Data Structures

Arrays

* If array is sorted use binary search

`mid = start + (end - start)/2 ;` → In the loop while(start < end)

Maximum tip calculator:

→ find difference (absolute differe) → c = abs(a - b) and sort array c, and continue.

Find if string is plaindrome.

→ input fgets(string, sizeof string, stdin);

(If you're taking another input before enter might be taken so call one extra fgets)

Also here if you enter simran, length is 8

so `length = strlen(string) - 2 ;` *

↳ index of last character

* to ignore all special characters and take only alphanumeric

```
if(!(>=65 && <=90 || >=97 && <=122 || >=48 && <=57)
    i++; continue; (ignore)
```

to convert to lowercase

```
if (>=65 && <= 90)
    s[i] = s[i] + 32;
```

if copying one string to other do character wise and at last string2[i] = '\0'

↳ last char → strlen(string2) - 1

Binary search Tree

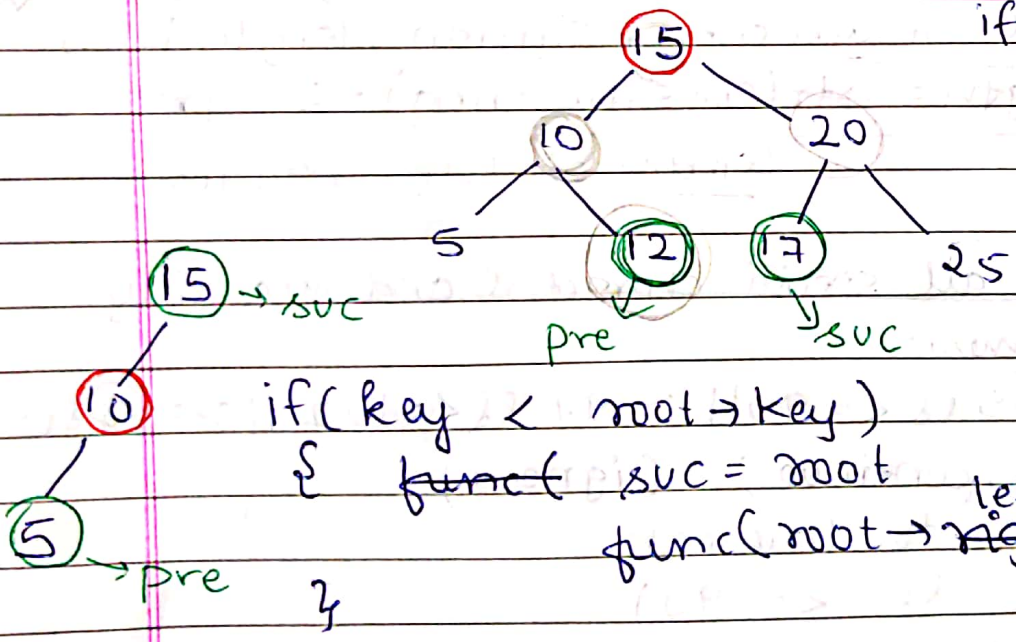
→ Inserting ✓ O(n)
if(node == NULL) create a new node (root)

→ Inorder traversal of BST always produces sorted output

→ So can always get inorder traversal by sorting any given traversal.

→ Deleting ✓ O(n)

→ Predecessor and successor



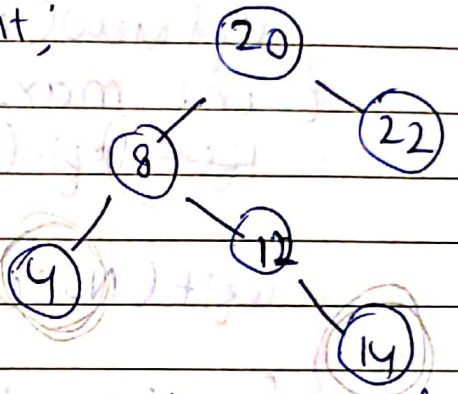
if(key < root→key)
{ ~~func~~ suc = root
func(root→~~right~~ left, ...)
}

