Algorithms - 4 Divide and conquer

Divide: into a number of subproblems that are smaller instances of the same problem
Conquer: solve subproblems recursively
  Base case: if the subproblems are small enough, solve by brute force
Combine: merge subproblem solutions to give a solution to the original problem

Recurrence Relations

> transform recurrence relations in to closed form - asymptotic solution
>
> **substitution**
>
> > guess
> > check: mathematical induction to find the constants and show it works)
> >
> > > name the constant in the asymptotic notation
>
> **recursion**
>
> > recursion tree
>
> **master method**

Master Method

$$T(n) = aT(n/b) + f(n)$$

$a = \#$ recursive calls ≥ 1 $\quad b > 1$

a,b are constants

$f(n) \geq 0$ the time to "divide" and "combine" steps

cases

1. $f(n) = O(n^{\log_b a - \epsilon})$ constant $\epsilon > 0$

   $T(n) = \Theta(n^{\log_b a})$ (cost dominated by the leaves)

2. $f(n) = \Theta(n^{\log_b a} \log^k n)$ constant $k \geq 0$

   $T(n) = \Theta(n^{\log_b a} \log^{k+1}(n))$ $\quad \log_b n$ levels (cost is same at each level)

3. $f(n) = \Omega(n^{\log_b a + \epsilon})$ constant $\epsilon > 0$ and $f(n)$ satisfies the regularity condition:

   $af(n/b) \leq cf(n)$ for some constant $c < 1$ and sufficiently large n

   You don't have to check this in this class

   $T(n) = \Theta(f(n))$ (cost dominated by the root)

Master Method (simplified)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1, \\ aT(n/b) + \Theta(n^d) & \text{otherwise} \end{cases}$$

$a \geq 1 \quad b > 1 \quad d \geq 0 \quad$ assume $n = b^k$

cases

1. if $\log_b a > d$ then $T(n) = \Theta(n^{\log_b a})$ (cost dominated by the leaves)

   $a > b^d$

2. if $\log_b a = d$ then $T(n) = \Theta(n^d \log(n))$ (cost is same at each level)

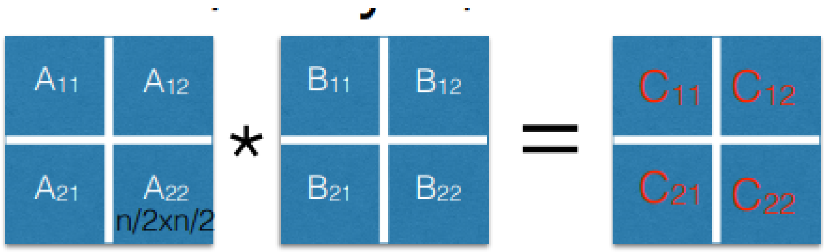
   $a = b^d$ $\quad \log_b n$ levels

3. if $\log_b a < d$ then $T(n) = \Theta(n^d)$ (cost dominated by the root)

   $a < b^d$

proof:

Matrix Multiplication

> normally: $O(n^3)$
>
> Recursively compute the matrix multiplication
> each matrix -> 4 squares
>
> $$T(n) = \begin{cases} 8T(\frac{n}{2}) + O(n^2) & else \\ O(1) & if\ n = 1 \end{cases}$$
>
> Still $O(n^3)$
>
> Strassen's Matrix
>
> > 
> >
> >
> > The 7 products!
> >
> > $P_1 = A_{11}(B_{12} - B_{22})$
> >
> > $P_2 = (A_{11} + A_{12})B_{22}$
> >
> > $P_3 = (A_{21} + A_{22})B_{11}$
> >
> > $P_4 = A_{22}(B_{21} - B_{11})$
> >
> > $P_5 = (A_{11} + A_{22})(B_{11} + B_{22})$
> >
> > $P_6 = (A_{12} - A_{22})(B_{21} + B_{22})$
> >
> > $P_7 = (A_{11} - A_{21})(B_{11} + B_{12})$
> >
> > $C_{11} = P_5 + P_4 - P_2 + P_6$
> >
> > $C_{12} = P_1 + P_2$
> >
> > $C_{21} = P_3 + P_4$
> >
> > $C_{22} = P_5 + P_1 - P3 - P_7$
> >
> > $T(n) = 7T(n/2) + \Theta(n^2)$
> >
> > in the unit cost model
> >
> > $$T(n) = \begin{cases} 7T(\frac{n}{2}) + O(n^2) & else \\ O(1) & if\ n = 1 \end{cases}$$
> >
> > $n^{2.80} \leq O(n^{log_2 7}) \leq n^{2.81}$
>
> Recurrence
> substitution method: guess and check method

Integer Multipllication

> original: x, y - n digit $O(n^2)$

Closest Pair of Points

```
CLOSET_PAIR_REC(px) # px are points sorted by x-value
    if |px| <= 3
        solve and return
    CONSTRUCT Qx and Rx
    (q1, q2) = CLOSEST_PAIR_REC(Qx)
    (r1, r2) = CLOSEST_PAIR_REC(Rx)
    theta = min{d(q1, q2), d(r1, r2)}
    x_mean = max x-coordianate in Qx
    CONSTRUCT Y = (s1, s2, ...sm) # sorted by y-value
    where si =(xi, yi) and |xi-xmean| <= theta
    for each si in Y
        for j = 1 to 7
            if d(si, si+j) < theta
                (s1', s2') = (si, si+j)
                theta = d(s'1, s'2)
```

T(n) = 2T(b/2) + O(nlongn) => O(nlog^2n)

I'th Statically

```
DETERMINISTICSELECT(A, n, i)
    # dind the ith smallest of n items in A
 divide the elements of the input array A into groups of 5
    find the medium of each group of 5 items and put them into another array B
 x = DETERMINISTICSELECT(B, n/4 , n/10)
 partition A-{x} into two sets A1, A2 such that
    A1 = {k|k<x}
        A = 2
```