November 9, 2022 10:51 AM

# Part III: Advanced Algorithmic Techniques

Can we solve APSP in slightly better than $n^3$ time?

or 3SUM " " $n^2$ "

or OV " " $n^2$ "

## Shaving Logs

$n^3 \to n^3/\log^c n$ ?

$n^2 \to n^2/\log^c n$ ?

## Ex1: Boolean Matrix Mult.

Aralazarov, Dinic, Kronrod, Faradzhev '70

$O(n^3/\cancel{\log n})$ $\log^2 n$

"Four Russians Alg'm"

(worse than Strassen but is "combinatorial")

idea 1 - "bit tricks"

assume RAM model of computation with w-bit words

where $\underline{w \geq \log n}$

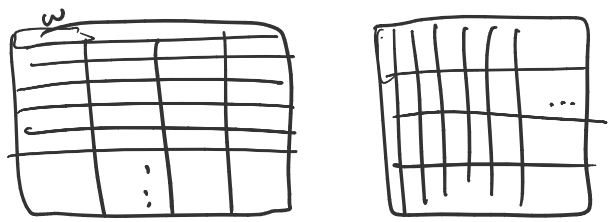
s.t. one index/ptr fits in a word

**Lemma** can compute Boolean product of $1\times w$ & $w \times 1$ matrix in $O(1)$ time (instead of $O(w)$)

Pf: by bitwise &

0110 & 1011 = 0010 ≠ 0

To multiply 2 n×n matrices,



reduces $n \cdot \frac{n}{w} \cdot n$ products of $1 \times w$ & $w \times 1$

$\Rightarrow$ total time $O(\frac{n^3}{w} \cdot 1) = \boxed{O(\frac{n^3}{\log n})}$

Rmk - what if bitwise & not supported?

use table lookup

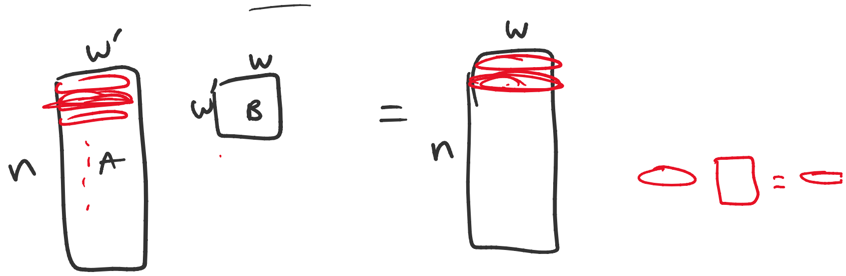
build table of size $O(2^w \cdot 2^w) = O(4^w) \le O(n^2)$.

set $w = \log n$

idea 2 - do more table lookup

a "4 Russians" trick

Lemma can compute Boolean product of $n \times w'$ & $w' \times w$ matrix with $w' = \delta \log n$ in $O(n)$ time (instead of $O(nw'w)$)




Pf: for each possible row $v$, precompute $v \cdot B$. & store in table

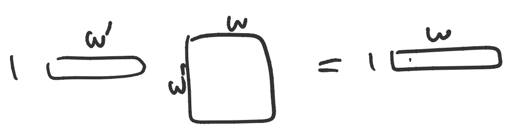
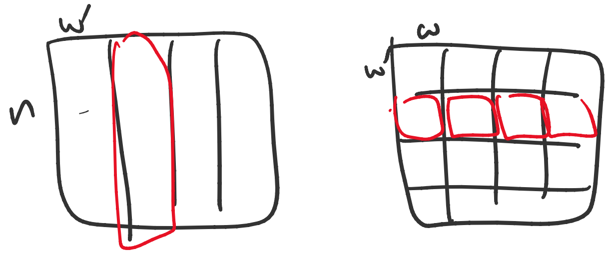



table has size $O(2^{w'})$

can build in $O(2^{w'} w' w) \ll o(n)$. time.

$\square$

To multiply 2 $n \times n$ matrices,




$\Rightarrow$ $\frac{n}{w'} \cdot \frac{n}{w}$ products of $n \times w'$ & $w' \times w$

$\Rightarrow$ total time $O\left(\frac{n}{w'} \cdot \frac{n}{w} \cdot n\right) \le \boxed{O\left(\frac{n^3}{\log^3 n}\right)}$.

Bansal, Williams '09: $O(n^3 / \log^{2.25} n)$.

C. '15: $O(n^3 / \log^3 n)$.

Yu '16: $O(n^3 / \log^4 n)$

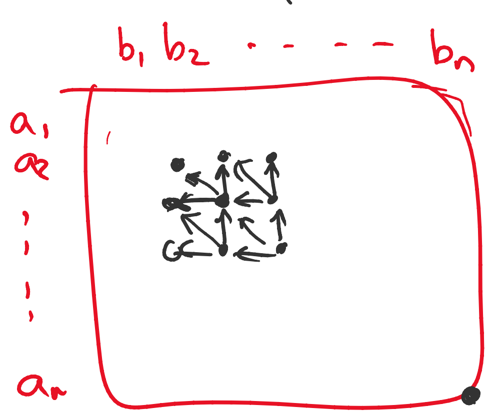
---

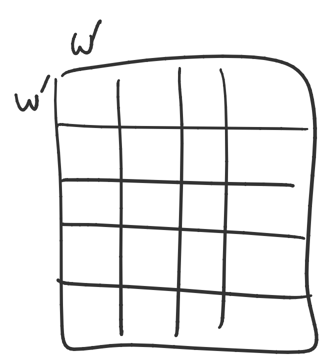
## Ex2: LCS / edit distance

Masek-Paterson '80 $\quad O\left(n^2 / \cancel{\log n}\ \log^2 n\right)$ when alphabet size $\sigma$ is const.

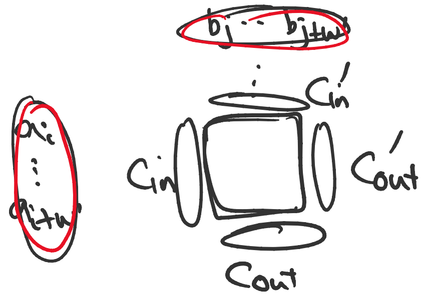
assump can be removed...

$$C[i,j] = \min \begin{cases} C[i-1,j]+1 \\ C[i,j-1]+1 \\ C[i-1,j-1] + d(a_i,b_j) \end{cases}$$




idea - bit tricks & table lookup again by storing diffs idea - bit tricks & table lookup again




function

$w'$ bits by storing diffs

$w' \log \sigma$ bits

$$f(C_{in}, C_{in}', a, b) = (C_{out}, C_{out}').$$

$(14, 14, 15, 16, 16, 17)$

$0\ 1\ 1\ 0\ 1$

build table for $f$

of size $O(2^{w'} \cdot 2^{w'} \cdot \sigma^{w'} \cdot \sigma^{w'})$

$\leq O(\sigma^{4w'}) \ll o(n)$

by setting $w' = \delta \log_\sigma n$

$$\Rightarrow \text{total time } O\left(\left(\frac{n}{w'}\right)^2\right) \leq \boxed{O\left(\frac{n^2}{(\log_\sigma n)^2}\right)}$$

---

## Ex 3: 3SUM for ints

Baran, Demaine, Patrascu '05 $\quad O\left(\frac{n^2}{\log^2 n}(\log\log n)^2\right).$

idea - hashing again!

$h(x) = x \bmod p.$