Review of foundational Mathematics

(i) Summation of Series

$$1 + 2 + 3 \cdots n = \frac{n(n+1)}{2}$$

$$1^2 + 2^2 + \cdots n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$1^3 + 2^3 + \cdots n^3 = \frac{n^2(n+1)^2}{4}$$

Q- $$I = \int_0^{10} \frac{[x]e^{[x]}}{e^{x-1}} \cdot dx$$

(ii) Arithmetic Progression

$a, \quad a+d, \quad a+2d, \cdots \cdots, \underbrace{a+(n-1)d}_{T_n : n^{th} \text{ term}}$

$$S_n = \frac{n}{2}(2a + (n-1)d)$$

(iii) GP

$a, ar, \cdots \underbrace{ar^{n-1}}_{T_n}$

$a, ar, ar^2, \cdots \infty$ terms

$$S_n = \frac{a}{1-r}$$

(iv) $S = 1 + 2x + 3x^2 + \cdots nx^{n-1}$ — (1)

$xS = x + 2x^2 + \cdots nx^n$ — (2)

(1) − (2)

$S(1-x) = \left(1 + x + x^2 + \cdots x^{n-1}\right) - nx^n$

$\Rightarrow S = \frac{1}{1-x}\left\{\frac{1-x^n}{1-x} - nx^n\right)$

H.W (a) $\frac{3+5+7\cdots \text{upto } n \text{ terms}}{5+8+11+\cdots \text{to 10 terms}} = 7$

find n

(b) $1 + \sqrt{3} + 3 + 3\sqrt{3} \cdots$ upto 10 terms

(c) $-\frac{5}{4} + \frac{5}{16} - \frac{5}{64} + \cdots \infty$

(d) $(\sqrt{2}+1) + 1 + (\sqrt{2}-1) \cdots \infty$

(e) $1^3 + 2^3 + \cdots 40^3$

(f) $11^2 + 12^2 + \cdots + 30^2$

(g) $2^3 + 4^3 + 6^3 + \cdots + 60^3$

Binomial Theorem

(i) $(1+x)^n = 1 + {}^nC_1 x + {}^nC_2 x^2 + \cdots + {}^nC_n x^n$

(ii) $(a+x)^n = a^n + {}^nC_1 a^{n-1} x + \cdots + {}^nC_n x^n$

(iii) $(1-x)^n = 1 - {}^nC_1 x^1 + {}^nC_2 x^2 + \cdots (-1)^n {}^nC_n x^n$

(iv) $(a-x)^n = a^n - {}^nC_1 x^1 a^{n-1} + {}^nC_2 x^2 a^{x-2}$

Special Cases

$(1+x)^n \simeq 1 + nx$ if $|x| \ll 1$

$(1-x)^n \simeq 1 - nx$ if $|x| \ll 1$

$$(1+x)^n = 1 + nx + \frac{n(n-1)x^2}{2} + \frac{n(n-1)(n-2)x^3}{\lfloor 3} + \frac{n(n-1)(n-2)(n-3)x^4}{\lfloor 4} \cdots$$

Ex-1 -

```
int main()
{
    int p, q = 10;      O(1)
    p = q * 25;         O(1)
    cout << p;          O(1)
}
```

T.C = O(1)

Ex-2

```
int main()
{
    int n;  Cin >> n;
    int a[n];
    int i; int sum = 0;
    for (i = 0; i < n; i++) {
        sum += a[i];
        cout << sum;
    }
```

Ex-3

```
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
        sum += j
    }
}
```

Ex-4
```
for (i = 0; i<n; i++)
{
  for (j = i; j<n; j++)
  {
    sum sum += i+j;
  }
}
```

Ex-5
```
for (i = 1; i<n; i = i * 2)
    sum += i
```

Ex-6
```
for (i=1; i*i <n; i++) {
    sum += i;
```

Ex-7
```
for (i = 0; i<n; i++){
    for (j = n; j>0; j = j/10)
        sum += i+j
    }
}
```

Q- Write an algorithm for finding the factorial of a number using recursion. Draw its flowchart and implement it using C++.
Compare and contrast the time and space complexity of recursive with iterative approach.

