Computer Science 50

Introduction to Computer Science I

Harvard College

Week 7

David J. Malan
malan@post.harvard.edu

Valgrind

http://valgrind.org/docs/manual/quick-start.html

```
% valgrind -v --leak-check=full a.out
...
==23596== Invalid write of size 4
==23596==    at 0x80486DF: f (memory.c:22)
==23596==    by 0x80486FC: main (memory.c:29)
...
==23596== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==23596==    at 0x4023595: malloc (vg_replace_malloc.c:149)
==23596==    by 0x80486D5: f (memory.c:21)
==23596==    by 0x80486FC: main (memory.c:29)
```

see
memory.c

Hexadecimal

Image from http://toughpigs.com/labels/fanaticism.html.

Endianness

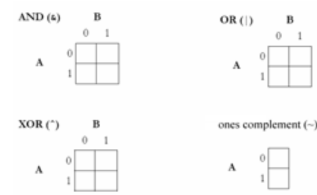
see
endian.c

Image from http://en.wikipedia.org/wiki/Endianness.

Bitwise Operators

- **&** bitwise AND
- **|** bitwise OR
- **^** bitwise XOR
- **~** ones complement
- **<<** left shift
- **>>** right shift

Bitwise Operators



see
binary.c, tolower.c, toupper.c

Bitwise Operators

Swapping Values

```
int FOO = 1;
int BAR = 4;

                // base-2 value in x       base-2 value in y
int x = FOO;    // 001
int y = BAR;    // 001                                   100

x = x ^ y;      // 001 ^ 100                             100
                // 101
y = x ^ y;      // 101                             101 ^ 100
                //                                       001
x = x ^ y;      // 101 ^ 001                             001
                // 100
```

see
swap2.c

Bitwise Operators

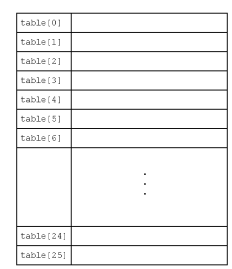
Swapping Values

```
int FOO = 1;
int BAR = 4;

                // value in x                    value in y
int x = FOO;    // FOO
int y = BAR;    // FOO                                  BAR

x = x ^ y;      // FOO ^ BAR                            BAR
y = x ^ y;      // FOO ^ BAR              (FOO ^ BAR) ^ BAR
                //                        FOO ^ (BAR ^ BAR)
                //                                  FOO ^ 0
                //                                      FOO
x = x ^ y;      // (FOO ^ BAR) ^ FOO                    FOO
                // FOO ^ BAR ^ FOO
                // FOO ^ FOO ^ BAR
                // (FOO ^ FOO) ^ BAR
                // 0 ^ BAR
                // BAR
```

see
swap2.c

Hash Tables

Linear Probing



Hash Tables
The Birthday Problem

In a room of *n* CS 50 students,
what's the probability that at least
two students share the same birthday?

Hash Tables
The Birthday Problem

$$\bar{p}(n) = 1\cdot\left(1-\frac{1}{365}\right)\cdot\left(1-\frac{2}{365}\right)\cdots\left(1-\frac{n-1}{365}\right) = \frac{365\cdot 364\cdots(365-n+1)}{365^n} = \frac{365!}{365^n(365-n)!}$$

Image from http://en.wikipedia.org/wiki/Birthday_paradox.

Hash Tables
The Birthday Problem

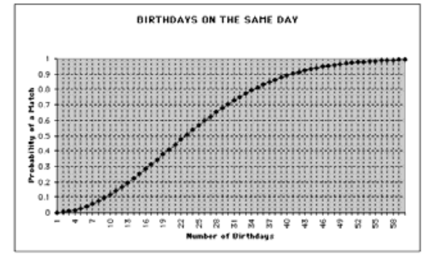


Image from http://www.mste.uiuc.edu/reese/birthday/probchart.GIF.

Hash Tables
Coalesced Chaining

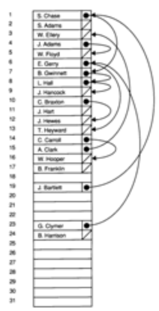


Figure from Lewis and Denenberg's *Data Structures & Their Algorithms.*

Hash Tables
Separate Chaining

Figure from Lewis and Denenberg's *Data Structures & Their Algorithms.*

Trees

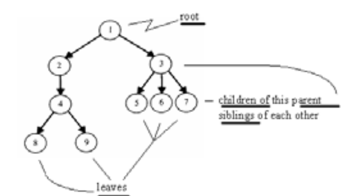


Figure by Larry Nyhoff.

Binary Search Trees

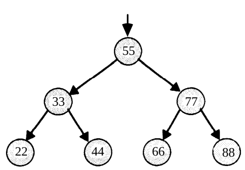


Figure from http://cs.calvin.edu/books/c++/ds/1e/.

