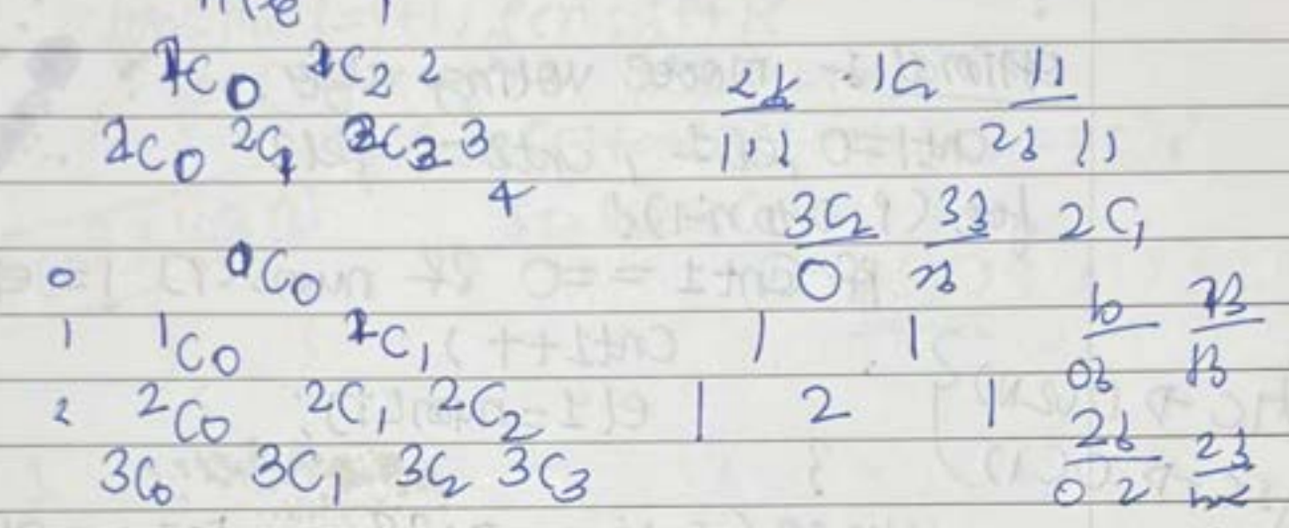
5 October 2023

Pascal's Triangle :-

$(a+b)^2 = a^2 + 2ab + b^2$



① given row and col selective val

R_1C

$R-1\ C_{C-1}$

{shortcut}

$10C_3 = \frac{10\times9\times8}{3\times2\times1}$

$\left\{\frac{10}{1}\times\frac{9}{2}\times\frac{8}{3}\right\}$

② Print particular row

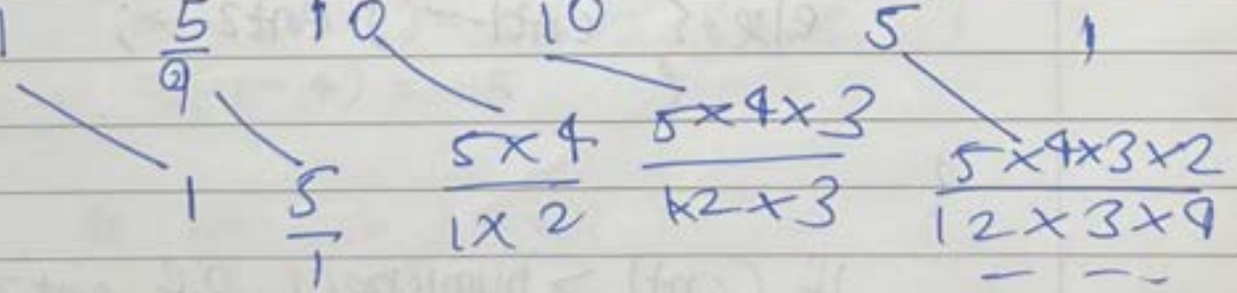
for (C=1 ; C<=n ;)

{ print(nCr(n-1, C-1));

}

TC ⊖(N×E) TC → O(1)

SC → O(1)



__/__/____

Majority element II [> n/3] times

Brute → O(N²)

Better :- <use map> O(N)

