April 3rd, 2014

On arXiv.org, the main repository of publically available theoretical computer science research, a paper was posted.

This paper disproved a widely-held conjecture about the algorithmic complexity of many problems...

Theorem: The 3SUM problem has an algorithm that runs in $O(n^2 \frac{(\log \log n)^{5/3}}{(\log n)^{2/3}})$ time.

\[ o(n^2) \]

[Grønlund, Pettie, “Threesomes, degenerates, and love triangles”, 2014]

This means that the lecture “Reductions” on March 13 is now out-of-date and contains results that are vacuously true.
**3SUM-HARDNESS**

**real RAM**

Problem (3SUM): Given an array of integers $S$, do there exist three integers $a, b, c \in S$ such that $a + b + c = 0$?

\[
\begin{bmatrix}
-20, 9, 4, -8, 1, 6, -10, -4, 3
\end{bmatrix}
\text{ YES}
\]

\[
\begin{bmatrix}
-4, -9, 8, 11, -1, 7, -20
\end{bmatrix}
\text{ NO}
\]

Algorithm for 3SUM:

1. Sort $S$. $O(n \log n)$

2. For $i$ from 1 to $S$.size(): $O(n)$
   - Construct array $R$, where $R = S + S[i]$. $O(n)$
   - Search $R$ for two elements $R[j], R[k]$ such that $R[j] + R[k] = S[i]$. $O(n)$ using two-finger algorithm

\[
S = \begin{bmatrix}
-20, 4, -8, 4, 13, 4, 9, 15
\end{bmatrix}
\]

\[
R = S + S[9] = \begin{bmatrix}
-5, 4, 7, 0, 16, 18, 19, 24, 30
\end{bmatrix}
\]

\[
\]

**Conjecture:** Any algorithm for 3SUM takes $\Omega(n^2)$ time.

Evidence is that people have tried hard and failed plus other stuff.

If I wanted to show that Problem B also takes $\Omega(n^2)$ time to solve using any algorithm, I can use a reduction from 3SUM to Problem B. Such a reduction implies that Problem B is 3SUM-hard.
3SUM-hard PROBLEMS

Problem (3-Points-on-a-line): Given a set of $n$ points in the plane, are any 3 collinear?

Reduction:
\[ \text{Take integers } S = \{x_1, x_2, x_3, \ldots, x_n\} \]
\[ \text{Create point set } P = \{(x_1, x_1^2), (x_2, x_2^2), \ldots, (x_n, x_n^2)\} \]
\[ \text{Feed } P \text{ to 3-P-O-A-L solver} \]
\[ \text{If 3-P-O-A-L says yes, then return yes. Else return no.} \]

3SUM solver works b/c \( ab + bc = 0 \) iff \((a, a^2), (b, b^2), (c, c^2)\) collinear. The reduction says “3-P-O-A-L is 3SUM-hard.”

If any 3SUM solver must take \( \Omega(n^2) \) time, then since this solver takes \( O(n + f(n) + 1) \) time, \( f(n) = \Omega(n^2) \).

3SUM-hardness \( \iff \) Takes \( \Omega(n^2) \) time if 3SUM takes \( \Omega(n^4) \) time.

This reduction also implies that if 3-P-O-A-L can be solved in \( O(f(n)) \) time, then 3SUM can be solved in \( O(n + f(n) + 1) \) time. A trivial \( f(n) \) is \( O(1) = O(n^2) \), but we already had \( O(n^2) \) 3SUM algo.
Problem (Minimum Area Triangle): given a set of $n$ points in the plane, what is the smallest area triangle induced by 3 of the points?

Thm: M-A-T is 3SUM-hard. How to show this?

Suppose we nested reductions:

**If** Problem A takes $\Omega(n^2)$ time, then Problem C takes $\Omega(n^2)$ time provided $r_1(n) + r_2(n) + r_3(n) + r_4(n)$ is $o(n^2)$.

"The $\Omega(n^2)$ time has to be spent somewhere."

Claim: M-A-T is 3SUM-hard

Proof:
The New State of 3SUM

1. Can be solved in \( o(n^3) \) time.
2. Unknown if solvable in \( O(n^{3-\varepsilon}) \) time for some \( \varepsilon > 0 \).
3. New conjecture: \( \Omega(n^{2-\varepsilon}) \) time if \( \varepsilon > 0 \)?

\( o(n^3) \) time algorithm idea: solve in \( O(n^{3/2}\sqrt{\log n}) \) time in comparison-cost model, then port algorithm to real RAM using tricks to obtain a slight speed-up over naive port.

Linear decision tree model: Pay only for comparing values. Get to specify a linear function of the values and get sign of function out. Can compare \( O(1) \) values at once.
3SUM Algorithm

Input: Array $A$ of $n$ numbers

1. Sort $A$. Partition into $\frac{n}{g}$ groups $A_0, A_1, ..., A_{\frac{n}{g}}$ of size $g$. $O(n \log n)$

2. Let $D = \{a - a' : a, a' \in A_i\}$. Sort $D$. $O(n \log n + gn)$ [Fredman, 1976]

3. Let $A_{i,j} = \{atb : a \in A_i, b \in A_j, i \neq j\}$. Sort all $A_{i,j}$. $O(1)$ using step 2 &
   $att \leftarrow d$ if $a - c < d - b$
   "Fredman's trick"

4. For $k$ from 1 to $n$: $O(n)$ iterations

4a. Let $d = 0, h = \frac{k}{g}$. // $keA_h$

4b. while $d < h$: $O(\frac{n}{g})$ iterations

4c. if $-A[k] \leq A_{j,h}$: return "Yes". $O(\log(\frac{n}{g}))$ via BS of $A_{j,h}$

4d. if $\max(A_j) + \min(A_h) > -A[k]$: $--h$, else: $++1$ $O(1)$

5. return "No"

Total comparisons: $O(n \log n + \frac{n^2 \log n}{g})$

Let $g = \sqrt{n \log n}$: $O(n \log n + \frac{n^2 \log(\log n)}{\sqrt{n \log n}}) = O(n^{3/2} \log n)$

How to port into $o(n^2)$ real RAM algorithm?