## first section for binary search():

1. ***binary search in integer &&type of mid*** 🡪***page();***
2. ***binary search in double&&type of mid 🡪page();***

***3) pointer firsr style***🡪***page();***

***4) 4-two pointer second style(window is range with start and end)***🡪***page();***

1. Fixed size window of length K In this windows, we have specific range and searching for  
   a range with specific property. Easy to handle
2. Variable size window In this windows, the window can be of any size. More  
   tricky

***5) //***

هتقدر تلعب بالشروط وتحط شروط براحتك

***Section for binary search in intage***

// return after you search on it if you sersh on the bager than last element return -1

//if you sersh on the first element return index after it

int arr[10000];

int bain\_2(int size, int q) {

int end = size - 1, index\_2 = -1,st=0;

while (st <= end) {

int mid = (st + end) / 2;

if (arr[mid] > q) {

index\_2 = mid;

end = mid - 1;

}

else {

st = mid + 1;

}

}

return index\_2;

}

//return before you search on it if you sersh on the first element return -1

//if you sersh on the element bager than all element return last element

int arr[10000];

int bain( int size, int q) {

int end = size - 1, index = -1,st=0;

while (st <= end) {

int mid = (st + end) / 2;

if (arr[mid] < q) {

index = mid;

st = mid + 1;

}

else {

end = mid - 1;

}

}

return index;

}

بترجع العنصر الى بتبحث عنة ولوفى تكرار هترجع اول تكرار(من الشمال) ولو اكبر من كل العناصر الى فى الارى هيرجع -1 ولو اصغرمن كل العناصر هيرجع اكبر واحد

int arr[10000];

int bain\_2(int size, int q) {

int end = size - 1, index\_2 = -1,st=0;

while (st <= end) {int mid = (st + end) / 2;

if (arr[mid] >= q) {

index\_2 = mid;

end = mid - 1;}

else {st = mid + 1;}}return index\_2;}

هيرجع العنصر الى بتبحث عنة ولو فى تكرار هيرجع اخر تكرار لو الرقم الى بتبحث اكبر من كل الارقام هيرجع اخر رقم لو اصغر من كله هيرجع سالب واحد

int bain( int size, int q) {

int end = size - 1, index = -1,st=0;

while (st <= end) {

int mid = (st + end) / 2;

if (arr[mid] <= q) {

index = mid;

st = mid + 1;}else {end = mid - 1;}}return index;}

important ex

bool ok(int l) {

l--;

bool flag = 0;

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

if (valid(i+l , j+l ) && (gr[i + l][j + l] - gr[i + l][j - 1] - gr[i - 1][j + l] + gr[i - 1][j - 1]) <= l\_b)

return 1;

}

}

return 0;

}

int solve(int end=n,int st=1) {

ans = 0;

while (st <= end) {

mid = (st + end) / 2;

if (ok(mid)) {

ans = mid;

st = mid + 1;

}

else end = mid -1;

}

return ans;

}

بص سعات بيحصل infinity loop

ف لما يحص هتمسك قمتين(مكانين) ورى بعض 1و2

وتجرب عليهم الصيغة دى وتشوف انهى تنفع

1-Mid=start+(end-start)/2=(srart+end)/2🡪the first imprtant

2- Mid=start+(end-start+1)/2;

***2) Section for binary search in double***

int f(int n) {return 0;}

double BS\_double(double st, double ed, double value) {

while ( fabs(ed-st)>EPS)

{

double mid = st+(ed - st) / 2;

if (function(mid) < value)st = mid;

else ed = mid;

}return st;}

***or //***

bool can(int m) {return 0;}

double BS\_double(double st, double ed, double value) {

forr (i,100) // loop as you want

{double mid = st+(ed - st) / 2;

if (function(mid) < value)st = mid;

else ed = mid;}

if (can(end))return st; return -1;}

***Section for two pointer first style***

Ex: given a sorted array have N inager ,you need find pair(I,j) have sum is equal x;

***solve***

1)nested loop and compare the sum :O(N^2);

2)binary searsh(BS) for each v search for x-v:O(nlon(n));

3)two pointer :O(N);

For trace(for num 3);

Arr={2 ,4,5,7,8,20},x=11;

P1=0,p2=5, y=2+20=22>11;

P1=0,p2=4, y=2+8=10<11;

P1=1,p2=4, y=4+4=12>11;

P1=1,p2=3. Y=4+7=11==11;found;

Code:

int arr[1000],n;

bool fun(int sum) {

int l = 0, r = n-1;

while (l < r) {

if (arr[l] + arr[r] == sum)return 1;

else if (arr[l] + arr[r] > sum)r--;

else l--;

}

return 0;

}

***Section for two pointer fsecond style***

***1)*** Recall: Fixed size sliding window

EX:

Given an array of N values, find M  
consecutive values that has the max sum?  
◼ A brute force to compute that is just O(NM)  
by starting from every index and compute M  
values. Matter of 2 nested loops  
◼ Observation: What is the relation between the  
first M values and 2nd M values?

