Computational Complexity

1. Time Complexity
2. Space Complexity

Measuring the Performance of algorithm

Two important measures

1. Time Complexity
2. Space Complexity

Running Time of Algorithm

= How much time the algorithm uses in terms of input size

= # of certain operations used in the algorithm where each operation take a constant time. Like: multiplications, additions, comparisons, assignments, shifts, …etc

≠ # of seconds or minutes used by algorithm, because this depends on machine and technology (O.S., Prog. language, …etc)

Input Size vs. Input Type

- The running time is expressed in terms of input size and we concentrate our analysis on large inputs.
- Input types (arrays, lists, strings, integers, ..etc) is not an issue here.

Algorithm Running Time

- Should be machine & technology independent.
- Should concentrate on the asymptotic times (large input size).
- Should concentrate on the main largest term (order of growth) and ignore the smaller ones.
- Sometimes we may even ignore the 1st constant (multiplicative) factor.
- The constant factors or the other smaller terms are important when comparing two algorithms of the same order of running time.
- Asymptotic Notations are used to describe asymptotic behavior of algorithm.

Examples of Running Times (RT)

- Worst-case RT of Linear Search = $\theta(n)$
- Worst-case RT of Binary Search = $\theta(\log n)$
  - Average however is still = $\theta(\log n)$
- RT of Selection Sort = $n(n-1)/2 = \theta(n^2)$
- RT of Insertion Sort = $\Omega(n)$ and $O(n^2)$
  - Average however is still = $\theta(n^2)$
- RT of Bottom Up Merge Sort = $\theta(n \log n)$
- # of comparisons = RT

Complexity of Running Time

- Could be
- Logarithmic = $\theta(\log n)$
- Linear = $\theta(n)$
- Quadratic = $\theta(n^2)$
- Cubic = $\theta(n^3)$
- Polynomial = $\theta(n^k)$ … all are efficient.

How to compute the RT

- Ugly: by going through the code & counting iterations and operations
- Beautiful: by doing smart abstract analysis based on the idea of the algorithm

Space Complexity

- Is defined to be the extra space used by the algorithm beside the space allocated to hold the input
- I.e., it is the work space used by the algorithm measured by the number of cells or words.

Examples

- Linear Search uses θ(1) (extra) space
- And so is Binary Search, ,Selection Sort, and Insertion Sort
- Merge Sort however uses θ(n) extra space

Optimal Algorithms

- These are algorithms whose worst-case performances meet the best-case performance of any algorithm that solves the same problem
- Optimal RT = min { RT of A : A algorithm
  solves the problem}

Example

- Merge Sort is optimal among all comparisons-based sorting algorithms, because it uses $\theta(n \log n)$.
- Theorem: Any comparisons-based sorting algorithm uses $\Omega(n \log n)$ cmps.
- There are however other non-cmp-based sorting algorithm that do better.

Worst-case vs. Average-case

- Which one is better
  - low worst-case with high average or
  - low average with high worst-case?
- The average is the average!

Input Size = n

- Sorting & searching: n = # of elements in the array
- Graphs: n, m = # of vertices & edges
- Integer Multiplications: n = # of bits
- Cryptography: n = # of bits