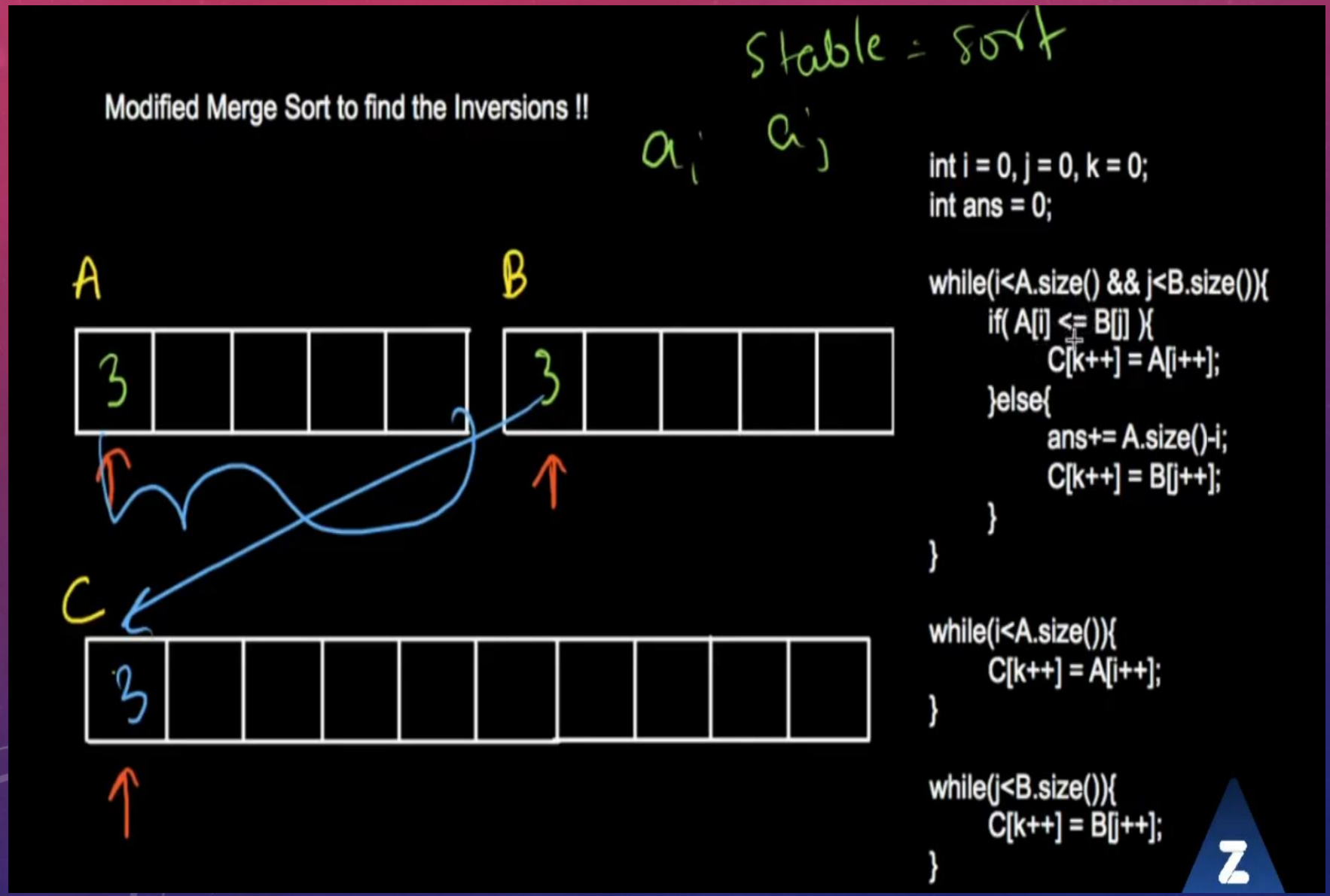
Fibonacci Numbers

① $\sum_{i=0}^{n} F_i = F_{n+2} - 1$

② $\sum_{i=0}^{n} F_i^2 = F_n \cdot F_{n+1}$

③ $gcd(F_n, F_m) = F_{gcd(n,m)}$

Inversions using merge sort



Code for recursive Merge Sort

https://drive.google.com/file/d/1G2eJiTZrms5XBpIO2Kcp4LM2Usc9opHO/view

Inversion:
a[i]>a[j] for i<j

① $(a+b)^n = {}^nC_0 a^0 b^n + {}^nC_1 a^1 b^{n-1} \dots {}^nC_n a^n b^0$

② Calculating ${}^nC_r$

- Linearly → ($n \le 10^6$) → Fact[0] = 1, Fact[i] = (Fact[i-1] * i) % prime
- Multiplication → ($n \le 10^{18}$, $n-r, r \le 10^6$) → $\prod_{i=0}^{r} \frac{(n-i)}{(i+1)}$

