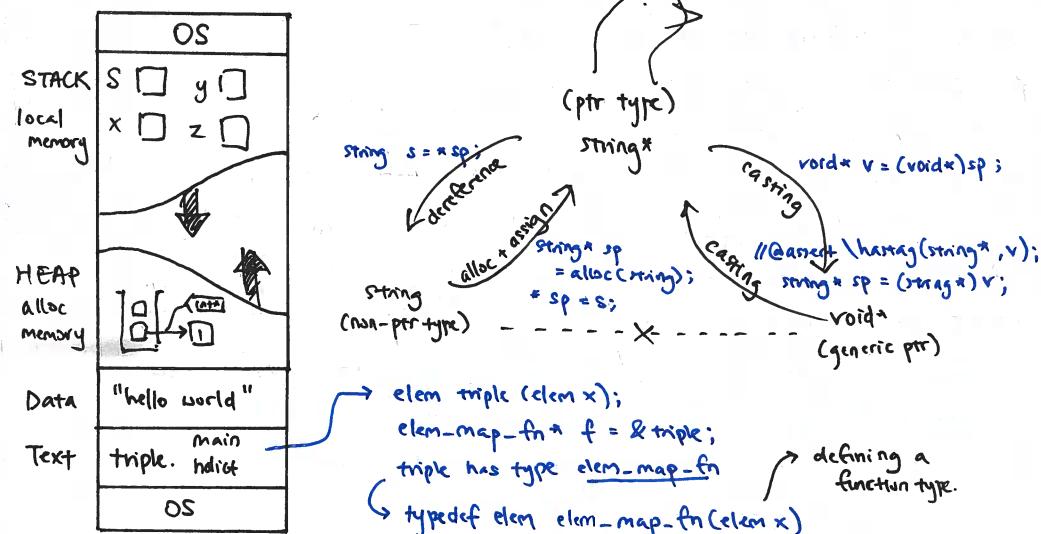


| Dec | Bin | Hex | Two's complement: every k-bit word corresponds to a int between -2^{k-1} and $2^{k-1} - 1$ $>, <, ==$ | |
|-----|------|-----|---|--------------------------------|
| 0 | 0000 | 0 | $\rightarrow \text{int_max}() + 1 == \text{int_min}()$ | need to account for overflow |
| 1 | 0001 | 1 | $\rightarrow -\text{int_min}() == \text{int_min}()$ | $\rightarrow -x == \sim x + 1$ |
| 2 | 0010 | 2 | $x \ll k = x \cdot 2^k$, $x \gg k = \lfloor \frac{x}{2^k} \rfloor$, $0 \leq k < 32$ | |
| 3 | 0011 | 3 | $a_0 2^0 + \dots + a_{31} 2^{31}$ | sign extension! |
| 4 | 0100 | 4 | | |
| 5 | 0101 | 5 | | |
| 6 | 0110 | 6 | | |
| 7 | 0111 | 7 | $\text{int_max}()$ | |
| 8 | 1000 | 8 | $\text{int_min}()$ | |
| 9 | 1001 | 9 | /2 repeatedly, reverse order of remainder | |
| 10 | 1010 | A | $(x/y) = y + (x \% y) = x$, rounds towards 0 | |
| 11 | 1011 | B | $\text{// @ requires } y != 0, !(x == \text{int_min}()) \& y == -1;$ | |
| 12 | 1100 | C | linear search | selection sort |
| 13 | 1101 | D | binary search | mergesort |
| 14 | 1110 | E | $\text{best: } O(1)$ | $O(n^2)$ |
| 15 | 1111 | F | $\text{avg.: } O(n)$ | $O(n \log n)$ |

① inplace: constant amt. of memory allocated

② stable: relative order of duplicate elements doesn't change after sorting



Half-Generic Dict \rightarrow 1 type in 1 file

typedef $_*$ entry \rightarrow client must split code into 2 files

typedef $_$ key

key entry-key (entry e)

int key-hash (key k)

bool key-equiv (key k1, key k2)

Generic Dictionaries

typedef void* entry

typedef void* key

typedef key entry-key-fn (entry e)

typedef int key-hash-fn (key k)

typedef bool key-equiv-fn (key k1, key k2)

① concrete type definition

struct a-header { ... }

typedef struct a-header a;