Let A = {1, 2, 3, 4, 5, 6, -3}, M = 4  
◼ 1st M values = **1**+2+3+4 = 10  
◼ 2nd M values = 2+3+4+**5** = 10**-1+5** = 14  
◼ 3rd M values = 3+4+5+6 = 14-2+6 = 18  
◼ 4th M values = 4+5+6-3 = 18-3-3 = 12  
◼ So answer is max(14, 18, 12) = 18

We create a **window** of **fixed** size M  
◼ cur window = last window - its first item + new mth item  
◼ Window start = pointer 1  
◼ Window end = pointer 2  
◼ P2 = P1+K-1

***Code:***

int arr[1000], n;

ll fun(int m) {

ll cur = 0,ans=0,p1=0;

for (int i = 0; i < n; i++) {

if (i+1 > m) {

ans = max(ans, cur);

cur = ans + arr[i] - arr[p1];

p1++;

}

else cur += arr[i];

}

return ans;

}

2)Variable size sliding window

EX:

Find a range with property  
◼ Given an array having N **positive** integers, find the  
contiguous subarray having sum as **max** as possible, **but**<= M(value)

***solve***  
◼ Let p1 = p2 = 0  
◼ Keep moving p2 as much as the window is **ok**◼ Once window is **!ok** = stop p2  
◼ keep moving p1 as long as window is !**ok**◼ Once window is **ok** = stop p1 and back to p2 again  
◼ For any **ok** window (here sum <= M), do your evaluations  
◼ Remember this strategy :)

***as a trace***

Let A = {2, 4, 3, 9, **6, 3, 1**, 5}, M = 10  
◼ P1 = 0, P2 = 0, Y = 2 = 2 <= 10. P2++  
◼ P1 = 0, P2 = 1, Y = 2+4 = 6 <= 10. P2++  
◼ P1 = 0, P2 = 2, Y = 2+4+3 = 9 <= 10. P2++  
◼ P1 = 0, P2 = 3, Y = 2+4+3+9 = 18 > 10. P1++  
◼ P1 = 1, P2 = 3, Y = 4+3+9 = 16 > 10. P1++  
◼ P1 = 2, P2 = 3, Y = 3+9 = 12 > 10. P1++  
◼ P1 = 3, P2 = 3, Y = 9 = 9 <= 10. P2++  
◼ P1 = 3, P2 = 4, Y = 9+6 = 15 > 10. P1++  
◼ P1 = 4, P2 = 4, Y = 6 = 6 <= 10. P2++  
◼ P1 = 4, P2 = 5, Y = 6+3 = 9 <= 10. P2++  
◼ P1 = 4, P2 = 6, Y = 6+3+1 = 10 .. max stop

***Code:***

int arr[1000], n,m;

cin >> n>>m;

for (int i = 0; i < n; i++)cin >> arr[i];

int l = 0, r = 0;

ll ans,sum=0;

while (l < n) {

while (r<n&&sum+arr[r]<=m) {

sum += arr[r];

r++;

}

ans = max(ans, sum);

sum -= arr[l];

l++;

}

EX2:

Another (critical) example  
◼ Given an array containing N integers, you need to find the  
length of the **smallest)num of element)** contiguous subarray that contains at  
least K **distinct** elements in it.  
◼ As we said, P1=P2 = 0. Shift P2, then P1, P2, P1….etc  
◼ But what makes a window ok?  
◼ As long as we don’t have k distinct numbers = **OK**◼ How to know current count?  
◼ Maintain a set & map datastructure of the current numbers  
◼ P2 adds its number, P1 remove its number

***Code***

int arr[1000], n,k;

map<int, int>cnt;

set<int>s;

cin >> n>>k;

for (int i = 0; i < n; i++)cin >> arr[i];

int l = 0, r = 0;

ll ans=OO,sum=0;

while (l < n) {

while (r<n &&s.size()<k) {

s.insert(arr[r]);

cnt[arr[r]]++;

r++;

}

if (s.size() >= k) {

ans = min(ans, ll(r - l));

}

if (cnt[arr[l]] == l) s.erase(arr[l]);

cnt[arr[l]]--;

l++;

}

Your turn  
Given two sorted arrays A and B, each having  
length N and M respectively. Form a new  
sorted merged array having values of both the  
arrays in sorted format.  
◼ This is 2 arrays not just one! They are also sorted  
◼ Let P1 = 0 on Array A  
◼ Let P2 = 0 on Array B  
◼ Let C is the new array  
◼ What is C[0]? A[p1] or B[P2]?  
◼ This is an important step of the merge sort algorithm  
Summary  
◼ Examples summary  
◼ So we maintain 2 (or more?) pointers on an array  
◼ Case: p1 = start, p2 = end  
◼ Case: p1 = start, p2 = start+fixed\_length  
◼ Case: p1 = start, p2 = start  
◼ Case: p1 = start of array, p2 = start of another array  
◼ Some popular algorithms are related,  
explicitly or implicitly, to 2-pointers  
◼ Reverse string (We can do that with 2 points (0, n-1) and  
do swapping)  
◼ Quick sort, Mrege sort, Z-function, Prefix function