Bin	Hex	Dec
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

128 64 32 16 8 4 2 1

Dec → Bin

122 | 0
61 | 1
30 | 0
15 | 1
7 | 1
3 | 1
1
read ↑

Arithmetic Errors

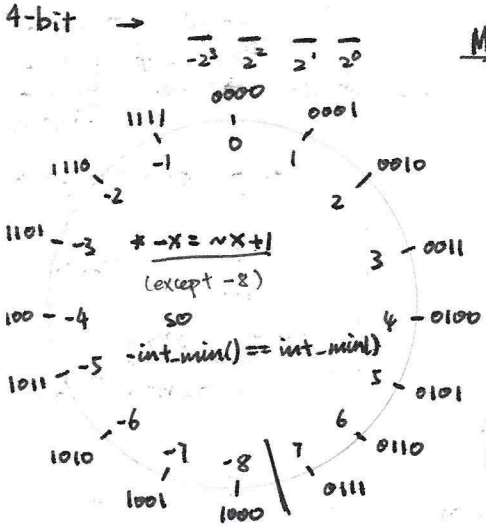
- x/0 or x%0
- int_min() / -1
- x << 32, x >> -1 sth. like that.

Bitwise

& → mask | → overlay
~ → flip ^ → difference(?)

2's Complement

4-bit →



* -x = ~x+1 (except -8)

-int_min() == int_min()

Bits	int_min()	int_max()
8	-128	127
16	-32768	32767
32	-2147483648	2147483647
n	-2^{n-1}	$2^{n-1}-1$

Overflow check

```
assert (int_max() - x >= y)
assert (int_max() / x >= y)
```

Safe avg

x + (y-x)/2

Modular Arithmetic

Always true:

- -(-x) == x
- x + (-x) == 0
- x * (y+z) == x*y + x*z
- x * 0 == 0
- int_min() * -1 == int_min()
- ~~x only when positive~~
- x >> k = $\lfloor \frac{x}{2^k} \rfloor$ * (negative rounding dir!)
- x << k == x * 2^k *

Mod: (x/y)* y + (x%y) == x

$0 \le |x\%y| < |y|$

Division round towards 0!

-5/3 = -1 -5/2 = -2
-5%3 = -2 -5%2 = -1
-5%-3 = -2 -5%-2 = -1

Struct

```
struct foo {
  int bar;
};
```

✓ `struct foo* x = alloc(struct foo);`

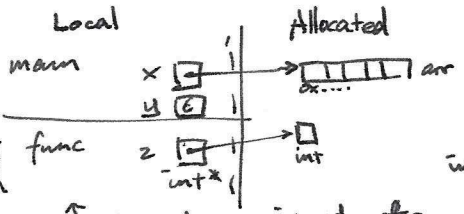
✓ `{ typedef struct foo faa; faa* x = alloc(faa);`

interface type:

`typedef faa* faa_t;`

Applies to C ~~++~~ in certain situations, if not all

Memory Model



↖ Gets decommissioned after func returns.

Stack

in → → out

Queue

in → out

Complexity

T(n) is cost func. Count steps.

$f \in O(g) \Leftarrow \exists n_0 \in \mathbb{N}, \exists c \in \mathbb{R}^+, \forall n \ge n_0, f(n) \le c\,g(n)$

$O(1) \to O(\log n) \to O(n) \to O(n \log n) \to O(n^2) \to O(2^n) \to O(n!)$

Array Util

is_sorted(A, lo, hi)
is_in (x, A, lo, hi)
lt_seg (x, A, lo, hi)
lt_segs (A, lo_1, hi_1, B, lo_2, hi_2)

n	2^n
0	1
1	2
5	32
8	256
10	1024

Rep invariants ↔ (aka) data structure invariants

BEWARE ONLY CHANGING REF INSIDE THE FUNCTION'S FRAME.

Deref

x → data , (*x).data

BigO proof

To show: $n^3 + 9n^2 - 7n + 2 \le Cn^3$

Let no = 1

$n^3 \le n^3$
$9n^2 \le 9n^3$
$-7n \le 0$
$2 \le 2n^3$
↓ add

$n^3 + 9n^2 - 7n + 2 \le 12n^3$

Let C = 12

Contracts

WHERE to LOOK

INIT
- requires
- code before loop

PRES
- Us true initially (assum
- LG
- Loop body

EXIT
- ¬LG
- Lis
- Code after loop body

TERM
- LG
- L1
- Changing vars

> The expression — strictly — at each iteration of the loop and can never become — than the constants — where the loop guard is false.

Lines to use
- Undeclared vars & statements thereabout
- Recent bool expressions
- Contracts
- Assignment in current block of code.

LOOP INVARIANTS CHECKED AFTER FINAL EXECUTION TOO!

STACK - LIFO QUEUE - FIFO

top ← back → | | | | | | → front

Generic types

Don't:
- \#hastag (void*, __);
- void* v = alloc(void);
- *v; ← void* pointer

Function:

`typedef int int_proc_fn (int x);`

(int: return type; int_proc_fn: typename; (int x): input)

REMEMBER "T = _____();" if func returns result rather than change orig.