③ $\sum_{i=0}^{n} {}^nC_i = 2^n$ $\quad \sum_{i=0}^{n} i \cdot {}^nC_i = n \cdot 2^{n-1}$

④ $\sum_{m=k}^{n} {}^mC_k = {}^{n+1}C_{k+1}$ eg. ${}^1C_1 + {}^2C_1 + {}^3C_1 = {}^4C_2 = 6$

⑤ $\sum_{i=0}^{k} {}^nC_i \cdot {}^mC_{k-i} = {}^{n+m}C_k$

⑥ $\sum_{i=0}^{n} ({}^nC_i)^2 = {}^{2n}C_n$

⑦ $x_1 + x_2 + x_3 \dots x_n = r$ where $x_i \ge 0$

no. of ways = ${}^{n+r-1}C_{n-1}$

⑧ $\sum_{k=0}^{[\frac{n}{2}]} \binom{n-k}{k} = Fib(n+1)$

Catalan Numbers → 1, 1, 2, 5, 14, 42

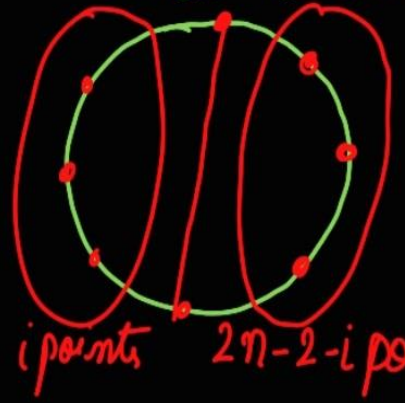
① $C_n = \left(\frac{1}{n+1}\right)\binom{2n}{n} = \left({}^{2n}C_n - {}^{2n}C_{n-1}\right)$

② Recurrence $C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i}$, $C_0 = 1$

