Homework 1. Dec. 17. 2020

5. Asymptotic Complexity Comparisons

(a) $f_1 = 3^n$ $f_2 = n^{\frac{1}{2}}$ $f_3 = 12$ $f_4 = 2^{\log_2 n}$ $f_5 = \sqrt{n}$

$f_6 = 2^n$ $f_7 = \log_2 n$ $f_8 = 2^{\sqrt{n}}$ $f_9 = n^3$

list: f_3 f_7 f_2 f_5 f_4 f_9 f_8 f_6 f_1

(b) (i) $f(n) = \log_3 n$ $g(n) = \log_4 (n)$ $f(n) = \Theta(g(n))$

(ii) $f(n) = n \log n^4$ $g(n) = n^2 \log n^3$ $f(n) = O(g(n))$

(iii) $f(n) = \sqrt{n}$ $g(n) = (\log n)^3$ $f(n) = \Omega(g(n))$

(iv) $f(n) = n + \log n$ $g(n) = n + (\log n)^2$ $f(n) = \Theta(g(n))$

6. Computing Factorials

$N! = 1 \times 2 \times \cdots \times N$

(a) N is $\log N$ bits long. Find an $f(n)$ that $N!$ is $\Theta(f(n))$ bits long.

~~$\log N! = \Theta(N \log N)$~~

multiply n bit by an n bit number: get an number of $(n+n)$ bit

$N!$ bit: at most $N \cdot \log N = N \log N$ bits.

at least: consider $N/2 \sim N$ $\frac{N}{2} \times 2 = N$ (1 bit)

each at least $(\log N - 1)$ bits

total: at least $(\log N - 1) \cdot \frac{N}{2} = \frac{N(\log N - 1)}{2}$

(b) give a simple algorithm to compute $N!$ Multiplying two n bit numbers cost $O(n^2)$.

```
f=1
for i=2:N
    f = f·i.
```

$T(n) = \Theta(N^2 \log^2 N)$

7. Decimal to Binary

Given the n-digit decimal of a number, convertin...

needs $O(n^2)$ steps. Give a divide & conquer algorithm to show it does not take much more time than Karatsuba's algorithm

$x = 10^{n/2} \cdot a + b$ a, b are two $n/2$ digits number.

Multiplication: $O(n^{\log_2 3})$

Addition: $O(n)$ $\log_b a = d = \log_2 3$

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

8. Maximum Subarray Sum

an array A of integers, the maximum subarray sum is

$$\max_{i \le j} \sum_{k=i}^{j} A[k]$$

design an $O(n \log n)$ algorithm that finds it.

Solution: Split A into two parts, L and R

$L = A[1 \dots n/2]$ $R = A[n/2+1 \dots n]$

two cases: ① contained entirely in L or R S_1, S_2

② cross the boundary { ending at $n/2$ $O(n)$ / starting at $n/2+1$ $O(n)$ } S_3

return

$\max(S_1, S_2, S_3)$ or return $\max\{A[1], 0\}$ if $n = 1$.

$T(n) \le 2T(n/2) + c \cdot n$

$T(n) = O(n \log n)$

9. Monotone Matrices

Matrix Monotone $A_{m \times n}$: $n \ge m$ each row of A has no duplicates if the minimum of row i is located at column j_i, then $j_1 < j_2 < \cdots < j_m$

eg. $A = \begin{pmatrix} \boxed{1} & 3 & 4 & 6 & 5 & 2 \\ 7 & 3 & \boxed{2} & 5 & 6 & 4 \\ 7 & 9 & 6 & 3 & 10 & \boxed{0} \end{pmatrix}$

Give an algorithm (better than $O(mn)$) that finds the minimum in each row of matrix A

Solution: ① one row, scan that row

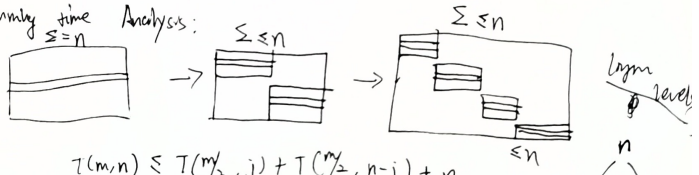
② more than one row:

i) find the smallest entry for $m/2$ row.

the column is j, then we only need to

scan recursively $A[1\dots m/2-1, 1:j-1]$ and $A[m/2+1:m, j+1:n]$

Running time Analysis:



$T(m,n) \le T(m/2, j) + T(m/2, n-j) + n$

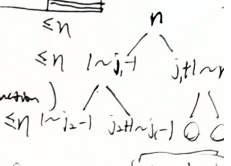
$T(m,n) = n(\log m + 1)$ (by induction)

recursion level number: $\log m$

n_k is the scanned column numbers for row-k.

we have $\boxed{\sum_k n_k \le n}$ $(k \neq m/2)$

total $T(n) \le n \cdot (\log m + 1)$



$\le n \cdot \log m$

$m/2$ th row