Use of Bits in 32 bit Word

Opava AP2, 06/04/2017

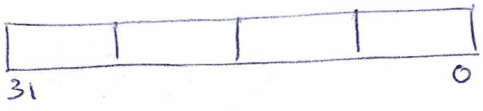
① What is int, short, long
② Operations on bits
③ Example: "Mini Dict".
④ Example: Blocks in Bloxorz.
⑤ Arithmetic by bits.

See:
www.cprogramming.com/tutorial/bitwise_operators.html

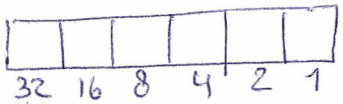
① Datatypes int, etc.

Manual: stores integer number between
-2147483647 & $+2147483647$.

Memory (chips) store blocks of 32 bits. (four Bytes)



Representation: binary coding!



The "value" of the i^{th} bit is 2^i, so eg 43 is
$32 + 8 + 2 + 1$ is 101011

With 6 bits used in this way, the max is 111111,
so $32+16+8+4+2+1$, which is $2^6 - 1$

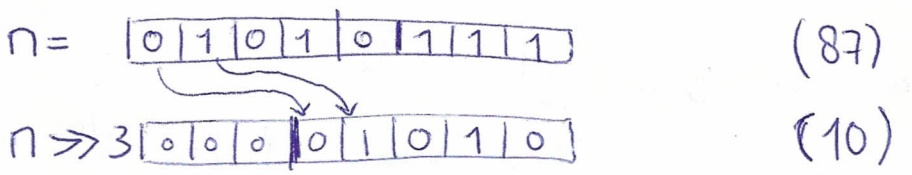
Representation int uses bit nr 31 as sign (0 pos, 1 neg)
so 31 are used to store powers of two.
$\Rightarrow$ max is $2^{31} - 1 = 2147483648$ or Two Billion

(2) Programming language hides this from you!
Write all programs, compute all kinds of things without knowing this!
Operators like + , / are mapped to processor instructions that treat the "word" as an integer.

See www.cprogramming.com/tutorial/bitwise_operators.html
(says: space is usually not a problem but I disagree!)
C, C++, C# and Java offer operations that work directly on bits: extremely fast & powerful!

It works the same for short, int and long but my examples are all just 8b long!

Shift Move bits to left or right, fill on other side with zeroes.



The last three bits are gone!
So $(n \gg 3) \ll 3$ is not always the same as n!

AND & takes two sequences, performs a pointwise AND

n	0 1 0 1 0 1 1 1	(87)
m	1 1 1 1 0 0 0 0	(240)
n & m	0 1 0 1 0 0 0 0	~~(246)~~ 80

87 & 240 = ~~246~~ 80

OR pointwise OR:

n	0 1 0 1 0 1 1 1
m	1 1 1 1 0 0 0 0
n | m	1 1 1 1 0 1 1 1 (243)

XOR Pointwise XOR: place 1 where there is exactly ONE 1

n	0 1 0 1 0 1 1 1
m	1 1 1 1 0 0 0 0
n ^ m	1 0 1 0 0 1 1 1

NOT: unary operator! Bitwise complement

n	0 1 0 1 0 1 1 1
~n	1 0 1 0 1 0 0 0

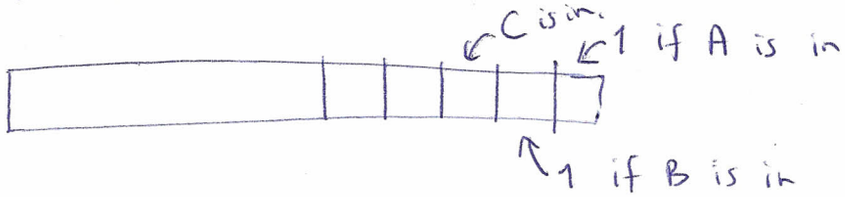
③ Subset dictionary

Program reasons about 20 objects A B C ..
and must store subsets. like { A D E F } set1

{ A B } set2

{ } (= ∅) emp.

