11 Week DSA Workshop by GeeksforGeeks

Instructor: Shashi Bhushan Kumar
https://www.linkedin.com/in/shashi-bhushan-coder/

==========

Week 1 - Day 1

==========

Time / Space Complexity

https://www.geeksforgeeks.org/analysis-of-algorithms-set-1-asymptotic-analysis/

https://www.geeksforgeeks.org/analysis-of-algorithms-set-2-asymptotic-analysis/

Arrays

https://www.geeksforgeeks.org/trapping-rain-water/

```
//Time - O(n*n) -- O(n)

int find_total_water(int arr[], int n){
  int total_water = 0; // ans
  for(int i=1;i<n-1;i++){ // ignoring the 1st (0th) and last(n-1th) index --- O(n)
    int left_max = find_left_max(arr,n,i); // O(n) -- O(1)
    int right_max = find_right_max(arr,n,i); // O(n) -- O(1)
```

```
    total_water = total_water + min(left_max, right_max) - arr[i];
  }
  return total_water;
}

int find_left_max(int arr[], int n, int i){
  int maximum = 0;
  for(int j=0;j<=i;j++){
    maximum = max(maximum,arr[i]);
  }
  return maximum;
}

int find_right_max(int arr[], int n, int i){
  int maximum = 0;
  for(int j=n-1;j>=i;j--){
    maximum = max(maximum,arr[j]);
  }
  return maximum;
}
```
==========

Basic Maths

Check for Prime No

n = 5, true
n = 24, false

```
// O(n)
bool is_prime(int n){
for(int i=2;i<=n;i++){
```

```
  if(n%i==0) return false;
}
return true;
}

// O(n/2) -- O(n)
bool is_prime(int n){
for(int i=2;i<=n/2;i++){
  if(n%i==0) return false;
}
return true;
}
// O(sqrt(n))
bool is_prime(int n){
for(int i=2;i<=sqrt(n);i++){
  if(n%i==0) return false;
}
return true;
}
```

==========

Print all the prime numbers from 1 to n

Seive of eratosthenes

https://www.geeksforgeeks.org/sieve-of-eratosthenes/

n = 100
2 3 5 7 11 13 17 .....

Method 1: (without using Seive)

```
// O(n * sqrt(n))
```

```
void printAllPrime_1(int n){
for(int i=2;i<=n;i++){ // O(n)
  if(is_prime(i)) print i; // O(sqrt(n))
}
}
```

Method 2:

```
// Time / Space
bool prime[n+1]; // array
   memset(prime, true, sizeof(prime)); // All these values are prime

   /*
   for(int i=0;i<n;i++){
   prime[i] = true;
 }
   */

   for (int p=2; p*p<=n; p++) // ??
   {
       // If prime[p] is not changed, then it is a prime
       if (prime[p] == true)
       {
           for (int i=2*p; i<=n; i += p) // multiples of p
               prime[i] = false;
       }
   }

   // Print all prime numbers
   for (int p=2; p<=n; p++) // O(n)
      if (prime[p]==true) // O(1)
         cout << p << " ";
```

==========

Printing All substrings of a given string

s = "abc"

n = 3,
Total No of substrings : (n*(n+1))/2 , 3*4/2 = 6

a
ab
abc
b
bc
c

s2 = "pqr"

i = 0 ('p') -- 1 to n-i (3 -0) == 3
p - 1
pq - 2
pqr - 3
q
qr
r
i = 2 ('r') -- 1 to n-i (3 -2) == 1

```
// Function to print all sub strings
void subString(string s, int n)
{
    for (int i = 0; i < n; i++)
        for (int len = 1; len <= n - i; len++){
          string subString = "";
          for(int j=i;j<i+len;j++){
            subString = subString + s[j];
```

```
        }
        print(subString);
      }
}
```

