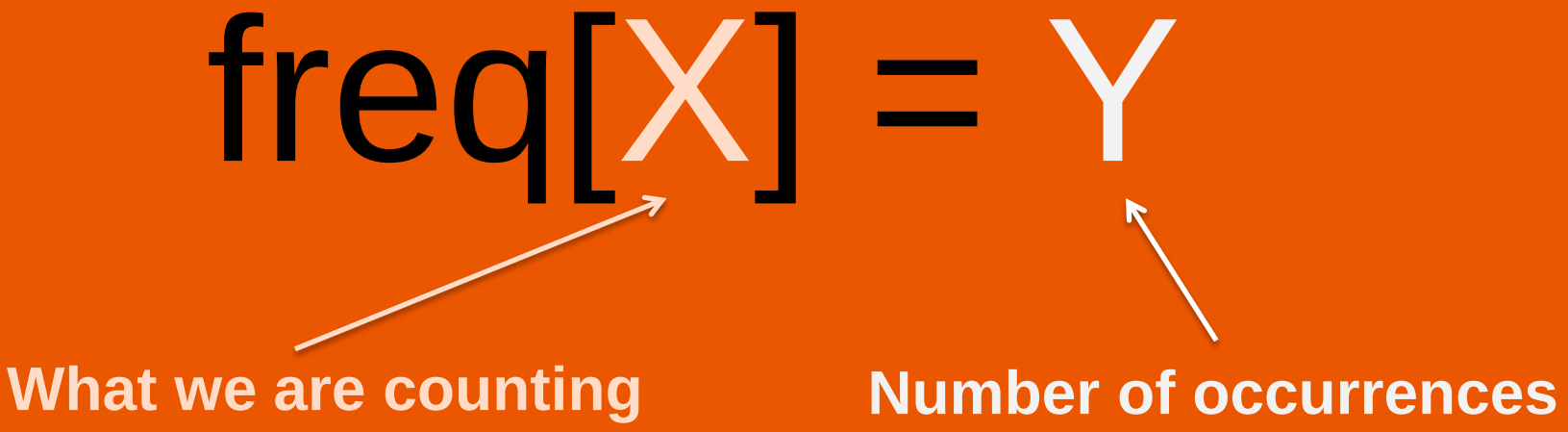
Programming Challenges I

Session 3

CSCI 485/4930 - Spring 2018

Frequency Array


freq[X] = Y
What we are counting
Number of occurrences

Frequency Array Example

Given an array of positive integers, count number of occurrences for each element in the array. Assume all elements in the array are less than 100,000.

Input	Output
7 1 2 1 7 100 7 7	1 occurred 2 times 2 occurred 1 time 7 occurred 3 times 100 occurred 1 time

Frequency Array Example

```
#include <iostream>
using namespace std;
int freq[100005];

int main() {
    int n;
    cin >> n;
    for(int i=0; i<n; i++){
        int cur;
        cin >> cur;
        freq[cur]++;
    }
    for(int i=0; i<=100000; i++){
        if(freq[i] > 0){
            cout << i << " occurred " << freq[i]
            cout << (freq[i] == 1 ? " time" : " times") << "\n";
        }
    }
    return 0;
}
```

Characters Frequency Array Example

ASCII value	Character	Control character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	(null)	NUL	032	(space)	064	@	096	
001	☺	SOH	033	!	065	A	097	a
002	☻	STX	034	"	066	B	098	b
003	♥	ETX	035	#	067	C	099	c
004	♦	EOT	036	$	068	D	100	d
005	♣	ENQ	037	%	069	E	101	e
006	♠	ACK	038	&	070	F	102	f
007	(beep)	BEL	039	'	071	G	103	g
008	◘	BS	040	(	072	H	104	h
009	(tab)	HT	041	)	073	I	105	i
010	(line feed)	LF	042	*	074	J	106	j
011	(home)	VT	043	+	075	K	107	k
012	(form feed)	FF	044	,	076	L	108	l
013	(carriage return)	CR	045	-	077	M	109	m
014	♫	SO	046	.	078	N	110	n
015	☼	SI	047	/	079	O	111	o
016	►	DLE	048	0	080	P	112	p
017	◄	DC1	049	1	081	Q	113	q
018	↕	DC2	050	2	082	R	114	r
019	!!	DC3	051	3	083	S	115	s
020	¶	DC4	052	4	084	T	116	t
021	§	NAK	053	5	085	U	117	u
022	▬	SYN	054	6	086	V	118	v
023	↨	ETB	055	7	087	W	119	w
024	↑	CAN	056	8	088	X	120	x
025	↓	EM	057	9	089	Y	121	y
026	→	SUB	058	:	090	Z	122	z
027	←	ESC	059	;	091	[	123	{
028	(cursor right)	FS	060	<	092	\	124	|
029	(cursor left)	GS	061	=	093	]	125	}
030	(cursor up)	RS	062	>	094	^	126	~
031	(cursor down)	US	063	?	095	_	127	⌂

azabab

Letter	Freq
‘a’	3
‘b’	2
…	
‘z’	1

Letter	Freq
0	3
1	2
…	
25	1

1990

Letter	Freq
‘0’	1
‘1’	1
…	
‘9’	2

Letter	Freq
0	1
1	1
…	
9	2

Characters Frequency Array Example

```
int freq[26];
string s;
cin >> s; // "lowercase"
for (int i = 0; i<s.size(); ++i)
    ++freq[s[i] - 'a'];
```

```
int freq[10];
cin >> s; // "0123456789"
for (int i = 0; i<s.size(); ++i)
    ++freq[s[i] - '0'];
```

