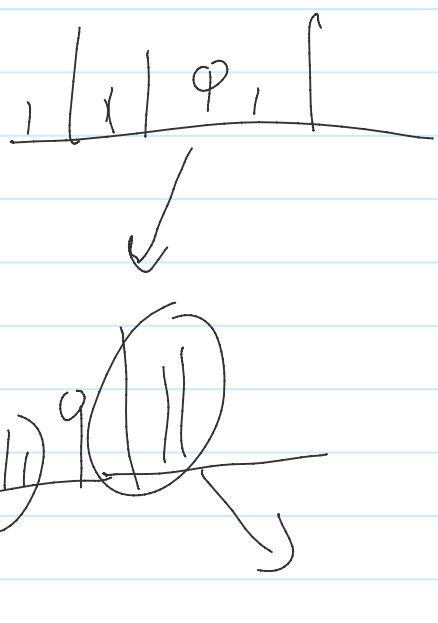
Divide and Conquer

Monday, March 26, 2018 6:35 PM

Definition.
Break a problem into two or more smaller sub-problems (potentially recursively) solve the subproblems and incorporate the solutions for the main problem.



Break up the problem
Solve (recursively)
Merge

Multiplication of long integers

8678,9321,5414

a[] = { 4, 5, 1, 2, 3, 9, 8, 7, 6, 8 }

b[] = { 3, 5, 3, 6, 2, 9 }

c[] = { 0, 0, 0, ..., 0 }

- for each digit in a_i
  - for each digit in b_j
    - $c[i+j] = a_i \cdot b_j$

do carry (c)

Kolmogorov conjectured that multiplication was $\Omega(n \cdot n)$ (a) (b)

Karatsuba

optimally $|a_1| \approx |a_0|$, $|b_1| \approx |b_0|$

$a = a_1 \cdot D + a_0$

$b = b_1 \cdot D + b_0$

$a \cdot b = (a_1 D + a_0)(b_1 D + b_0)$

$a \cdot b = a_1 b_1 D^2 + (a_1 b_0 + b_1 a_0) D + a_0 b_0$

$= c_2 D^2 + c_1 D^1 + c_0 D^0$

$\not{*}$ $c_2 = a_1 b_1$ ← $O((\frac{n}{2})(\frac{n}{2}))$

$\not{*}$ $c_0 = a_0 b_0$ 2 operations of $*$

$c_1 = a_1 b_0 + a_0 b_1$

$c_1 = a_1 b_0 + a_0 b_1 + a_1 b_1 - a_1 b_1 + a_0 b_0 - a_0 b_0$

$c_1 = a_1 b_0 + a_1 b_1 + a_0 b_1 + a_0 b_0 - a_1 b_1 - a_0 b_0$

$= a_1 (b_0 + b_1) + a_0 (b_0 + b_1) - a_1 b_1 - a_0 b_0$

$= (a_1 + a_0)(b_0 + b_1) - a_1 b_1 - a_0 b_0$

$= (a_1 + a_0)(b_0 + b_1) - c_2 - c_0$

1 operation; c_2 already done; c_0 already done

from 4 total operations to 3

$O(4 \cdot (\frac{n}{2})(\frac{n}{2})) = O(n \cdot n)$

$O(3 (\frac{n}{2})(\frac{n}{2})) = (n \cdot n)$

originally $S(n) = 4 \cdot S(\frac{n}{2}) \cdot S(\frac{n}{2}) + O(n)$

originally

$$t(n) = 3 \cdot t\left(\frac{1}{2}\right) \cdot t\left(\frac{1}{2}\right) + O(n)$$

$$t(n) \in O(n^{\log_2(3)})$$

Subset sum

Given a set of values $A = \{a_1, a_2, a_3, \cdots, a_n\}$

determine if $\exists$ a subset of A is equal to a target value

Backtracking $i \geq 0$

```
BSubSum(t, list[], current, p)
    if(current == t) return true;
    if(current > t) return false;
    if(p > list.size) return false;

    if(BSubSum(t, list, current, p+1))   // skip value
        return true;

    new sum = current + list[p];
    if(BSubSum(t, list, new sum, p+1))   // add value
        return true;

    return false;
```