# 2-pointers

**Pointers** mean an index in an array.

Q. You are given 2 sorted arrays: A of size n and B of size m. Merge them into one sorted array.

Eg.

Input :

A = [1, 6, 9, 13, 18, 18]

B = [2, 3, 8, 13, 25]

Output:

[1, 2, 3, 6, 8, 9, 13, 13, 18, 18, 25]

Link: <https://codeforces.com/edu/course/2/lesson/9/1/practice/contest/307092/problem/A>

Sol:

| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  int32\_t main() {  int n, m;  cin >> n >> m;   vector<int> a(n), b(m);  for (int i = 0; i < n; i++) {  cin >> a[i];  }   for (int i = 0; i < m; i++) {  cin >> b[i];  }   int i = 0, j = 0;  vector<int> c;   while (i < n && j < m) {  if (a[i] <= b[j]) {  c.push\_back(a[i]);  i++;  } else {  c.push\_back(b[j]);  j++;  }  }   while (j < m) {  c.push\_back(b[j]);  j++;  }   while (i < n) {  c.push\_back(a[i]);  i++;  }   for (int i = 0; i < m + n; i++) {  cout << c[i] << ' ';  }   return 0; } |
| --- |

TIme Complexity: O(n + m)

Q: <https://cses.fi/problemset/task/1641>

A[] = {2,7,5,1}

X = 8

2 -> sumLeft = 8-2 = 6

A[] = {1,2,3,4,5,6,7,8}

1 -> sumLeft = x-1 = 6 //x=7

Sum = 10

\*Sum == sumLeft -> we have found a triplet -> 1,2,8

\*sum<sumLeft - > we can ignore the minimum number i.e. the leftmost number

\*sum>sumLeft -> we can ignore the maximum number i.e. the righmost number .

2,7,5,1 -> 1,3,4

1,2,3,4

1,2,5,7 -> 1,2,3

Sol:

| int n,x;  cin>>n>>x;  vector<pii > v(n);  for(int i=0;i<n;i++){  cin>>v[i].fi;  v[i].se = i;  }  sort(v.begin(),v.end());  vector<int> ans;  for(int i=0;i<n;i++){ //v[i].fi is the 1st number of the triplet  int sumLeft = x-v[i].fi;  int l=i+1,r=n-1;  while(l<r){  int sum = v[l].fi+v[r].fi;  if(sum==sumLeft){ //a pair has been found  // triplet = {v[i].se,v[l].se,v[r].se}  ans.pb(v[i].se);  ans.pb(v[l].se);  ans.pb(v[r].se);  break;  }else if (sum<sumLeft){  l++;  }else{  r--;  }  }  if(ans.size()!=0){  break;  }  }  sort(ans.begin(),ans.end());  if(ans.size()!=0){  cout<<ans[0]+1<<" "<<ans[1]+1<<" "<<ans[2]+1;  }else{  cout<<"IMPOSSIBLE";  } |
| --- |

Q: <https://codeforces.com/contest/279/problem/B>

N books numbered from 1 to n

Ith book takes a[i] minutes

Free Time available: T minutes

Basically, we need to find largest continuous segment of the array such that the sum of elements in this segment <=t