eg. no. of triangulation ways for convex polygon of (n+2) sides → C_n

$f(n+1) = \sum_i f(i)\, f(n-i)$

eg. no. of ways to connect 2n points on circle such that no 2 intersect



$f(2n) = \sum f(i) * f(2n-2-i)$

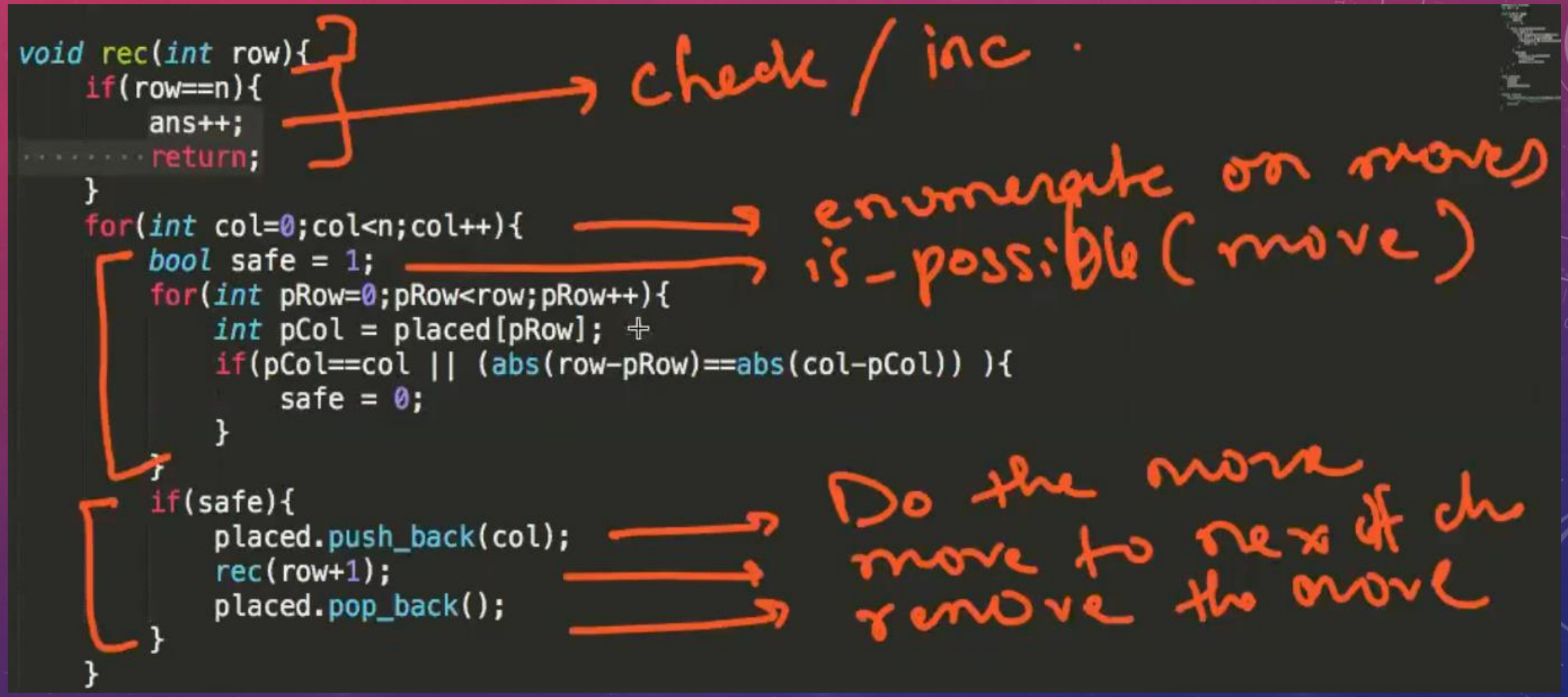
SEGMENT TREE

```
void build(int index, int l, int r)
{
    if(l==r)
    {
        t[index]=a[l];
        return;
    }
    int mid=(l+r)/2;
    build(index*2,l,mid);
    build(index*2+1,mid+1,r);
    t[index]=t[2*index]+t[2*index+1];
}
```

```
void update(int index, int l, int r, int pos, int val)
{
    if(pos<l || pos>r) return;
    if(l==r)
    {
        t[index]=val;
        a[l]=val;
        return;
    }
    int mid=(l+r)/2;
    update(2*index,l,mid,pos,val);
    update(2*index+1,mid+1,r,pos,val);
    t[index]=t[2*index]+t[2*index+1];
}
```

