Design and Analysis of Algorithms

L05: Analysis Framework

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

- Text book 1: Levitin

Algorithm Analysis

- Analysis
  - detailed examination of the elements or structure of something, typically as a basis for discussion or interpretation.
  - *Mathematics:* the part of mathematics concerned with the theory of functions and the use of limits, continuity, and the operations of calculus.
- Analysis of algorithms:
  - Investigation of an algorithm's efficiency w.r.t. running time and memory space.

Algorithm Analysis

- Issues:
  - correctness
  - time efficiency
  - space efficiency
  - optimality
- Approaches:
  - theoretical analysis
  - empirical analysis
- From pactical point of view:
  - Efficiency concerns are primary
  - Space (memory) is no more an issue today
    - Difference: secondary, primary, cache
  - We will mostly study time efficiency

Time vs Space Efficiency

- Consider multiplication of any two digit numbers
  - Example: Multiply 79 * 67
- Time efficiency requires computation
  - Number of multiplications (single digit): 4
  - Number of additions (single digit): 5
  - Total mathematical operations: 9
  - Total space requirement: 14 digits
- Space (memory) efficiency
  - Store all multiplication values in an 100x100 array.
  - Then do a lookup.
  - Time requirement: 2 lookup
  - Space requirement: 10000 memory locations

Measuring Input Size

- Representing a polynomial:
  - $P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$
  - I/p size: n+1
    - or n (+1 is fixed and ignored, inconsequential)
- At times choice of parameter specifying the input size is important
  - Consider nxn matrix multiplication
  - Is input size n or n^2?
  - If considering matrix order: size is n
  - If consider number of elements: size is n^2
  - Latter is preferred as it works for nxm matrix too

Input Size and Operation Examples

Problem	Input size measure	Basic operation
Searching for key in a list of n items	Number of list's items, i.e. n	Key comparison
Multiplication of two matrices	Matrix dimensions or total number of elements	Multiplication of two numbers
Checking primality of a given integer n	number of digits (in binary representation)	Division
Typical graph problem	number of vertices and/or edges	Visiting a vertex or traversing an edge

Empirical Analysis of Time Efficiency

- Select a specific (typical) sample of inputs
- Choices:
  - Physical unit of time (e.g., milliseconds), or
  - Count actual number of basic operation's executions
- Physical unit of time varies depending upon computer being used, operations available
  - Count of basic operations is considered.
    - This is further approximated, need not be actual value
- Analyze the empirical data

Theoretical Analysis: Time Efficiency

- Time efficiency is analyzed by determining the number of repetitions of the *basic operation* as a function of *input size*
- Basic operation: the operation that contributes the most towards the running time of the algorithm
    - $T(n) \approx c_{op} C(n)$
        - c_{op} is cost of operation
        - $C(n)$ represents number of operations

Time Efficiency

- Consider sum of n numbers
  - Total sum operations $C(n) = n(n+1)/2 \approx n^2$
  - $T(n) = c_{op}C(n) = c_{op}n^2$
  - $T(2n) = c_{op}C(2n) = c_{op}(2n)^2 = c_{op}4n^2$
  - $T(2n)/T(n) = 4$
  - $T(10n)/T(n) = 100$
- Summary: c_{op} does not play a role when comparing the performance on two inputs.
  - **Order of growth** is more important.
    - Lower order terms and constants multiple are not important
    - These are subsumed in **order of growth.**

Best, Worst and Average case

- For some algorithms, efficiency depends on form of input:
- Worst case: $C_{worst}(n)$ – max over inputs size n
- Best case: $C_{best}(n)$ – min over inputs of size n
- Avg case: $C_{avg}(n)$ – "average" over inputs of size n
  - Number of times the basic operation will be executed on typical input
  - NOT the average of worst and best case
  - Expected number of basic operations considered as a random variable under some assumption about the probability distribution of all possible inputs.
  - So, avg = expected under uniform distribution.

