two pointer

বিশেষ করে codeforces এর অনেক প্রবলেম এর ট্যাগে দেখা যায় "**two pointer**" ট্যাগ করা আছে । স্বাভাবিক ভাবেই যেহেতু পয়েন্টার কথাটা আছে নামের সাথে আমি অনেকটা সময় ধরে ভাবতাম এই প্রবলেমগুলা হয়তো পয়েন্টার ব্যাবহার করে করা হয় । অনেকটা ছয় অন্ধের হাতি দেখা গল্পের মত । পরে কোডফরসেস এর একটা কমেন্ট এ একজন এর এক্সপ্লেনেশন দেখি । এরপর একদিন বুয়েট ওনিয়ন টিমের সাকিব ভাইয়াকেও জিজ্ঞাসা করছিলাম এই জিনিসটা কি । ভাইয়া একটা প্রবলেম দিয়ে এর সলুশ্যন বলছিলেন কিভাবে হচ্ছে , এই ট্যাকনিক টাই two pointer । এইখানে বলে রাখা ভাল অনেক এর এই ট্যাকনিকে স্লাইডিং উইন্ডো (যেহেতু একটা বাউন্ডারির মধ্যে কাজ করতে হয় এবং বাউন্ডারিটা একটা রেঞ্জ এর মধ্যে উত্তর দেয় তাই) বলা হয় , দুইটা আসলে একই জিনিস । two pointer সম্পর্কে আরও কিছু বলার আগে আমরা একটা প্রবলেম দেখি ।

আমাদের বলা হল আমাদের কাছে দুইটা সর্টেট array আছে , আমাদের বলতে হবে এই দুইটা array থেকে একটা একটা ভ্যালু নিয়ে আমরা কতভাবে একটা নাম্বার M বানাতে পারি যাদের কোন মধ্যে কোন ডুপলিকেট ভ্যালু নেই ।

যদি আমরা একটু Naive Method দেখি -

```
const int NX = 1e5 + 10 ; // limit of the array
int A[NX] , B[NX] ; // two array , both are ascendingly sorted
int M ; // value that we need to match
int N ; // Size of the Array


void solve()
{
     int ans = 0 , i , j ;
     for ( i = 0 ; i < N ; i++ )
     {
          for ( j = 0 ; j < N ; j++ ) if( A[i] + B[j] == M ) ans++;
     }
  return ans ;
}
```

view rawone.cpp hosted with ❤ by **GitHub**

এইখানে আমরা inner এবং outer for loop এর দুইটা relation এর মাধ্যমে খুব সহজে Ans বের করে দিতে পারি ।

আমরা এইখানে N পর্যন্ত দুইটা লুপ চাল্লাচ্ছি । তাই আমাদের টোটাল রানটাইম হয়ে যাচ্ছে O(N ^ 2) । আমরা যদি একটু Modification করি আমরা এইটা কমিয়ে O(N) এ নিয়ে আসতে পারি । আমরা দুইটা পয়েন্ট নেই । Say low and high । low পয়েন্ট করতেছে আমাদের A array এরটার starting point কে এবং high পয়েন্ট করতেছে আমাদের B array এর ending পজিশনটাকে ।

Observation number one ::

আমরা যদি দেখি A[low] + B[high] > M তাহলে আমরা একটা জিনিস সিউর ভাবে বলতে পারব । আমাদের অবশ্যই B এর ভ্যালু কমাতে হবে । কারণ যেহেতু A[low] হচ্ছে A এর সবথেকে ছোট ভ্যালু সুতরাং তাকে আর কমিয়ে কখনই B[high] এর সাথে add করে M বানানো যাবে না ।

Observation number two :::

আমরা যদি দেখি A[low] + B[high] < M তাহলে আমাদের অবশ্যই low এর ভ্যালু বাড়াতে হবে । কারণ high হচ্ছে B এর সবথেকে বড় ভ্যালু এর থেকে বড় ভ্যালু নাই । high থেকে আমরা কোন ভ্যালু বাড়াতে পারছি না । তাই অবশ্যই আমাদের এখন low থেকে বাড়াতে হবে ।