→ If a given Binary tree is BST

~~do preorder~~ inorder traversal store in array, check if array sorted xD

Lowest common Ancestor in a BST

```
{
while(root != null)
{
    if( n1 < root->data && n2 < root->data)
    {       root = root->left;
    else if( n2 > root->data && n1 > root->data)
            root = root->right;
    else
        break;
}
return root;
}
```



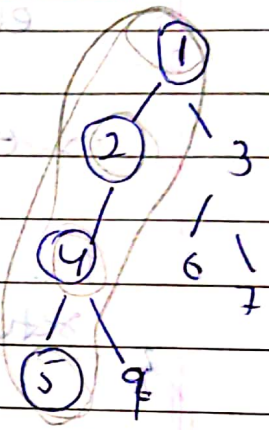
The value always lies b/w n1 and n2 and is the first such value even if its n1 or n2

★ Distance b/w n1 and n2 = distance from root to n1 + distance from root to n2 - 2 * distance from root to LCA

Unsorted array, find max of three elements

sort array return max(a[0], a[1], a[n-1] OR
a[n-1] & a[n-2] a[n-3])

Left view of a binary ~~search~~ tree.




```
left view(Node * root)
{ int max_level = 0;
  left(root, &max_level, 1)

left(Node * root, int * max_level,
                  int level)
{ if(root == NULL)
      return;
   if(* max_level < level)
   {
      printf(root → data)
      *max_level = level;
   }
  left(root→left, &max_level, level+1);
  left(root→right, max_level, level+1);

}
```

Middle of a linked list (slow, fast pointer)

while(fast != NULL && fast->next != NULL)

Aptitude

~~Prime no. exp to p^2 if not divisible~~

* Find smallest ~~pr~~ number n such that $n^2 \geq p$, check if p divisible by a prime no $\leq n$

* Divisibility test

→ By 3 → ~~sum~~ sum divisible by 3
→ By 4 → last two digits divisble by 4
→ By 6 → if divisible by 2 or 3
→ By 7 → remove last digit, subtract double of last digit from remaining no, if left 0, 7 ✓
→ By 8 → last 3 digits divisby by 8
→ By 9 → sum of digits
→ By 11 → sum of ele at even position − odd posi. = multiple of 11 or 0

$\frac{x^n - a^n}{x-a}$ divisible $\forall$ all $\frac{x^n - a^n}{x+a}$ $x \in$ even

$\frac{x^n + a^n}{x+a}$ ✓ for $\forall\ x \in$ odd

HCF | LCM (×)

Page No.:
Date : / /

LCM and HCF.

→ For 2 numbers a and b LCM × HCF = a × b

→ Find the greatest number which on dividing 70 and 50 leaves remainders 1 & 4 respectively
→ Number exactly divides 69 and 46
HCF (69, 46) = 23

→ Find the ~~HCF~~ largest number which divides 64, 136 and 238 to leave the same remainder in each case
→ HCF ((136 − 64), (238 − 136), (238 − 64))
= 6

→ Find the ~~LCM~~ least number which when divided by 5, 7, 9 and 12 leaves remainder 3 in each case
→ LCM (5, 7, 9, 12) = 1260
= 1260 + 3 = 1263

→ Find the largest four digit number exactly divisible by 15, 21 and 28
→ LCM (15, 21, 28) = 420
largest 4 digit number = 9999
⇒ 9999 % 420 → remainder 339
⇒ 9999 − 339 = 9660 //

→ The policeman at 3 different places on a ground blow a whistle after every 42 sec, 60 sec and 78 sec respectively. If they all blow whistle at 9:30:00, when will they blow whistle again together?

→ LCM(42, 60, 78) = 5460 = 1 hour 31 min at 11:01

→ Find the least no. which when divided by 6, 7, 8 leaves remainder 3, but 0 when divided by 9

→ LCM (6, 7, 8) = 168

= 168m + 3 = 171

→ 3 rectangular fields area 60m², 84m², 108m² are to be divided into identical rectangular flower beds, each having length 6m, breadth = ?

