Unix and AFS Concepts

- AFS File permissions
- Commands
  filename globbing, commands, pipes, redirection, stderr, stdin, stdout

\[
\rightarrow \text{wc} \ /\text{usr}/\text{Pool}/* .*
\]
\[
\rightarrow \text{cat} \ /\text{usr}/\text{Pool}/* \mid \text{wc}
\]
\[
\text{man} \ fs
\]
\[
\text{man-K AFS}
\]
\[
\text{wc} \ < \ /\text{usr}/\text{Pool}/*
\]
\[
\text{make} \ > \ \text{nakealog} \rightarrow \text{stdout} \rightarrow \text{stderr} \rightarrow \text{stdout} \rightarrow \text{redirects stdout}
\]
\[
\text{go make} \ > \ & \text{makealog} \rightarrow \text{append (log)}
\]

Makefile Concepts

resolving targets

  target: dep2 dep2 dep3
Java Concepts

exceptions

private, public, protected - why?

Bike

protected gears;

data classes and interfaces

memory allocation

Data Structures.

extend things you are familiar with

- use in new ways

MyClass

\[ \text{MyClass} \]

\[ \text{ MyClass } \]

Class Bag

\[ \text{ Class Bag } \]

private Queue q;

public isEmpty() { return q.isEmpty(); }

public add(\text{item}) { q.enqueue(\text{item}); }

class Bag

\[ \text{ public remove(\text{item})} \]

\[ \text{ Queue \text{tmp} = \text{while} (\text{q.count}() > 0) } \]

\[ \text{ tmp \text{.enqueue}() ; q \text{.dequeue}() } \]

\[ \text{ compose to item} \]

\[ \text{ public remove(\text{item})} \]

\[ \text{ Queue \text{tmp} = \text{while} (\text{q.count}() > 0) } \]

\[ \text{ tmp \text{.enqueue}() ; q \text{.dequeue}() } \]

\[ \text{ compose to item} \]
Stacks -
common stack operations

$3 \times 2 + 4 \frac{2}{3} - 2$

See number -
push on stack

 operators come after operands

operator -
pop 2 things

use operator

push answer right back

$\begin{align*}
4 & \quad + & \quad 2 & \quad 3 & \quad \frac{3}{2} & \quad 6 & \quad \frac{7}{2} \\
3 & \quad 12 & \quad 14 & \quad 14 & \quad \frac{14}{8}
\end{align*}$

Queues -
how to operate, array & reference implementations
draw ref. implementations

Trees -
orders of traversal

Nomenclature

binary trees - data structures, addition & deletion

Hash Tables
- hash functions
  integers, strings
- chaining
  other methods
Evaluating speed of Algorithms

time is a function of input size

eq. list(n)

\[ \text{time} = f(n) = n^2 \]

10 (100)

100 (10,000)

\[ \therefore \]

Average performance

\[ \text{e.g. list}(n) \]

sequential search

\[ \frac{1}{2}n = \frac{n}{2} \]

\[ f(n) = c \]

O(\_)

Big-Oh notation

O(f) \rightarrow \text{function grows no faster than } f.

O(n^2)

O(n)

O(2^n)
\[ f_3(x) > f_4(x), \quad x > y \]

- \( f_3 \) is \( O(x) \)
- \( f_4 \) is \( O(x) \)

1. Small values greater
2. Two functions \( O(-) \)
   - but one is greater
   - constant factors do not matter

Do not assume:

\[ O(n^2) > \frac{1}{n} \text{ NO!} \]

\( O(n) \) is faster

\[
\begin{array}{c|c|c|c|c}
\text{Input Size} & 33n & 13n^2 & 2^n & \frac{1}{1,000,000} \\
10 & .00033 \text{ms} & .0013 & \text{centi} & \\
100 & & .0013 & \text{sec} & \\
1,000 & & 13 \text{sec} & & \\
10,000 & & & \text{centi} & \\
100,000 & 3.3 \text{sec} & 1.5 \text{days} & & \\
\end{array}
\]

Calculates time in \( \mu \text{secs} \)