You can use a map
or a string, but it
is very efficient to use int!



So set1 = 1 + 8 + 16 + 32 = 57

6	5	4	3	2	1	0
0	0	1	1	1	0	0	1
G	F	E	D	C	B	A

set2 = 11 = 3

emp = 0 0 0 0 0 0 0 0 = 0.

3.1 See if object i is in: ((set >> i) & 1 == 1)

e.g. set1 contains D, object nr 3!

set1 is 0 0 1 1 1 0 0 1
set1>>3 is 0 0 0 0 0 1 1 1
1 is 0 0 0 0 0 0 0 1
AND 0 0 0 0 0 0 0 1 which is indeed == 1

set2 does not contain D.

3.2 Insert item i in a set:

set2 = set | (1 << i)

eg set1 ∪ {2} C} You "want" set1[2]=True

1	is	00000001
1<<2	is	00000100
set1	is	00111001
or of these two		00111101,

so looks "the same as" set1 but now with bit2 =1.

3.3 Delete item i:

set3 = set2 & ~(1 << i)

How does this work?

1<<i	is	00001000 (i)
~(1<<i)	is	11110111

If I & this with some bitrow, all bits will remain except bit i which will be forced to 0.

set2 = 00111101 &

set3 = 00110101

3.4 ~~Check~~ Compute a∪b : a | b

3.5 Check if a⊆b (a & ~b) == 0

} Think of examples your self!

(4) Packing values: Blocks of bits.

In Bloxorz, we have three parameters (i, j, or), four if we count the switches!) that are each small.

Assume i and j are < 8 (3 bits)
or < 4 (2 bits).

Typical example | i = 5 j = 7 or = 2 |

As bits: i = 0 0 0 0 0 | 1 0 1

↑ Because we KNOW that $i < 8$, these bits are ALWAYS 0 so it isn't neccessary to store them!

j = 0 0 0 0 0 | 1 1 1

↑ Five irrelevant bits of j

or = 0 0 0 0 0 0 | 1 0

Storing the three values as three numbers wastes a lot of space! Use an int (8b in my examples) to store the relevant bits only: 3+3+2 is 8 so this fits!

p = 101 | 111 | 10 (i, j, or)

(190 = 32 × 5 + 4 * 7 + 2)

How can we form this number quickly?

I can left-shift i over 5 positions without losing information because I know the left 5 bits are 0:

$i \ll 5$ is 1 0 1 | 0 0 0 0 0 — 160

Similar

$j \ll 2$ is 0 0 0 | 1 1 1 | 0 0 — 28

and

or is 0 0 0 | 1 0 0 0 | 1 0 — 2 +

190.

Addition works differently from OR, but as long as no column contains more 1's the outcome is the same!

I can OR these strings:

p = 1 0 1 | 1 1 1 | 1 0

So $p = (i \ll 5) \mid (j \ll 2) \mid or$

$= (((i \ll 3) \mid j) \ll 2) \mid or$

Find back j: $(p \gg 2)$ & m3

where m3 is a "mask" of three 1's: 0000 0111

which is the value $2^3 - 1$, or 7.

⑤ Arithmetic

Some simple arithmetic can be done with bit operations, especially when powers of 2 are involved!

Even $(x \% 2) == 0$ requires division ¢¢ $

$(x \& 1) == 0$ only looks at last bit.

Multiply by 2 $2i$ is $2 * i$ (multiply)
or $i + i$ (add)
or $i \ll 1$ bitshift

$2i+1$ is $(i \ll 1) | 1$

Multiply by 2^t : $2^t * a$ is $a \ll t$

Divide by 2^t $a / 2^t$ is $a \gg t$ (integer!)

Round down to multiple of 2^t eg with $t=3$, $85 \to 80$

$a \& \sim m3$ or $(a \gg t) \ll t$

⑥ See all bits:

```
for (int i = 31; i >= 0; i--)
    if (((n >> i) & 1) == 1)   print '1'
                else           print '0'.
```