এইখানে একটা খটকা লাগতে পারে Observation one এ যেহেতু A[] array তে left to right যাওয়া হচ্ছে current low ভ্যালু মানে low যাকে right now point করছে A[] array তে তার থেকেও তো কম ভ্যালু আমার array তে থাকতে পারে । আমরা নিচের কোডটা একটু ভাল মত খেয়াল করলেই দেখতে পাব যে যদি থেকে থাকে এবং তার সাথে যদি high B[] array এর এড যদি M এর সমান হয় তাহলে তা আগেই Ans এর সাথে এড হয়ে আসবে । একদমই একই কাজ হচ্ছে B[] array তেও । এইখানে আমরা condition দিয়ে দুইটা পয়েন্টকে নিয়ন্ত্রণ করছি ।

```
const int NX = 1e5 + 10 ; // limit of the array



int A[NX] , B[NX]  ; // given array , ascending order sorted
int N ; // limit of the array
int M ; // We need to make M by using this two array



void solve()
{
    int ans = 0 , low = 0 , high = N - 1 ;
    while( low < N )
    {
          while( A[low] + B[high] > M && high > 0 ) high--;
          if( A[low] + B[high] == M ) ans++;
          low++;
    }
   return ans ;
}
```

view rawtwo.cpp hosted with ♥ by **GitHub**

এইখানে আমরা low and high দুইটা পয়েন্ট merge করে আগাচ্ছি । এই ধরনের ট্যানিক এ প্রবলেম সল্ভ করাই হচ্ছে two pointer method .

এখন আমরা আরেকটা প্রবলেম এর মাধ্যমে two pointer এর মাধ্যমে কিভাবে প্রবলেম সল্ভ করা হয় এইটা দেখব ।
problem এইখানে আমাকে N টা নাম্বার দেওয়া থাকবে এবং একটা ভ্যালু দেওয়া থাকবে say S । আমাকে minimum length এর consecutive sub sequence বের করতে হয়ে যাতে এই consecutive sub sequence এর sum, S এর থেকে বড় বা সমান হয় ।

এই প্রবলেম অনেক এপ্ররোচে আমাদের করা যাবে মনে হইতে পারে । কিন্ত o(n) runtime আনা ব্যাতিত্ব এই প্রবলেম সল্ভ হবে না । o(n) runtime আমরা two pointer ট্যাকনিক এর মাধ্যমে আনতে পারি ।

একই ভাবে আগের প্রবলেম এর মত এইখানে আমরা দুইটা পয়েন্ট নিব । low , high । প্রাথমিক ভাবে দুইটা পয়েন্ট এই given number array এর starting point indicate করে । যতক্ষণ পর্যন্ত না low থেকে high এর

sum , M এর ভ্যালু ক্রস না করে আমরা high এর ভ্যালু বাড়াইয়া যাব । যখনই ক্রস করবে বা সমান হবে তখনই আমরা current length check করব (high - low + 1) যদি minimum ans update করা যায় তাহলে update করব । যখনই sum এর ভ্যালু M এর থেকে বড় হয়ে যাবে তখন আমরা low এর ভ্যালু বাড়ানো সাথে সাথে sum এর ভ্যালু ও adjust করতে থাকব ।

