Exercise 1 (7 points) Recall the following formal definition of subcubic reductions: Let $A$ and $B$ be computational problems with a common size measure $n$ on inputs. We say that there is a subcubic reduction from $A$ to $B$ if there is an algorithm $A$ with oracle access to $B$ satisfying three properties:

- For every instance $x$ of $A$, $A(x)$ solves the problem $A$ on $x$.
- For some $\gamma > 0$, $A$ runs in $O(n^{3-\gamma})$ time on instances of size $n$.
- For every $\varepsilon > 0$ there is a $\delta > 0$ such that for every instance $x$ of $A$ of size $n$ we have $\sum_i n_i^{3-\varepsilon} \leq n^{3-\delta}$, where $n_i$ is the size of the $i$th oracle call to $B$ in $A(x)$.

We use the notation $A \leq B$ to denote the existence of a subcubic reduction from $A$ to $B$. Prove that subcubic reductions are transitive. In other words, prove that if $A \leq B$ and $B \leq C$ then $A \leq C$.

Solution:
By definition, we have:

$$(D_1)$$ There is some $\gamma > 0$ such that for every $\epsilon > 0$ there exists a $\delta > 0$ so that for large enough $n$ there exist $\{n_i\}$ with $\sum_i n_i^{3-\varepsilon} \leq n^{3-\delta}$ and an oracle algorithm $A_B$ for $A$ which on instances of size $n$ runs in $O(n^{3-\gamma})$ time and makes oracle calls to $B$ with sizes $n_i$. 

(D2) There is some $\gamma' > 0$ such that for every $\epsilon' > 0$ there exists a $\delta' > 0$ so that for all large enough $n_i$ there exist $\{n_{ij}\}$ with $\sum_j n_{ij}^{3-\epsilon'} \leq n_i^{3-\delta'}$ and an oracle algorithm $B_C$ for $B$ which on instances of size $n_i$ runs in time $O(n_i^{3-\gamma'})$ and makes oracle calls to $C$ with sizes $n_{ij}$.

(D3) We will show that: There is some $\gamma'' > 0$ such that for every $\epsilon'' > 0$ there exists a $\delta'' > 0$ so that for all large enough $n$ there exist $\{n_{ij}\}$ with $\sum_{ij} n_{ij}^{3-\epsilon''} \leq n^{3-\delta''}$ and an oracle algorithm $A_C$ for $A$ which on instances of size $n$ runs in time $O(n^{3-\gamma''})$ and makes oracle calls to $C$ with sizes $n_{ij}$.

Let $\epsilon'' > 0$ be given. Now we just replace oracle calls for $B$ in $A_B$ by oracle algorithm $B_C$. This gives us the desired oracle algorithm $A_C$, note that it only uses oracle calls to $C$. The running time of $A_C$ is $O(n^{3-\gamma''} + \sum_i n_i^{3-\gamma'})$ and it makes oracle calls to $C$ with input sizes $n_{ij}$. By $D_1$, there exists some $\alpha > 0$ such that $\sum_i n_i^{3-\gamma'} \leq n^{3-\alpha}$. By setting $\gamma'' = \min(\alpha, \gamma')$, we get that $A_C$ runs in time $O(n^{3-\gamma''})$.

Now we consider the quantity $\sum_i \sum_j n_{ij}^{3-\epsilon''}$. Let $\beta > 0$ be the value corresponding to $\epsilon'' > 0$, as in $D_2$. Thus gives us that $\sum_j n_{ij}^{3-\epsilon''} \leq n_i^{3-\beta}$. Thus $\sum_i \sum_j n_{ij}^{3-\epsilon''} \leq \sum_i n_i^{3-\beta}$. Now we pick $\epsilon = 3\beta$ in $D_1$. This gives us a $\delta''$ such that $\sum_i \sum_j n_{ij}^{3-\beta} \leq n^{3-\delta''}$. Thus $\sum_i \sum_j n_{ij}^{3-\epsilon''} \leq n^{3-\delta''}$.

Exercise 2 (8 points) The Metricity Problem is defined as follows: Given an $n \times n$ matrix $A$ with entries in $\{0, \ldots, |v|^c\}$ for some constant $c > 0$, decide whether $\forall i, j, k \in [n]: A_{ij} \leq A_{ik} + A_{kj}$. Prove that Metricity Problem is equivalent to APSP under subcubic reductions.

**Hint:** Solve it using Min-Plus Product and reduce Negative Triangle to it.

**Solution:**
In the Min-Plus problem we compute $\min_k(A_{ik} + A_{kj})$ for all $i, j \in [n]$. Hence to check $\forall i, j, k \in [n]: A_{ij} \leq A_{ik} + A_{kj}$, it is enough to check if $A_{ij} \leq \min_k(A_{ik} + A_{kj})$. This checking only takes $O(n^2)$ time if we have solved the Min-Plus problem. Thus there is a subcubic reduction from Metricity Problem to Min-Plus problem.

Now we show the other direction. Let $G = (V, E)$ be a given graph with edge weights $w : E \rightarrow Z$ such that for all $e \in E$, $w(e) \in [-M, M]$ for some $M > 0$. Build a tripartite graph with $n$ node partitions $I, J, K$ and edge weights $W(\cdot)$ so that for any $i \in I, j \in J, k \in K$, $W(i, j) = 2M + w(i, j), W(j, k) = 2M + w(j, k)$ and $W(i, k) = 4M - w(k, i)$. For all pairs of distinct nodes $a, b$ so that $a, b$ are in the same partition, let $W(a, b) = 2M$. Finally, let $W(x, x) = 0$ for all $x$. For any three vertices $x, y, z$ in the same partition $W(x, y) + W(y, z) = 4M > 2M = W(x, z)$.