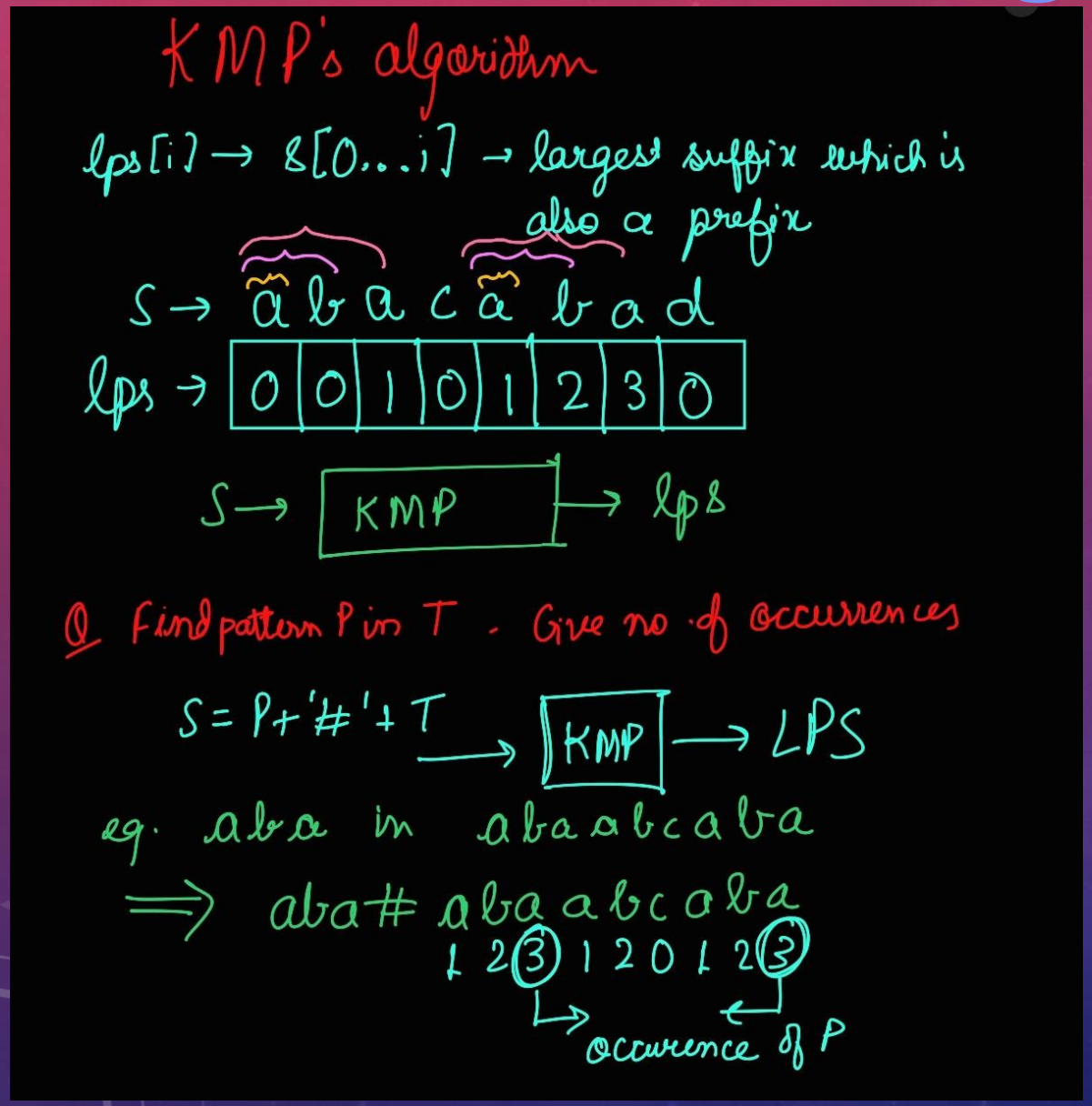
```
int query(int index, int l, int r, int lq, int rq)
{
    if(l>rq || lq>r) return 0;
    if(lq<=l && r<=rq) return t[index];
    int mid=(l+r)/2;
    return query(2*index,l,mid,lq,rq)+ query(2*index+1,mid+1,r,lq,rq);
}
```

N QUEENS



We start from row 0. We try placing in each column. If we can place in that column, we store that column, call for the next row and remove that column. The number of times we reach row n will be the answer.

KMP Algorithm




```
void solve(){
    string s;
    cin>>s;
    int n = s.length();

    int lps[n+1];
    int i=0,j=-1;lps[0]=-1;
    while(i<n){
        while(j!=-1&&s[j]!=s[i])j=lps[j];
        i++;j++;lps[i]=j;
    }
```

To find the period in a string -> N-lps[N]