CSE 417: Algorithms and Computational Complexity

2: Complexity

Winter 2002

Lecture by: Bill Pentney

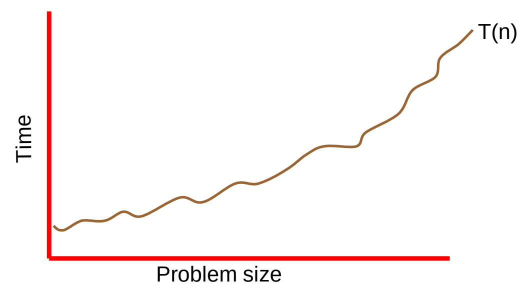
Complexity analysis

- Problem size n
  - **Worst-case complexity**: **max** # steps algorithm takes on any input of size n
  - **Best-case complexity:** **min** # steps algorithm takes on any input of size n
  - **Average-case complexity**: **avg** # steps algorithm takes on inputs of size n

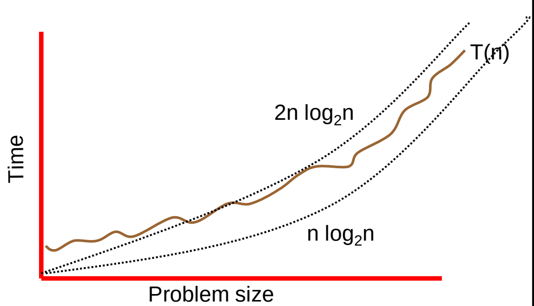
Complexity

- The complexity of an algorithm associates a number **T(n)**, the best/worst/average-case time the algorithm takes, with each problem size **n**.
- Mathematically,
  - **T: $N^+ \rightarrow R^+$**
  - that is **T** is a function that maps positive integers giving problem size to positive real numbers giving number of steps.

Complexity



Complexity



O-notation etc

- Given two functions **f** and **g:N→R**
  - **f(n)** is **O(g(n))** iff there is a constant **c**>0 so that **c g(n)** is eventually always $\geq$ **f(n)**
  - **f(n)** is **Ω(g(n))** iff there is a constant **c**>0 so that **c g(n)** is eventually always $\leq$ **f(n)**
  - **f(n)** is **Θ(g(n))** iff there is are constants c_1 and c_2>0 so that eventually always $c_1 g(n) \leq f(n) \leq c_2 g(n)$

Examples

- **$10n^2$-16n+100** is **$O(n^2)$** also $O(n^3)$
  - $10n^2-16n+100 \le 11n^2$ for all $n \ge 10$
- **$10n^2$-16n+100** is **$\Omega(n^2)$** also $\Omega(n)$
  - $10n^2-16n+100 \ge 9n^2$ for all $n \ge 16$
  - Therefore also **$10n^2$-16n+100** is **$\Theta(n^2)$**
- **$10n^2$-16n+100** is **not $O(n)$** also **not** $\Omega(n^3)$
- **Note:** I don't use notation **f(n)=O(g(n))**

7

Domination

- **f(n)** is **o(g(n))** iff **$\lim_{n\to\infty}$ f(n)/g(n)=0**
  - that is **g(n)** **dominates** **f(n)**
- If $\alpha \le \beta$ then **n^α** is **$O(n^\beta)$**
- If $\alpha < \beta$ then **n^α** is **$o(n^\beta)$**
- **Note:** if **f(n)** is **Θ(g(n))** then it cannot be **o(g(n))**

8

Working with O-Ω-Θ notation

- Claim: For any a, b>1 $\log_a n$ is $\Theta(\log_b n)$
  - $\log_a n = \log_a b \log_b n$ so letting $c=\log_a b$ we get that $c\log_b n \le \log_a n \le c\log_b n$
- Claim: For any a and b>0, $(n+a)^b$ is $\Theta(n^b)$
  - $(n+a)^b \le (2n)^b$ for $n \ge |a|$
    $= 2^b n^b = cn^b$ for $c=2^b$ so $(n+a)^b$ is $O(n^b)$
  - $(n+a)b \ge (n/2)^b$ for $n \ge 2|a|$
    $=2^{-b} n^b = c'n$ for $c=2^{-b}$ so $(n+a)^b$ is $\Omega(n^b)$

9

General algorithm design paradigm

- Find a way to reduce your problem to one or more smaller problems of the same type
- When problems are really small solve them directly

10

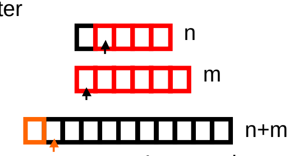
Example

- Mergesort
  - on a problem of size at least 2
    - Sort the first half of the numbers
    - Sort the second half of the numbers
    - Merge the two sorted lists
  - on a problem of size 1 do nothing

11

Cost of Merge

- Given two lists to merge size n and m
  - Maintain pointer to head of each list
  - Move smaller element to output and advance pointer



Worst case n+m-1 comparisons
Best case min(n,m) comparisons

12

Recurrence relation for Mergesort

- In total including other operations let's say each merge costs 3 per element output
- $T(n)=T(\lceil n/2 \rceil)+T(\lfloor n/2 \rfloor)+3n$ for $n \geq 2$ ("ceiling" round up, "floor" round down)
- $T(1)=1$
- Can use this to figure out T for any value of n
  - $T(5)=T(3)+T(2)+3x5$
    $=(T(2)+T(1)+3x3)+(T(1)+T(1)+3x2)+15$
    $=((T(1)+T(1)+6)+1+9)+(1+1+6)+15$
    $=8+10+8+15=41$

Insertion Sort

- For i=2 to n do
  j←i
  while(j>1 & X[j] > X[j-1]) do
  swap X[j] and X[j-1]

- i.e., For i=2 to n do
  Insert X[i] in the sorted list
  X[1],...,X[i-1]

May need to add extra conditions - Insertion Sort

- Original problem
  - **Input:** $x_1,...,x_n$ with same values as $a_1,...,a_n$
  - **Desired output:** $x_1 \leq x_2 \leq ... \leq x_n$ containing same values as $a_1,...,a_n$

- Partial progress
  - $x_1 \leq x_2 \leq ... \leq x_i, x_{i+1},...,x_n$ containing same values as $a_1,...,a_n$

Recurrence relation for Insertion Sort

- Let **T(n,i)** be the **worst case cost** of creating list that has first **i** elements sorted out of **n.**
  - We want **T(n,n)**
- The insertion of **X[i]** makes up to **i-1** comparisons in the worst case

- **T(n,i)=T(n,i-1)+i-1** for **i>1**
- **T(n,1)=0** since a list of length 1 is always sorted
- Therefore **T(n,n)=n(n-1)/2** (next class)