Divide and Conquer Algorithms

Vidhya Balasubramanian

Introduction

- Solves computational problem by
  - Dividing it into subproblems of smaller size
    - Solve each problem recursively
  - Merging solutions to sub-problems to produce solution
- e.g.,
  - Merge Sort
  - Quick Sort
- Complexity analyzed using recurrence relations

Divide and Conquer

- Divide Step:
  - If input size smaller than certain threshold solve by straightforward method
  - Divide input data into two or more disjoint subsets
- Recur:
  - Recursively solve subproblems associated with the subsets
- Conquer
  - Take the solutions to the subproblems and merge them into a solution to the original problem

Sorting

- Merge Sort
  - Divide: Trivial
  - Conquer: Recursively Sort each sub-array
  - Combine: Linear-time merge
- Quick Sort
  - Divide: Split array based on pivot
  - Conquer: Recursively split
  - Combine: Trivial

Powering a Number

- Compute a^n
- Naive algorithm: $\Theta(n)$
- Divide and Conquer Algorithm
  - $$a^n = \begin{cases} a^{n/2} . a^{n/2} \; if\ n\ is\ even \\ a^{(n-1)/2} . a^{n/2} \; if\ n\ is\ odd \end{cases}$$
  - $T(n) = T(n/2)+\Theta(1) \Rightarrow T(n) = \Theta(\lg n)$

Matrix Multiplication

- Input: $X = [x_{ij}]$, $Y = [y_{ij}]$, $i,j = 1,2,3,\ldots,n$
- Output: $Z = [z_{ij}] = X\text{x}Y$
  - $$z_{ij} = \sum_{k=1}^{n} x_{ik} \cdot y_{kj}$$
  - $\Theta(n^3)$
- *Divide and conquer can reduce cost*

Divide and Conquer Strategy

- Idea
  - n x n matrix = 2x2 matrix of (n/2)x(n/2) submatrices
  - Rewrite Z = X x Y as
  - $\begin{pmatrix} i & j \\ k & l \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} x \begin{pmatrix} e & f \\ g & h \end{pmatrix}$
  - i= ae+bg
  - j= af+bh
  - k= ce+dg
  - l= cf+dh

Divide and Conquer Strategy

- Analysis
  - 8 multiplications of $(n/2)\times(n/2)$ submatrices
  - 4 additions of $(n/2)\times(n/2)$ submatrices
- $T(n) = 8T(n/2)+\Theta(n^2)$
- This is $\Theta(n^3)$ – not better than brute force

Strassen's Method

- Multiply 2×2matrices with 7 recursive multiplications
- $P1 = a \cdot (f - h)$
- $P2 = (a + b) \cdot h$
- $P3 = (c + d) \cdot e$
- $P4 = d \cdot (g - e)$
- $P5 = (a + d) \cdot (e + h)$
- $P6 = (b - d) \cdot (g + h)$
- $P7 = (a - c) \cdot (e + f)$
- $r = P5 + P4 - P2 + P6$
- $s = P1 + P2$
- $t = P3 + P4$
- $u = P5 + P1 - P3 - P7$
- 7 multiplications, 18 adds/subs
- No reliance on commutativity of multiplication