→ HCF(60, 84, 108) = 12

6 × x = 12 x = 2 //

WORK & WAGES

Comparision of work and efficiency

$$\frac{M_1 D_1 H_1 E_1}{W_1} = \frac{M_2 D_2 H_2 E_2}{W_2}$$

M_1 = # workers

D_1 = days

H = hours

E = efficiency

W = units of work

Q. A → 10 days to complete B → 15 days ? together ?

→ LCM (10, 15) = 30

A → 3 units/day
B → 2 units/day
→ 5 units per day

30/5 = 6 days

Q. Two friends A & B working together complete work in 4 days. A does in 12 days, B = ?

→ Lets say work = 12 units

A & B = 3 units/day

A → 12/12 → 1 unit/day

B → 2 unit, 12/2 = 6 days

Q A & B - 18 B & C - 24 A & C - 36

A, B, C = ? (together & individual)

LCM (A&B, B&C, A&C) = (18, 24, 36) = 72

A & B — 4 units/day

B + C - 3

A + C — 2

2(A + B + C) = 9 A + B + C = 4.5 /day

72/4.5 = 16 days

Overview of data structures

Array

O(n) insertion [at the beginning]
O(n) deletion

Linked list

Circular linked list (any node can be made starting element)

Access time O(n)
Search O(n)
Insertion, deletion O(1) [depending on implementation]

Stack

access time O(n)

Queue same.

Circular queue (if implemented using array space efficient as can add elements at a[0], a[1] if not present.

Q Print all elements of linked list in reverse order

```
void func1(void * head)
{ if (head == NULL) return;
  func1(head->next);
  printf("%d ", head->data);
}
```

Write code to reverse doubly linked list.

```
void fun( struct node **head )
{
    struct node *temp=NULL;
    struct node * curr = *head;
    while (curr != NULL)
    {
        temp = curr -> prev;
        curr -> prev = curr -> next;
        curr -> next = temp;
        curr = curr -> prev;
    }
    if (temp != NULL)
        *head = temp -> prev;
}
```

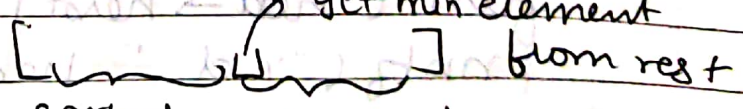
Sort a linked list?

Merge sort and insertion sort

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

Sorting algorithms

Selection sort



$O(n^2)$ time

$O(1)$ auxilary space

* → Never takes more than $O(n)$ swaps, use when memory write is costly operation

In place

Bubble sort

compares adjecent elements, swaps last element in place. $O(n^2)$

WORST CASE — array is reverse sorted.

BEST CASE — $O(n)$

In an almost sorted array, (swap of just two elements) can be fixed in linear time $O(2n)$

Merge Sort

Divide and Conquer.

$O(n \log n) \rightarrow$ always

★ Sort linked list in $O(n \log n)$

merge operation can be implemented in without extra space.

Merge sort access data sequentially so can be applied on linked list, quick sort needs random access.

```
void Mergesort(arr[], l, r)
{   if(l < r)
    {
        m = l+r/2;
        mergeS(arr, l, m);
        merge S(arr, m+1, r);
        merge (arr, l, m, r);
    }
}
```

Teacher's Signature.....

```
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    n1 = m - l + 1;
    n2 = r - m;
    0 ⟶ n1:
        L[i] = arr[l+i]
    0 ⟶ n2
        R[j] = arr[l+m+1+j];
    i=0  j=0  k=l
    while(i<n1 && j<n2)
    {
        if(L[i] <= R[j])
            arr[k] = L[i]
            i++
        else
            arr[k] = R[j]
            j++
        k++

    while(i<n1)
        arr[k++] = L[i++]
    while(j<n2)
        arr[k++] = R[j++]

}
```

Used to find inversion count

```
inv_c = ms
inv_c += ms
inv_c += m
```

Prefix of strings

```
char arr[n][100];
scanf(" %s", arr[i]); → to get input
scanf("%[^\n]", arr[i]; → on new line
```