Exc-

$$f(n) = \sum_{i=0}^{n} (n-i) * f[i] \; ; \; f[0] = 1$$

$f[0] = 1$

$$f[1] = \sum_{i=0}^{1} (1-i) * f[1-i]$$

$= (1-0) * f[0] + (1-1) * f[1]$

$= 1 * 1 = 1$

$f[2] = (2-0) * f[0] + (2-1) * f[1]$

$= 2 * 1 + 1 * 1$

$= 2 + 1$

$= 3$

$f[3] = (3-0) * f[0] + (3-1) * f[1] + (3-2) * f[2]$

$= 3 * 1 + 2 * 1 + 1 * 3$

$= 3 + 2 + 3$

$= 8$

$f[4] = (4-0) * 1 + 3 * 1 + 2 * 3 + 1 * 8$

$= 4 + 3 + 6 + 8$

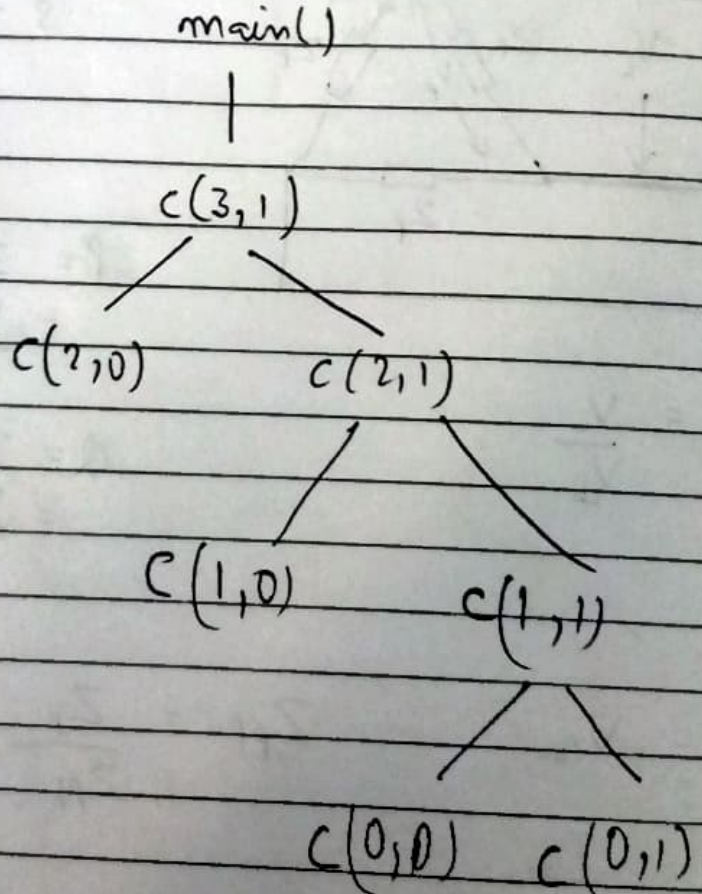
$= 21.$

Prove that $nC_m = n-1C_m + n-1C_{m-1}$.

