COS 126 **Spring, 1999**

COS 126 Main Objectives

- Programming
  - programming skills universal, same basic features found in many languages (C, Java, PostScript, Maple, Matlab, TeX, HTML)
  - can address interesting and important problems with basic skills and without relying on "packaged" solutions
  - fundamental programming tools: array, linked list, stack, queue, tree, ADT, binary search, recursion, divide-and-conquer
- TOY and machine language
  - von Neumann machine
- How is a machine built?
  - use layers of abstration
  - fundamental building block = switch (transistor, relay, vacuum tube)
  - machine sees only 0's and 1's $\Rightarrow$ need to understand Boolean functions
  - build Boolean circuits, decoder, multiplexer, memory bit from AND, OR, NOT gates
  - build arithmetic circuits (adder) using Boolean circuits
  - incorporate time with sequential circuits
- How powerful is my machine?
  - formal languages used to describe abstract machines (FSA, PDA, Turing machine)
  - deterministic vs. nondeterministic machine
  - Chomsky hierarchy delineates fundamental machine-grammar relationship and classifies machines according to power
  - TOY and everyday computers equivalent to Turing machine
  - all abstract and real machines have fundamental limitations
- What is an algorithm?
  - Church-Turing thesis says intuitive notion of algorithm is a Turing machine
  - some problems unsolvable even on Turing machine
- How good is my algorithm?
  - complexity, polynomial vs. exponential
  - NP-completeness and intractability, P $\neq$ NP conjecture
- Systems programming
  - machine language - 0's and 1's
  - assembly language - symbolic variables (use BST for symbol table)
  - compiler translates from C to machine language (uses grammar)
  - interpreter emulates one machine on another (reuse old programs)
  - multiprogramming and windows (single machine simulating many machines)