Session A 08

Tuesday, 14 October 2025 9:06 AM

nums = [4, 4, 8, 4, 12, 5, 3, 3, 1]

x = 0;

x = x ^ nums[i];

100
100
000
010
010
001
011

011
010
001
101
100
011
111

111
011
100
001
101 → 5

Q 338 (Counting Bits)

n = 5;

0	1	2	3	4	5
0	1	1	2	1	2

000
001
010
011
100
101

```
for( i=0; i<=n ; i++)
{
    x = i;
    while(x != 0)
    {
        count = count + (x&1);
        x = x>>1;
    }
    ans[i] = count;
}
```

n = 11

(x) & bit storage

n = 0000 1011 & 1

0000 0001

0000 0001

if number is odd

n = n>>1; | n = n/2;

count = count + (n&1);

count 0 x2

0000 1011

& 0000 0001

0000 0001

0000 1010 ✓

& 0000 0001

0000 0000

n = n/2 | n = n>>1;

count 3

0000 0001

0000 0000

n = 0

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* countBits(int n, int* returnSize) {
    int* result = (int*)malloc(sizeof(int) * (n + 1));
    *returnSize = n + 1;
    for (int i = 0; i <= n; i++)
    {
        int x = i;
        int count = 0;
        while (x)
        {
            x = x & (x - 1);
            count++;
        }
        result[i] = count;
    }
    return result;
}
```

(int*) malloc((sizeof(int)*(n+1)))

```
/**
 * Note: The returned array must be malloced, assume caller calls free().
 */
int* countBits(int n, int *returnSize)
{
    int *result = (int *)malloc(sizeof(int) * (n + 1));
    *returnSize = n + 1;
    for (int i = 0; i <= n; i++)
    {
        int x = i;
        int count = 0;
        while (x)
        {
            x = x & (x - 1);
            count++;
        }
        result[i] = count;
    }
    return result;
}
```

count = count + (x & 1);

x = x/2; | x = x>>1;

leCode :- 2220 (minimum bit flips)

start = 10 → 0000 1010

Goal = 7 → 0000 0111

0 = 0011 01

char *s="abcd";

s[0] = 'a'

s[1] = 'b'

s[2] = 'c'

s[3] = 'd'

t = "abcde";

t[0] = 'a'

t[1] = 'b'

t[2] = 'c'

t[3] = 'd'

char x='\0';

for(int i=0 ; s[i]!='\0'; i++)

{

x = x ^ s[i] ^ t[i];

}

x = x ^ t[i];

```
char findTheDifference(char *s, char *t)
{
    char xor = '\0';
    int i;
    for (i = 0; s[i] != '\0'; i++)
    {
        xor = xor ^ s[i] ^ t[i];
    }
    xor = xor ^ t[i];
    return xor;
}
```