New1.c

Level order Tree Traversal

* Breadth first traversal

Find the height of a tree

```
int height(struct node *node)
{   if(node == NULL)
        return 0;
    lheight = height(node->left);
    rheight = height(node->right);

    if( lheight > rheight)
        return (1 + lheight);
    else return(1 + rheight);
};

void print_level(struct node *root)
{   int h = height(root);
    for(i=1; i<=h; i++)
        printlevelnew(root, i); }
void printlevelnew(root, int level)
{   if (root == NULL) return;
    if( level == 1) print(root->data);
    else if( level > 1)
        printlevelnew(root->left, level-1);
        printlevelnew(root->right, level-1);
}
```

Teacher's Signature.....

Largest Sum Contiguous Subarray

```
int max( a[], size)
{
    int max = INT_MIN
    int max_here = 0;
    for(i=0; i<n; i++)
    {   max_here += a[i]
        if( max < max_here)
            max = max_here;
        if(max_here < 0)
            max_here = 0;
    }
    return max
}
```

Largest Product contiguous Sub array

```
int max( int * arr, size)
{
    int max = arr[0]
    int min = arr[0]   int maxP = arr[0]
    for(i=1; i<n; i++){
        if (arr[i] < 0)
            swap(max, min);
        max = max(arr[i], arr[i] * max)
        min = min(arr[i], arr[i] * min)
        maxP = max(maxP, max);
    }
    return maxP
}
```

Teacher's Signature.....

Find the Missing Number

Method 1 sum = n(n+1)/2 subtract.

array a , n sum = (n+1)(n+2)/2

method 2

```
x1 = arr[0];
x2 = 1;
for(i=1; i<n; i++)
    x1 = x1 ^ arr[i]
for(i=2; i<=n+1; i++)
    x2 = x2 ^ i;
return x1 ^ x2
```

★★ Reverse words in a given string

input hey what you doin → gets()

```
int t; scanf(&t);
getchar(); → ** after scanf
int main()
{   char s[100];
    gets(s);   }
```

PROGRAM GO THROUGH GFG

Sort an array containing 0, 1, 2

Teacher's Signature.....

Find permutation of a string

l=0 r=n-1

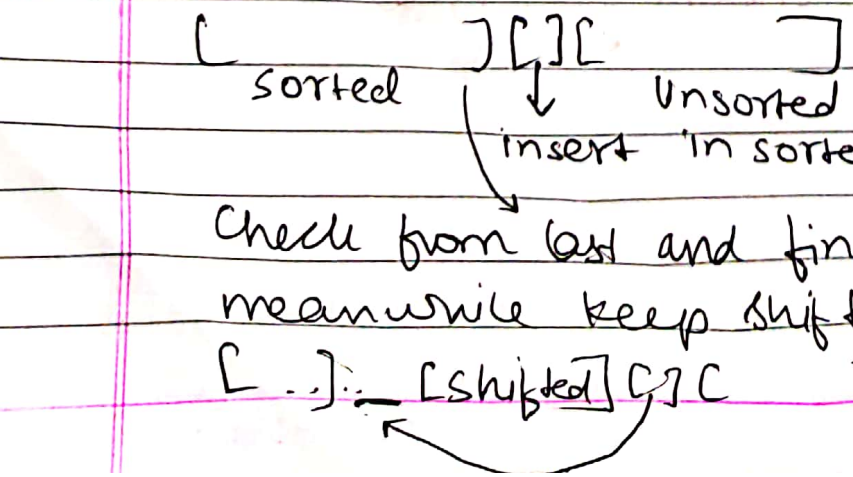
```
void perm(char *s, int l, int r)
{
    if(l==r)
        printf("%s", s);
    else
    {
        for(i=l; i<=r; i++)
        { swap(a+i, a+l);
          perm(s, l+1, r);
          swap(a+i, a+l);
        }
    }
}
```

Count no of set bits.

```
count=0
while(n)
{
    count = count + (n & 1);
    n = n >> 1
}
```

Insertion sort

Can be used for linked list



Teacher's Signature.....

