Design and Analysis of Algorithms

CS202

Grading Policy

- Weekly Quizzes: 20 marks
- Programming + Written Assignments: 20 marks
- Exams: 3*20 = 60 marks

1. Introduction

2. Inductive Design

3. Divide and Conquer Paradigm

Algorithmic Problems

1. Given a collection of webpages, and a keyword, find the webpages most relevant to the keyword.
2. Given my current location, find all petrol pumps near me.
3. Schedule time-table for CSE courses at IITH.
4. Given a positive integer n, check if n is a prime.
5. Given a sequence of elements, arrange them in increasing order.

Goals:

1. Design correct and efficient algorithms
2. Understand and apply standard algorithms
3. Learn broad design techniques

E.g. 1: Closest pair problem

Input: $(x_1,y_1),(x_2,y_2),\ldots,(x_n ,y_n)$

Output: A pair (x_i,y_i), (x_j,y_j) such that $\mathrm{sqrt}[(x_i-y_i)^2+(x_j-y_j)^2]$ is minimized.

Instance: (0,0),(0.3,0.4),(-0.5,3.6),(3,4)

Input Size: n

Algorithm 1: Closest pair

For each pair of points, compute the distance between them.
Find the minimum of all these distances, and return the corresponding points.

Algorithm 1: Closest pair

```
Min_so_far = -∞
For each i in 1 to n-1
    For each j>i
        Find d= sqrt[(x_i-y_i)^2+(x_j-y_j)^2]
        Min_so_far=min(d,Min_so_far)
```

Time Complexity of algorithm A

Input size: n

The maximum number of instructions made by A over all inputs of size n-> T(n)

[Worst-case complexity]

Time Complexity of Linear Search

Input: A[1,…,n], x
Output: Some position of x in A, if present

Lin_Search: For i=1 to n, if x=A[i] return i.

T(n)=n

Analysis of correctness and time complexity

Number of instructions<3n^2
Processor which can execute 3*10^9 instructions per second.

n: 10^6 Time: 1000s (3*10^12/3*10^9) ~ 16 min
n: 10^7 Time: 1600 min ~1.16 days
n: 10^8 Time: 116 days
Closest pair admits O(n log n) time algorithm.
n: 10^6 Time: 20 ms
n: 10^7 Time: 0.23 s
n: 10^8 Time: 2.7 s

RAM model

Input Size:

Input: A positive integer n
Output: Find $\sqrt{n}$

Accurate size: Number of bits

$$\log_2 n$$

Input Size:

Input: Two positive integers m,n

Output: Find mn

Log_2 m + log _2 n

Input size in terms of space (#bits) to store m,n.

1421424

8342414 12   k bits: 00000 (k times) to 11111 (k times): 0 to 2^k-1 m<=2^k-1 =>

56

#bits to represent an integer m: log_2 (m+1)

RAM Model:

1. Random access: Unit cost for read/write of A[i]
2. Each word of data is limited in size (#bits)
3. Each instruction is unit cost: arithmetic operations (+,-,*), comparison.

Find the input size

1. Input: x, y in {1,2,…,n}. Output: xy

 2 log_2 n

2. Input: A,B: n by n matrix. Output: AB (n^2)

 Input size: 2n^2 (number of elements)* size of each element (measured in bits)

Find the time complexity of the natural algo

1. Input: x, y in {1,2,…,n}. Output: xy

2. Input: A,B: n by n matrix. Output: AB

Paradigm 0: Induction

Examples: Linear search, Adding x to all the elements, calculating Fibonacci numbers, finding the minimum element of an unsorted array.

Inductive Design

Input: A[1,2,…,n]
Output: f(A[1,2,…,n])

Algorithm:
For i=1 to n,
    Update value of f on seeing A[i]
Compute f(A[1]),f(A[1,2]),f(A[1,2,3]),…

E.g. 2: Finding the Minimum in an array

Input: A[1,2,…,n]
Output: Min(A[1,2,…,n])

Example Input: 14,3,-7,5,0,-2,8,5,10,4
Example Output: -7

Algorithm to find minimum

Min_so_far=A[1]
For i=2 to n
Set Min_so_far=min(Min_so_far,A[i])

Divide & Conquer Strategy

E.g. 1:

Given a positive integer n, find √n.

```
i=1
While (i*i<n)
   Increment I
Return (i-1)
```

E.x. 1:

Finding a local minimum

3,7,0,-2,12,8,10,1,4,2

A local minimum is an element A[i] such that it is:

A[i]<=A[i-1] and A[i]<=A[i+1]