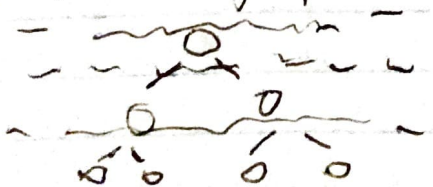
Trees

- tree = directed acyclic graph, composed of nodes
- 1 path between nodes
- Tree = NULL
  | Node Tree Tree } points to 2 trees (rooted binary tree)
- Node = smallest tree entry w/ some value
- Binary tree nodes have up to 2 children
- k-ary tree nodes have up to k children, is binary tree
- subtree = tree rooted at some node n
  - must contain all descendents of n
  - can't have same root as entire tree
- leaf = node w/ no children (is NULL)
- Traversal: ① Preorder ② Inorder ③ Postorder ④ Level Order
  ① visit root, go leftmost, then right, repeat (print at each visit)
  ② visit root, visit to leftmost, print nodes on path to right once leaf reached, repeat
  ③ visit root, visit leftmost, print leaf once reached, visit right, print leaf, repeat (print root last)
  ④ BFS in graph, nodes visited level by level w/ queue



Binary Search Tree (BST)

- ordered tree
- keys < node value go under left subtree
- keys > " " right subtree

IGNORE DUPLICATES!

- Process
  1. create empty tree
  2. find min node (root to leftmost)
  3. find max node (root to rightmost)
  4. find tree height
  5. check if tree balanced → balanced O(log n)
     → unbalanced O(n)
  6. find key in tree → 3 cases
  7. insert new key → 3 cases
  8. remove key → 3 cases
  9. delete tree → postorder traversal (recursively)

⇒ ⑥: - if current node > key, recursively left subtree
  - if " " < key, " " right subtree
  - if " " = key, found ✓

⇒ ⑦: - if current node = NULL, create new node as root
  - if " " > key, recursively add key in left sub
  - if " " < key, " " right sub

⇒ ⑧:- if node to remove is missing left child } replace w/ other child
- if " " right child }
- if " " both children
- find inorder successor, copy value, remove successor

Scripting

- for files & processes
- sh = og Unix shell (Bourne shell)
- other shells - csh, tcsh, bash, etc.
  - act as command interpreters
  - interactive & batched commands to OS
    as prompt or command line — as script or file

Commands: line = entire command
line split into different tokens/words by whitespace
first word on line = command to execute, rest = arguments for command

→ types: aliases → only interactively
builtins → bash provided commands
functions → sequence of commands
executables → specifies executable programs executed by file path

- $PATH = list of directory names
- script = sequence of commands in (executable) file
- first line in Bash script = interpreter directive (#!/bin/bash for example)
  ↳ = hashbang or shebang

- .bashrc = script, executed when interactive
- .bash_profile = script, executed at start of new login shell

- variable = parameter, assigned w/ = sign

- expansion operator is $

- Special Parameters
  - $n nth argument passed to script or function
  - $# # of positional parameters
  - $$ process ID (PID) of current shell
  - $_ last argument of last executed command
  - $? exit status of last executed command

- function = sequence of commands, multiple in 1 script
- array = list of strings, made w/ explicit indices

- Glob matches strings
  - * any str
  - ? any single character
  - [chars] any character in chars
  - [a-z] any character between a & z

testing ints: $x -lt $y <
-le ≤
-eq =
-ne !=
-ge ≥
-gt >

- testing files: -f $file is regular file?
-d is file directory?
-e does file exist?
-r can user read file?
-w " " write file?
-x " " executed file?

- until is valid loop in Bash
- file redirection: stdin file descriptor 0
stdout " " 1
stderr " " 2

- Pipes
  - operator |
  - connects stdout of 1 process to stdin of other process
  - programs written w/ | chained together

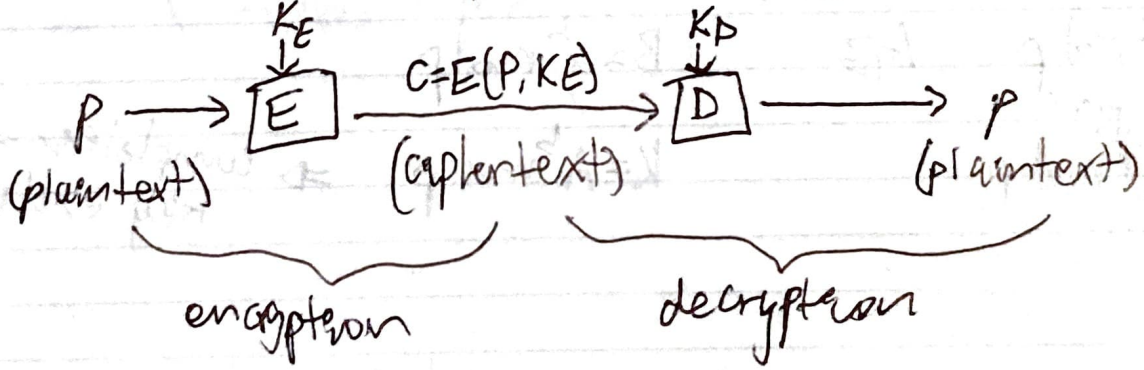
- awk = program for simple parsing of text
  - scans infile, splitting lines into fields automatically
  - output processed further w/ other tools

- Useful commands:

diff	compares files line by line
tee	copies stdin to stdout
sort	sort/merge lines of files
uniq	report/filter duplicate lines
wc	report word/line/byte/char count
mv	move/rename file
cp	copy file
rm	remove file

Cryptography

- goal = make message comprehensible to recipient only
- 2 inputs: data & key (only to authorized users)
- ciphertext = info available to world
- plaintext = known to ppl w/ keys only



K_E = encryption key, K_D = decryption key

- Unbreakable codes: xor message w/ random key a bit at a time
- Data Encryption Standard (DES)
  - 56 bit key
- Current AES algorithms use 128 bit keys

- Energy to flip bit in E: E > kT in 2 Joules

$k = 1,38 \times 10^{-23}$

$T \sim 293$

- Simon block ciphers w/ n bit words & block length 2n used by NSA → examples in slides

Public Key Cryptography

- K_U public & K_R private

- encryption: $E(K_U, E(K_R, M)) = E(K_R, E(K_U, M)) = M$

Diffie-Hellman Key Exchange

Alice	Public Channel	Bob
a	p, g	b
$A = g^a \bmod p$	A, B	$B = g^b \bmod p$
$K = B^a \bmod p$		$K = A^b \bmod p$

p, g = prime & generator both public

⇒ completes secure key exchange

RSA

- large primes p and q
- $n = p \cdot q$
- $\varphi(n) = (p-1)(q-1)$
- select e such that $GCD(e, \varphi(n)) = 1$
- calculate d such that $e \cdot d \equiv 1 \pmod{\varphi(n)}$
- $E(m) = m^e \bmod n$ (encryption)
- $D(c) = c^d \bmod n$ (decryption)

- Attack RSA w/ Generalized Number Field Sieve
- public key requires n (public modulus) & e (public exponent)
- private key requires ints n & d
  - ↳ d itself is the priv key but n is needed for decryption

GMP

- mpz_t = GMP int object
- %zd & %zu decimal int
- mpz_inits() & mpz_clears() must have NULL
- pkg-config --libs <libs> for Makefile