Example: Sequential Search

- Algorithm: SequentialSearch(A[0..n-1],K)

```
// searches for a value K in a given array by sequential searching
// Input:  an array A[0..n-1], and key K
// Output: Index of the element that matches K
//            -1, if element is not found
i←0
while i < n and A[i] ≠ K do
    i ← i+1
if i < n then
    return i
else
  return -1
```

- What is Worst, Best and Average case analysis?

Sequential Search: Avg Case Analysis

- Let `p` be the probability of finding key `K`
  - Thus, probability of not finding `K` is `(1-p)`
- Probability of finding `K` at position `i` is same i.e. `p/n`
- The exepected number of searches $C_{avg}(n)$ is given by

  `1*p/n+2*p/n+...+n*p/n+(1-p)*n`

  `= (p/n)(1+2+...+n) + (1-p)*n`

  `= (p/n)*n(n+1)/2 + (1-p)*n`

  `= p*(n+1)/2 + (1-p)*n`

- When `p=1` i.e. search is always successful

  $C_{avg}(n) = (n+1)/2 \approx n/2$

- When `p=0` i.e. search is failure

  $C_{avg}(n) = n$

Sequential Search Analysis

- Computation of average case analysis is lot more complex than best case and worst case.
- For this specific problem

  $C_{avg}(n) = (C_{worst}(n) + C_{best}(n))/2$

  - This is not true in general and can't be simplified this way. Not a legitimate way.

Amortised Efficiency

- So far, efficiency is related to a single run of algorithm.
- In some cases, single run can be very expensive, but subsequent run can be much cheaper
  - Real life example:
    - To drink water, dig a well.
      - First time very costly, subsequently minimal cost
    - Giving a lecture first time on new topic
      - Subsequent lectures on same topic much easier
- Thus, amortise the cost over n operations

Order of Growth

n	$\log_2 n$	$\sqrt{n}$	n	$n\log_2 n$	n^2	n^3	2^n	n!
10	3.3	3.16	10	33.2	10^2	10^3	10^3	$3.6*10^6$
10^2	6.6	10	10^2	664	10^4	10^6	$1.3*10^{30}$	$9.3*10^{157}$
10^3	10	31.6	10^3	9965	10^6	10^9		
10^4	13.2	100	10^4	$1.3*10^5$	10^8	10^{12}		
10^5	16.5	316.2	10^5	$1.6*10^6$	10^{10}	10^{15}		
10^6	20	1000	10^6	$2.0*10^7$	10^{12}	10^{18}		

Summary : Analysis Framework

- Both time and space efficiencies are measured as functions of the algorithm's input size
- Time efficiency is measured by counting the number of times the base operation of algorithm is executed.
- Space efficiency is measured by counting the number of extra memory units consumed by algo.
- Efficiency for same algorithm may vary significantly for inputs of same size.
  - Worst case, Best case, and Average case
- Framework primary interest in order of growth
  - Running time of algorithm

Exercises - A

- For the following algorithms (problems), identify
  - Natural size metric
  - Basic operation
  - Count of basic operation (average case)

- P01: Computing sum of n numbers
- P02: Computing factorial(n)
- P03: Finding largest element of n numbers
- P04: List all prime numbers <n using Sieve method
- P05: Multiplying 2 numbers each of n digits

Exercises-B

- Glove selection: There are 22 gloves in a drawer: 5 pairs of red gloves, 4 pairs of yellow, and 2 pairs of green. You select the gloves in the dark and can check them only after a selection has been made. What is the smallest number of gloves you need to select to have (Hint: Gloves are left and right side)

  - At least one matching pair? (worst case)
  - At least one matching pair in the best case?
  - At least one matching pair of each color? (worse case)

Exercises-C

Missing socks: Imagine that after washing 5 distinct pairs of socks, you discover that two socks are missing. Of course, you would like to have the largest number of complete pairs remaining. Thus, you are left with 4 complete pairs in the best-case scenario and with 3 complete pairs in the worst case. Assuming that the probability of disappearance for each of the 10 socks is the same, find the probability of

- The best-case scenario;
- The worst-case scenario;
- The number of pairs you should expect in the average case.

.