Tries

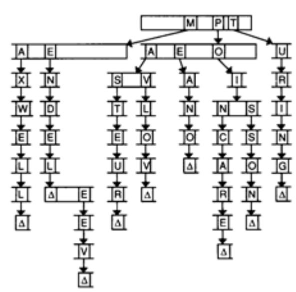
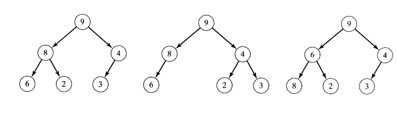


Figure from Lewis and Denenberg's *Data Structures & Their Algorithms.*

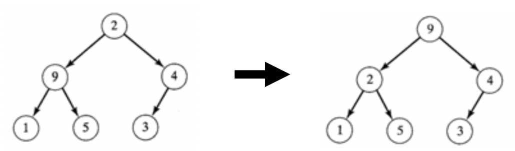
Heaps

- A **heap** is a binary tree that
  - is **complete** (*i.e.*, every level of the tree is completely filled with nodes except for, perhaps, the bottommost level, whose nodes are in the leftmost locations)
  - satisfies the **heap-order property** (*i.e.*, each node's value is greater than or equal to that of each of its children, if any)



Figures by Larry Nyhoff.

Heapifying an Almost Heap



Figures by Larry Nyhoff.

Heapifying a Binary Tree

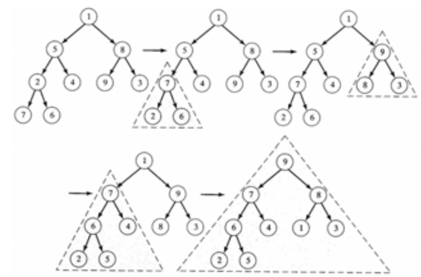


Figure by Larry Nyhoff.

Heapsort

35 15 77 60 22 41

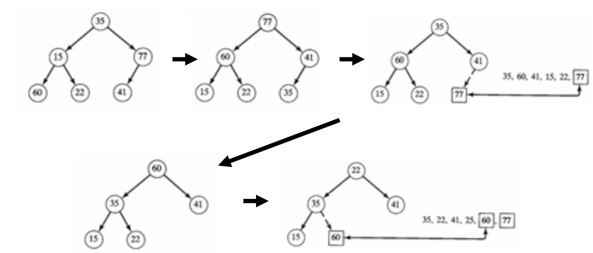


Figure by Larry Nyhoff.

Heapsort

35 15 77 60 22 41

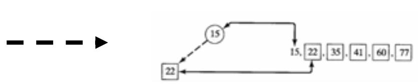


Figure by Larry Nyhoff.

Morse Code

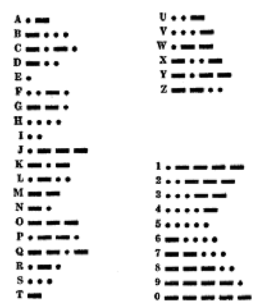
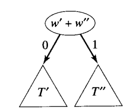


Image adapted from Wikipedia.

Huffman Coding

Immediate Decodability

1) Initialize a list of one-node binary trees containing weights $w_1, w_2, \ldots, w_n$, one for each of the characters $C_1, C_2, \ldots, C_n$.
2) Do the following $n - 1$ times:
   a) Find two trees T' and T'' in this list with roots of minimal weight w' and w''.
   b) Replace these two trees with a binary tree whose root has weight $w' + w''$ and whose subtrees are T' and T''; label the pointers to these subtrees 0 and 1, respectively:



3) The code for character C_i is the bit string labeling the path from root to leaf C_i in the final binary tree.

Pseudocode by Larry Nyhoff.

Huffman Coding

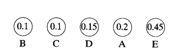
Example

"ECEABEADCAEDEEEECEADEEEEEDBAAEABDBBAAEAAAC DDCCEABEEDCBEEDEAEEEEEAEEDBCEBEEADEAEEDAEBC DEDEAEEDCEEAEEE"

character	A	B	C	D	E
frequency	0.2	0.1	0.1	0.15	0.45

Huffman Coding

Example



Figures by Larry Nyhoff.

Huffman Coding

Example

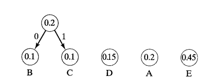


Figure by Larry Nyhoff.

Huffman Coding
Example

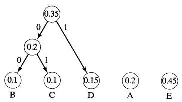


Figure by Larry Nyhoff.

Huffman Coding
Example

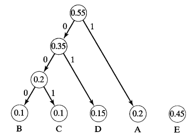


Figure by Larry Nyhoff.

Huffman Coding
Example

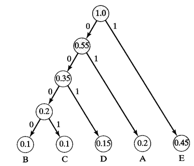


Figure by Larry Nyhoff.

Huffman Coding
Example

Character	Huffman Code
A	
B	
C	
D	
E	

Huffman Coding
Problem?

0 1 0 1 0 1 1 0 1 0

Huffman Coding
In C

```
typedef struct node
{
    char symbol;
    int frequency;
    struct node *left;
    struct node *right;
}
node;
```


Computer Science 50
Introduction to Computer Science I

Harvard College

Week 7

David J. Malan
malan@post.harvard.edu