Subarray with a given sum.

```
sum → given sum_ = a[0], start = 0
for (i=1; i<=n ; i++)
{
    if( while( sum_ > sum && start < i-1)
        { sum_ = sum_ - a[start];
          start++;
        }
    if ( sum_ == sum)
        printf( start+1, i )
    if (i<n)
        sum_ += a[i]
}
```

Trapping Rain Water

0 1 0 2 1 0 1 3 2 1 2

find left_max, right_max

0 1 1 2 2 2 2 3 3 3 3

3 3 3 3 2 3 3 3 2 2 2

sum = sum + min(leftmax[i], rightmax[i]) - arr[i]

Pythagorean triplet in an array.

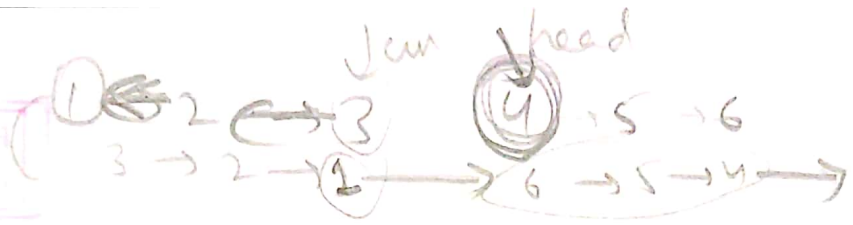
3 1 4 6 5

find sq. 9 1 16 36 25

sort 1 9 16 25 36

```
for(int i=n-1; i>=2; i--)
{ l=0, r=i-1;
  while (l<r)
  { if(arr[l]+arr[r] == arr[i])
        return true.
    (arr[l]+arr[r]>arr[i])?r--:l++;
  }
return false;
```

Teacher's Signature.....



Page No.:
Date : / /

Reverse a linked list in groups of given size.

Permutation and Combination

Permutation: arrangement. (A, B) → AB, BA

$^nP_r = n!/(n-r)!$

Combination: (A, B) → AB

$n!/(n-r)!r!$

$^nC_r = {}^nC_{n-r}$

Words from DRIVER all vowels together.

D, (R), I, E, V, (R)

$= \frac{5 \times 4 \times 3 \times 2 \times 1}{2!} \times 2$

(R is repeated)

2 vowels can be interchanged

In how many ways can 20 boys and 18 girls make a queue such that no 2 girls are together?

$20! \times {}^{21}P_{18}$

There are 5 floating stones on a river. A man wants to cross the river. He can move 1 or 2 steps at a time. Find the number of ways in which he can cross the river?

Total steps

[6 unit steps] $^5C_4$

[4 unit, 1 double] $^5C_2$

[3 unit, 2 double]

[3 double]

Out of 7 boys and 4 girls, how many queues of 3 boys and 2 girls can be formed?

$\Rightarrow {}^{7}C_3 \times {}^{4}C_2 \times 5!$

$= 210 \times 120 = 25200$

Merge two sorted linked lists

```
if(head2 -> data < head1 -> data)
  & swap (head1, head2);

  Node * head = head1;
  while( head1 != NULL && head2 != NULL)
  {
      if(head1 -> data <= head2 -> data)
      {
          prev = head1
          head1 = head1 -> next
      }
      else
      {
          prev -> next = head2;
          prev = head2;
          temp = head2 -> next;
          head2 -> next = head1;
          head2 = temp;
      }
  }
  if( head2 == NULL)
  {   while(head2 != NULL){
      prev -> next = head2;
          prev = prev -> next;
          head2 = head2 -> next; }
```

return head;

2 pipes together fill bucket in 12 mins
First pipe 10 min faster than second pipe
Second pipe alone?

time taken by first pipe = t
second = $t+10$

$$\frac{1}{t} + \frac{1}{t+10} = \frac{1}{12}$$

$$\frac{2t+10}{t^2+10t} = \frac{1}{12}$$

$$24t + 120 = t^2 + 10t$$

$$t^2 - 14t - 120 = 0 \quad (t-20)(t+6) = 0$$

$$t = 20$$

B takes $(20+10) = 30$ mins

TSD (Apptitude)

km/hour ⟶ $\times 5/18$ to m/s
m/s ⟶ $\times 18/5$ to km/h

Average speed $= \frac{n}{\Sigma(1/s_i)}$

A person travels 10km 3 times at speed 4 km/h, 5 km/h, 6 km/h.

