PROBLEM SESSION

2024-01-26

6.21 $f(n)$ Simply exp but not $\Theta(C^n)$ for any $C>1$

$$f(n) = \begin{cases} 2^n & \text{if } n \text{ even} \\ 3^n & \text{if } n \text{ odd} \end{cases}$$

Prec. ex: $c=1$

6.27 $(\forall A>1, \underline{c>0})(\forall B)\left(y^B = o\left(A^{y^c}\right)\right)$

$x = y^c$ $\qquad$ $y = x^{1/c}$

$$x^{B/c} = o(A^x) \quad ✓$$

2

6.54 abc computing Fib

(a) $b(F_n)$

$$F_n \sim \frac{\phi^n}{\sqrt{5}} \qquad \phi = \frac{1+\sqrt{5}}{2}$$

$$\log_2 F_n \sim n \log_2 \phi$$

$b(m) =$

$\lceil \log_2(m+1) \rceil$

$= 1 + \lfloor \log_2 m \rfloor$

$7 \quad \underbrace{111}_{3} = \log_2 8$

$8 \quad \underbrace{1000}_{4} = \log_2 16$

(b) F_n cannot be computed in poly time — can't even be written down

INPUT: n

$\underline{b(n) \sim \log_2 n}$

$b(F_n) \sim \underline{cn} \qquad c = \log_2 \phi$

~~$F_n \to \phi^n$ exp. beats $(cn)^k$ for any fixed k~~

$\underline{\underline{c \cdot 2^w}} = cn$ beats $b(n)^k \sim (\underbrace{\log_2 n}_{})^k \leftarrow \underline{\underline{w^k}}$

$cn = c \cdot 2^w \to$ beats $\to$

(c) If n is tiny

input length : n

$F_0 = 0$, $F_1 = 1$
for $i = 2$ to n
  $F_i = F_{i-1} + F_{i-2}$

addition of
k-bit numbers
$O(k)$

total
$$c\sum_{k=1}^{n} k = c\,\frac{n(n+1)}{2}$$
$$= O(n^2)$$

trickier method

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad A^k = \begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix}$$

repeated squaring

cost: last step
mult. of k-digit int

up to $2^k = \Theta(n)$ ← 2^k-digit numbers

↓ cost of
$A \mapsto A^2$

$O(n^2)$ (4

trickier method

$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ $A^k = \begin{bmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{bmatrix}$

repeated squaring

↓ cost of $A \mapsto A^2$

cost: last step
mult. of k-digit int | up to $2^k = \Theta(n) \leftarrow$ 2^k-digit numbers

Schoolbook method $O(k^2) = O(n^2)$

improved by Karatsuba $O(n^{\log_2 3})$

6.82 complexity of Knapsack w tiny weights

$\underbrace{w_1 \ldots w_n}_{\mathbb{N}}$ $\underbrace{v_1 \ldots v_n}_{\mathbb{R}}$ $\underbrace{W}_{\mathbb{N}}$ input

$\mathbb{N}$

also tiny

$O(nW)$

poly. time

WLOG $W \le \sum_{i=1}^{n} w_i$

$N = \text{size input} = \sum b(v_i) + \underbrace{\sum w_i}_{\ge n} + W \ge n + W$