Optimal :- moore voting algo.

```
cnt1=0, el1, cnt2=0, el2
for (i 0 to n-1){
    if (cnt1==0 && nums[i] != el2){
        cnt1++;
        el1=nums[i];
    }
    else if (cnt2==0 && nums[i] != el1){
        cnt2++;
        el2=nums[i];
    }
    else if (nums[i]==el1)
        cnt1++;
    }
    else if (nums[i]==el2){
        cnt2++;
    }
    else { cnt1--; cnt2--;
    }
}
if (cnt1 > nums.size()/3 && cnt2 > nums.size()/3)
}
```

TC → O(2N)
SC → O(1)

3Sum 6 October 2023

Brute { triplet sum zero, must unique }

```
for(int i=0; i<n; i++){
    for(int j=i+1; j<n; j++){
        for(int k=j+1; k<n; k++){
            if(arr[i]+arr[j]+arr[k]==0){
                st.insert(v);
                v(3);
            }
        }
    }
}
```

T.C → O(n^3) log n

n^2

arr[i] + arr[j] + arr[k] = 0

arr[k] = -(arr[i] + arr[j]) { search in hash table }

arr[] = {-1, 0, 1, 2, -1, -4}

i j k

= -(-1+0) = 1

= -1+1 = 0

= (-1+2) = 1

= -(-1-1) = 0

= (-1-4) = 5

5
2
1
0

4 {-1, 0, -1}

do for each i and j.

```
for(i 0->n){        set<int> hashmap;
    for(j=i+1 to n){
        int third = -(arr[i] + arr[j]);
        if(hashmap.find(third) != end()){
            // triplet found vector(3)
        }
        else{
            hashmap.insert(third)
        }
    }
}
```

set

TC → O(N²) × log N

SC → O(N) + O(unique triplets).

Optimal :- {two pointer approach}

```
arr[] = {-2,-2,-2,-1,-1,-1,0,0,0,
          i  j                2,2,2,2}
                                    k
func(){   vector<vector<int>> ans;
sort(num.begin(), num.end());
for(int i=0; i<n; i++){
    if(i>0 and num[i]==num[i-1]){
        continue;
    }
    int j=i+1; k=n-1;
    while(j<k){
        int sum = num[i] + num[j] + num[k];
        if(sum<0){ j++;}
        else if(sum>0){
            k--;
        }
    }
```

__/__/__

```
else {
    vector<int> temp = { nums[i], nums[j], nums[k] };
    j++;                ans.push_back(temp);
    k--;
    while (j<k and nums[j] == nums[j-1]) j++;
    while (j<k and nums[k] == nums[k+1]) k--;
  }
}
}
return ans;
}
```

TC → N log N + O(N×N)
SC → O(no. of Triplets)

→ 4sum — 7 octomber

nums[i] + nums[j] + nums[k] + nums[l] = target

{ i ≠ j ≠ k ≠ l }

Brute →

```
Int()
for (i=0 → n) {
    for (j = i+1 → n) {
        for (k = j+1 → n) {
            for (l = k+1 → n) {
                long long sum = nums[i];
                sum += nums[j] + nums[k];
                sum += nums[l];
                if (sum == target) {
                    Run in set
                }
            }
        }
    }
}
```

TC → O(N⁴)
SC → O(no. of four)

Better

→

```
funct(){
    for(i=0 to n){
        for(j=i+1 to n){
            set<long long> hashmap;
            for(k=j+1 to n){
long long Fourth = target-(nums[i]+
                           nums[j]+
                           nums[k])
                if(hashmap.find(Fourth)!=hashmap.end)
                {
                    Push all four value
                    nums[i] nums[j] nums[k] & Fourth
                }
                hashmap.insert(nums[k]);
            }
        }
    }
```

TC → O(n3 x log n)

SC → O(n) + O(quads)