Avg speed:- $\frac{3}{\left(\frac{1}{4}+\frac{1}{5}+\frac{1}{6}\right)}$

→ Same direction := relative speed $(x-y)$ km/h
→ Opp direction := $x+y$ km/hr

$x-y$, $x+y$:- b and c the distance b/w them after 1 hour

Q. 25 km/hr → ← 4 km/hr

total time = 2 hour 54 min
total distance = ? (one side)

$d_1 = d_2$

$25 \times t = 4 \times (2.9 - t)$

Q. Walking at speed 5 km/hr ↓ 7 min late
if 1 km/hr faster ↑ 5 min early
Distance ??

case 1 time $= \frac{d}{5}$ case 2 time $= \frac{d}{6}$

difference b/w these time = (7 min − (−5 min))
= 12 mins.

$\frac{d}{5} - \frac{d}{6} = 0.2$ ↓ 0.2 hours

$d = 6$ //

Q. A ← 465 km → B

10 AM → 65 km/hr ← 11 am

35 km/hr ←

when do they (V) meet?

A ——— 400 ——— B

11 AM 11 am

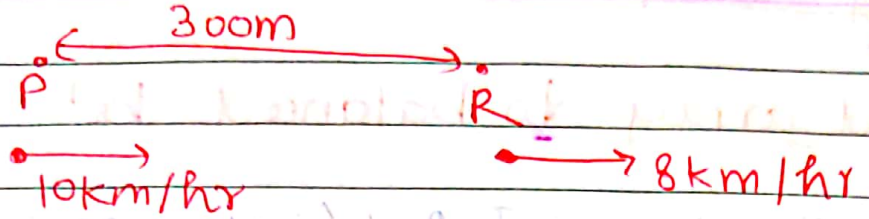
65 km/hr 35 km/h.

Relative speed = 35 + 65 = 100 km/hr

time = $\frac{400 \text{ km}}{100 \text{ km/hr}}$ = 4 hours

= 11 AM + 4 hours = 3 pm

Q.



How long will R run before being caught?

⇒ relative speed = 2km/hr.

time $= \frac{0.3}{2} = 0.15$ hours.

In 0.15 hours R travels

distance = $0.15 \times 8 = 1.2$ km.

OR

both run the same time.

time1 = time2.

$\Rightarrow \frac{x}{8} = \frac{0.3+x}{10}$

$x = 1.2$ //

$5x(x+7) = 6x(x-5)$

$\frac{75}{100} =$

$d = 6 \times (0.75 - \frac{5}{100})$

6×0.25

Binary Tree problems

Sorted array to balanced bst

```
struct TNode* soToBst(int arr[], int start,
                      int end)
{
  if(start>end)
      return NULL;
  int mid = (start+end)/2;
  struct TNode *root = newNode(arr[mid]);
  root->left = soToBst(arr, start, mid-1);
  root->right = soToBst(arr, mid+1, end);
  return root;
}
```

Root to leaf path sum

```
bool hasPathSum(Node *node, int sum)
{
  if(node==NULL)
    return(sum==0);
  else
  {
    bool ans;
    int dsum = sum - node->data;
    if(node->left)
      ans = ans || hasPathSum(node->left, dsum);
    if(node->right)
      ans = ans || hasPathSum(node->right, dsum);
    return ans;
  }
}
```

if(dsum==0 && node->left==NULL && node->right==NULL) return 1;

j+col[k] >=0 & < MAX && A[i+row[k]][j+col[k]]