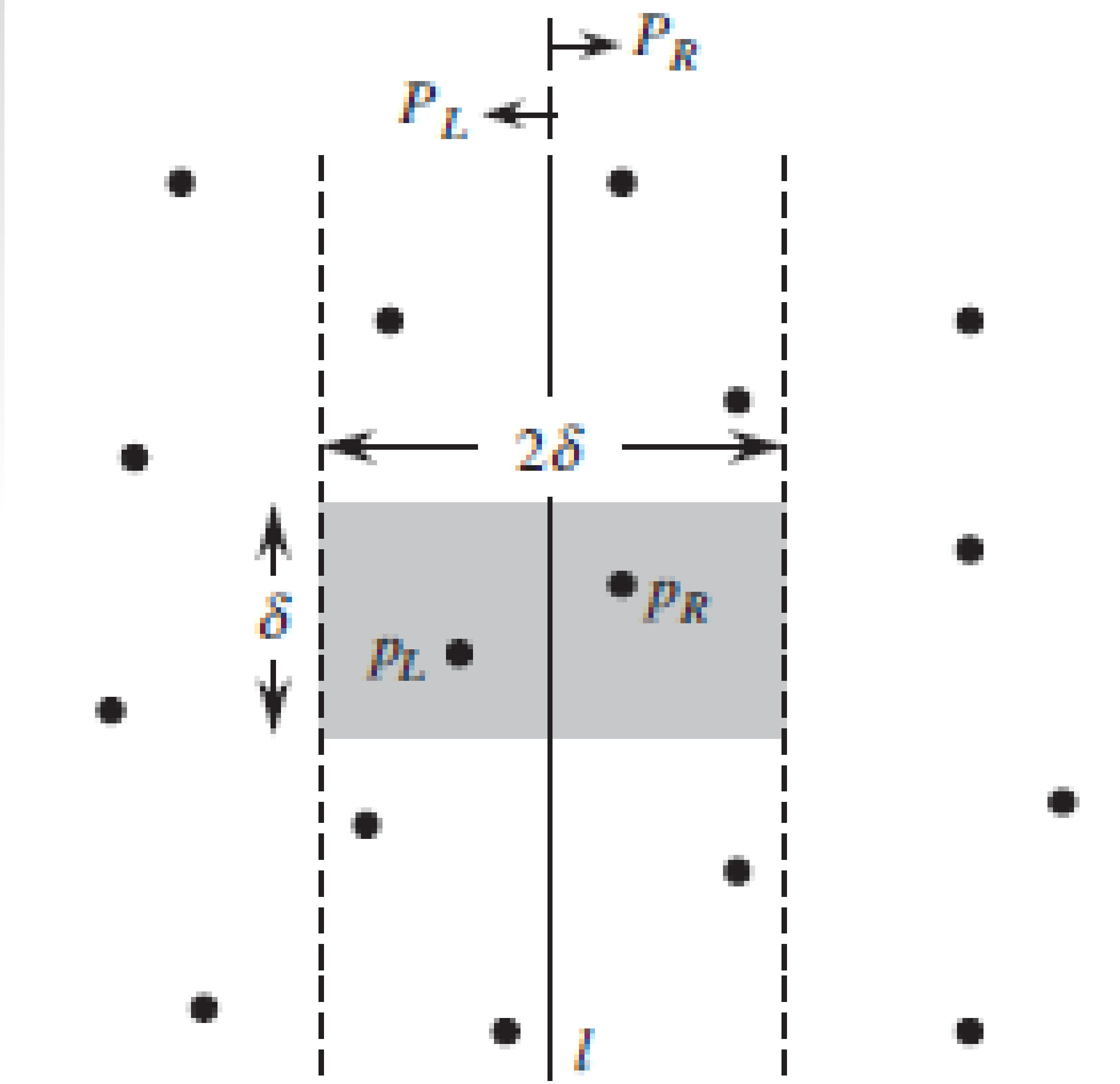
Strassen's Algorithm

- Partition: A and B into $(n/2)\times(n/2)$ submatrices.
  - Form terms to be multiplied using + and –.
- Conquer: Perform 7 multiplications of $(n/2)\times(n/2)$ submatrices recursively
- Combine: Form C using +and –on $(n/2)\times(n/2)$ submatrices.
- $T(n) = 7T(n/2)+\Theta(n^2)$
- This is $\Theta(n^{2.81})$

Closest Pair Problem

- Input: n points in a plane, each given by a pair of real numbers
- Output: Pair of points with shortest distance between them
- Brute Force
  - Compute distance between every pair and find the minimum distance pair
  - $\Theta(n^2)$
- Apply divide and conquer

Divide and Conquer Strategy



Divide and Conquer Strategy

- Idea
  - Draw a vertical line that has n/2 points on each side
  - Apply algorithm recursively to find closest pair on each side
- Finding the correct vertical line
  - Sort points by x-coordinate and pick median coordinate line
    - Points on this line assigned to one side
    - Sort points also by y-coordinate to improve performance
  - $\Theta(n \log n)$
- Closest pair may involve one point on each side of the line

Divide and Conquer Strategy

- Checking pairs that cross the line
  - Consider a pair that crosses the line iff its distance is lesser than the smallest distance that does not cross the line
    - Finding the set of points satisfying this by x-axis alone is costly
  - Sort the identified points along y-axis as well
    - Then only O(n) points have to be checked
    - Initially sorting the points by y-axis helps improve cost
- Complexity: $T(n) = 2T(n/2) + \Theta(n)$

Problems

- Given an array having first n ints and next n chars

  A= $i1\ i2\ i3 \ldots i$n, $c1\ c2\ c3 \ldots c$n

  Write an in-place algorithm to rearrange the elements of the array as: A = $i1\ c1\ i2\ c2 \ldots i$n cn
- Give an algorithm to divide an integer array into 2 sub-arrays s.t their averages are equal i.e average of values of left array must be same as average of right array