Part III: Advanced Algorithmic Techniques

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

or $3$SUM " $n^2$ "

or $O(V " n^2"$

Shaving Logs

$n^3 \rightarrow \frac{n^3}{\log n}$?

$n^2 \rightarrow \frac{n^2}{\log n}$?

Ex1: Boolean Matrix Mult.

Avalovazov, Dinic, Kromod, Faradzev '70

$O(\frac{n^3}{\log n})$ "Four Russians Alg" (worse than Strassen but is "combinatorial")

"bit tricks"

Assume RAM model of computation with $w$-bit words

where $w = \log n$

st. one index/pointer fits in a word

Lemma: can compute Boolean product of

$1 \times 1$ & $w \times 1$ matrix in $O(1)$ time

instead of $O^*(w)$

\[
\begin{array}{c}
1 \quad w \\
\hline
w & 1
\end{array}
\]

Pf: by bitwise &

$0110 \ & 1011 = 0010 \neq 0$

To multiply 2 $n \times n$ matrices,
Reduces $n \times w \times n$ products of $1 \times w \& w \times 1$ to total time $O\left(\frac{n^3}{w}\right)$.

Remark: What if bitwise $\&$ not supported?

- Use table lookup
  - Build table of size $O(2^w \cdot 2^w) = O(4^w) \leq O(n^2)$.
  - 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 $w' = 8 \log n$ in $O(n)$ time (instead of $O(nw'w)$).

Proof: For each possible row $u$, compute $u \cdot B$ & store in table.

$$1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1$$

Table has size $O(2^{w'})$ can build in $O(2^{w'}w') = O(n)$ time.

QED
To multiply 2 \( n \times n \) matrices,

\[
\Rightarrow \frac{n}{w} \cdot \frac{n}{w} \quad \text{products of } n \times w \cdot w \times w
\]

\[
\Rightarrow \text{total time } O\left(\frac{n}{w} \cdot \frac{n}{w} \cdot n\right) \leq O\left(\frac{n^3}{\log^2 n}\right)
\]

Bansal, Williams '09: \( O\left(\frac{n^3}{\log^{2.25} n}\right) \).

C. '15: \( O\left(\frac{n^3}{\log^3 n}\right) \).

Yu '16: \( O\left(\frac{n^3}{\log^4 n}\right) \).

---

**Ex2: LCS / edit distance**

Masek-Paterson '80 \( O\left(\frac{n^2}{\log n}\right) \) when alphabet size \( \sigma \) is constant.

**Assumption**: can be relaxed...

\[
C[i,j] = \min \left\{ \begin{array}{l}
C[i+1,j] + 1 \\
C[i,j-1] + 1 \\
C[i-1,j-1] + d(a_i, b_j)
\end{array} \right.
\]

---

id a - hit tricks & table lookup again
idea - bit tricks & table lookup again w' bits

\[ f(C_{in}, C_{in}', a, b) = (C_{out}, C_{out}') \]

\((14, 14, 15, 16, 16, 17, 0, 1, 10, 1)\)

build table for \( f \) of size \( O(2^w' \cdot 2 \cdot \sigma \cdot \sigma') \)

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

by setting \( w' = \delta \log_2 n \)

\[ \Rightarrow \text{total time } O\left(\left(\frac{n}{w'}\right)^2\right) \leq O\left(\frac{n^2}{(\log_2 n)^2}\right) \]

**Ex 3: 3SUM for ints**

Baran, Demaine, Patrascu '05

\( O\left(\frac{n^2}{\log^2 n} (\log \log n)^2\right) \).

idea - hashing again!

\( h(x) = x \mod p \).