Lemma (Transitivity) If \( A \leq B \) and \( B \leq C \), then \( A \leq C \).

If \( A \leq B \), \( B \leq C \), \( C \leq A \), then \( A, B, C \) are "subcubic equivalent".

Consequence: If one of \( A, B, C \) admits an \( O(n^{3/8}) \)-time algorithm, then all of \( A, B, C \) do.

Def. A subcubic reduction from problem \( P \) to problem \( Q \) exists \((P \leq Q)\) if there is an alg \( A \) with oracle access to \( Q \) s.t.

- For every instance \( x \) of \( P \), \( A(x) \) solves problem \( P \) on \( x \).
- Excluding oracle calls, \( A \) runs in time \( O(n^{3/8}) \) on instances of size \( n \).

For every \( \epsilon \) there is a \( \delta > 0 \) s.t. for every instance \( x \) of \( P \) of size \( n \), \( \sum_{i=1}^{n} i \leq n^{3/8} \) if \( n = n_1 + n_2 \).
Given weighted directed graph $G = (V, E)$

**APSP**

Task: Given a directed graph, compute the length of the shortest path from $u$ to $v$

Given $n 	imes n$ matrices $A, B$

Compute matrix $C$

$C[i, j] = \min (A[i, k] + B[k, j])$

$C = A \odot B$

Given a directed graph with node set $V = I \cup J \cup K$

Task: For every $i \in I$ and $j \in J$, decide if there is a negative-weight path $i \rightarrow j$

Task: Decide if there is a triangle $V \rightarrow V \rightarrow V$ of total negative weight

Observation: $\forall v \in V: \text{dist}(v^{(1)}, v^{(4)}) < 0$

$\iff G$ contains a negative-weight triangle including $v$

Reduction:

- Construct $G'$
- Compute APSP on $G'$
- Look at $\text{dist}(v^{(1)}, v^{(4)})$ for every node $v$
- Output 'yes' if $v < 0$ for some node $v$
- Output 'no'
From All-Pairs Neg Triangle to Neg Triangle

First assume neg triangle alg. also outputs one neg triangle (if it exists)

- Initialize C as an all-zero matrix

- Split each of I, J, K into \(\leq \frac{s}{3}\) parts of size \(s\)
  
  \[I = I_1, I_2, \ldots, I_{\frac{s}{3}}; J = J_1, J_2, \ldots, J_{\frac{s}{3}}; K = K_1, K_2, \ldots, K_{\frac{s}{3}}\]

  \(\Rightarrow\) \((\frac{n}{s})^3\) triples of the form \((I_x, J_x, K_z)\)

- For each triple \((I_x, J_x, K_z)\)
  
  While \(G[I_x, J_x, K_z]\) contains a neg triangle
  
  Find a neg triangle \((i, k, j) (k, i, j) (j, i)\)

  Set \(C[i, j] = 1\)

  Remove \((j, i)\) from the graph

Correctness: algorithm terminates (can remove at most \(n^2\) edges)

- if \(i, k, j\) is neg triangle, it will be found

Running Time

\(#\) oracle calls: \(\text{FindNT}(s)\)

\(= (\#\text{triples} \times \#\text{triangles found}) \cdot \text{FindNT}(s)\)

\(= (\frac{n^3}{s} + n^2) \cdot \text{FindNT}(s)\)

[Assume \(\text{FindNT}(n) = n^2 \cdot \varepsilon\)]

Set \(s = n^{\frac{3}{\varepsilon}}\)

\(= O(n^2 \cdot (n^{\frac{3}{\varepsilon}})^{3\varepsilon}) = O(n^3 - \varepsilon)^{\varepsilon}\)
From All-Pairs Neg Triangle to Neg Triangle

Now: Finding Neg Triangle using decision algorithm.

Partition each node set into two sets of same size each:

$I = I_1 \cup I_2$

$J = J_1 \cup J_2$

$K = K_1 \cup K_2$

For each triple $(I_a, J_b, K_c)$:

check if subgraph $G[I_a \cup J_b \cup K_c]$ contains any $\Delta$

For one triple $(I_a, J_b, K_c)$ that contains a neg triangle:

Recurse on $G[I_a \cup J_b \cup K_c]$

[Base case: If $I, J, K$ constant size then use brute-force approach to find and output neg $\Delta$]

Running Time:

$T_{FindNT}(n) \leq 2^3 T_{DecideNT}(\frac{n}{2}) + T_{FindNT}(\frac{n}{2}) + \alpha n^2$

$= O(T_{DecideNT}(\frac{n}{2}))$

$= O(\sum_{\text{DecideNT}(\frac{m}{2})}) = O(T_{DecideNT}(n) + n^2)$