



1. Given a matrix having 0-1 only where each row is sorted in increasing order, find the row with the maximum number of 1's.

binary search >  assignment1.cpp >  main()

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int arr[] = {1,2,2,3,3,3,5,7,8,8};
5      int n = 10;
6      int x = 8;
7      int low = 0 ;
8      int high = n-1;
9      bool flag = false;
10     while(low<=high){
11         int mid = low + (high - low)/2;
12         if(arr[mid]==x){
13             if(arr[mid+1]!=x){
14                 flag = true;
15                 cout<<mid;
16                 break;
17             }
18             else low = mid +1;
19         }
20         if(arr[mid]<x)    low = mid + 1;
21         if(arr[mid]>x)    high = mid - 1;
22     }
23     if(flag == false) return -1;
24 }
```

2. Given a sorted binary array, efficiently count the total number of 1's in it.

binary search > [assignment2.cpp](#) > ...

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int arr[] = {0,0,0,0,0,1,1,1,1,1,1,1};
5      int n = 124;
6      int x = 1;
7      int low = 0 ;
8      int high = n-1;
9      int f=-1;
10 while(low<=high){
11     int mid = low + (high - low)/2;
12     if(arr[mid]==x){
13         if(mid==0){
14             f=mid;
15             break;
16         }
17         else if(arr[mid-1]!=x){
18             f=mid;
19             break;
20         }
21         else high = mid -1;
22     }
23     if(arr[mid]<x)    low = mid + 1;
24     if(arr[mid]>x)    high = mid - 1;
25 }
26 if(f==-1) cout<<"0";
27 else cout<<n-f;
28 }
```

3. Given a matrix having 0-1 only where each row is sorted in increasing order, find the row with the maximum number of 1's.


binary search > [G+](#) assignment3.cpp > ...

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int arr[5][6]= {{0,0,0,1,1,1},{0,0,1,1,1,1},{0,0,0,0,1,1},{0,1,1,1,1,1},{0,0,0,0,0,1}};
5      int m = 5;
6      int n = 6;
7      int x = 1;
8      int row = 0;
9      int maxcount = 0;
10     for(int i=0;i<m;i++){
11         int low = 0;
12         int high = n-1;
13         int count = 0;
14         int f = -1;
15         while(low<=high){
16             int mid = low + (high - low)/2;
17             if(arr[i][mid]==x){
18                 if(mid==0){
19                     f=mid;
20                     break;
21                 }
22                 else if(arr[i][mid-1]!=x){
23                     f=mid;
24                     break;
25                 }
26                 else high = mid -1;
27             }
28             if(arr[i][mid]<x) low = mid + 1;
29             if(arr[i][mid]>x) high = mid - 1;
30         }
```

```
30     }
31     if(f==-1) count = 0;
32     else count = n-f;
33     if(maxcount<count){
34         maxcount = count;
35         row = i;
36     }
37 }
38 cout<<row<<" "<<maxcount;
39 }
```

4. Given an array of integers `nums` containing $n + 1$ integers where each integer is in the range $[1, n]$ inclusive in sorted order.

There is only one repeated number in `nums`, return this repeated number.

binary search >  assnignment4.cpp > ...

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int arr[] = {1,2,3,4,5,6,6,7,};
5      int n = sizeof(arr)/sizeof(arr[0]);
6      int low = 0;
7      int high = n-1;
8      while(low<=high){
9          int mid = low + (high - low)/2;
10         if(arr[mid]==mid+1) low = mid+1;
11         if(arr[mid]==mid){
12             if(arr[mid]==arr[mid-1]){
13                 cout<<arr[mid];
14                 break;
15             }
16             else high = mid - 1;
17         }
18     }
19 }
```