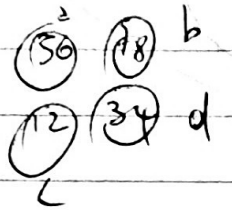
Algos — WEEK 1

Karatsuba multiplication

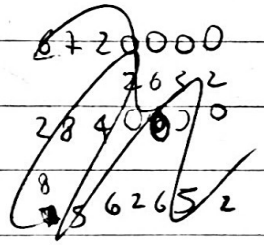


1 $a \cdot c = 672$

2 $b \cdot d = 2652$

3 $(a+b)(c+d) = 6164$

4 $3 - 2 - 1 = 2840$



$\leadsto$ ① · 10000 + ② + ④ · 100

Why? Because: expressing x and y as $(10^{n/2}a + b)$ and $(10^{n/2}c + d)$

$$(10^{n/2}a + b)(10^{n/2}c + d) = 10^n ac + 10^{n/2}(ad + bc) + bd$$

Only 3 recursive multiplications!

Divide and conquer design algorithm design paradigms
- Integer Multiplication
- Sorting
- Matrix multiplication
- Closest pair

Primitives for graphs
- Connectivity information
- Shortest paths
- Structure of information
- Social Networks

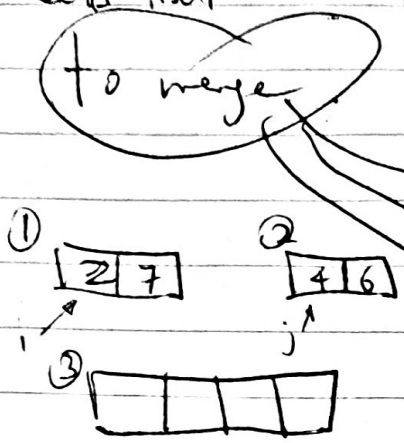
Randomization in algorithm design
- Quick sort
- Primality testing

Use and implementation of Data Structures
- Heaps
- Balanced binary search Trees

Merge Sort

recursive

Calls itself on halves of the array until it works to merge



Either The minimum element of ③ has to be either at the beginning of ① or of ②!

(compare i and j)

if it's in ②, move it to the right, repeat. (if it's in ②, move it)

Running Time:

C = output length n
A = 1st sorted array length n/2
B = 2nd sorted array length n/2
i = 1
j = 1
) init ①

for k = 1 to n ① assignment of k
if A(i) < B(j) : ② comparison (1)
C(k) = A(i) ③ assignment (1)
i++ ④ increment (1)
else [B(j) < A(i)]:
C(k) = B(j)
j++
end

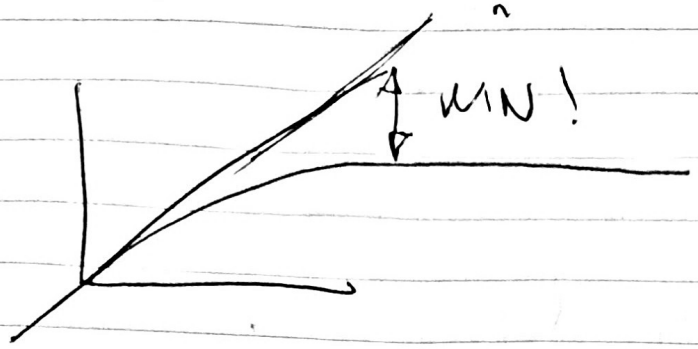
2 ① 2 operations

B(n) for loop is executed n times,
each of these times:
② assignment of k
① comparison
③ assignment
④ increment

(number of calls is given by the log(n) of the array split 4 times, 2 times, 1 time)

+ 4n+2 < 6n ② for Each merge number of merges exists by log, so

Comparing n^2/n to $\frac{6n\log n}{n}$:



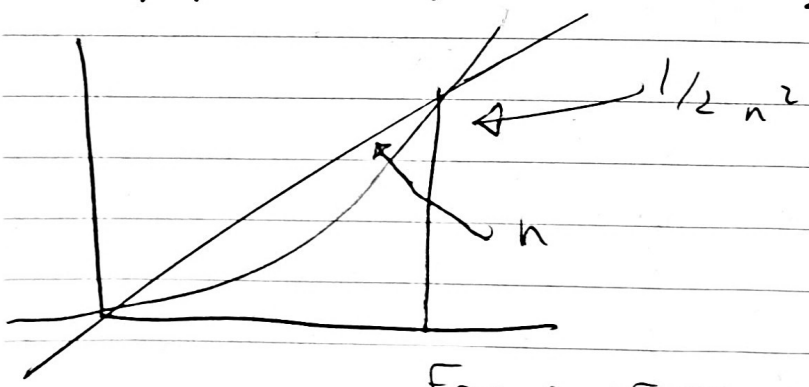
Principles:

Worst case Analysis!

A large case analysis and benchmarks requires previous knowledge of the domain the algorithm will be applied to.

Ignore small things

Asymptotic Analysis: focus on large input sizes n



FOCUS AFTER THIS BIT! SO

$6n\log(n)$ just becomes $n\log(n)$

Asymptotic Analysis

The Gist

Integers $O(n \log(n))$

Suppress constant factors and lower order terms,
- constant factors: too system dependent
- lower order terms: irrelevant for large inputs

Examples

1 - One Loop:

Does Array A contain integer t? $O(n)$

2 - Two loops:

Do Arrays A and B contain integer t? $O(n)$

3 - Two Nested Loops:

Do arrays A and B have a number in comon? $O(n^2)$

4 - Two Nested Loops (II)

Does array A have duplicate entries? $O(n)^2$

Big Oh ~~not~~ is the upper bound.

