Overview

- MOTIVATION

- ALGO DEF.

- ANALYSIS - Seq. Search
- TIME
  WORK DONE
  WORST, AV CASE

  SPACE

- OPTIMALITY
- ORDER NOTATION
- Design - Bin Search.

- MOTIVATION

  — Algorithms: How to do things on a computer?

  — Other application areas study specialized algorithms.

  e.g.
  Operating Systems
  Compiler Design
  Artificial Intelligence
  Data Base Management Systems

  — Algorithm Design Techniques
  Divide and conquer, greedy, Dynamic Programming etc.

  — Basic Algorithms
  sorting, graph algo, matrix multiplication, NP-Complete problems, parallel algos.

- HISTORY
  Phrase Algorithm; Persian Author

  — Abu Jafar Mohammed ibn Musa *al Khowarizmi* (825 AD)

  — wrote a Math textbook

  — al Khowarizmi: from the town of Khowarazm (now Khiva, Uzbekistan)

- ALGORITHM DEFINITION
  An algorithm is composed of a finite number of steps, each of which may require one or more operations.

  — each OPERATION executable on a computer.

  — compute 5/0 is not WELL DEFINED.

  — algorithms must TERMINATE after a finite number of operations.

  — PROCEDURE: a nonterminating algorithm.
  e.g. OS.

- GOAL - Distinguish Algorithms.

- TIME - Computer Dependent.
  AMOUNT OF WORK - COST

  — proportional to the number of **basic operations**.

  — computer independent.

  e.g. matrix multiplication
  basic operation - multiplication & addition.

Example Problem: Sequential Search

Problem: Given an array $L[1..n]$, containing n DISTINCT entries, find the index of x, if $x \in L$, else return 0.

Input: array L, array size n, search item x.

Output: If $x \in L$ then index of x in L, else 0.

Algorithm: .
Abstract Level Description: Scan L left to right looking for x in L & return index.

Pseudocode

1. index $\leftarrow 1$
2. **While** index $\leq n$
3.   **if** $L[\text{index}] = x$
4.   return (index)
5.   **else**
6.   index $\leftarrow$ index $+ 1$
7. return (0)

Work done

Steps 1. assignment to a register variable

While loop, if $x = L[k]$

2. k comparisons of register variables
3. k comparisons of x with L entry.
6. $k - 1$ increments of register variable.

Since step 3 is costliest & other steps are executed no more times, basic operation is *comparison of a memory and a register variable.*

cost $= k, \qquad \text{if } x = L[k]$
cost $= n, \qquad \text{if } x \notin L$

Time Complexity

Best Case time

- $k = 1, \quad x = L[1],$ best-case input.
- $B(n) = 1,$ as a function of input size.

Worst Case time

- $x = L(n)$ or $x \notin L$
- $W(n) = n$

Average-Case cost?

- Let $x \in L$
- Further, let x is equally likely to be in any position 1 through n.
- E_k: event that $x = L[k]$
- Probability of E_k happening

$$\begin{aligned} P(E_k) &= 1/n, \qquad 1 \leq k \leq n \\ t(E_k) &= \text{cost or time when } L[k] = x \\ &= k \end{aligned}$$

Thus, average cost

$$\begin{aligned} A(n) &= P(E_1)t(E_1) + P(E_2)t(E_2) + \ldots + P(E_n)t(E_n) \\ &= \sum_{k=1}^{n} P(E_k)t(E_k) \\ &= \sum_{k=1}^{n} \frac{1}{n}k \\ &= \frac{1}{n} \sum_{k=1}^{n} k \\ &= \frac{1}{n}(\frac{n(n+1)}{2}) \\ &= \frac{n+1}{2} \end{aligned}$$

Did we average over all possible inputs?

Let us allow the possibility that x might not be in L.

$$E_0 = \text{event that } x \notin L$$

Let

$$\begin{aligned} P(x \in L) &= q \\ P(x \notin L) &= 1 - q = P(E_0) \end{aligned}$$

For $k \geq 1$,

$$\begin{aligned} P(E_k) &= P(x \in L \ \text{ and } \ x = L[k]) \\ &= P(x \in L)P(x = L[k] \text{ given that } x \in L) \\ &= q\,\frac{1}{n} = \frac{q}{n} \end{aligned}$$

Thus,

$$\begin{aligned} A(n) &= \sum_{k=0}^{n} P(E_k)t(E_k) \\ &= \sum_{k=1}^{n} P(E_k)t(E_k) + P(E_0)t(E_0) \\ &= \sum_{k=1}^{n} \frac{q}{n}k \ + \ (1-q)n \\ &= \frac{q}{n}\sum_{k=1} k \ + \ (1-q)n \\ &= \frac{q}{n}\frac{n(n+1)}{2} \ + \ (1-q)n \\ &= q\frac{n+1}{2} + (1-q)n \end{aligned}$$

Thus,

- if $q = 1, \quad x \in L, \quad A(n) = \frac{n+1}{2}$
- if $q = 0, \quad x \notin L, \quad A(n) = n$
- if $q = 1/2$,

$$\begin{aligned} A(n) &= \frac{1}{2}\,\frac{n+1}{2} + \frac{1}{2}n \\ &= \frac{n+1+2n}{4} \\ &= \frac{3n}{4} + \frac{1}{4} \end{aligned}$$

Cost, Work = time complexity.

Space Complexity

Extra space used apart from the input and the program (and the output, if required by specification).

$$S(n) = 1$$
$$\text{(1 register variable)}$$

Trade-off between time & space

If number are between 1 and 100, then a prepared index array$A[1 \ldots 100]$ such that

$A[i] = j$, if $L[j] = i$; else $A[i] = 0$

can yield $W(n) = O(1)$.

($characteristic\ array$)

Optimality

- Is the sequential search algo. optimal?
- Is there another algo. which solves the same problem using fewer number of comparisons?
- Theorem: In the worst case,

$$W(n) = n$$

- Proof by contradiction

  — If there is another algo B whose $W(n) < n$, then the element not seen by B may be x & B would be incorrect.

  ($O(\log n)$ on an ordered list & $O(1)$ in a characteristic vector alternatives only when performing multiple searches.)

  — **Lower Bound** on the number of comparisons for searching in an unordered list is n.

Problem Complexity

Def. (Worst case) Complexity of a problem is the min number of operations needed to solve the problem in the worst case using particular resources.

eg. Matrix Multiplication
$C_{nn} = A_{\text{n x n}} * B_{\text{n x n}}$
$C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j} + \cdots + A_{in}B_{nj}$

for $i \leftarrow 1$ to n
  for $j \leftarrow 1$ to n
    $C_{ij} \leftarrow 0$
    for $k \leftarrow 1$ to n
      $C_{ij} = C_{ij} + A_{ik}B_{kj}$

- Basic operation: Multiplication (addition can be ignored)
- $W(n) = n^3$
- lower bound on number of multiplications $= n^2$
  lower bound on the problem complexity of matrix multiplication $= n^2$.
- best algorithm found has complexity $n^{2.367}$

Average-case problem complexity

Space complexity of a problem • matrix multiplication $O(1)$

• sorting on comparison model $O(1)$.

• matching parentheses $O(n)$.