$n \cdot W < (n+W)^2$

NOT $o(N^2)$

$N = \Theta(n)$

make $W = \Theta(n)$, $v_i = 1$

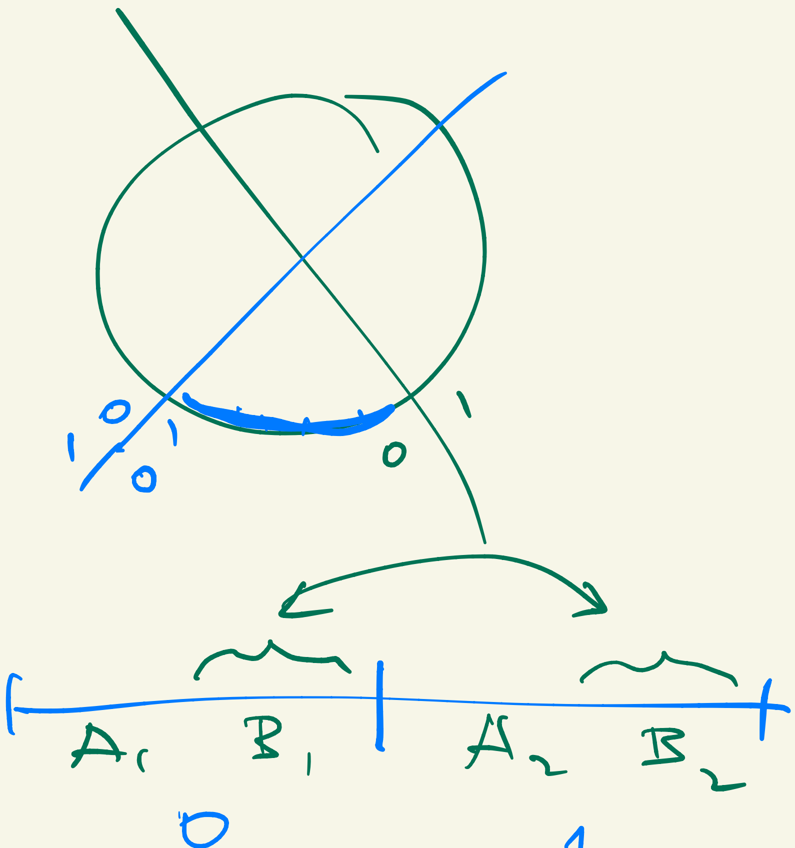
$n \cdot W = \Theta(n^2)$

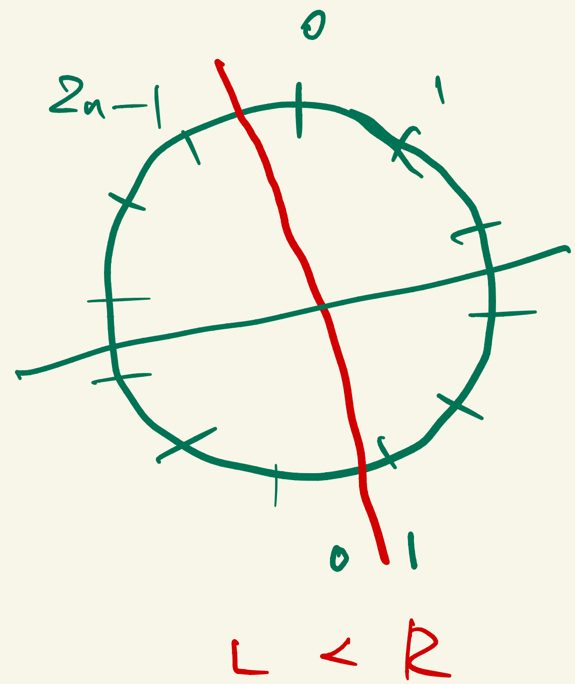
BON 4.38 Evenly splitting fake coins

$2n$ coins, some fake (lighter)

k fake coins

1st sol'n





2nd sol'n

6.5.7 $(F_k \bmod m)$ poly time in bit model

input $\sim \log_2 k + \log_2 m$

$$\overline{A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad \overline{A}^k = \begin{bmatrix} \overline{F_{k+1}} & \overline{F_k} \\ \overline{F_k} & \overline{F_{k-1}} \end{bmatrix}}$$

$\overline{x} := (x \bmod m)$

$\overline{A}$: entry-wise mod m

$$\overline{x+y} = \overline{\overline{x} + \overline{y}}$$

$$\overline{x \cdot y} = \overline{\overline{x} \cdot \overline{y}}$$

use repeated squaring to get $\overline{A^k}$

$$A \mapsto \overline{A^2}$$

$$A \mapsto \overline{X \cdot A}$$

cost of each round: multipl. of $\log_2 m$-bit numbers

total $\log_2 k \cdot (\log_2 m)^2 = O(n^3)$ #rounds $\log_2 k$

4.64 MERGE in $m+k-1$ comparisons

$A[1..m]$, $B[1..k]$ sorted arrays $\to C[1..m+k]$

(a) optimal if $m=k$

NTD: find two inputs that give same answer to all questions while output must be diff.

BAD CASE: $A[1] < B[1] < A[2] < B[2] < \cdots$

Claim We must compare all pairs of neighbors

If only $2n-2$ questions asked

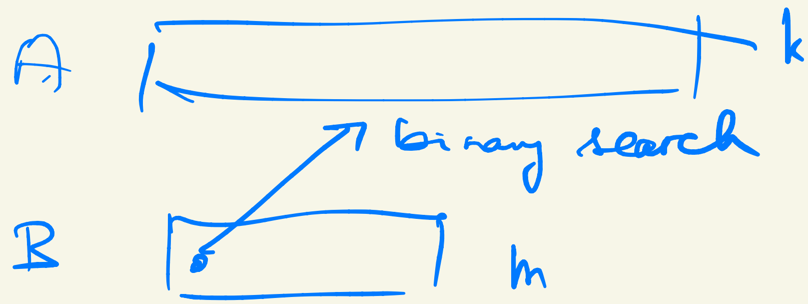
$\to$ $\exists$ pair of consecutive entries, not compared

swap these two to get another input that gives the same answer to all questions

4.67 if $m = o\left(\frac{n}{\log n}\right)$ then

MERGE far from optimal:

$\forall m, n$ we can merge in $m \cdot \lceil \log_2 (k+1) \rceil$



$A[i] \qquad A[i+1]$

$\underbrace{\qquad\qquad\qquad}_{B[j] \dots B[j']}$

for $i = 1$ **to** m

INSERT $B[i]$ into A by binary search

6.79 KNAPSACK dynam. prog. alg.

NOT poly time all inputs: integers

need to find a set of bad inputs

$v_i = 1$, $w_i = 1$ $W = 2^t$

$N = 2n + t$ make $n \le t$

$\downarrow$

$N = \Theta(t)$ time $\Omega(2^t)$

Common sense: $W \le \sum w_i$ ✗

take $v_i = 1$, $w_i = 1$ except $w_n = 2^t$

$N = \Theta(t)$ time $\Omega(2^t)$

6.85 KNAPSACK variant:

min weight for target value V

values $\in \mathbb{N}$ incl V

$O(nV)$

dynamic programming

$$m[i, j] = \min_{I \subseteq [n]} \left\{ \sum_{k \in I} w_k \;\middle|\; \sum_{k \in I} v_k \geq j \right\}$$

$0 \leq i \leq n$ $\quad$ $0 \leq j \leq V$

$\uparrow$

#items