azabab

Letter	Freq
‘a’	3
‘b’	2
…	
‘z’	1

Letter	Freq
0	3
1	2
…	
25	1

1990

Letter	Freq
‘0’	1
‘1’	1
…	
‘9’	2

Letter	Freq
0	1
1	1
…	
9	2

Cumulative Array

1D cummulative

index	0	1	2	3	4	5	6	7	8
original		2	0	3	-1	4	2	3	-4
cumulative	0	2	2	5	4	8	10	13	9

```
Sum(i, j) = s[j] - s[i - 1]
```

1D Cumulative Array

```cpp
#include <iostream>
using namespace std;

int cumm[100005];
int main(){
    int n;
    cin >> n;
    for(int i=1; i<n; i++){
        cin >> cumm[i];
        cumm[i] += cumm[i-1];
    }
    //cout << cumm[end] - cumm[start-1];
    return 0;
}
```

2D cummulative

2D Cumulative Array

```cpp
#include <iostream>
using namespace std;

int cumm[1005][1005];
int main(){
    int n, m;
    cin >> n >> m;
    for(int i=1; i<=n; i++){
        for(int j=1; j<=m; j++){
            cin >> cumm[i][j];
            cumm[i][j] += cumm[i-1][j] + cumm[i][j-1] - cumm[i-1][j-1];
        }
    }
    // sum of the rectangle with (sx, sy) and (ex, ey) corners
    cout << cumm[ex][ey] - cumm[ex][sy-1] - cumm[sx-1][ey] + cumm[sx-1][sy-1];
    return 0;
}
```

Bit Masks

Revision

Decimal to Binary Conversion

	Current Number	Current Number % 2
	44	0
44 / 2	22	0
22 / 2	11	1
11 / 2	5	1
5 / 2	2	0
2 / 2	1	1
END	0	

101100

Bitwise Operations

83	1010011
54	0110110
83 & 54 == 18	0010010
83 | 54 == 119	1110111
83 ^ 54 == 101	1100101

83	…0001010011
~83	…1110101100

Bitwise Operations

5	101
5 << 1 == 10	1010
5 >> 1 == 2	10
5 << 2 == 20	10100
5 >> 2 == 1	1

Bitwise Operations

5	101
5 << 1 == 10	1010
5 >> 1 == 2	10
5 << 2 == 20	10100
5 >> 2 == 1	1

Even ?

x % 2 == x & 1

9			
1	0	0	1
2^3	2^2	2^1	2^0
8	4	2	1

if the least significant bit is set to 1,
then the number is odd

Set Bit 1

Set the bit with index **3** to 1 in the number 37 (100101)

37	100101
1 << **3**	001000
37 | (1 << 3)	101101
101101 = 45	

```
int setBit1(int num, int ind) {
        return num | (1<<ind);
}
```

Set Bit o

Set the bit with index **2** to 0 in the number 37 (100101)

37	100101
1 << **2**	000100
~ (1 << 2)	111011
37 & ~(1 << 2)	100001
100001 = 33	

```
int setBit1(int num, int ind) {
        return num & ~(1<<ind);
}
```

Flip Bit

Flip the bit with index **2** in the number 37 (100101)

37	100101
1 << **2**	000100
37 ^ (1 << 2)	100001
100001 = 33	

```
int flipBit(int num, int ind) {
        return num ^ (1 << ind);
}
```

Get Bit

Get the bit with index **2** in the number 37 (100101)

37	100101
37 >> 2	100100
((37 >> 2) & 1)	1
The bit at index 2 is 1	

```
int getBit(int num, int ind) {
        return (num >> ind) & 1;
}
```

Counting ones in binary representation

int cnt = __builtin_popcount(num);

```
int popCount(int num){
    int res = 0;
    while(num > 0){
        res += num & 1;
        num >>= 1;
    }
    return res;
}
```

Bit Masks

Bit Masks

32 int: bits take indices from **0** to **31**

64 int: bits take indices from **0** to **63**

Bit Masks

Power Set:

given a set S = {x, y, z} , write all possible subsets from S.

0	→	0	0	0	→	{ }
1	→	0	0	1	→	{ z }
2	→	0	1	0	→	{ y }
3	→	0	1	1	→	{ y, z }
4	→	1	0	0	→	{ x }
5	→	1	0	1	→	{ x, z }
6	→	1	1	0	→	{ x, y }
7	→	1	1	1	→	{ x, y, z }

Bit Masks

```
string xyz = "xyz";
for(int i = 0; i < (1 << 3; i++){
    for(int j = 0; j < 3; j++)
        if((i >> j) & 1)
            cout << xyz[j];
    cout << endl;
}
```

x
y
xy
z
xz
yz
xyz

Knapsack Problem

Given a Knapsack of a maximum capacity of W and N items each with its own value and weight, throw in items inside the Knapsack such that the final contents has the maximum value.

Input

4 10
100 11
8 6
15 7
8 4

Output

16

Knapsack problem solution using bitmasks

https://ideone.com/RFHVwT

Complexity: O(2^n)

Goint over all subsets (power set), number of different subsets = 2^n.