COP3502 - 4/15/22

Bitwise Ops Applications

~ NOT

Seeing if bit is on.

9 8 7 6 5 4 3 2 1 0

int x = 1 0 0 1 0 1 1 0 1 0 (in binary)

is bit # k on?

if ((x & (1<<k)) != 0)

// bit k is on

(1<<k) → like x 2^k left shift k bits

```
  1 0 0 1 0 1 1 0 1 0      x
& 0 0 0 1 0 0 0 0 0 0
-----------------------
```

num bits on

```
int res = 0;
for (int i = 0; i < 20; i++)
    if ((x & (1 << i)) != 0)
        res++;
return res;
```

lowestBitOn
highestBitOn
~~exercise~~ exercises to code

1. (1) Correct Answer Recovery / Test Grading
2. (2) Fence Painting
3. (3) Babysitting (subset of jobs)
4. (4) Knapsack (small # of items)

Old way to try all 2^n ans keys
⇒ recursion fill item k → F / → T

With bitwise ops:

```
for (int key = 0; key < (1<<n); key++) {
    // each int is a unique answer key
    // 000, 001, 010, 011, 100, 101, 110, 111
```

}

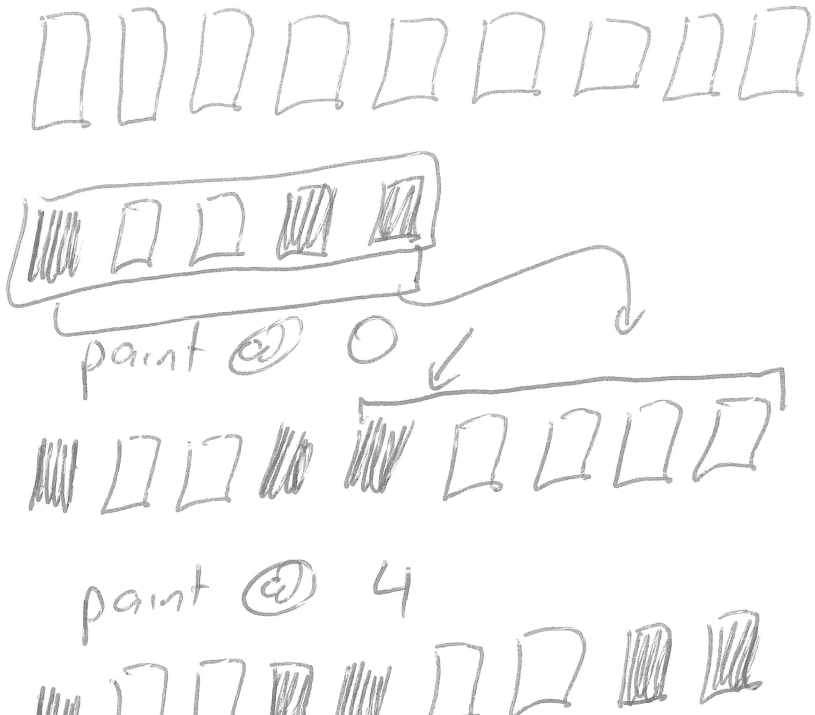
How to score a test

```
myans = 10110101
^ key = 01110111
--------------------
        11000010
```

→ XOR is 1 each bit I got wrong

```
correct = n - bitsOn(myans ^ key);
```

Fence Painting



Input: mold (0-31) → 16+8+1 = 25
positions for painting → [0, 4, 5, 12, 14] paint

Output: final state of the fence as a bitmask

```
int fence = 0
for (int i=0; i<paintLen; i++)
    fence |= (mold << paint[i]);
return fence;
```

Baby Sitting

To check no intersection

(a) (job1 & job2) == 0

(b) (job1 ^ job2) == job1 + job2

(c) (job1 | job2) == job1 + job2

baby sitter illustration

Job (0 based)	Days (0 based)	Money
0	0, 3	30
1	2	25
2	1, 3	70
3	1, 2	35

money

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value	0	30	25	55	70	-1	95	-1	35	65	-1	-1	-1	-1	-1	-1

dayList

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
value	0	9	4	13	~~12~~ 10	x	14	x	6	15	x	x	x	x	x	x
		0,3	2	0,2,3	1,3		1,2,3		1,2	0,1,2,3						

When filling out index 9, we see the ~~lowestOneBit~~ leastBitOn is 0, then 9 - (1 << 0) = 8 so to build the set of job 9, we can do job 0 added to set of jobs 8 (which is just job 3) Job 3 uses days 1, 2, stored as dayList[8] = 6 $= 2^1 + 2^2$. Job 0 uses days 0, 3 which is 9. Since 6 & 9 equals 0, there is no conflict. The new subset of busy days is 6 | 9 = 15. The new money earned is 35 + 30 = 65

35: money for job set 8; 30: money job 0.