==========

Week 1 - Day 2

Arrays

https://www.geeksforgeeks.org/array-rotation/

```
// Method -1
// Time - O(n), Space : O(d)
void rotateArray(int arr[], int n, int d){
  int tmp[d]; // 1 2
  for(int i=0;i<d;i++){
    tmp[i] = arr[i];
  }
  for(int i=0;i<n-d;i++){ // 1 2 3 4 5 6 7 -- 3 4 5 6 7 6 7
    arr[i] = arr[i+d];
  }
  for(int i=0;i<d;i++){ //  3 4 5 6 7 6 7 --3 4 5 6 7  1 2
    arr[i+n-d] = tmp[i];
  }
}

Method 2:
void rotate(int arr[], int n){ // O(n)
  int tmp = arr[0];
  for(int i=0;i<n-1;i++){ // 1 2 3 4 5 --- 2 3 4 5 1
    arr[i] = arr[i+1];
  }
```

```
  arr[n-1] = tmp;
}
// Time - O(n*d) , Space - O(1)
void rotateArray(int arr[], int n, int d){
for(int i=0;i<d;i++){ // O(d)
  rotate(arr,n); // O(n)
}
}
```

method 3:

```
/*Function to reverse arr[] from index start to end*/
void reverseArray(int arr[], int start, int end)
{
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

/* Function to left rotate arr[] of size n by d */
void leftRotate(int arr[], int d, int n)
{
    if (d == 0)
        return;
    reverseArray(arr, 0, d - 1);
    reverseArray(arr, d, n - 1);
    reverseArray(arr, 0, n - 1);
}
```

==========

Searching

2 4 8 12 15 10 7 , x = 15, output: 4
x = 6, output: -1

Linear Search

```
// Time : O(n), Space : O(1)
int search(int arr[], int x){
  for(int i=0;i<n;i++){
    if(arr[i]==x) return i;
  }
  return -1;
}
```

Sorted Array

Binary Search - logn

```
// Time - O(logn), space: O(1)
int binary_search(int arr[], int n, int x){
  int l = 0;
  int r = n-1;
  while(l<=r){
    int mid = (l+r)/2;
    if(arr[mid]==x) return mid;
    if(arr[mid]<x) l = mid+1;
    else r = mid-1;
  }
  return -1;
}
```

==========

square root of a no (Amazon)

n = 25, 5
n = 27, 5
n = 35, 5

n= 10^18 , square_root(n) = 10^9 , O(sqrt(n)) = O(10^9)
O(logn) = O(64)
Method1:

```
// Time - O(sqrt(n)) - O(logn)
int square_root(int n){
  for(int i=1;i<=n;i++){
```

```
    if(i*i==n) return i; // n=25 , i=5
    if(i*i>n) return i-1; // n=27, i= 6
  }
}
```

==========

Game: Guess The No

https://www.geeksforgeeks.org/array-rotation/
https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/
Searching

https://www.geeksforgeeks.org/square-root-of-an-integer/
https://www.geeksforgeeks.org/find-first-and-last-positions-of-an-element-in-a-sorted-array/

==========

Week 2- Day -1

Binary Search:

https://www.geeksforgeeks.org/find-first-and-last-positions-of-an-element-in-a-sorted-array/
Count Freq of a No.
https://www.geeksforgeeks.org/find-row-number-binary-matrix-maximum-number-1s/
https://www.geeksforgeeks.org/search-an-element-in-a-sorted-and-pivoted-array/

Sorting :
https://www.geeksforgeeks.org/fractional-knapsack-problem/

https://www.geeksforgeeks.org/maximise-the-number-of-toys-that-can-be-purchased-with-amount-k/

==========

Row with maximum No of ones in a matrix

method 2 / 3:

```
// O(m*n) - O(m*logn)
int find_row_with_max_one(int mat[][], int m, int n){
  int ans = 0;
  int max_count_one =0;
  int count = 0;
  for(int i=0;i<m;i++){
    count = count_no_of_ones_on_ith_row(mat,i,n); // O(n) - O(logn)
    if(count > max_count_one){
      ans = i;
      max_count_one = count;
    }
  }
}
```

method 4: O(m+n)

5 5 , x= 5

2 3 5 5 6 7 7 8 9 10 15, x=5

2 3 3 4 5 5 5 5 5 5 7 9 , x = 5

```
int first_occ(int arr[], int n, int x){ // logn
  int l=0, r= n-1;
  while(l<=r){
    int mid = (l+r)/2;
    if(arr[mid]==x && (mid==0 || arr[mid-1]!=x)) return mid;
    if(arr[mid]>=x) r = mid-1;
    else l = mid+1;
  }
  return -1;
}

int last_occ(int arr[], int n, int x){ // logn
  int l=0, r= n-1;
  while(l<=r){
    int mid = (l+r)/2;
    if(arr[mid]==x && (mid==n-1 || arr[mid+1]!=x)) return mid;
    if(arr[mid]>x) r = mid-1;
    else l = mid+1;
  }
  return -1;
}
```