[1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5]
i & k l

TC → O(n² × n) ≃ O(n³)
SC → O(quads);

```
Funct(){
    vector<vector<int>>ans;
    int n = nums.size();
    sort(nums.begin(), nums.end());
    for(int i=0 ; i<n ; i++){
        if (i!=0 and nums[i]==nums[i-1]){
            continue;
        }
        for(int j=i+1; j<n ; j++){
            if(j != (i+1) and nums[j] ==
                nums[j-1]){
            continue;
        }
        int k = j+1;
        int l = n-1;
        while(k<l){
            long long sum = nums[i];
                sum t= nums[j];
                sum t= nums[k];
                sum t= nums[l];
            if(sum==target){
                Push all four in {ans}
                k++
                l--;
} k same} while } k++ } 4
} l same} while } l-- }
            else if (sum<target){ k++; }
            else { l--; }
```

__/__/____

8 October 23

Longest subarray with 0 sum :-

Brute :-
O(N^3)

Better / Optimal :-

```
int length(int arr[]){
    unordered_map<int, int> mapp; int sum=0, maxi;
    for(int i=0; i<n; i++){
        sum += arr[i];
        if(sum==0){
            maxi = max(maxi, i+1);
        }
        else if(mapp[sum] != 0){
            maxi = max(maxi, i - mapp[sum]);
        }
        else{
            mapp[sum] = i;
        }
    }
    return maxi;
}
```

TC → O(NlogN)
SC → O(N)

__/__/__

→ Count subarrays with XOR as K

→ Brute

arr[] = {4, 2, 2, 6, 4} K = 6

{4, 2}
{6}
{2, 2, 6}
{4, 2, 2, 6, 4}

```
int func(){
  for(int i=0; i<n; i++){
    for(int j=i; j<n; j++)  int xore=0;
      for(int k=i; k<=j; k++){
        xore ^= arr[k];
      }
      if(xore == k) cnt++;
    }
TC → O(N3) }
SC → O(1)  return cnt;
```

Better

```
for(i→n){
  int xore=0;
    for(j=i ton){
      xore = xore ^ arr[j];
      if(xore==k) cnt++;
    }  }
```

TC → O(N²)
SC → O(1)

Optional :-

```
int funct(){
    int xore=0;
    unordered_map<int,int>mapp;
    mapp[xore]++;
    int cnt=0;
    for(int i=0 ; i<n ; i++){
        xore^=a[i];
        if ( int x = xore ^ k;
            if (mapp[x]){
                cnt+=mapp[x];
            }
            mapp[xore]++;
            mapp[x]++;
        }
    return cnt;
}
```

→ merge overlapping Intervals :-

(1,3), (2,4), (2,6) (8,9), (8,10), (9,11), (15,18)

(1,6) (8,11) (16,17)

(15,17)

```
vector<vector<int>> overlapping(){
    int n = arr.size();
    sort(arr.begin(), arr.end());
    vector<vector<int>> ans;
```

```
for(int i=0; i<n; i++){
    int start = arr[i][0];
    int end = arr[i][1];
    if(!ans.empty() and end <= ans.back()[1])
    { continue;
    }
    else{
      for(int j=i+1; j<n; j++){
          if(arr[j][0] <= end){
              end = max(end, arr[j][1]);
          }
          else{
              break;
          }
      }
      ans.push_back({start,end});
    }
    return ans;
}
```

TC → N log N + O(N2) (touching each element twice)
SC → O(N)

i++ → {already taken} i, j++ j j(break) i

{(1,3), (2,4), (2,6), (8,9), (8,10), (9,10), (15,18), (16,17);

① i ② j j j (break) j break

{touched 2}

__/__/____

Optimal

```
void function(){
    int n = arr.size();
    vector<vector<int>> answer;
    sort(arr.begin(), arr.end());
    for (int i=0; i<n ; i++) {
        if (!answer.empty() and start <=
                                answer.back()[1]){
            answer.back()[1] = max(end,
                                answer.back()[1]
        }
        else {
            answer.push_back({start, end});
        }
    }
}
```

sort

TC → O(nlogn) + n

SC → O(n).

9 October 2023

→ merge 2 sorted arrays without extra space :-

Brute make 3rd array put element.

