COP 3502 4/4/24

Today: base conversion
bit wise ops

Tues/Thurs (4/9, 4/11) - Group Project Games

4/16 - Travis on Binary Search Apps

4/18 - Travis Foundation Exam Info + Final Exam Review

Convert base b to base 10

23161_8

$= 2\times8^4 + 3\times8^3 + 1\times8^2 + 6\times8^1 + 1\times8^0$

base b: symbols 0, 1, 2, .. b-1
b symbols
0, 1, 2, 3, .. 9, a, b, c, ... 'z'

$247_{10} = 2\times10^2 + 4\times10^1 + 7\times10^0$

---

base 10 to base 10

$247_{10} \rightarrow$ ----$_5$

$247 = \underbrace{d_3\times5^3 + d_2\times5^2 + d_1\times5^1}_{\text{div is by 5}} + \underbrace{d_0\times5^0}_{247\,\%5}$

remainder when divided by 5

$\frac{247}{5} = d_3\times5^2 + d_2\times5^1 + d_1\times5^0$

last "digit" is obtained via % base
INT DIVISION HAS REST OF # TO CONVERT

$5 \overline{)247}$

$5 \overline{)49}$ R 2

$5 \overline{)9}$ R 4

$5 \overline{)1}$ R 4

0 R 1

$247_{10} = 1442_5$

Convert btw base powers of 2

$3FAC6_{16} = \_\_\_\_\_ _8$

Since $2^4 = 16$, a base 16 "digit" can be replaced with 4 bits base 16 → base 2

$0011|1111|1010|1100|0110_2$

$1\ 7\ 5\ 3\ 0\ 6_8$

$0110 = 0\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0$ (0-15)

$1100 = 1\times2^7 + 1\times2^6 + 0\times2^5 + 0\times2^4$

$= 16^1(1\times2^3 + 1\times2^2 + 0\times2^1 + 0\times2^0)$

---

base X to base Y

→ base 10

Bitwise Operators

& x & y bitwise and

$$x = 10110011$$
$$\& \; y = 01010110$$
$$00010010$$

$11 \times 16 + 3 = 179$
$5 \times 16 + 6$

$$179 \;\&\; 86 = 18$$

| x | y bitwise or

255 - 8

$$x = 10110011$$
$$| \; y = 01010110$$
$$11111011 = \boxed{247}$$

---

btw, in computer ints 2's complement so we can store neg #s. Most significant bit is negative $-2^{31}, 2^{30}, 2^{29} \ldots 2^0$

range int $[-2^{31}, 2^{31}-1]$

---

^ x ^ y bitwise XOR

$14 \times 16 + 5$

256 − 27 = 9

$$x = 10110011$$
$$\wedge \; y = 01010110$$
$$11100101 = 229$$

left shift

x << k

1011000000

x = 1011 k = 5

move # to left by k bits
mathematically multiplying by 2^k

---

right shift

x >> k

x = 1011010011 => 10110

k = 5 move # to the right by k bits
chopping the k least significant bits off
int div by 2^k.

~x flips all bits in x

#1 use is to simulate a ~~bool~~ boolean array of a small size.

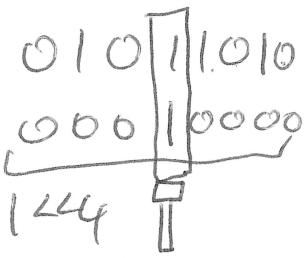
How to access if a bit is on or off

```
if (x & (1<<k))
    // kth bit of x is on
else
    // kth bit of x is off
```

x = 0101101010

0000100000

144



Example Applications

(1) Combinations w/ Knapsack Problem.

Value[subset] = happiness of buying the subset of items delineated here provided it's not too expensive!

(2) Scoring T/F exam

ans key = 10100110 1 = wrong

^ student = 00001111 0 = correct

10101001 count # bits set to 1

↳ incorrect ans

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

Baby sitter

(1) loop through each subset of jobs

example : 01101 (job 0, 2, 3)

for this subset keep onemask = current busy days

job 0 days: 10001
job 2 days: 01010
job 3 days: 00100

if a job sets added check if

busy & job[k] == 0

if not ⟹ not possible

else busy = busy | job[k]

Separately add up amt money

FE Jan 2024 Part D Q 3

x = 0 1 0 0 1 1 1 1 0 1 0 1 1 0

+1 → 10000

+1+1 → ↑ by 1

4 1's to 011 ⟹ 100

1 1

best ans = cur # on bits − longest run + 1