We can take 2 pointers - l and r

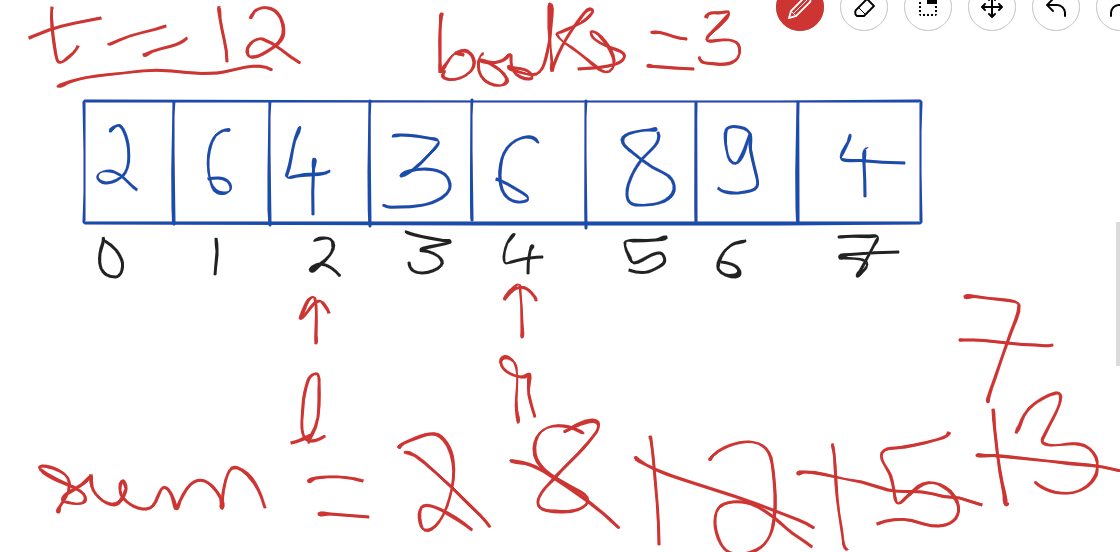
If sum within the range [l, r] <=t, then we can increment r.

Otherwise if sum>t, we can increase l until sum becomes <=t

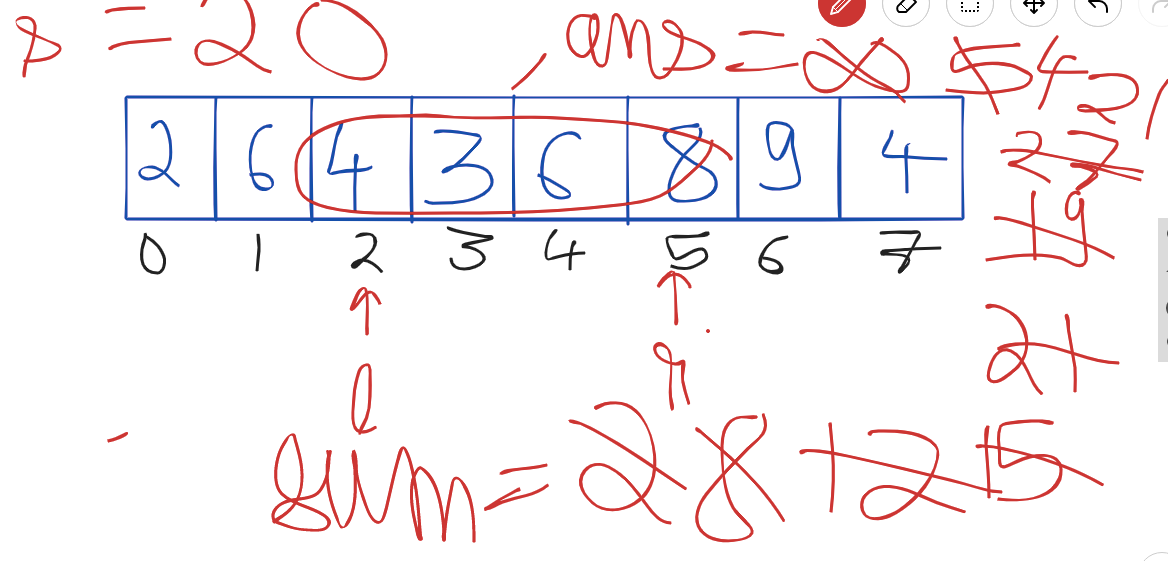
Sol:

| #include <bits/stdc++.h> #define int long long  using namespace std;  typedef long long ll;  int32\_t main() {  int n, t;  cin >> n >> t;   vector<int> a(n);   for (int i = 0; i < n; i++) {  cin >> a[i];  }   int sum = 0, ans = 0;  int l = 0;  for (int r = 0; r < n; r++) {  sum = sum + a[r];  while (sum > t) {  sum -= a[l];  l++;  }  ans = max(ans, r - l + 1);  }  cout << ans;   return 0; } |
| --- |

