**Convolution 3SUM**

**Def. (Convolution 3SUM)**

*Input*: array $A[0..n-1]$ of integers


**Theorem**: If there is an algorithm with running time $O(n^2 \cdot c)$ for convolution 3SUM, then there is an algorithm with running time $O(n^2 \cdot s)$ for 3SUM.

**Proof**: Consider 3SUM instance $WIS[1..U]$

*Preprocessing*: check if there is a 3SUM witness $x + x = z$ by sorting and binary search.

Pick, uniformly at random, hash function $h$ from almost balanced and almost linear family $h: [U] \rightarrow [R]$ (param $R$).

For sake of demonstration, assume linear here.

1. Apply hash function to each $a \in I$.
2. For each $x \in [R]$ such that $|h^{-1}(x)| > 3 \frac{n}{R}$ ("heavy")
   - Check if there is a 3SUM witness $a + b = c$ for each $a \in h^{-1}(x)$.
3. For each $a \in h^{-1}(x)$ and $b \in W$, store $a + b$ in a set data structure.
4. For each $a \in h^{-1}(x)$, check if $c$ contained in set data structure.

- Time $O(|h^{-1}(x)| \cdot n)$ hash set
- Time $O(|h^{-1}(x)| \cdot n \log n)$ binary search tree

$\rightarrow$ Time $\sum_{x \in h^{-1}(x)} |h^{-1}(x)| > 3 \frac{n}{R}$

$= O(R \cdot n)$ in expectation (almost balanced property).
2) Assume that each $h^{-1}(x)$ (where $x \in [R]$) contains $\leq 3\frac{n}{R}$ elements.

For all triples $i, j, k \in [3\frac{n}{R}]$ create array $A$ of length $2 \cdot 8 \cdot R$.

For $x \in [R]$ set $0 = 8R$

Put $i$-th element of $h^{-1}(x)$ to $A[8(x-1)+1]$ and $A[8(x-1)+2]$.


Put $k$-th element of $h^{-1}(x)$ to $A[8(x-1)+5]$ and $A[8(x-1)+6]$.

Set all other entries to $\infty$ (or sufficiently large value).

Return yes if at least one non-$\infty$ instance returns yes.

Correctness: 
If $a = b$, then found in preprocessing. Otherwise $\forall a \neq b \implies h(a) + h(b) = h(c)$ (mod $R$) (linearity)

either $h(a) + h(b) = h(c)$ or $h(a) + h(b) = h(c) + R$

$\implies$ Let $i$ be $i$-th element of $h^{-1}(h(a))$, $j, k$ accordingly

$\implies$ There is a triple $i, j, k$, st. in array $A$ $A[8(i-1)+1], A[8(i-1)+2]$

or $A[8(i-1)+3], A[8(i-1)+4], A[8(i-1)+5], A[8(i-1)+6]$


Then we must have $i = 8 \cdot t_a + 1$ and $j = 8 \cdot t_b + 3$

(Because $x + y = z$ has unique solution over $[1, 3, 4]$ and $A[i] \neq A[j]$)

$\implies$ must find $a \in h^{-1}(i+1), b \in h^{-1}(j+1), c \in h^{-1}(i+j+1)$

s.t. $a + b = c$

Running Time: $O(n \log n + nR + (\frac{n}{R})^3, n^2 - \epsilon)$

Set $R = \frac{\epsilon}{2}, n^{1/4} \implies O(n^{2 - \epsilon})$
From Convolution 3SUM to $O$-weight triangle

**Definition:** ($O$-weight triangle)

**Input:** Weighted directed graph $G$ with positive/negative edge weights

**Task:** Decide if $G$ contains a triangle of weight exactly $0$

**Theorem:** If there is an algorithm for $O$-weight triangle on graphs with $n$ nodes with running time $O(n^{3-\epsilon})$ (for some constant $\epsilon > 0$), then there is an algorithm for Convolution 3SUM on an array of length $n$ with running time $O(n^{2-s})$ (for some constant $s > 0$)

**Proof:**

Consider Convolution 3SUM instance $A, B, C$

For each $i \in [m]$: create bipartite graph $G_i$:

- $\text{L}_i = A_t[A_{i-1}m^3 + i]$,
- $\text{R}_i = A[S_{i-2}m^3 + i]$,
- $\text{C}_i = [S_{i-2}m^3 + i]$,
- $w(\text{L}_i, \text{R}_i, \text{C}_i) = 0$ if $i = 1$,
- $w(\text{L}_i, \text{R}_i, \text{C}_i) = \text{C}_i[\text{eq}-1]$ if $i > 1$

**Claim:**

There is a $O$-weight triangle $L_i[S], R_i[T], S_i[Q]$ for some $i \in [n]$

$\iff$ there are $m_1, m_2, m_3$ s.t. $A[m_1] + B[m_2] = C[m_3]$
\[ \Rightarrow A_{i+j} + B_{i+j} - C_{(i+j)} = 0 \]
\[ \Rightarrow A_{(i-1)j} + B_{(i-1)j} - C_{(i-1)} = 0 \]

**Proof:**

1. Consider some \( m_1, m_2 \). Let \( A[m_{1+n}] + B[m_{2+n}] = C[m_{1+n}] \)
2. Let \( m_2 = (i-1)j + t \) for some \( t \in \{0, \ldots, j-1\} \). Let \( m_1 = (s-1)j + t' \) for some \( t' \in \{0, \ldots, j-1\} \).
3. Consider \( G_i^L \in R_i^{[t]} \).
4. \( w[R_i^{[t]}, S_i, [m_{1+n}], [m_{2+n}]] = B_{[m_{2+n}]} = B_{[m_{2+n}]} \)
5. \( -C[(s-1)j + t'] = -C[(s-1)j + t + (i-1)j] = -C[m_{1+n} + m_{2+n}] = 0 \)
6. Weight of triangle: \( A[m_{1+n}] + B[m_{2+n}] - C[m_{1+n} + m_{2+n}] = 0 \)

Running time: By assumption, \( O \) weight through graph with \( O(n^{1/2}) \) edges takes time \( O(n^{1/2} \log n) \) \( O(n^{1/2} - \epsilon) \)

Repeat for each \( i \in [n] \) and \( t \in [n^{1/2}] \).