5. Count the freq of a No in a sorted Array - logn

2 3 3 4 5 5 5 5 5 5 7 9 , x = 5

first_occ = 4
last_occ = 10
count = last_occ - first_occ +1 = 10 -4 + 1= 7

Week 2- Day - 2

Hashing:

Hashing

C++

```
unordered_map<string, int> map;

map["Ramu"] = 30; // O(1)
map["Rohan"] = 40; // insert

search / get

return map["Ramu"]; // 30
```

Java

```
HashMap<String,Integer> map;
```

https://leetcode.com/problems/contains-duplicate/solution/

https://www.geeksforgeeks.org/given-an-array-a-and-a-number-x-check-for-pair-in-a-with-sum-as-x/

https://www.geeksforgeeks.org/find-winner-election-votes-represented-candidate-names/

Week 3 - Day 1

==========

Matrix

Boolean Matrix Question

https://www.geeksforgeeks.org/a-boolean-matrix-question/

Method 1 (Use two temporary arrays):

```
void modifyMatrix(bool mat[R][C])
{
   bool row[R];
   bool col[C];

   int i, j;

   /* Initialize all values of row[] as 0 */
   for (i = 0; i < R; i++)
   {
   row[i] = 0;
   }

   /* Initialize all values of col[] as 0 */
   for (i = 0; i < C; i++)
   {
   col[i] = 0;
   }

   // Store the rows and columns to be marked as
   // 1 in row[] and col[] arrays respectively
   for (i = 0; i < R; i++)
   {
      for (j = 0; j < C; j++)
      {
         if (mat[i][j] == 1)
         {
            row[i] = 1;
            col[j] = 1;
         }
      }
   }

   // Modify the input matrix mat[] using the
```

```
    // above constructed row[] and col[] arrays
    for (i = 0; i < R; i++)
    {
        for (j = 0; j < C; j++)
        {
            if ( row[i] == 1 || col[j] == 1 )
            {
                mat[i][j] = 1;
            }
        }
    }
}
```

//Time Complexity: O(M*N)
//Auxiliary Space: O(M + N)

Method 2 (A Space Optimized Version of Method 1):

```
void modifyMatrix(int mat[R][C])
{
    // variables to check if there are any 1
    // in first row and column
    bool row_flag = false;
    bool col_flag = false;

    // updating the first row and col if 1
    // is encountered
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            if (i == 0 && mat[i][j] == 1)
                row_flag = true;

            if (j == 0 && mat[i][j] == 1)
                col_flag = true;
```

```
            if (mat[i][j] == 1) {

                mat[0][j] = 1;
                mat[i][0] = 1;
            }
        }
    }

    // Modify the input matrix mat[] using the
    // first row and first column of Matrix mat
    for (int i = 1; i < R; i++) {
        for (int j = 1; j < C; j++) {

            if (mat[0][j] == 1 || mat[i][0] == 1) {
                mat[i][j] = 1;
            }
        }
    }

    // modify first row if there was any 1
    if (row_flag == true) {
        for (int i = 0; i < C; i++) {
            mat[0][i] = 1;
        }
    }

    // modify first col if there was any 1
    if (col_flag == true) {
        for (int i = 0; i < R; i++) {
            mat[i][0] = 1;
        }
    }
}
```

//Time Complexity: O(M*N)
//Auxiliary Space: O(1)

String

Q1. Check Whether two strings are anagram of each other

https://www.geeksforgeeks.org/check-whether-two-strings-are-anagram-of-each-other/

Method 1(Sorting):

```
bool areAnagram(string str1, string str2)
{
    // Get lengths of both strings
    int n1 = str1.length();
    int n2 = str2.length();

    // If length of both strings is not same, then they
    // cannot be anagram
    if (n1 != n2)
        return false;

    // Sort both the strings
    sort(str1.begin(), str1.end());
    sort(str2.begin(), str2.end());

    // Compare sorted strings
    for (int i = 0; i < n1; i++)
        if (str1[i] != str2[i])
            return false;

    return true;
}
```

//Time Complexity: O(NlogN)

