Advanced Algorithms
Lecture 2

Lecture by Shay Mozes
Typeset by Steven Karas

2016-11-13
Last edited 18:17:13 2016-12-28

1 Divide and Conquer

Algorithm design pattern with three steps: Divide, Conquer, Combine. Basically, divide the problem into several independent sub-instances of the same problem. Solve each sub-problem (potentially in parallel), and then combine the results. Typically, problems are subdivided several times before being tractable enough to solve.

1.1 Example: Binary Search

Given a sorted sequence of names and telephone numbers (sorted by name), we want to find the telephone number that belongs to a specific person.

Divide the sequence into a middle element and two sub-sequences. Conquer by

Analysis

$$T(n) = \begin{cases} c_1 & \text{if } n = 1 \\ T(n/2) + c_2 & \text{if } n > 1 \end{cases}$$

1.2 Example: Merge Sort

Given a sequence, order the elements according to some total ordering.

Divide: the sequence into two subsequences. Conquer: sort recursively. Combine: Merge lists

Analysis

$$T(n) = \begin{cases} O(1) & \text{if } n = 1 \\ 2T(n/2) + cn & \text{if } n > 1 \end{cases}$$

1.3 Example: Counting inversions

Music site wants to match song preferences (find users with similar tastes). Two users have ranked n songs as $1, ..., n$ and $a_0...a_n$. Songs i, j are inverted if $i < j$, yet $a_i > a_j$.

Brute force For each ranking, compare them and search for the matching song.

Analysis $O(n^2)$

D&C Divide: Split into half sequences.
Conquer: NOP
Combine: Assume each half is sorted. Count inversions, and merge into sorted whole.

Analysis Divide: $O(1)$
Conquer: $O(2T(n/2))$
Combine: $O(n)$

1.4 Master Theorem

Divide and Conquer algorithm complexity is affected by 3 criteria:

1. The number of sub-instances (a). Note that $a \geq 1$
2. The ratio of the initial problem (b). Note that $b > 1$
3. The complexity required to divide the problem ($D(n)$), and the complexity to combine the results ($C(n)$)

General Master Theorem

$$T(n) = aT(n/b) + \underbrace{D(n) + C(n)}_{f(n)}$$

Of course, $T(n) = \Theta(1)$ for small enough n bounded by some constant factor.

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a} \log^k n)$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ and if $a \cdot f(n/b) \leq c \cdot f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

Relaxed version for $f(n) = \Theta(n^k)$ As a special case, when $f(n) = \Theta(n^k)$ we get:

1. if $a > b^k$, then $T(n) = \Theta(n^{\log_b a})$
2. if $a = b^k$, then $T(n) = \Theta(n^k \log n)$
3. if $a < b^k$, then $T(n) = \Theta(n^k)$

Examples

1. $T(n) = T(2n/3) + 1$, then $T(n) = \Theta(\log n)$
2. $T(n) = 9T(n/3) + n$, then $T(n) = \Theta(n^2)$

Merge Sort: $T(n) = 2T(n/2) + cn = O(n \log n)$ Binary Search: $T(n) = T(n/2) + c = O(\log n)$

General f Intuition: Compare $f(n)$ to $n^{\log_b a}$

If $f(n) = O(n^{log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some constant $\varepsilon > 0$ and if $a \cdot f(n/b) \leq c \cdot f(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

Intuition Each layer has a times more instances than the layer above. Each subinstance is $1/b$ smaller than the layer above it. Thus, the tree has $\log_b n$ layers. In the bottom layer, each subinstance takes $\Theta(1)$, but there are $\Theta(n^{\log_b a})$.

Formally

$$a^{\log_b n} = 2^{\log a \frac{\log n}{\log b}} = n^{\log_b a}$$

$$\Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b a - 1} a^j f(\frac{n}{b^j})$$

1.5 Example: Integer Multiplication

Given two n-bit numbers

Naive (Elementary) Perform n additions of $O(n)$ bit numbers: $\Theta(n^2)$.

Naive Divide and Conquer Divide and Conquer: Assume X, Y given in binary, with a, c being the higher significance bits of X, Y respectively, and b, d the low bits.

$$X = a2^{n/2} + b \qquad Y = c2^{n/2} + d$$

$$XY = ac2^n + (ad + bc)2^{n/2} + bd$$

Thus, $T(n) = 4T(\frac{n}{2}) + \Theta(n)$, where $a = 4$, $b = 2$, $k = 1$. Therefore, $O(n^2)$.

Karatsuba

Gauss Equation

$$ad + bc = (a + b)(c + d) - ac - bd$$

$$XY = ac2^n + (ad + bc)2^{n/2} + bd$$

Giving a=3, b=2, k=1

$$T(n) = \begin{cases} 1 & \text{if n=1} \\ 3T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

$$T(n) = \Theta(n^{log_2 3}) = \Theta(n^{1.58})$$

1.6 Example: Matrix Multiplication

Dot product Given two n-vectors a and b, compute $c = a \cdot b = \sum_{i=1}^{n} a_i b_i$

Naive

$$c = a \cdot b = \sum_{i=1}^{n} a_i b_i$$

This takes $\Theta(n)$ time, which is optimal.

Matrix multiplication Given two n by n matrices A and B, compute $C = AB$

Naive

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$

This takes $\Theta(n^3)$ time.

Analysis

Block Multiplication Split each matrix into 4 regions:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Also $T(n) = 8T(n/2) + \Theta(n^2) = \Theta(n^3)$

Fast Multiplication Strassen 1969
Also $T(n) = 7T(n/2) + \Theta(n^2) = \Theta(n^{2.81})$

1.7 Example: Convex Hull

Given a set A of n points on the plane, the convex hull of A is the smallest convex polygon that contains the points in A.

Every line that connects any two points in the set is inside the convex hull.

For convenience, assume that no points are identical, and have unique x,y coordinates. Our desired output is a set of vertices in clockwise order.

Let $A = p_1, ..., p_n$. Let CH(A) be the convex hull of A.

1. Sort the points of A by a_x 2. if $n \leq 3$, solve directly. Otherwise: 3. Divide $A = L \cup R$ and repeat 4. Combine the convex hulls

Find some pair of vertices in CH(L) and CH(R), check if there are any points below/above the line between them. It's sufficient to check the neighbors of the candidate point.

Complexity is O(n) comparisons of O(1)

Analysis Preprocessing: O(nlogn) Recursion: Each step takes O(n), O(1) for each point.

Therefore:

FILL in from slide 87 MUST recognize that 2T(n/2)+cn is O(nlogn)

Lower bound Any algorithm for calculating the convex hull takes $\Omega(n \log n)$ time.

Proof Given n positive numbers $x_1, ..., x_n$, let $A = \{(x_i, x_i^2) | x_i \in X\}$, and find a convex hull of the n points.

TOOD: fill in from slide 88

1.8 Example: Closest Pair

Given a set of points $P = \{p_1, ..., p_n\}$ in two dimensions, output the pair of points p_i, p_j with minimal Euclidean distance between them.

NOTE: n^2 pairs of points. Naive.

1-dimensional problem Reduce to one dimension. Sort the points, then scan the set for subsequent pairs.

O(n)

1-dimensional divide & conquer Divide into two sets. Find the smallest pairing, and combine according to slide 93

2-dimensional Divide & Conquer Divide into two sets by x. Find smallest in each.

To combine: check only points inside 2δ band around division point. Sort the points by

COPY from slide 105

Analysis O(n) 2T(n/2) O(n)