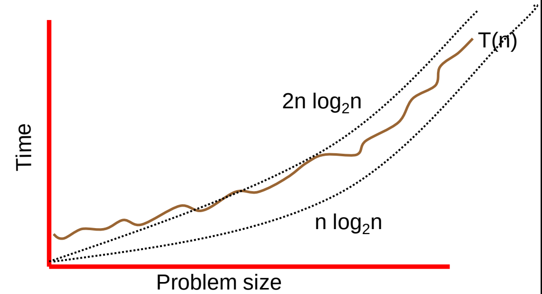
CSE 417: Algorithms and Computational Complexity

3: Complexity (cont.)

Winter 2002
W. L. Ruzzo

Complexity



O-notation etc

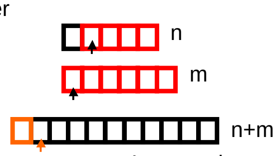
- Given two functions **f** and **g:N→R**
  - **f(n)** is **O(g(n))** iff there is a constant **c**>0 so that **c g(n)** is eventually always ≥ **f(n)**
  - **f(n)** is **Ω(g(n))** iff there is a constant **c**>0 so that **c g(n)** is eventually always ≤ **f(n)**
  - **f(n)** is **Θ(g(n))** iff there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

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

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

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 $n \ge 2$ ("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]

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**

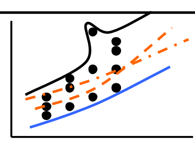
Solving recurrence relations

- e.g. T(n)=T(n-1)+f(n) for n ≥ 1
  T(0)=0
  - solution is T(n)= $\sum_{i=1}^{n} f(i)$
- Insertion sort: $T_n(i)=T_n(i-1)+i-1$
  - so $T_n(n)= \sum_{i=1}^{n}(i-1) = n(n-1)/2$

Arithmetic Series

- S= 1 + 2 + 3 + ... + (n-1)
- S= (n-1)+(n-2)+(n-3)+ ... + 1
- 2S=n + n + n + ... + n {n-1 terms}
- 2S=n(n-1) so S=n(n-1)/2
- Works generally when f(i)=ai+b for all i
- Sum = average term size x # of terms

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

Why Worst-Case Analysis?

- Appropriate for time-critical applications, e.g. avionics
- Unlike Average-Case, no debate about what the right definition is
- Analysis often easier
- Result is often representative of "typical" problem instances
- Of course there are exceptions…