 15750 + 2625 = 18375$

K=2

$M[2,4] = M[2,2] + M[3,4] + p_1 p_2 p_4$

$= 0 + 750 + 35 \times 15 \times 10$

$= 750 + 5250$

$= 6000$

Date

K=3

$M[2,4] = M[2,3] + M[4,4] + P_1 P_3 P_4$

$= 2625 + 0 + 35 \times 5 \times 10$

$= 2625 + 1750$

$= 4375$

K=3

$M[3,5] = M[3,3] + M[4,5] + P_2 P_3 P_5$

$= 0 + 1000 + 15 \times 5 \times 20$

$= 1000 + 1500 = 2500$

K=4

$M[3,5] = M[3,4] + M[5,5] + P_2 P_4 P_5$

$= 750 + 0 + 15 \times 10 \times 20$

$= 750 + 3000 = 3750$

K=4

$M[4,6] = M[4,4] + M[5,6] + P_3 P_4 P_6$

$= 0 + 5000 + 5 \times 10 \times 25$

$= 5000 + 1250 = 6250$

K=5

$M[4,6] = M[4,5] + M[6,6] + P_3 P_5 P_6$

$= 1000 + 0 + 5 \times 20 \times 25$

$= 1000 + 2500 = 3500$

K=1

$M[1,4] = M[1,1] + M[2,4] + P_0 P_4 P_1$

$= 0 + 4375 + 30 \times 10 \times 35$

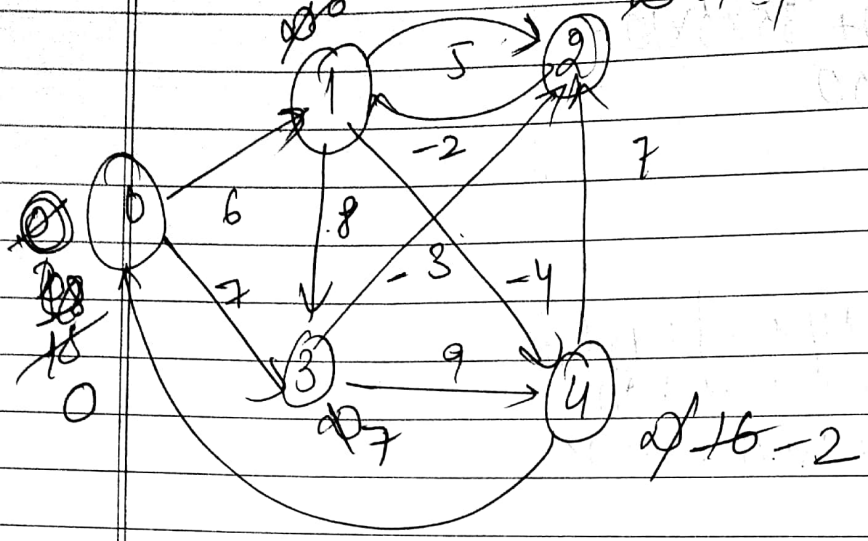
$= 4375 + 10500 = 14875$

Page No.

Date ___/___/_____

Bellman Ford Method

[MST-2]



Vertex 0 – 0
1 – 2
2 – 4
3 – 7
4 – -2