Number of set bits

```
while(n)
{
   n = n & (n-1)
   count++
}
```

100 & 011 = 000

110 & 101 = 100 & 011 = 000 → 3

1110 & 1011 = 1010 & 1001 = 1000 & 0111 = 0000 → 3

1111 & 10000 = 0000 → 1

11 → 2

7 → 3

DFS → When find connected components (DFS also)

n = 3

① 111 & 110 = 110

② 101 & 10 = 0 → 2

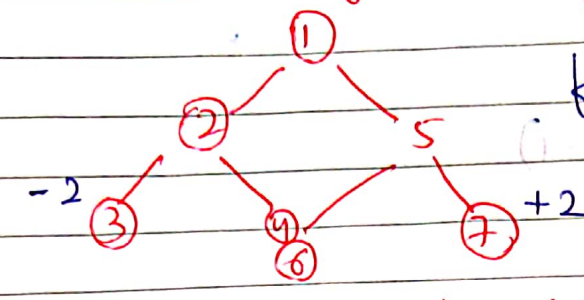
** memset(array, 0, sizeof(array))

Find num of islands (components)

```
int visited[MAX][MAX] = {0};
int find( int A[MAX][MAX], int N, int M)
{
    int count = 0;
    for( i=0; i<MAX; i++)
    {  for(j=0; j<MAX; j++)
       {  if (A[i][j] && !visited[i][j])
          {  DFS(A, i, j);
             count++ }
       }  }
    memset(visited, 0, sizeof(visited));
} return count;
int DFS( )
{  visited[r][c] = 1;
   static int row = {-1,-1,-1, 0, 0, 1, 1, 1};
              col = {-1, 0, 1,-1, 1,-1, 0, 1};
   for(int k=0; k<8; k++)
   {  if(isvalid)
      {  if( !visited[i+row[k]][j+col[k]])
            DFS(A, i+row[k], j+col[k]);
   }  }
```

Teacher's Signature.....

Vertical order of a binary tree



find horizontal distances

Output 3 2 1 4 6 5 7

```
void verticalOrder(Node *root)
{
    int min=0, int max=0
    find(root, &min, &max, 0);
    for(i=min; i<=max; i++)          i → level
    {
        print(root, i, 0);
    }
}
void find(Node *root, int *min, int *max, int hd)
{
    if(root==NULL) return;
    if(hd < *min)
        *min = hd
    else if(hd > *max) *max = hd;
    find(root->left, min, max, hd-1);
    find(root->right, min, max, hd+1);
}
void print(Node *root, int level, int hd)
{  if(root==NULL) return
    if(level == hd) print(node->data)
    print(root->left, level, hd-1);
    print(root->right, level, hd+1);
}
```

Teacher's Signature.....

Find next larger element (Using stack)

```
void printNGE (int arr[], int n)
{   int i = 0
    stack s;
    push(&s, a[0]);
    for(i = 1; i < n; i++)
    {
        next = a[i]
        if(isempty(&s) == false)
        {
            ele = pop(&s);
            while(ele < next)
            {
                printf("%d ---> %d", ele, next);
                if(isempy(&s)
                    break;
                ele = pop(&s);
            }
            if(ele > next)
                push(&s, ele);
        }
        push(&s, next);
}

while(isempty(&s) == false)
{
    ele = pop(&s)
    next = -1;
    pf("%d ---> %d", ele, next);
}
```

Add 2 numbers represented by linked lists

5 → 6 → 3

8 → 4 → 2

1 → 4 → 0 → 5

If same size following

```
node * addSameSize(Node * head1, Node * head2, int *carry)
{
    if(head1 == NULL)
        return NULL;
    Node * result = (Node *)malloc(sizeof(Node));

    result->next = addSameSize(head->next, head2->next, carry)
    sum = head1->data + head2->data + *carry
    carry = sum/10;
    sum = sum%10;
    result->data = sum;
    return result;
}
```

3 → 5 → 6 → 3 → 2

8 → 6 → 9 call addsame

then pass 3 → 5 (head1) and carry and add

and insert node at front of list. (result)

Teacher's Signature.....

0 → 9 L A → Z ↓ 10 → 15

Intersection and union of LL.

Use merge sort first

TODO : Merge Sort LL

BIT Manipulation

Find nth magic number

→ either a power of 5, or sum of powers of it.

5, 25, 130 (25+5) ....