Method 2(Count Characters):

```
bool areAnagram(char* str1, char* str2)
{
    // Create 2 count arrays and initialize all values as 0
    int count1[NO_OF_CHARS] = { 0 };
    int count2[NO_OF_CHARS] = { 0 };
    int i;

    // For each character in input strings, increment count
    // in the corresponding count array
    for (i = 0; str1[i] && str2[i]; i++) {
        count1[str1[i]]++;
        count2[str2[i]]++;
    }

    // If both strings are of different length. Removing
    // this condition will make the program fail for strings
    // like "aaca" and "aca"
    if (str1[i] || str2[i])
        return false;

    // Compare count arrays
    for (i = 0; i < NO_OF_CHARS; i++)
        if (count1[i] != count2[i])
            return false;

    return true;
}
```

Method 3 (count characters using one array):

```
bool areAnagram(char* str1, char* str2)
{
```

```
    // Create a count array and initialize all values as 0
    int count[NO_OF_CHARS] = { 0 };
    int i;

    // For each character in input strings, increment count
    // in the corresponding count array
    for (i = 0; str1[i] && str2[i]; i++) {
        count[str1[i]]++;
        count[str2[i]]--;
    }

    // If both strings are of different length. Removing
    // this condition will make the program fail for strings
    // like "aaca" and "aca"
    if (str1[i] || str2[i])
        return false;

    // See if there is any non-zero value in count array
    for (i = 0; i < NO_OF_CHARS; i++)
        if (count[i])
            return false;
    return true;
}
```

//Time Complexity: O(N)

Method 4 (Taking Sum):

```
bool isAnagram(string c, string d)
{
    if (c.size() != d.size())
        return false;
    int count = 0;

    // Take sum of all characters of first String
    for (int i = 0; i < c.size(); i++) {
        count += c[i];
    }
```

```
    // Subtract the Value of all the characters of second
    // String
    for (int i = 0; i < d.size(); i++) {
        count -= d[i];
    }

    // If Count = 0 then they are anagram
    // If count > 0 or count < 0 then they are not anagram
    return (count == 0);
}
```

//Time Complexity: O(N)
//Auxiliary Space: O(1)

Q2. Sum of Two Large Numbers

https://www.geeksforgeeks.org/sum-two-large-numbers/

Method 1:

```
string findSum(string str1, string str2)
{
    // Before proceeding further, make sure length
    // of str2 is larger.
    if (str1.length() > str2.length())
        swap(str1, str2);

    // Take an empty string for storing result
    string str = "";

    // Calculate length of both string
    int n1 = str1.length(), n2 = str2.length();

    // Reverse both of strings
```

```
    reverse(str1.begin(), str1.end());
    reverse(str2.begin(), str2.end());

    int carry = 0;
    for (int i=0; i<n1; i++)
    {
        // Do school mathematics, compute sum of
        // current digits and carry
        int sum = ((str1[i]-'0')+(str2[i]-'0')+carry);
        str.push_back(sum%10 + '0');

        // Calculate carry for next step
        carry = sum/10;
    }

    // Add remaining digits of larger number
    for (int i=n1; i<n2; i++)
    {
        int sum = ((str2[i]-'0')+carry);
        str.push_back(sum%10 + '0');
        carry = sum/10;
    }

    // Add remaining carry
    if (carry)
        str.push_back(carry+'0');

    // reverse resultant string
    reverse(str.begin(), str.end());

    return str;
}
```

Method 2(Optimized):

We can avoid the first two string reverse operations by traversing them from end.

```
string findSum(string str1, string str2)
{
    // Before proceeding further, make sure length
    // of str2 is larger.
    if (str1.length() > str2.length())
        swap(str1, str2);

    // Take an empty string for storing result
    string str = "";

    // Calculate length of both string
    int n1 = str1.length(), n2 = str2.length();
    int diff = n2 - n1;

    // Initially take carry zero
    int carry = 0;

    // Traverse from end of both strings
    for (int i=n1-1; i>=0; i--)
    {
        // Do school mathematics, compute sum of
        // current digits and carry
        int sum = ((str1[i]-'0') +
                   (str2[i+diff]-'0') +
                   carry);
        str.push_back(sum%10 + '0');
        carry = sum/10;
    }

    // Add remaining digits of str2[]
    for (int i=n2-n1-1; i>=0; i--)
    {
        int sum = ((str2[i]-'0')+carry);
        str.push_back(sum%10 + '0');
        carry = sum/10;
```

```
    }

    // Add remaining carry
    if (carry)
        str.push_back(carry+'0');

    // reverse resultant string
    reverse(str.begin(), str.end());

    return str;
}
```

Week 3 - Day 2

==========

Puzzles

Q1. Cut Silver bar

https://github.com/love1024/spoj-solution-with-explanation/blob/master/SILVER.cpp

The idea here is any number can be represented as power of 2 and we can create any number from these combinations.