কোডটা দেখলে ব্যাপারটা ক্লিয়ার হয়ে যাবে

```
 //BISMILLAHIRRAHMANIRRAHIM
/*
 manus tar shopner soman boro
 Author :: Shakil Ahmed
.............AUST_CSE27.........
 prob   ::
 Type   ::
 verdict::
 */


#include <bits/stdc++.h>
#define pb push_back
#define mp make_pair
#define pi acos(-1.0)
#define ff first
#define ss second
#define re return
#define QI queue<int>
#define SI stack<int>
#define SZ(x) ((int) (x).size())
#define all(x) (x).begin(), (x).end()
#define sqr(x) ((x) * (x))
#define memo(a,b) memset((a),(b),sizeof(a))
#define G() getchar()
#define MAX3(a,b,c) max(a,max(b,c))


double const EPS=3e-8;
using namespace std;


#define FI freopen ("input.txt", "r", stdin)
#define FO freopen ("output.txt", "w", stdout)
```

```
typedef long long Long;
typedef long long int64;
typedef unsigned long long ull;
typedef vector<int> vi ;
typedef set<int> si;
typedef vector<Long>vl;
typedef pair<int,int>pii;
typedef pair<string,int>psi;
typedef pair<Long,Long>pll;
typedef pair<double,double>pdd;
typedef vector<pii> vpii;


// For loop


#define forab(i, a, b)       for (__typeof (b) i = (a) ; i <= b ; ++i)
#define rep(i, n)            forab (i, 0, (n) - 1)
#define For(i, n)            forab (i, 1, n)
#define rofba(i, a, b)       for (__typeof (b)i = (b) ; i >= a ; --i)
#define per(i, n)            rofba (i, 0, (n) - 1)
#define rof(i, n)            rofba (i, 1, n)
#define forstl(i, s)  for (__typeof ((s).end ()) i = (s).begin (); i != (s).end (); ++i)


template< class T > T gcd(T a, T b) { return (b != 0 ? gcd<T>(b, a%b) : a); }
template< class T > T lcm(T a, T b) { return (a / gcd<T>(a, b) * b); }


//Fast Reader
template<class T>inline bool read(T &x){int c=getchar();int
sgn=1;while(~c&&c<'0'||c>'9'){if(c=='-')sgn=-
1;c=getchar();}for(x=0;~c&&'0'<=c&&c<='9';c=getchar())x=x*10+c-'0'; x*=sgn; return ~c;}


//int dx[]={1,0,-1,0};int dy[]={0,1,0,-1}; //4 Direction
//int dx[]={1,1,0,-1,-1,-1,0,1};int dy[]={0,1,1,1,0,-1,-1,-1};//8 direction
//int dx[]={2,1,-1,-2,-2,-1,1,2};int dy[]={1,2,2,1,-1,-2,-2,-1};//Knight Direction
//int dx[]={2,1,-1,-2,-1,1};int dy[]={0,1,1,0,-1,-1}; //Hexagonal Direction
```

```
/* ************************************  My code start here
**************************************** */
typedef long long ll ;
const int MX = (int) 1e6+100;
ll inp[MX] , S;
int n ;
int main()
{
   // ios_base::sync_with_stdio(0); cin.tie(0);
    while(scanf("%d %lld",&n,&S)==2)
    {
        int low = 0 , high = 0 , ans = n+1 ;
        int i ;
        for ( i = 0 ; i < n ; i++ ) read(inp[i]);
        ll sum = inp[0];
        // Algorithm
        /*



        */
        while( high < n )
        {
            if( sum >= S )
            {
                int temp = high - low + 1 ;
                if( temp < ans ) ans = temp ;
            }
            if( sum >= S && low < high )
            {
                sum -= inp[low];
                low++;
            }
            else if( sum < S )
            {
                high++;
                if( high < n )
                sum += inp[high];
            }
        }
        if( ans == n+1 ) ans = 0 ; // value possible na
        printf("%d\n",ans);
    }
```

```
        return 0;
    }
```

view rawnine.cpp hosted with ❤ by **GitHub**

এখন যদি আমরা এই কোডটার রান টাইম দেখি । কোডটা high = 0 থেকে high < len পর্যন্ত চলছে । মানে O(n) এ ।

two pointer এর আরো একটা problem হচ্ছে এইটা । এইখানে আমাদের given array দেওয়া হ্যনি । আমাদের given formula দিয়ে input ready করে নিতে হবে । এইটা বাদে almost same প্রবলেম হিসাবে মিলে যাচ্ছে আগের প্রবলেমটার সাথে ।

two pointer এর আরো প্রবলেম আইডি এইখানে কমেন্ট জানালে আমি এড করে দিতে পারব । বা কোন ট্যাকনিক কমেন্ট এ দিলে সবাই জানতে পারবে ।