② representation invariant

bool is-ssa(ssa* A) { ... }

\hookrightarrow used in implementation

③ implementation of interface functions

after client interface & before library interface.

| | unsorted array | array sorted by key | linked list | hash table | bst | AVL | heaps |
|----------|---------------------|---------------------|-------------|------------|--------|-------------|-------------|
| lookup | $O(n)$ | $O(\log n)$ | $O(n)$ | $O(1)$ | $O(n)$ | $O(\log n)$ | $O(1)$ |
| insert | $O(1)$ amortized | $O(n)$ | $O(1)$ | $O(1)$ | $O(n)$ | $O(\log n)$ | $O(\log n)$ |
| find-min | $O(n)$ | $O(1)$ | $O(n)$ | $O(n)$ | $O(n)$ | $O(\log n)$ | $O(1)$ |

AVL: ordering invariant ($L < x < R$) + height invariant ($|h_L - h_R| \leq 1$)

dictionary with n entries, γ load factor is n/m

table with capacity M look-up: worst case $O(n/m)$

collisions unavoidable \rightarrow if $n > m$, pigeonhole principle \rightarrow certainty

if $n > 1$, birthday paradox \rightarrow probabilistic result

INIT: code before loop, preconditions, (loop) initialization

PRES: LI (prev. iteration), Loop guard, code in loop (current iteration)

EXIT: LIs, negation of Loop guard, code after loop

TERM: the expression strictly increases / decreases on each iteration, but cannot go above / below on which the LG is false & loop terminates.

SAFETY: $A[i]$, $0 \leq i \leq n$, $p \neq \text{NULL}$, $y \neq 0$

CORRECTNESS: preconditions imply post conditions, function behaves as expected.

- loop runs constant # of times / each operation has constant rate / each run takes $O(\dots)$ / loop takes time in $O(\dots)$ each iteration
- 1 taken to add element, 1 taken to allocate this position when adding new node.

CO/CI: #use <util>, #use <canio>, header files (macro definitions) compilation

C: #include <stdio.h>, #include "lib/xalloc.h", #define, #if def DEBUG ... #endif

↳ pointer arithmetic: $A[i] \equiv *(A+i)$, $p \rightarrow \text{next} \equiv (*p).next$

Stack-allocated arrays: int E[8], int F[] = {2,4,6} ← s-a structs: access fields with dot notation

| | Writable | allocation | dealloc | |
|-------|----------|----------------|-------------------|---|
| DATA | NO | auto | N/A | char* s1 = "1" struct point q = { .x = 15 } |
| Stack | YES | auto | auto (fn returns) | char* s2 = xmalloc(strlen(s1)+1); strcpy(s2, s1); free(s2); |
| Heap | YES | manual (alloc) | manual (free) | char s3[] = "world"; char s4[] = {'s','l','k','y','\0'} |

sizeof(char) = 1 byte = 8 bits

(new-t) exp → of old-t ① char x = -3 (0xFD)

↓ ② int → char

0xF...FOF → 0x0F

new-t can represent YES value of exp

↓ new-t signed

↓ YES → ID, often discard more signif. bits. ③

new-t larger than old-t YES → bits sign-extended ④

↓ NO → least significant bits retained

enum seasons { A, B, C }; ③ char → uint
switch (A) { 0xFD → 0xF...FD

case A:

break;

default:

printf("");

④ char → uchar
0xFD → 0xFD

BST: ordering invariant

AVL: height + ordering

Heap: shape + ordering

≤ 1 violation



— = maintained.

≤ 2 violations

swap with child is highest priority (smallest for min-heap)

for node i: left child is 2i, right child 2i+1, parent i/2

! unsigned nums: can't compare with 0!

library fn: ensures $IS_stack(S)$! also if add-item, $size++$!

AM → dense (has edge) AL → sparse (sparse)

space O(v²) O(v+e)

hasedge O(1) O(min(v,e))

addedge O(1) O(1)

getnbr O(1) O(1)

iterate over nbrs O(mn(v,e)) O(mn(v,e))

DFS: stack, BFS: queue, A* → priority Q.

implement as a heap* (type of binTree)

implemented as array

highest priority → smallest number for minheaps.

↓ lowest.

* also any priority queue with integer priorities.