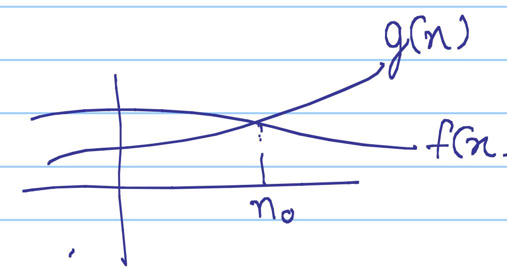
Note Title 2/6/2009

5 Feb 2009 6.006 Recitation 2

* Lecture 2 review
    - Merge-Sort
    - Complexity analysis

* Divide-and-Conquer
    - Maximum contiguous subset sum problem
    - Karastuba Multiplication

Complexity Analysis



Big-O notation

$f(n) = O(g(n))$ as $n \to \infty$

iff $\exists k > 0$, real n_0 s.t. $f(n) \le k\,|g(n)|$ for $\forall\, n > n_0$.

Big-Ω notation

$f(n) = \Omega(g(n))$ as $n \to \infty$

$f(n) \geq k\,|g(n)|$ $\quad \exists k, n_0, \quad n > n_0$

Θ notation

$f(n) = \Theta(g(n))$, $\exists k_1, k_2, n_0$: $\forall\, n > n_0$ $|g(n) \cdot k_1| < |f(n)| < |g(n) \cdot k_2|$

Maximum Contiguous Subset Sum Problem

Input : Array of n floating point numbers
Output : Maximum sum found in any contiguous subvector of the input

X

31	-41	59	26	-53	58	97	-93	-23	84

2 6

$X[2..6]$, 187

If all numbers in X are positive ? Take the whole array
If all numbers in X are negative ? Take nothing
If contiguous sum is not required ? Take all the positive elements

Algorithm1

Take sum of all pairs $0 \le i, j \le n$ $x[i..j]$ and take maximum of them

Algorithm 1

```
maxsofar = 0
for i = [0,n)            → n steps
    for j = [i,n)        → n
        sum = 0
        for k = [i,j]    → n
            sum = sum + x[k]     /* sum is sum of x[i..j]
        maxsofar = max(maxsofar, sum)
```

$O(n^3)$

Algorithm 2

```
maxsofar = 0
for i = [0,n)  ← n
    sum = 0
    for j = [i,n)  ← n
        sum = sum + x[j]     /* sum(x[i..j]) = sum(x[i..j-1]) + x[j]
        maxsofar = max(maxsofar, sum)
```

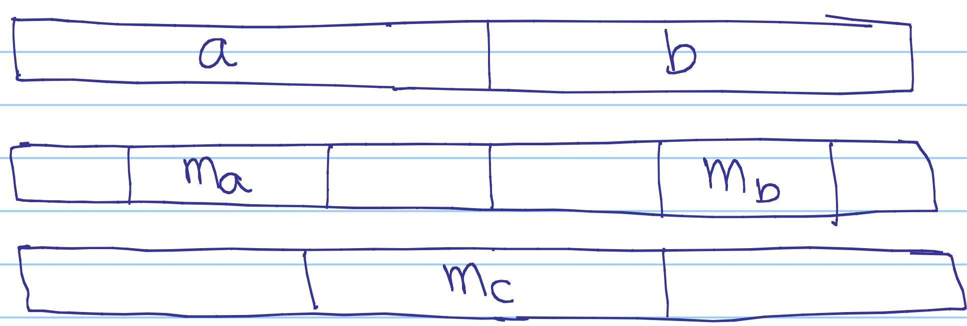
$O(n^2)$

Algorithm 3

```
cumarr[-1] = 0                                    n
for i = [0,n)
    cumarr[i] = cumarr[i-1] + x[i]        /* cumarray[i] = Σ_{j=0}^{i} x[j] = sum(x[0...i]) */
maxsofar = 0                                      n
for i = [0,n)                                     n
    for j = [i,n)
        sum = cumarr[j] - cumarray[i-1]   /* sum(x[i..j]) = sum(x[0...j]) - sum(x[0..i-1])
        maxsofar = max(maxsofar, sum)                       = cumarray[j] - cumarray[i-1]
```

$O(n^2)$

DIVIDE AND CONQUER :-)



$\max(m_a, m_b, m_c)$

Algorithm 4

```
float maxsum3(l,u)
  if (l>u)          /* zero elements */
     return 0
  if( l==u)         /* one element */
     return max(0, x[l])
  m = (l+u) / 2
  /* find max crossing to left */
  lmax = sum = 0             n
  for (i=m ; i>=l ; i--)
       sum = sum + x[i]
       lmax = max(lmax, sum)
  /* find max crossing to right */
  rmax = sum = 0          n
  for   i = [m,u]
       sum = sum + x[i]
       rmax = max(rmax, sum)
  return  max( lmax + rmax, maxsum3( l,m) , maxsum3(m+1, u))
```

$T(n) = 2T(n/2) + O(n)$

$O(n\log n)$

Algorithm 5

maxsofar = 0
maxendinghere = 0 $\qquad O(n)$

for i = [0, n)
    maxendinghere = max(maxendinghere + x[i], 0)
    maxsofar = max(maxsofar, maxendinghere)

KARASTUBA'S ALGORITHM

Multiply two n-bit numbers X and Y.

$X = x_{n-1} x_{n-2} \dots x_1 x_0$

$Y = y_{n-1} y_{n-2} \dots y_1 y_0$

Algorithm 1

sum = 0 $\qquad O(n^2)$
for i = [0, 2n-1] ← 2n
    for j = [0, i] ← n
        k = i - j
        sum = sum + x[j] × y[k]
    result[i] = sum mod 2, sum = $\lfloor sum/2 \rfloor$

Karastuba's Algorithm

$X = X_1 2^{n/2} + X_0$ $\quad\quad$ $X_0 = [x_{n/2-1} x_{n/2-2} \cdots x_1 x_0]$

$Y = Y_1 2^{n/2} + Y_0$ $\quad\quad$ $X_1 = [x_{n-1} x_{n-2} \cdots x_{n/2}]$

$$X * Y = (X_1 2^{n/2} + X_0)(Y_1 2^{n/2} + Y_0)$$
$$= X_1 * Y_1 2^n + (X_1 Y_0 + X_0 Y_1) 2^{n/2} + X_0 Y_0$$

$$X_1 Y_0 + X_0 Y_1 = (X_0 + X_1)(Y_0 + Y_1) - X_0 Y_0 - X_1 Y_1$$

$$X * Y = X_1 Y_1 2^n + [(X_0 + X_1)(Y_0 + Y_1) - X_0 Y_0 - X_1 Y_1] 2^{n/2} + X_0 Y_0$$

$T(n) = 3T(n/2) + O(n)$ $\quad\quad$ $O(n^{\log_2 3})$

Schönhage and Strassen $\quad$ $O(n \log n \log(\log n))$