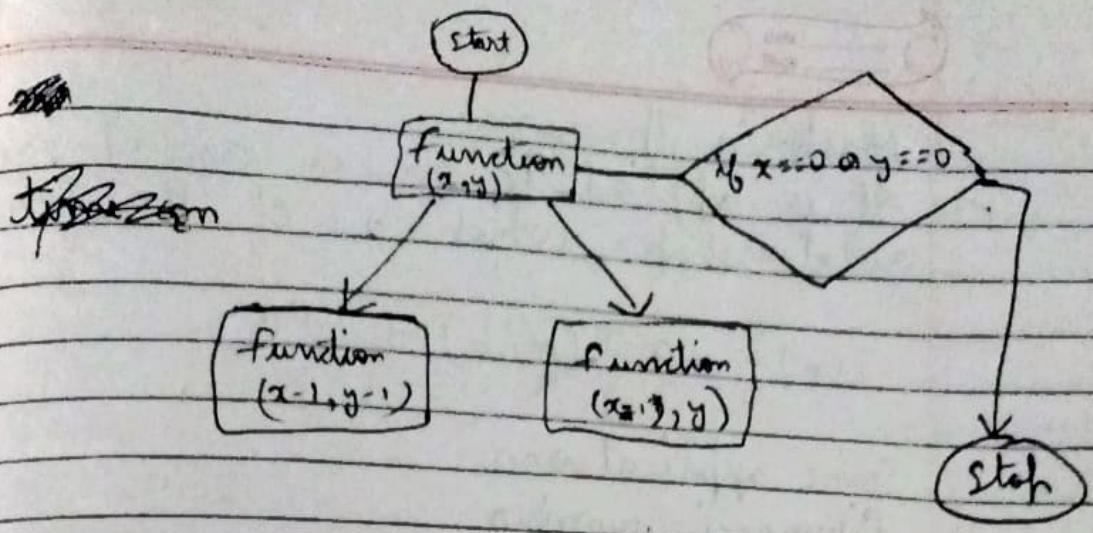
$$^nC_m = n \frac{(n-m)!\, n!}{m!}$$

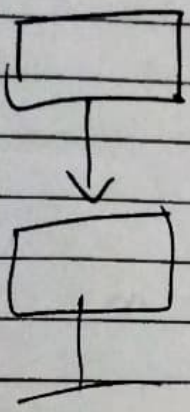
$$\frac{(n-m-1)!\,(n-1)!}{m!} +$$

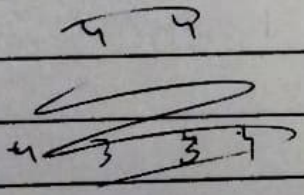
Recursion tree for C(3,1)



time complexity :- $O(2^n)$







Divide and Conquer

→ Break the original problem into smaller subproblem which are similar to the original.

→ Solve the subproblems recursively.

→ Combine the solution to the subproblem to get the solution to the original problem.

Master's Theorem

→ It is applied to special cases of recurrence relationship, which are of the form:-

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

Some applications:

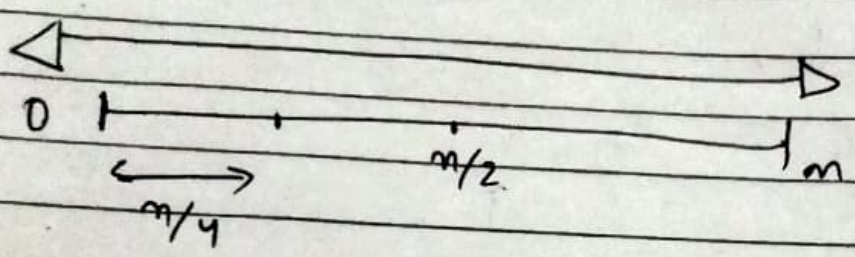
→ fibonacci number

→ Merge Sort.

For merge sort:-

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

Let us try to understand study the binary search version of integer square root by closing



→ $\frac{n}{2} * \frac{n}{2} == n$

→ $\frac{n}{2} * \frac{n}{2} > n$ $[0, n/2]$

→ $\frac{n}{2} * \frac{n}{2} < n$ $[n/2, n]$

~~fo~~ while left < right:

~~& ans~~

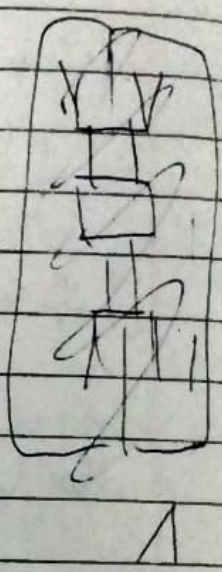
mid = (left + right) // 2

~~ans~~

if ~~@~~ right * right > n

Q- Compute the exact bound for the time complexity of the following code:

```
f(n){
    for ( )
        { } → log2 n
    return T√n + T√n
}
```



