Building Software

- Process in software development:
  - define problems (specify requirements, I/O)
  - develop (design and analyze) algorithms
  - implement, test and evaluate software

Software Engineering

Programming

Core of Software: Algorithms

- Algorithm ~ a sequence of computer instructions to solve a given problem

\[ \text{input} \xrightarrow{\text{Data Structures}} \text{Algorithm} \xrightarrow{\text{output}} \]

- A problem may have none or more than one algorithmic solution \( \rightarrow \text{Theory of Computation} \)

- Some algorithmic solution may take far too long to compute \( \rightarrow \text{Analysis of Algorithms} \)

Why learn about algorithms?

- Core of computation \( \rightarrow \)
  - CS areas:
    - Hardware/System - ?
    - Security - ?
    - Software - ?
    - Network/AI/DB - ?
  - Non-CS areas:
    - Biology -?
    - Other engineering - ?
    - Manufacturing - ?

Why analyze algorithms?

- To know if algorithm is correct
  \textit{What does “correct” really mean?}

- To understand when finding algorithmic solutions with certain properties can be infeasible

- To understand resources required
  \textit{What are types of resources?}

- To be able to compare performance among algorithms so that we can select the most appropriate solution
  \textit{Do we always pick the most efficient algorithm?}
Why learn about algorithm design?

- Different designs result in different algorithms
- You may need to solve new problems that require brand new design

Example 1: Power of SW vs. HW

- Problem: To sort $n$ numbers
  - Algorithm $A$: requires $2n^2$ instructions
  - Algorithm $B$: requires $50n \ln n$ instructions
  - Super computer $S$: executes $100 \times 10^6$ instructions/sec
  - PC $P$: executes $10^6$ instructions/sec

  Which is Faster: $A$ or $B$?
  Which is Better: $A&S$ vs. $B&P$?

  Say, $n = 10^6$ numbers

  $A&S \text{ Time} = \frac{2(10^6)}{10} \text{ instrs} \approx 2 \times 10^8 \text{ sec} \approx 5.56$ hours

  $B&P \text{ Time} = \frac{50 \times 10^6 \times \ln(10)}{10} \approx 50 \times 6 \times 3 \approx 10^3$ sec \approx 16.7 mins

Lesson learned

- Total performance depends on choosing efficient algorithms just as much as choosing fast hardware!!!
Cosmic Time Scale

- **0 sec**: Big bang
- **$10^{18}$ sec**: Now
- **$10^{24}$ sec**: Stars lose planets
- **$10^{26}$ sec**: All stars burned out
- **$10^{39}$ sec**: Protons decay; solid matter vanishes
- **$10^{71}$ sec**: Black hole starts to vanish
- **$10^{100}$ sec**: End of everything – last black holes vanish

- **GUT freezing**: strong forces separate out
- **Electro magnetic force separate from weak force**
- **Quark $\rightarrow$ Protons $\rightarrow$ Nuclei $\rightarrow$ Atoms $\rightarrow$ Stars, Planets

- 10 sec is a very long time!
- Even if we could get each operation done faster, say 10 operations/sec, operation takes less time than light travels in one angstrom unit ~ 10^{-10} meters – which you can’t!
- $B$ still takes ~ 10 / 10 = 10 sec .... by then your computer will disintegrate
- But $A$ takes only 10 operations ~ 10 sec which is fine

Lesson learned

- Don’t underestimate the power of SW
- The importance of this course!

Understanding your problems

- **Computational Problems**
  - **Tractable problems**
    - Realistically computable
    - Polynomial time
    - Example: Sorting problem
  - **Intractable problems**
    - No algorithmic solution
    - Theoretically computable but not realistically computable
    - Exponential time or no algorithm
    - Example: Traveling Salesman problem

Problems Characteristics

- Knowledge intensive
- Incomplete data
- Uncertain situations
- Combinatorial explosive choices that may lead to solutions

Can we understand Algorithmic Solution before Implementing it?