Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

TUF

Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

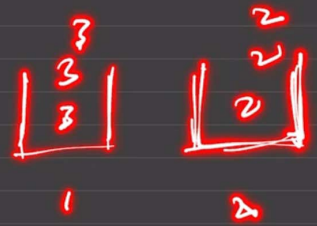
Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

TUF

Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]



Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

4

Fruit into Baskets

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

5

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

TUF

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

1, 2

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

3

set

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 2 1 1 2 3 3 4]

2
3
set

3

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

2

3

set

```
for ( i = 0 → n )
    set < int > st

    for ( j = i → n )
        st.add ( arr[i] );

        if ( st.s
```

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = 0

for (i = 0 → n)
{
    set < int > st

    for (j = i → n)
    {
        st.add (arr[j]) ;

        if (st.size () <= 2)
            maxlen = max (maxlen , j-i+1)

```
arr[] = [3 3 3 1 2 1 1 2 3 3 4]
                        maxlen = 0
for( i = 0 -> n)
{
    set < int > st

    for( j = i -> n)
    {
        st.add(arr[j]);

        if (st.size() <= 2)
            maxlen = max(maxlen, j-i+1)
        else
            break;
    }
}
```

```
for ( i = 0 → n )
{
    set < int > st

    for ( j = i → n )
    {
        st.add ( arr[i] ) ;

        if ( st.size () <= 2)
            maxlen = max (maxlen , j-i+1)
        else
            break ;
    }
}
```

Fruit into Baskets

2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = 0

for (i = 0 → n)

TC → N × N

set < int > st

for (j = i → n) → log N

st.add (arr[i]) ;

if (st.size () <= 2)

maxlen = max (maxlen, j-i+1)

else

break ;

}

}

log 3 = 1

Fruit into Baskets

2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = 0

for (i = 0 → n)

set < int > st

for (j = i → n) → log N

st.add (arr[i]) ;

if (st.size() <= 2)

maxlen = max(maxlen, j-i+1)

else

break ;

}

}

TC → N × N

SC → O(3)

log 3 = 1

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l

r

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = 0 1

1

3, 2

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 1
1, 1
3, ~~2~~ 3

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 1
1, 1
3, 1

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l r

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 1
1, 1
3, 0

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 1
1, 1

2

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 1
1, 2

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

2, 2
1, 3

num, freq

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l r

maxlen = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

2, 1
3, 1

num, freq

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

```
func ( arr)
{
    l=0 , r=0 , maxlen=0 ,  mpp

    while ( r < arr.size())
    {
        mpp[
```

```
func ( arr )
{
    l = 0 , r = 0 , maxlen = 0 ,  mpp

    while ( r < arr.size() )
    {
        mpp[ arr[r] ] ++ ;

        if ( mpp.size() <= k )
        {
            max
```

```
while (r < arr.size())
{
    mpp[arr[r]]++;

    if (mpp.size() > k)
    {
        while (mpp.size() > k)
        {
            mpp[arr[l]]--;
            if (mpp[arr[l]] == 0) mpp.erase(mp
```

```
mpp[arr[r]]++;

if (mpp.size() > k)
{
    while (mpp.size() > k)
    {
        mpp[arr[l]]--;
        if (mpp[arr[l]] == 0)
            mpp.erase(arr[l]);
```

```
if (mpp.size() > k)
{
    while (mpp.size() > k)
    {
        mpp[arr[l]]--;
        if (mpp[arr[l]] == 0)
            mpp.erase(arr[l]);

        l++;
    }

}
```

```
            l++;
        }

    }
    if ( mpp.size() <= k)
    {
        maxlen = max(maxlen, r-l+1);
    }

    right++;
```

```
            maxlen = max(maxlen, r-l+1);
        }

        right++;
    }

    return maxlen;
}
```

Fruit into Baskets

max length subarray with at most 2 types of numbers.

```
                r                       r
arr[] = [3 3 3 1 2 1 1 2 3 3 4]
```

TC - N + N

```
func (arr)
{
    l = 0, r = 0, maxlen = 0, mpp

    while (r < arr.size())
    {
        mpp[arr[r]]++;

        if (mpp.size() > k)
        {
            while (mpp.size() > k)
            {
                mpp[arr[l]]--;
```

```
func ( arr)
{
    l=0 , r=0 , maxlen=0 ,  mpp
    while ( r < arr.size())
    {
        mpp[arr[r]]++;
        if (mpp.size() > k)
        {
            while ( mpp.size() > k)
            {
                mpp[arr[l]]--;
                if (mpp[arr[l]]==0)
                    mpp.erase(arr[l]);
                l++;
            }
```

TC - N + N

SC - O(3)

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

```
func ( arr)
{
    l=0, r=0, maxlen=0, mpp

    while ( r < arr.size())
    {
        mpp[arr[r]]++;

        if (mpp.size() > k)
        {
            while (mpp.size() > k)
            {
                mpp[arr[l]]--;
                if (mpp[arr[l]]==0)
                    mpp.erase(arr[l]);
```

TC - N + N

SC - O(3)

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l r

max len = 4

1
2
3

3

2N → N

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l r

2, 1
1, 1
3, 3

3

maxlen = 4

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 X 3 1 2 1 1 2 3 3 4]

l r

maxlen = 4

2, 1
1, 2
3, 2

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

l r

maxlen = 4

2, 1
1, 3
3, 1

Fruit into Baskets

max length subarray with at most 2 types of numbers.

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

maxlen = 4

2, 1
1, 3
3, 1

3

```
while (r < n)
{
    mpp{arr[r]]++

    if (mpp.size() > k)
    {
        mpp{arr[l]]--
        if (mpp{arr[l]]==0) mpp.erase(
```

```
mpp{arr[r]]++

if (mpp.size() > k)
{
    mpp{arr[l]]--
    if (mpp{arr[l]] == 0) mpp.erase(mpp{arr[l]]
    l++;
}

if (m
```

Once

```
if ( mpp.size() > k)
{
    mpp[arr[l]]--
    if (mpp[arr[l]] == 0) mpp.erase(mpp[arr[l]])
    l++;
}
if (mpp.size() <= k) maxlen = max(maxlen, r-l+1);
r++;
}
```

Once

```
if ( mpp.size() > k)
{
    mpp[arr[l]]--
    if (mpp[arr[l]] == 0) mpp.erase(mpp[arr[l]]
    l++;
}
if (mpp.size() <= k) maxlen = max(maxlen, r-l+1);
r++;
}
return maxlen.
```

Once

arr[] = [3 3 3 1 2 1 1 2 3 3 4]

TC → O(N)
SC → O(1)

```
func (arr, k)
{
    l = 0, r = 0, maxlen = 0, mpp

    while (r < n)
    {
        mpp{arr[r]}++

        if (mpp.size() > k)
        {
            mpp{arr[l]}--
            if (mpp{arr[l]} == 0) mpp.erase(mpp{arr[l]})
            l++;
        }

        if (mpp.size() <= k) maxlen = max(maxlen, r-l+1);
        r++;
    }
```

Once