$2T(n/2) + n$

(i) $T(n) = 2T(n/4) + 1$

(ii) $T(n) = 3T(n/6) + n$

(iii) $T(n) = 3T(n/4) + n$

(iv) $T(n) = 6T(n/4) + n^2$

(v) $T(n) = 6T(n/2) + n^2$

i) $a = 2 ; b = 4$

$k = \log_b a = \log_4 2 = \frac{1}{2}$

$n^k = n^{1/2} = n^{1/2}$

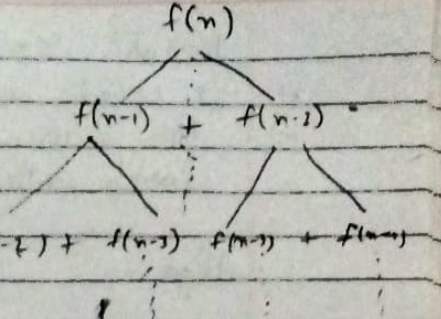
Comparing

(ii) $T(n) = a = 3 ; b = 6$

$K = \log_6 3$

$= \frac{1}{2}$

$n^K = \sqrt{n}$ & $n^{\log_6(3)}$



(iii) $\log_2 a = 3$, $b = 4$

$K = \log_a(3)$

n

(iv) $n^{\log_4(6)}$

(v) n^2

Binary Search vs Linear Search

```
int linear_search ( int A[], int size, int
                    value ) {

    int i;

    for( i=0; i< size ; i++){
        if (A[i] == value ) {
            return i; }

    return -1;
}
```

Binary search version

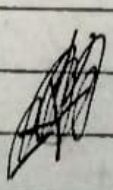
→ It works only for sorted array only.
→ after every three interval interation it's search interval is halved.

Algo :-

→ Compare value with middle element.
→ If be == value;

```
int ls(int arr[], int element, int size) {

    for(int i=0; i< size; i++) {
        if (arr[i] == element) {
            return i;
        }
    }
    return -1;
}
```



```
int bs( int arr[]; int element, int size){
    int left = 0;
    int right = size -1;
    int mid = (left + right) /2;
    while ( left < right ) {
        if (arr[mid] == element) {
            return mid;
        }
```

```
        if (arr[mid] > element) {
            left = mid + 1;
        }
        else {{
            right = mid - 1;
        }
    }
    return -1;
}
```

Integer Square root

ex:- $\sqrt{27} = 5$
$\sqrt{49} = 7$
$\sqrt{29} = 5$
$\sqrt{104} = 10$

largest integer x such that
$x * x \leq n$

Standard Algo

```
for (i=0; i*i<=n; i++);
    cout i-1;
```

Graph based Algorithms

- Shortest Path
  - Single source shortest path to all vertices
  - All pairs shortest path
- Minimum Spanning Tree

Tree - A connected graph without cycle

Spanning tree → A tree which includes all vertices.

Cost - refers to the weight on the graph

→ the objective of any algo is to minimize the cost.

App of shortest path tree :-

(a) Navigation
(b) Routing in computer networks

All pair shortest paths :- shortest path for all the pairs.

Ex Suppose Task - to connect locality with pipelines.

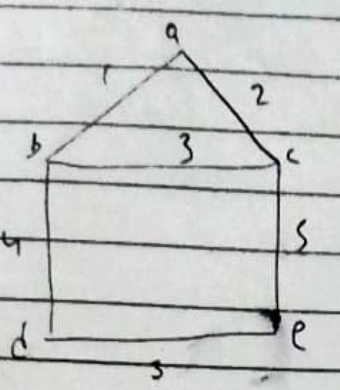
Outcome - to minimize the pipe length with all the localities

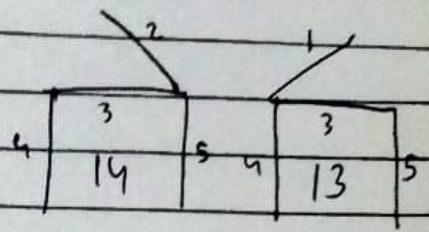
Goal - to minimize the pipeline length with all the locations joined.