$T(n) = O(f(n))$ if and only if there exists constants $c, n_0 > 0$ s.t. $T(n) \leq c \cdot f(n)$ for all $n \geq n_0$

c, n_0 cannot depend on n

If $T(n) = a_k n^k + \dots + a_1 n + a_0$ Then

$T(n) = O(n^k)$

Proof: ? I've done this in Analysis 1

$p(x) \leq c \cdot x^{n_{max}}$

$a_n x^n + a_{n-1} x^{n-1} \dots + a_0 x^0 \leq |a_n| x^n + |a_{n-1}| x^n \dots \leq c \cdot x^n$

Big Omega and Theta notation

Greater than or equal to (Best case) — equal to (Average)

$T(n) = \Omega(f(n))$: if and only : if $\exists$ constants c, n_0 such that $T(n) \geq c \cdot f(n)$

$T(n) = \Theta(f(n))$: if and only : if $T(n) = O(f(n))$ and : if

$T(n) = \Omega(f(n))$

Little o notation

$T(n) = o(f(n))$ if and only : if for all constants $c > 0$, $\exists$ a constant n_0 s.t. $T(n) \leq c \cdot f(n) \; \forall \; n \geq n_0$

Big O says there exist a constant c and 1 value n_0 for which this is true, o says for all constants c, there exists an n_0 for which it is true

example $2x \to O(4x)$, since at $n_0 = 8$ for constant $c = 1$ it's

$2x \to O(x^2)$, since no matter the choice of c, there is always an n_0 for which it is true.

Additional Examples

1) $2^{n+10} = O(2^n)$

Need to find c, n_0, s.t.

$2^{n+10} \leq c \cdot 2^n \quad \forall n \geq n_0$

Note: $2^{n+10} = 1024\, 2^n$.

$1024\, 2^n \leq 1025\, 2^n$ for $n \geq n_0 = 0$

$\| \; c$

2) 2^{10n} is not $O(2^n)$

Proof by contradiction. IF $2^{10n} = O(2^n)$ then $\exists c, n_0$ s.t. $2^{10n} \leq c \cdot 2^n \; \forall n \geq n_0$

But then $2^{9n} \leq c \quad \forall n \geq n_0$, which is false!

Problem Problem Terribly

$f(n) \quad g(n)$

$2^n = O(2^n)$

$x^2 + x = O(x^2)$

$2^n \log_2(2^n) = O(2^n \cdot \log_2(2^n)) \rightarrow cn2^n = O(n2^n)$

$x^2 \log_2(x^{2c}) = O(x^2 \log_2(x^2))$

$f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$

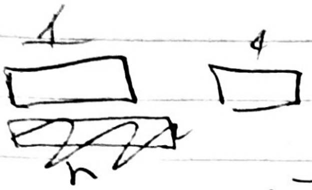
$f_n \cdot \log_2(f(n)^x) \leq \quad \forall n \geq n_0$ Time

$f(n) \leq c\, g(n)$ for $\forall n \geq n_0$

$2x^2 \quad x^2$

$2\log_2(3n) = O(\log_2 n)$

$2^{2x^2} = 4^{x^2} \leq \not< 2^{x^2}$



Then .... $2n + n$ steps

$2n$ for 2

$2n + n \quad < 3n + 2n = 5n$

$3n + n$ steps $= 4n \quad = 3n$

$4n + n$ steps $\quad 5n \quad = 14$

$6n \quad = 20n$

$2n + Kn \qquad 27n$

$(2+K)n \quad (2 + K \ldots)n$

Kn

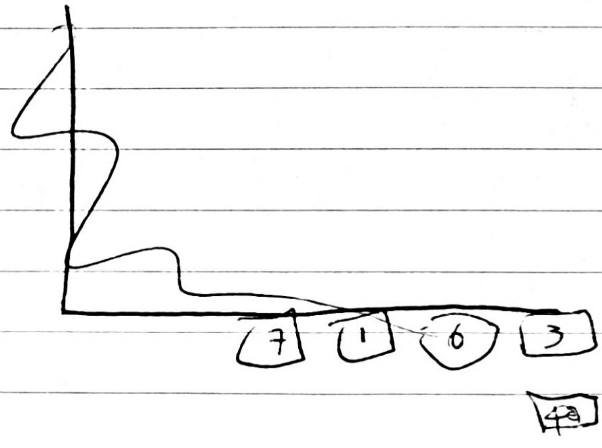
K

$1+2+3+4+5+6$

$Kn \quad \Sigma \quad (2+3+4+5+6+7)\,n$

$K \frac{K}{2}$

$\left(K \cdot \frac{K-1}{2}\right) +$

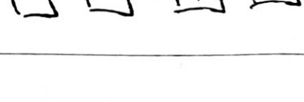


3 4 7 6

3

14

18



8

5 − 2 = 3

1000000

5 // 2 = 2

58

62

62 453

58 237

$(10^3(ac) + b)$

$(10^3(c) + d)$

$= 10^6 ac + bd + 10^3(ad + bc)$

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