1. Modified Cadens for starting and Ending Index

```
 int l = 0, r = 0, maxsofar = A[0], gmax = A[0], start = 0, end =
0;
    for(int i=1;i<n;i++){
        if(maxsofar + A[i] >= A[i]){   // equal for largest
subarray when two subarray                 with equal sum possible
            maxsofar = A[i] + maxsofar;
        }
        else{
            s =  i;
            maxsofar = A[i];
        }
           if(maxsofar > gmax){
            start = s, end = i;
            gmax = maxsofar;
        }
    }
```

2. Z algorithm
   https://www.youtube.com/watch?v=bS33M8pKFNU
   0 1 2 3 4 5 6 7 8 9 10 11
   a b c a b c a b c a b c x y a b c
   - 0 0 9 0 0 6 0 0

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
a	b	c	a	b	c	a	b	c	a	b	c	x	y	a
-	0	0	9	0	0	6								

```
 vector<int> z_function(string s) {
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; ++i) {
        if (i <= r)
            z[i] = min (r - i + 1, z[i - l]);
        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
            ++z[i];
        if (i + z[i] - 1 > r)
            l = i, r = i + z[i] - 1;
    }
    return z;
}
```

Application check string is cyclick or not
Create z array
Find the leftmost index with i + z[i] == n and n % i == 0.
Ans = cycle of length i

Reason
abcabcabc

0	1	2	3	4	5	6	7	8
a	b	c						
-	0	0	6					

At index 3 z[3] = 6 that means first six character and last 6 characters are same. So we can fill index 3 4 5 with a b c. similarly 6 7 8 a b c

```
int n = a.length(), l = 0, r = 0;
   for(int k=1;k<n;k++){
       if(k > r){
           l = r = k;
           while(r < n && a[r-l] == a[r])
               r++;
           z[k] = r - l;
           r--;
       }
       else{
           if(z[k-l]+k <= r)
               z[k] = z[k-l];
           else {
               l = k;
               while(r < n && a[r-l] == a[r])
                   r++;
               z[k] = r - l;
               r--;
           }

       }

   }
```

3. Next smaller element

```
int leftsmaller[n],
rightsmaller[n];
    for(int i=0;i<n;i++){
        int p = i - 1;
        while(p >= 0 && A[p] >=
A[i])
            p = leftsmaller[p];
        leftsmaller[i] = p;
    }
```

```
  for(int i=n-1;i>=0;i--){
        int p = i + 1;
        while(p <= n-1 && A[p]
>= A[i])
            p =
rightsmaller[p];
        rightsmaller[i] = p;
    }
```

4. Longest Increase Subsequence - long

```
int lengthOfLIS(vector<int>& nums) {
    vector<int> res;
    for(int i=0; i<nums.size(); i++) {
        auto it = std::lower_bound(res.begin(), res.end(), nums[i]);
        if(it==res.end()) res.push_back(nums[i]);
        else *it = nums[i];
    }
    return res.size();
}
```

5. Catalon Number

```
G[0] = G[1] = 1;

  for(int i=2; i<=n; ++i) {
    for(int j=1; j<=i; ++j) {
      G[i] += G[j-1] * G[i-j];
    }
  }
  return G[n];
```

6. Segment Tree

```
#define leftchild(l) 2*l+1
#define rightchild(r) 2*r+2
#define mid(l,r) (l+r)/2

int buildSeg(int l, int r, int pos, int arr[]){
    if(l == r){
        return seg[pos] = arr[l];
    }
    else{
        return seg[pos] =  min(buildSeg(l, mid(l,r), leftchild(pos), arr),
                           buildSeg(mid(l,r)+1, r, rightchild(pos), arr));
    }
}

int query(int l, int r, int pos, int arr[], int ql, int qr){

    if(l >= ql && r <= qr) return seg[pos];
    if(qr <= mid(l,r)) return query(l, mid(l,r), leftchild(pos), arr, ql, qr);
    if(ql > mid(l,r))  return query(mid(l,r)+1, r, rightchild(pos), arr, ql, qr);
    return  min(query(l, mid(l,r), leftchild(pos), arr, ql, qr),
         query(mid(l,r)+1, r, rightchild(pos), arr, ql, qr));

}

int update(int l, int r, int pos, int arr[], int index, int value){
    if(l == r && r == index){
        return seg[pos] = value;
    }

    if(index > r || index < l)
        return seg[pos];
    return seg[pos] = min(update(l, mid(l,r), leftchild(pos), arr, index, value),
                     update(mid(l,r)+1, r, rightchild(pos), arr, index, value));

}
```