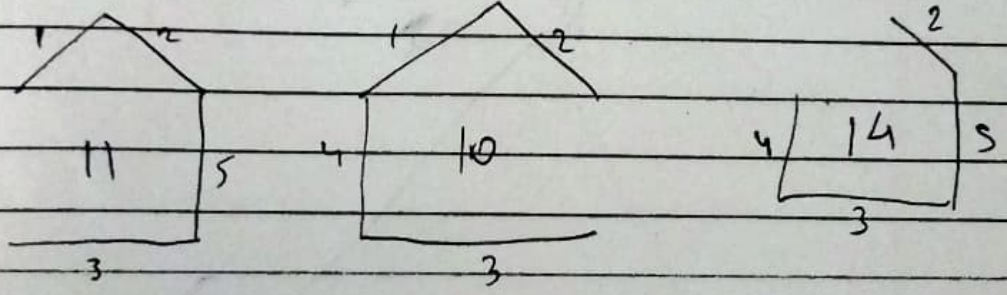
This is a minimum spanning tree problem

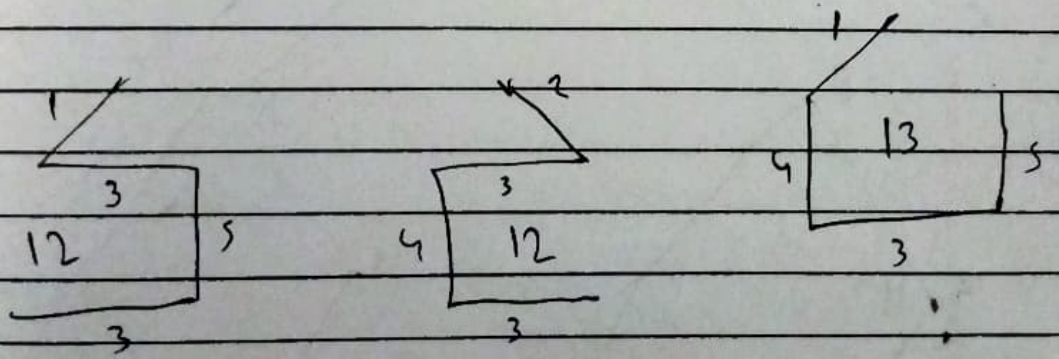
Solution - is to select a graph (tree) with minimum total weight.

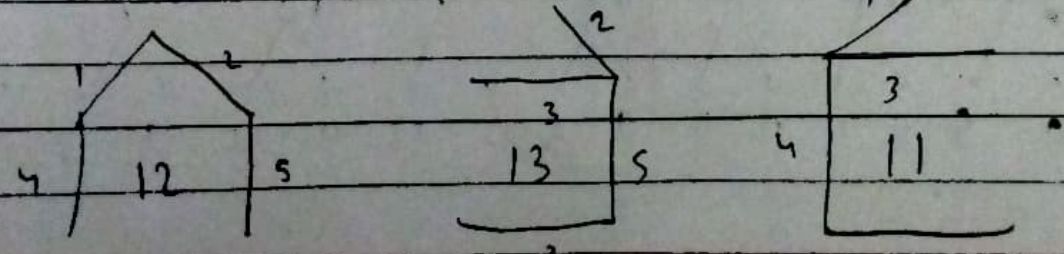
Q For the connected graph, draw all spanning tree:-











Teacher's Signature

Minimum Spanning Tree
- Prim's Algo
- Kruskal's Algo

Kruskal's :-

- Sort edges in increasing order of their weight (O(E log E))
- Add the edge 'e' in the graph, only if it is not forming a cycle (O(E))

(union find data structure is used)

Let E' be the number of edges

Time Complexity :-
O(E log E) + O(E)

Proof of correctness for Kruskal's

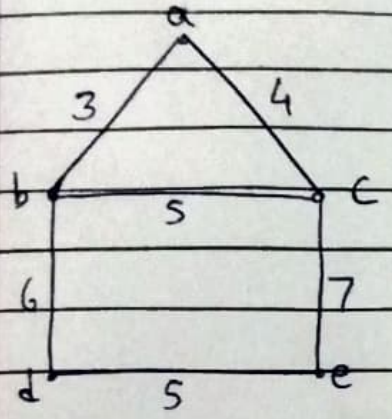
→ Since there are no cycle, ∴ it is a tree.

→ Since every edge is considered ∴ if the original graph was connected, this'll also generate MST.

Teacher's Signature .....

→ Assume that Kruskal's algo does not generate MST, i.e., there exists another MST having lesser weight

→ This is only possible in case of a cycle, which is contradicting to the concept of MST.