Time complexity: O(n)



Q. <https://codeforces.com/edu/course/2/lesson/9/2/practice/contest/307093/problem/B>



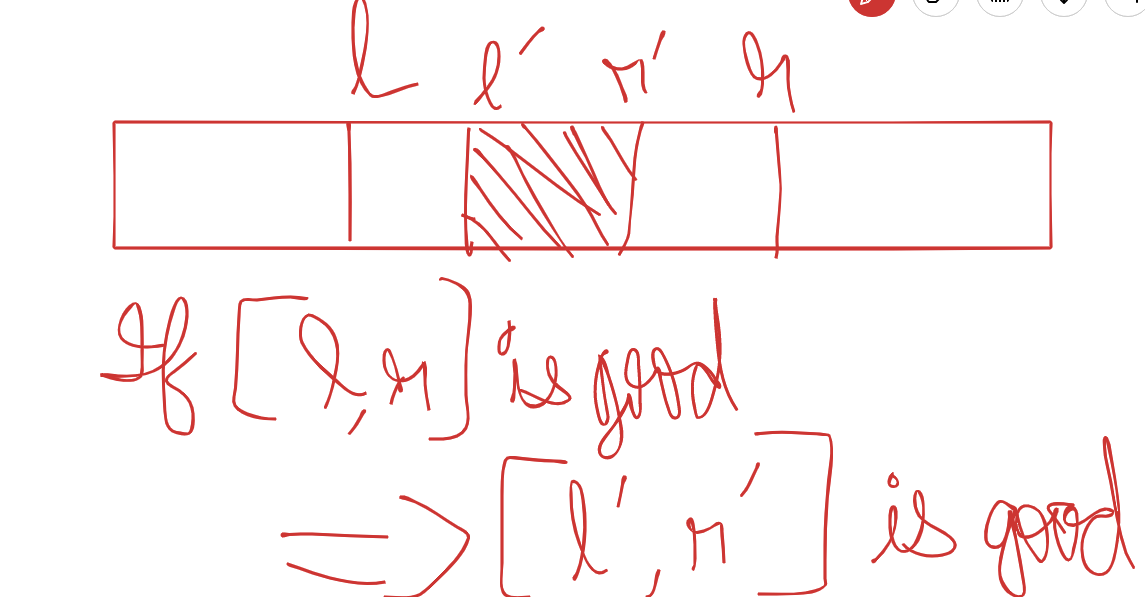
| #include <bits/stdc++.h> #define int long long  using namespace std;  typedef long long ll;  int32\_t main() {  int n, s;  cin >> n >> s;   vector<int> a(n);   for (int i = 0; i < n; i++) {  cin >> a[i];  }   int sum = 0, ans = 1e5 + 1;  int l = 0;   for (int r = 0; r < n; r++) {  sum = sum + a[r];  while (sum - a[l] >= s) {  sum = sum - a[l];  l++;  }  if (sum >= s) ans = min(ans, r - l + 1);  }   if(ans == 1e5 + 1)  cout<<-1;  else  cout << ans;   return 0; } |
| --- |