Better a = 1, 3, 2, 0 b = 7, 5, 2, 6, 8, 9

↑ ↑ ↑ ↑ ↑ ↑ ↑ break

Break

{then sort a and b}

__/__/____

n m

TC → min(a.size(), b.size()) + O(n log n)
SC → O(1) + O(m log m)

```
void func(){
    int left=n-1;
    int right=0;
    while(left>=0 && right<n){
      if(a[left] >= a[right]){
          swap(a[left], a[right]);
          left--;
          right++;
      }
      else{
          break;
      }
    }
    sort(a)
    sort(b)
}
```

→ gap method |shell sort 9/2 = 4.5] = 5

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

① left left left left ⑤ right right

right > (n+m) gap = ⌈gap/2⌉ (ceil)

__/__/___

```
void meege(){
    int len = m+n
    int gap = (len/2) + len%2;
    while(gap>0){
        int left=0;
        int right = left+gap;
        while(right < len){
            // arr1 and arr2
            if (left<n and right>=n){
                swap(arr1[left], arr2[right]);
            }
            // arr2 and arr2
            else if (left >=n){
                swap(arr2[left], arr2[right]);
            }
            else{
                swap(arr1[left], arr[right]);
            }
            if(gap ==1){ break;}
            gap *= gap/2];
        }
    }
```

TC → O(n+m) * O(log(m+m))
SC → O(1).

__/__/__

→ {Reapting & missing} 10 October 2023

{5 1 2 3 4 1}
return {1, 6}

→ Brute :- used hashmap.

→ Optimal → (maths)

Sarr - Sn = x - y {sum of}
Sarr² - Sn² = x² - y² {sum of squares}
□ = (x - y)(x + y)
□ = Sarr - Sn

TC → O(N)
SC → O(1)

Optimal
bit manipulation

11 October 23

→ {Count Inversion}
arr[] = [5, 3, 2, 4, 1] i < j & a[i] > a[j]

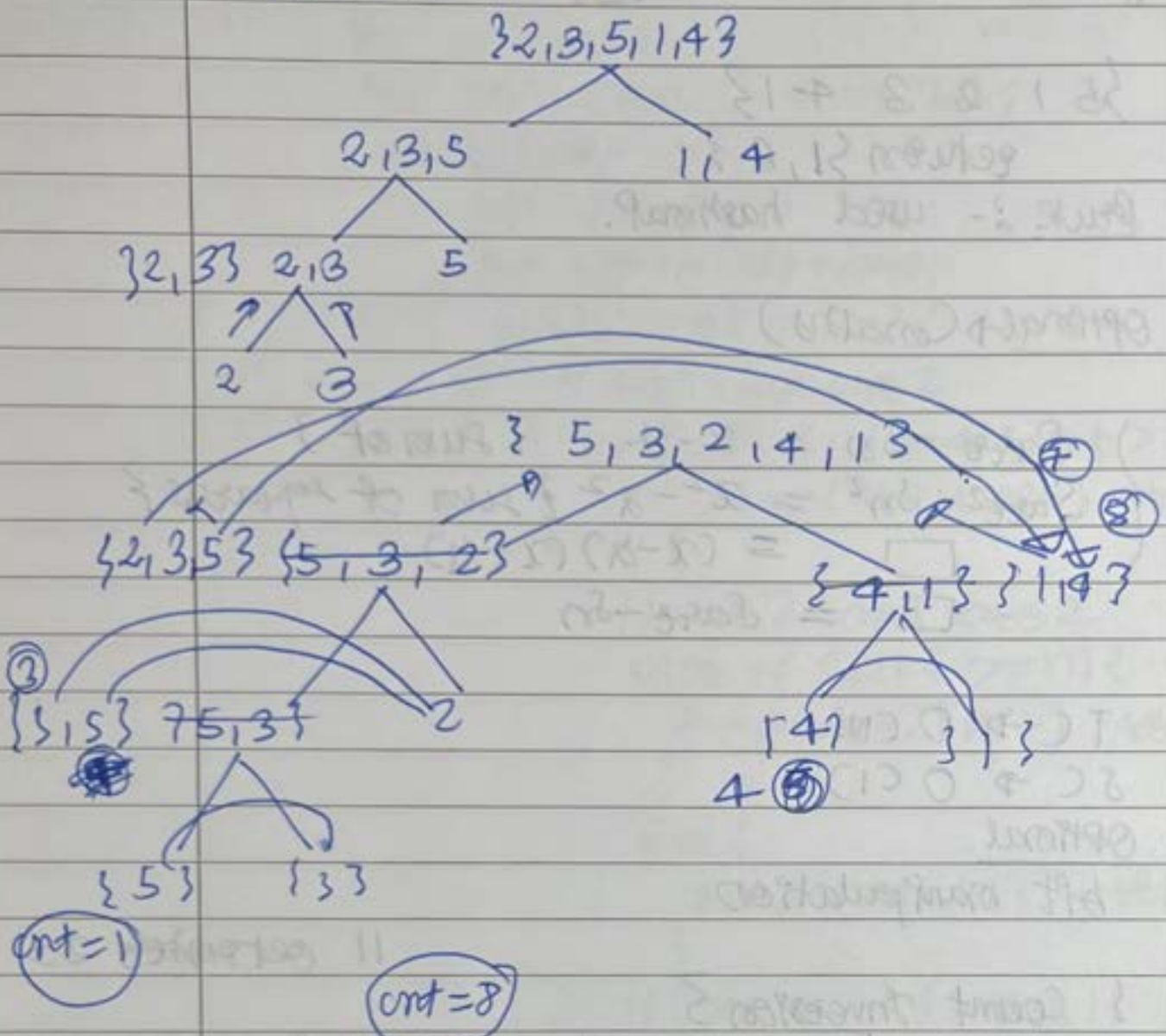
→ Brute

```
for (int i=0; i<n-1; i++){
    for (int j=i; j<n; j++){
        if (a[i] > a[j]) cnt++;
    }
}
return cnt
```