7th number → 1 1 1 $5^2+5^1+5^0$ 5th → 1 0 1 $5^2+5^1+5^0$

Swap bits (1 0 0 0 1)

(0 1 1 1 0 0 1 0)

Get all even bits unsigned int even = x & 0xAAAAAAAA

odd = x & 0x55555555

even = even >> 1

odd = odd << 1

ans = even | odd.

Find the element that appears once.

(3 3 3 1 5 5 5 1 1 2) ... 32

```
for( i=0 ; i< INT_SIZE ; i++)
{   sum = 0
    l = 1 << i ;
    for( j=0 ; j< n ; j++)
        if( arr[j] & l)
            count++;
    if( sum % 3),
        result = result | X ;
}
return result;
```

Teacher's Signature.....

Find binary representation of a number.

```
for(i = 1<<31; i>0; i = i/2)
    (x & i)? printf(1) : print(0);
```

OR

```
bin(n)
{ if(n>1)
    bin(n/2)
  printf("%d", n%2);
```

Rotate bits in a number.

```
left rotation (n, d) by d
    return (n<<d) | (n>>(32-d));

right rotation(n, d)
    return (n>>d) | (n<<(32-d));
```

Count no. of bits to be flipped to convert A → B

do XOR then count set bits

Count triplets with sum smaller than a given element.

sort array & [see gfg x D]

Find $a^b \% m$

x^y in O(log n) time.

```
while (y > 0)
{
    if (y & 1)
        res = res * x;
    y = y >> 1;
    x = x * x;
}
```

Convert infix to postfix

```
for(i = 0 ; k = -1 ; exp[i] ; i++)
{
    if ( isOperand (exp[i]))
        push(stack, exp[i]);
    else if (exp[i] == ')')
    { while(!isEmpty(stack) && peek(stack) != '(')
        exp[++k] = pop(stack)
      pop(stack);
    }
    else
    { while(!isEmpty(S) && prec(exp[i]) <= precpeek(stack))
        exp[++k] = pop(stack)
      push(stack, exp[i]);
    }
}
while (!isEmpty(S))
  exp[++k] = pop(S);
exp[++k] = '\0'
printf("%s", exp);
```

elsif ('(') push

Evaluate postfix expression

```
evaluate ( exp char [] exp )
for(i=0; i< strlen(exp); i++)
{
    if( isdigit(exp[i])
        push(s, exp[i]-'0')
    else
        val1 = pop(S)
        val2 = pop(S)
        switch exp[i]
        {
            '+': push(&, val2 + val1);
        }
    printf(pop(S))
}
```

No of digits in a number

ceil(log10(n))

Permutation in lexicographically inc order.

first second

A G S B

find first so that a[first] < a[first+1]

find second, greater that a[first]

but the smallest

swap sort after a[first]

A S G B

A S B G //

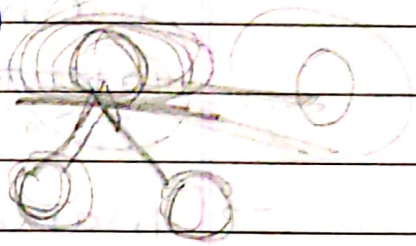
DFS → inorder, pre, post

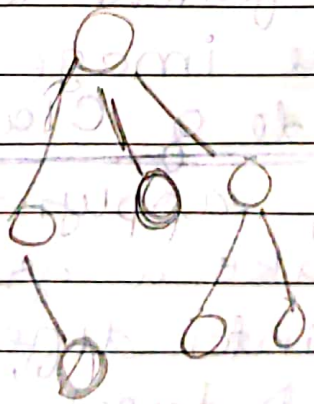
BFS → level order.

Printminlevel/Leaves sum

① Find the minimum level

```
int minLevel(Node *root, level)        → 1
{
  if(root == =NULL)
      return 0;
  if(!root→left && !root→right)
      return level;                    → int left, right = INT_MAX;
  if(root→left != NULL)
  left = minLevel(root→left, level+1)
  if(root→right) = NULL)
   right = minLevel(root→right, level+1)
  return min(left, right);
}
```





ceil(log2(3))