Algorithms II, CS 502

Algorithms Basics

Ugur Dogrusoz
Computer Eng Dept, Bilkent Univ
What is an Algorithm?

- Procedure that always halts with a correct solution to the problem at hand
Why study Algorithms?

- Analyze performance to determine “feasible vs. not”
- Algorithmic mathematics (e.g. big-O notation) allows comparing performance of two algorithms for the same problem
- Build a repertoire of algorithms for future use
- Learn various algorithm design paradigms and apply to new problems
Kinds of analyses

- **Worst case (usually):**
  - $T(n) =$ maximum time it takes for an algorithm for *any* input of size $n$

- **Average case (sometimes):**
  - $T(n) =$ expected time of algorithm over all inputs of size $n$
  - Need to know statistical distribution of inputs
  - Harder

- **Best case (rarely):**
  - Can always cheat with a slow algorithm that works fast on *some* input
Asymptotic notation

- Use for running time or memory requirement analysis
- Ignore machine-dependent constants, look at growth in $T(n)$ as $n$ goes to infinity
- When input size gets large enough, a quadratic algorithm always beats a cubic one
O-notation

Formally

\[ O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that} \]
\[ 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}. \]

Informally

- drop low order terms, ignore leading constants to form an upper bound

\[ 3n^3 + 90n^2 - 5n + 6046 = O(n^3) \]
\( \Omega \)-notation

- Formally

\[
\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq c g(n) \leq f(n) \text{ for all } n \geq n_0 \}.
\]

- Informally

- drop low order terms, ignore leading constants to form a lower bound

\[
3n^3 + 90n^2 - 5n + 6046 = \Omega(n^3)
\]
\( o \)- and \( \omega \)-notation

**Strict versions of \( O \) and \( \Omega \)**

\[
3n^3 + 90n^2 - 5n + 6046 = O(n^3)
\]

\[
3n^3 + 90n^2 - 5n + 6046 \neq o(n^3)
\]

\[
3n^3 + 90n^2 - 5n + 6046 = o(n^{3.01})
\]

\[
3n^3 + 90n^2 - 5n + 6046 = \Omega(n^3)
\]

\[
3n^3 + 90n^2 - 5n + 6046 \neq \omega(n^3)
\]

\[
3n^3 + 90n^2 - 5n + 6046 = \omega(n^{2.99})
\]
Θ-notation

Formally

\[ \Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \} . \]

Informally

- drop low order terms, ignore leading constants to form a tight (both lower and upper) bound

\[ 3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3) \]
Algorithm design paradigms

- Divide-and-conquer
- Dynamic programming
- Greedy
- Branch-and-bound
- ...

Ugur Dogrusoz
Methods for running time complexity

- **Master Method**
  - Applies to limited types of algorithms

- **Substitution Method**
  - Difficult to make the guess that works
  - Might not work (lead to induction that works)

- **Recursive Tree Method**
  - Difficult to get tight complexity
Example: Fibonacci numbers

- Calculate $n^{th}$ Fibonacci number
  - $F_0=0$, $F_1=1$, $F_i=F_{i-1}+F_{i-2}$ for $i \geq 2$

- Divide-and-conquer solution
  - Running time?
  - How to improve?