TC → O(N²)
SC → O(1).

__/__/____

merge sort



TC → O(NlogN)

SC → O(N)

__/__/____

{Reverse Pair} 11 October 23

arr[] = {40, 25, 19, 12, 9, 6, 2} → {15 Pairs}

i < j and a[i] > 2 × arr[j]

Brute:-

```
int cnt=0;
for (int i=0 to n-1){
    for (int j=i+1 to n){
        if (a[i] > 2*a[j]) cnt++;
    }
}
return cnt;
```

TC → O(N²)

SC → O(1)

TC → 2N log(2N)

SC → O(N)

40, 25, 19, 12, 9, 6, 2 compare (cnt = 15)

~~40, 25, 19, 12~~ {12, 19, 25, 40} ~~12, 6, 2~~ {2, 6, 9}

~~40, 25~~ {25, 40} ~~19, 12~~ {12, 19} {6, 9} ~~9, 6~~ {2}

40 25 19 12 {6} {9}

9 > 6×2 ✓

6 > 2×2 cnt = 4

9 > 2×2 cnt = 5

[40) [25)

40 > 25×2 ✗ {cnt = 0}

19 > 12×2 ✗ {cnt = 0}

{25, 40} {12, 19}

25 > 12×2 cnt = 1 (pointer at)

25 > 19×2 ✗

40 obviously breaks cnt += (i - 0 + 1)

i - 0 + 1 {cnt = 3}

__/__/____

} maximum Product Subarray } 12 October 23

arr[] = { 2, 3, -2, 4 } {ans = 6}

→ Brute

```
for(i to n)
    for(j=i to n){
        for(k=i k<j ){
            ans *= arr[k];
        }
        ans1 = max(ans1, ans);
    }
return ans1;
```

TC → O(N3)

SC → O(1)

→

```
for(i to n)
    int ans = 1;
    for(j=i to n){
        ans *= arr[i];
    }
    ans1 = max(ans1, ans);
}
return ans1;
```

TC → O(N2)

SC → O(1)

① all +ve
② even negative
③ odd negative
④ if no zeros

{-2, 3, 4, -1, 0, -2, 3, 1, 9, 0, 4, 6, -1, 4}

{2, 3, 4, -1}

```
ans = INT_MIN, pref = 1                    0 → n-1
        pref *= arr[i]                     1 → n-2
        suff *= arr[n-i-1];                2 → n-3

        ans = max(ans, max(pref, suff));
    }

For(int i=0 to n){
    if(pref==0) pref=1;
    if(suff==0) suff=1;
    pref = pref * arr[i];
    suff = suff * arr[n-i-1];
    maxi = max(maxi, max(pref, suff));
}
return maxi;
```

TC → O(N)
SC → O(1)