**Time Complexity**: O(n)

## 

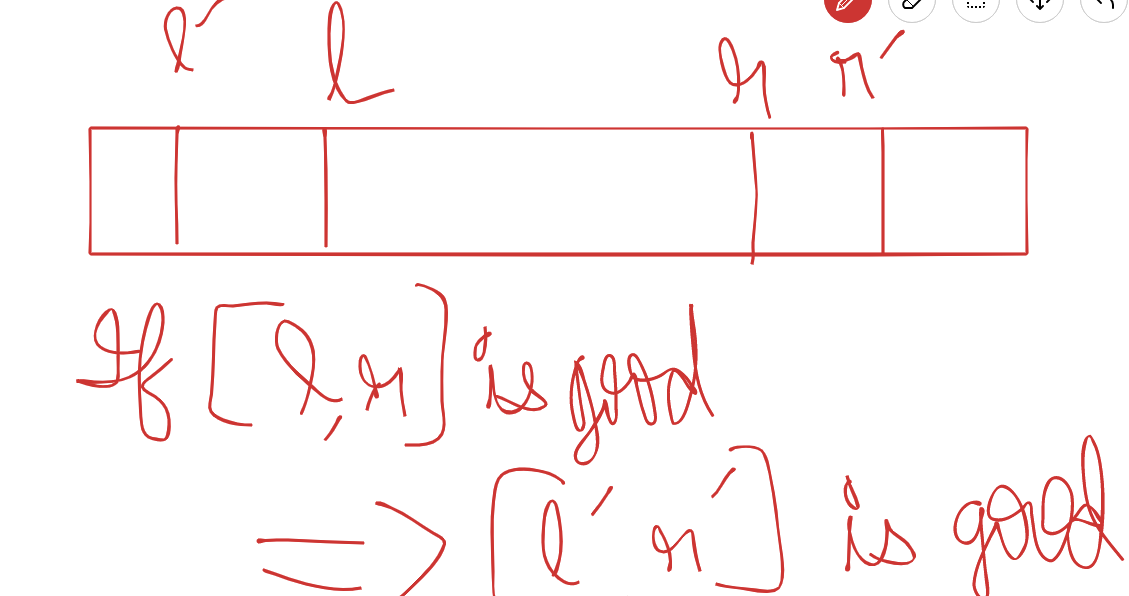
## When we can use 2-pointers?

**Case 1:**



**Example:**  
Sum of elements <= t ( as solved in previous question - B. Books)

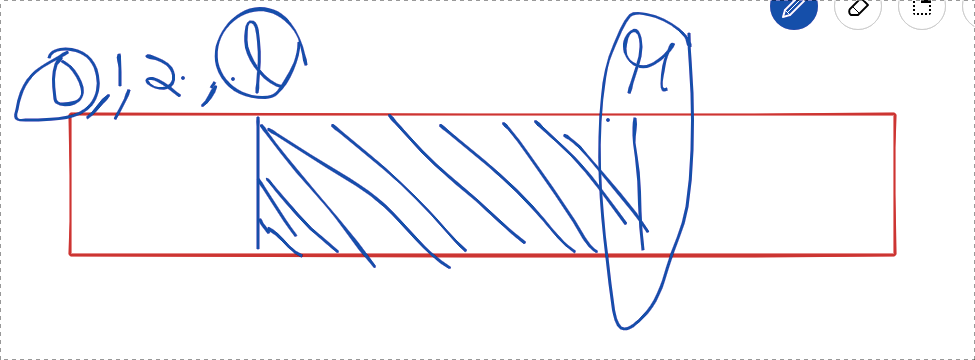
**Case 2:**



**Example:**

Sum of elements >= s (as solved in the previous )

Q. <https://codeforces.com/edu/course/2/lesson/9/2/practice/contest/307093/problem/D>



| #include <bits/stdc++.h> #define int long long  using namespace std;  typedef long long ll;  int32\_t main() {  int n, s;  cin >> n >> s;   vector<int> a(n);   for (int i = 0; i < n; i++) {  cin >> a[i];  }   int sum = 0, ans = 0;  int l = 0;   for (int r = 0; r < n; r++) {  sum = sum + a[r];  while (sum - a[l] >= s) {  sum = sum - a[l];  l++;  }  if (sum >= s) ans = ans + l + 1;  }  cout << ans;   return 0; } |
| --- |

**Time Complexity**: O(n)

**Practice Questions:**

1. <https://codeforces.com/edu/course/2/lesson/9/2/practice/contest/307093/problem/C>

2. <https://codeforces.com/edu/course/2/lesson/9/2/practice/contest/307093/problem/E>

3. <https://codeforces.com/edu/course/2/lesson/9/1/practice/contest/307092/problem/B>

4. <https://cses.fi/problemset/task/1640>

5. <https://codeforces.com/problemset/problem/702/C>

6. Try to solve the problems at:

<https://codeforces.com/edu/course/2/lesson/9/3/practice>