For eg: 7 = 1+2+4.

```
int cuts(n)
{
        int cut = 0;
        while(n>1)
        {
                n = n/2;
```

```
            cut+=1;
        }
        return cut;
}
```

Q2. Minimum number of weights

https://prismoskills.appspot.com/lessons/Brain_Teasers/Minimum_no_of_weights.jsp

Q.3 Defective Ball Problem

https://www.geeksforgeeks.org/puzzle-8-balls-problem/

https://www.geeksforgeeks.org/weight-heavy-ball/

==========

Week 4 - Day 1

==========

Recursion

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function.

Q1. Sum of natural numbers using recursion.

https://www.geeksforgeeks.org/sum-of-natural-numbers-using-recursion/

```
int recurSum(int n)
{
    if (n <= 1)
        return n;
    return n + recurSum(n - 1);
}
```

Q2. Factorial of a number

https://www.geeksforgeeks.org/program-for-factorial-of-a-number/

```
int factorial(unsigned int n)
{
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}
```

Q3. Count digits in an integer

https://www.geeksforgeeks.org/program-count-digits-integer-3-different-methods/

```
int countDigit(long long n)
{
    if (n == 0)
        return 0;
    return 1 + countDigit(n / 10);
}
```

Q.4 Count number of a’s in the given string

https://www.geeksforgeeks.org/count-occurrences-of-a-substring-recursively/

```
int count_a(s, n)
{
        if(n==0) return 0;

        if(s[n-1]==’a’)
                return 1+count_a(s, n-1) ;

        return count_a(s, n-1);

}
```

Q.5 Generate all the binary strings of N bits

https://www.geeksforgeeks.org/generate-all-the-binary-strings-of-n-bits/

```
void generateAllBinaryStrings(int n, int arr[], int i)
{
    if (i == n) {
        printTheArray(arr, n);
        return;
    }

    // First assign "0" at ith position
    // and try for all other permutations
    // for remaining positions
    arr[i] = 0;
    generateAllBinaryStrings(n, arr, i + 1);

    // And then assign "1" at ith position
    // and try for all other permutations
    // for remaining positions
    arr[i] = 1;
    generateAllBinaryStrings(n, arr, i + 1);
}
```

Q.6 Count ways to reach the nth stair. (Staircase Problem)

https://www.geeksforgeeks.org/count-ways-reach-nth-stair/

```
int findStep(int n)
{
       if (n == 1 || n == 0)
           return 1;
       else if (n == 2)
           return 2;

       else
           return findStep(n - 2) + findStep(n - 1);
 }
```

Week 4 - Day 2

==========

Bitwise

https://www.geeksforgeeks.org/bitwise-algorithms/

Q.1 Decimal to Binary

https://www.geeksforgeeks.org/program-decimal-binary-conversion/

```
int decToBinary(int n)
{
    // Size of an integer is assumed to be 32 bits
    for (int i = 31; i >= 0; i--) {
```

```
        int k = n >> i;
        if (k & 1)
            cout << "1";
        else
            cout << "0";
    }
}
```

Q.2. Count set bits in an integer

https://www.geeksforgeeks.org/count-set-bits-in-an-integer/

```
int countSetBits(unsigned int n)
{
    int count = 0;
    while (n) {
        count += n & 1;
        n >>= 1;
    }
    return count;
}
```

Q.3 Check if number is odd or not

https://www.geeksforgeeks.org/check-if-a-number-is-odd-or-even-using-bitwise-operators/

```
bool isOdd(int n)
{
    // n&1 is 1, then odd, else even
    return ((n & 1));
}
```

Q.4 Check whether K-th bit is set or not

https://www.geeksforgeeks.org/check-whether-k-th-bit-set-not/

```
void isKthBitSet(int n, int k)
{
    if ((n >> (k - 1)) & 1)
        cout << "SET";
    else
        cout << "NOT SET";
}
```