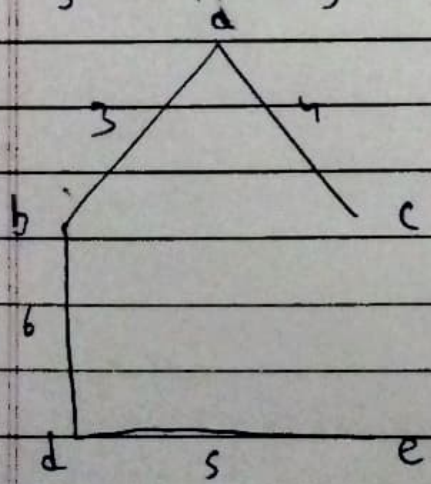


~~3 4~~

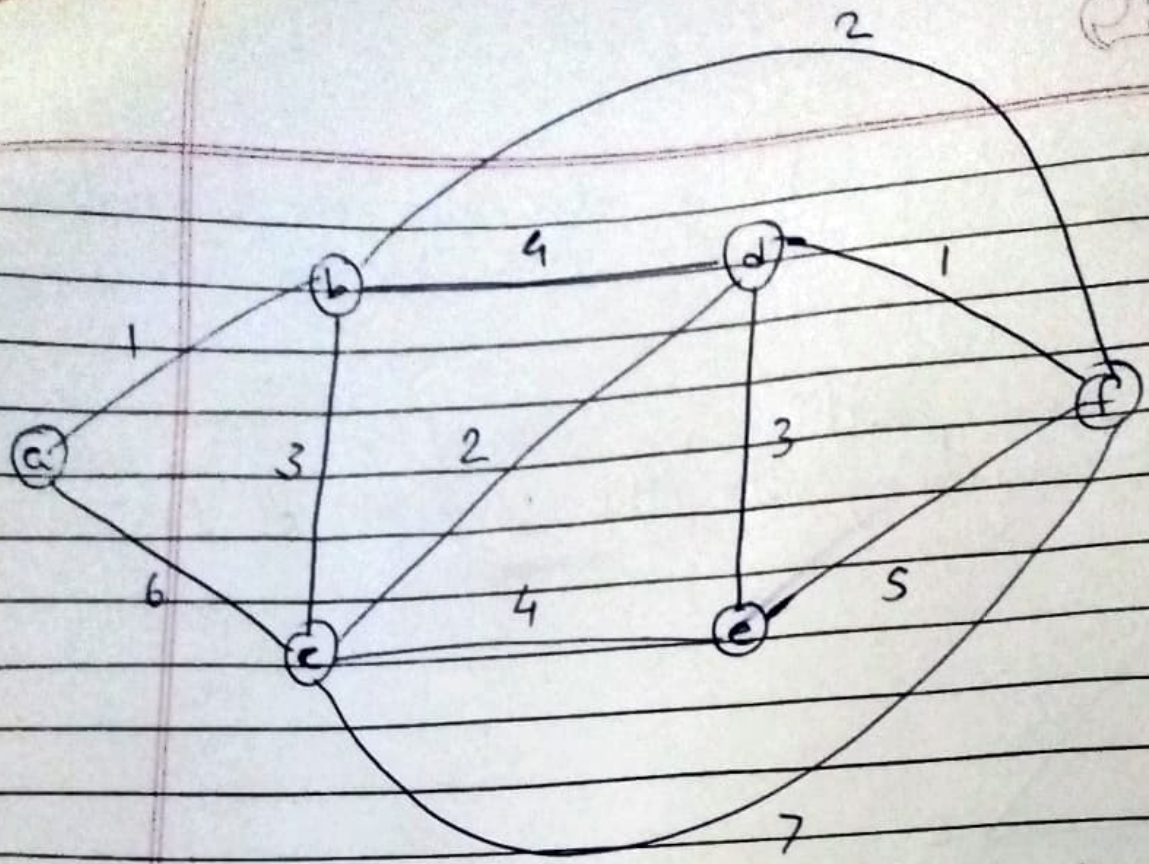
~~ab bc de b~~

Sort → ab ac bc de bd ce
3 4 5 5 6 7

Addy → ab ac de bd
3 4 5 6



Teacher's Signature.....



(i) ab - df - bf - cd - de

(ii) df - ab - dbf - cd - de

(iii) df - ab - cd - bf - de,

(iv) ab - df - bf - de - cd

(v) ab - df - cd - de - bf