UBA

	Best	Worst	Amortised
Add	O(1)	O(n)	O(1)
Del	O(1)	O(n)	O(1)
Get	O(1)	O(1)	O(1)

HASH DICT (with good hash fn)

	Best	Worst
Fixed size	$O(\frac{n}{m})$	$O(n)$
UBA resize	$O(1)$	$O(n)$

↑ average amortised.

Some Complexity

	Unsorted UBA arr	Sorted by key	Linked list	Hash table	AVL
Lookup	$O(n)$ lin search	$O(\log n)$ bin search	$O(1)$ lin search	$O(1)$ average	$O(\log n)$
Insert	$O(1)$ amortised	$O(n)$ move things out of way	$O(1)$ put at end	$O(1)$ average amortised	$O(\log n)$
Find min	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(\log n)$

Compilling with 2's complement

gcc -fwrapv

Sort	Best	Worst	Avg	In-place
Quick	nlogn	~~nlogn~~ n^2	nlogn	✓
Merge	nlogn	nlogn	nlogn	X
Selection	n^2	n^2	n^2	
Bubbel	n	n^2	n^2	

Graph complexity

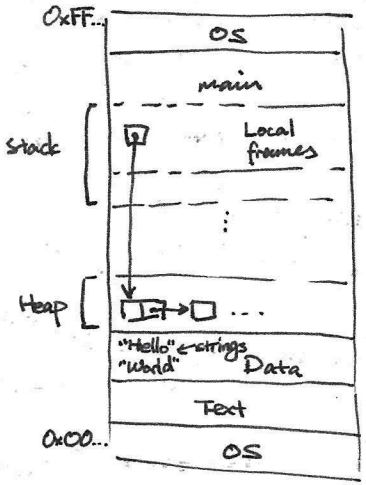
$* O(v+e) = O(\max(v,e))$

	hase	adde	getnbor	BFS/DFS	print	space
Adj	min(v,e)	1	1	v+e	v+e	v+e
Matrix	1	1	v	v^2	v^2	v^2

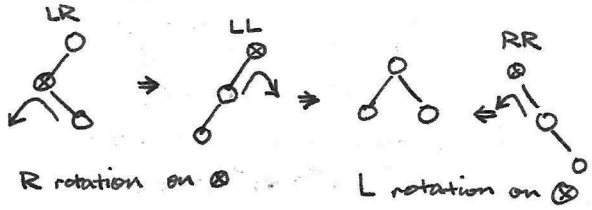
PQ (heap)

add	rem	peek, full, empty, new
logn	logn	1

More Memory



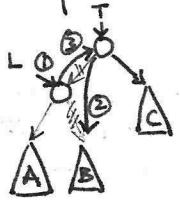
Rotation



Graph

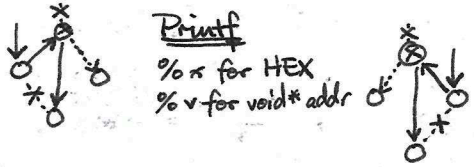
- $e \in [0, \frac{v(v-1)}{2}]$
- acyclic $\#e \leq v-1$
- dense iff $e \propto v^2$

Implementation



Printf

%x for HEX
%p for void* addr



--alloc

xcalloc(,)
xmalloc(*)

Int types [C]

- char — 1 byte
- size of int, short, size_t all ID
- sign/unsigned char ID
  - >> on signed get sign extension
  - >> on unsigned fill 0's.
- Pointer to int cast ID
- signed min max look: 0x80…0, 0x7F…F.

Ranges [C]

	min	max
int8_t	$-128 = -2^7$	$127 = 2^7 - 1$
uint8_t	0	$255 = 2^8 - 1$
int16_t	$-32768 = -2^{15}$	$32767 = 2^{15} - 1$
uint16_t	0	$65535 = 2^{16} - 1$
int32_t	$-2,\ldots,\ldots,648 = -2^{31}$	$2,\ldots,\ldots,647 = 2^{31} -$
uint32_t	0	$9\ldots\ldots = 2^{32} - 1$

Place of allocation [C]

```
char* A = xcalloc(...)        heap
char B[] = {'h','i','\0'};
char[7] B;                    } stack
char* c = "__"                read-only
struct A {__ __};
  struct A a;                 } ← stack
  a.x = __
  a.y = __
```

Int type casting [C]

DON'T PANIC

↓

can represent → Yes: preserve value

No and unsigned
- Smaller/equal: chop off
  - int 0x7FFFFFFE
  - uint8_t 0xFE
- Bigger: sign extended
  - ~~unsigned~~ char 0xFD
  - unsigned int 0xFF…FD
  - * unsigned char 0xFD (actually there)

No and signed: ID (often chop off)

Definedness [C]

Undef:
- read write sth not allocated
- read after free
- Double free
- Free at wrong place
- NULL →
- Out of bound A[i]
- Reading unitialised
- << negative value or too large
- signed int overflow (unless with 2's complement)
- Write to read-only
- / or % by 0
- INT_MIN / or % by -1

Defined:
- unsigned overflow — modular arith.
- free(NULL)