Q.5 Program to add two binary strings

https://www.geeksforgeeks.org/program-to-add-two-binary-strings/

```
string addBinary(string a, string b)
{
    string result = ""; // Initialize result
    int s = 0;          // Initialize digit sum

    // Traverse both strings starting from last
    // characters
    int i = a.size() - 1, j = b.size() - 1;
    while (i >= 0 || j >= 0 || s == 1)
    {
        // Comput sum of last digits and carry
        s += ((i >= 0)? a[i] - '0': 0);
        s += ((j >= 0)? b[j] - '0': 0);

        // If current digit sum is 1 or 3, add 1 to result
        result = char(s % 2 + '0') + result;

        // Compute carry
        s /= 2;

        // Move to next digits
        i--; j--;
    }
    return result;
```

```
}
```

Linked List

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.

https://www.geeksforgeeks.org/data-structures/linked-list/

Q.1 Insertion

https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/

```
// A linked list node
class Node
{
    public:
    int data;
    Node *next;
};
```

-> Insertion at front.

```
void push(Node** head_ref, int new_data)
{
    /* 1. allocate node */
    Node* new_node = new Node();

    /* 2. put in the data */
    new_node->data = new_data;

    /* 3. Make next of new node as head */
    new_node->next = (*head_ref);
```

```
    /* 4. move the head to point to the new node */
    (*head_ref) = new_node;
}
```

-> Insertion at end

```
void append(Node** head_ref, int new_data)
{

    // 1. allocate node
    Node* new_node = new Node();

    // Used in step 5
    Node *last = *head_ref;

    // 2. Put in the data
    new_node->data = new_data;

    // 3. This new node is going to be
    // the last node, so make next of
    // it as NULL
    new_node->next = NULL;

    // 4. If the Linked List is empty,
    // then make the new node as head
    if (*head_ref == NULL)
    {
        *head_ref = new_node;
        return;
    }

    // 5. Else traverse till the last node
    while (last->next != NULL)
        last = last->next;
```

```
    // 6. Change the next of last node
    last->next = new_node;
    return;
}
```

Q.2 Detect loop in a linked list

https://www.geeksforgeeks.org/detect-loop-in-a-linked-list/

```
bool detectLoop(struct Node* h)
{
    unordered_set<Node*> s;
    while (h != NULL) {
        // If this node is already present
        // in hashmap it means there is a cycle
        // (Because you we encountering the
        // node for the second time).
        if (s.find(h) != s.end())
            return true;

        // If we are seeing the node for
        // the first time, insert it in hash
        s.insert(h);

        h = h->next;
    }

    return false;
}
```

//Time Complexity: O(N)
//Auxiliary Space: O(N)

Q.3 Add two numbers represented by linked lists

https://www.geeksforgeeks.org/sum-of-two-linked-lists/

```
void addList(Node* head1, Node* head2, Node** result)
{
    Node* cur;

    // first list is empty
    if (head1 == NULL) {
        *result = head2;
        return;
    }

    // second list is empty
    else if (head2 == NULL) {
        *result = head1;
        return;
    }

    int size1 = getSize(head1);
    int size2 = getSize(head2);

    int carry = 0;

    // Add same size lists
    if (size1 == size2)
        *result = addSameSize(head1, head2, &carry);

    else {
        int diff = abs(size1 - size2);

        // First list should always be larger than second
        // list. If not, swap pointers
        if (size1 < size2)
            swapPointer(&head1, &head2);

        // move diff. number of nodes in first list
        for (cur = head1; diff--; cur = cur->next)
```

```
            ;

        // get addition of same size lists
        *result = addSameSize(cur, head2, &carry);

        // get addition of remaining first list and carry
        addCarryToRemaining(head1, cur, &carry, result);
    }

    // if some carry is still there, add a new node to the
    // front of the result list. e.g. 999 and 87
    if (carry)
        push(result, carry);
}
```

//Time Complexity: O(M+N)

Q4. Reverse a linked list

https://www.geeksforgeeks.org/reverse-a-linked-list/

```
void reverse()
    {
        // Initialize current, previous and
        // next pointers
        Node* current = head;
        Node *prev = NULL, *next = NULL;

        while (current != NULL) {
            // Store next
            next = current->next;

            // Reverse current node's pointer
            current->next = prev;

            // Move pointers one position ahead.
```

```
        prev = current;
        current = next;
    }
    head = prev;
}
```