Consider triples $x, y, z$ of vertices so that $x$ and $y$ are in the same partition and $z$ is in a different partition. We have: $W(x, z) + W(z, y) \geq M + M = 2M = W(x, y)$ and $W(x, z) - W(y, z) \leq 2M = W(x, y)$.

With little more effort, we can also show that the only possible triples which could violate the triangle inequality are triples with $i \in I, j \in J, k \in K$, and $W$ is not a metric iff there exist $i \in I, j \in J, k \in K$ such that $W(i, j) + W(j, k) < W(i, k)$. That is, $W$ is not a metric if and only if $w(i, j) + w(j, k) + w(k, i) < 0$ and $i, j, k$ is a negative triangle in $G$.

Exercise 3 (9 points) Recall the following problem defined on the previous exercise sheet:
Hitting Set Problem: Given two lists of \( n \) subsets over a universe \( U \) of size \( d \), determine if there is a set in the first list that intersects every set in the second list, i.e. a “hitting set”.

The HSH (Hitting set Hypothesis) states that the Hitting Set Problem cannot be solved in time \( O(n^{2-\epsilon}) \cdot \text{poly}(d) \). Prove that HSH implies OVH.

Hint: In the lecture, we showed a reduction from All-Pairs-Negative-Triangle to Negative-Triangle. The same kind of reduction can work here.

Solution:
Assume that OVH fails. Thus Orthogonal vectors can be solved in time \( O(n^{2-\epsilon} \cdot \text{poly}(d)) \).

Let \( A, B \) be an instance of Hitting Set where \( |A| = |B| = n \). Let \( s \) be a parameter to be set later. Partition \( A \) into \( s \) sets \( A_1, \ldots, A_s \) of size at most \( \lceil \frac{n}{s} \rceil \) each. Similarly, partition \( B \) into \( B_1, \ldots, B_s \) of size at most \( \lceil \frac{n}{s} \rceil \) each. Now, for every choice of \( i, j \in [s] \) in turn: while \( (A_i, B_j) \) contains an orthogonal pair \((u,v)\), remove \( u \) from \( A \) (and hence from all \( A_k \)) and ask about \((A_i, B_j)\) again; if no orthogonal pair is found, continue to the next choice of \((i,j)\). If at the end of this procedure \( A \) contains some \( u \), then \( u \) must be nonorthogonal to all vectors in \( B \), and hence the this Hitting Set instance is a “yes” instance. Otherwise, if \( A \) is empty, then every \( u \in A \) was orthogonal to some \( v \in B \) and the Hitting Set instance is a “no” instance.

The running time is as follows: every call to the Orthogonal vectors problem either returns an orthogonal pair \((u,v)\) or determines that no such pair exists in \( A_i \times B_j \). The number of times an orthogonal pair can be returned is at most \( s \) since when \((u,v)\) is discovered, \( u \) is removed from \( A \). On the other hand, each \((A_i, B_j)\) instance can be a “no”-instance of Orthogonal vectors at most once, so that the number of calls that return a “no” is at most \( s^2 \). Thus, if we set \( s = \sqrt{n} \), the number of instances created of Orthogonal vectors is at most \( 2n \) and their sizes are all at most \( 2\sqrt{n} \). Thus Hitting Set can be solved in time \( \sum_{i=1}^{2n} (\text{poly}(d) \cdot (\sqrt{n})^{2-\epsilon}) \leq O(n^{2-\frac{\epsilon}{2}} \cdot \text{poly}(d)) \).

Thus failure of OVH implies failure of HSH. Hence HSH implies OVH.

Exercise 4 (16 points) Recall the following problem from the lecture. Let \( A \) be a matrix with entries in \( \{[n^c],-[n^c]+1, \ldots, [n^c]\} \) for some constant \( c > 0 \). In the Maximum Submatrix problem the task is to find the maximum sum of all entries of any submatrix of \( A \). A submatrix here means a choice of some consecutive rows and some consecutive columns.

a) (7 points) Describe an \( O(n^3) \) time algorithm for the maximum submatrix problem (solve it directly without any reductions).

Solution:
Let \( A \) be the given matrix. We first compute all \( n^2 \) prefix sums of \( A \), i.e., all sums of submatrices with one corner equal to the upper left corner of \( A \). This can be done in time \( O(n^2) \). We shall construct \( \binom{n}{2} + n \) one dimensional arrays of length \( n \). Each such array corresponds to a choice of a pair \((k,l)\) of integers from \([n]\), where \( k \leq l \). Array corresponding to pair \((k,l)\) would equal to sum of rows of \( A \) from \( k \) to \( l \). In case \( k = l \), this array would be just the \( k \)-th row of \( A \). All these arrays can be constructed in \( O(n^3) \) time, using the precomputed prefix sums. Now it is clear that maximum sub-sub-matrix would correspond to a maximum sum subarray from one of these \( O(n^2) \) arrays. And this problem of maximum subarray can be solved in \( O(n) \) time using Kadane’s algorithm. See [http://en.wikipedia.org/wiki/Maximum_subarray_problem](http://en.wikipedia.org/wiki/Maximum_subarray_problem) for this algorithm. Thus the whole algorithm runs in time \( O(n^3) \). See also
b) (9 points) A submatrix of $A$ is called **centered** if it contains the center (entry $A_{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor}$) of matrix $A$. Now the problem is to find a **Maximum Centered Submatrix**. Show that if APSP has subcubic algorithm then so does this problem, i.e., show that there is subcubic reduction from this problem to APSP.

**Solution:**

A centered submatrix can be described by quadruple $(x_1, y_1, x_2, y_2)$ of integers from 1 to $\frac{n}{2}$. The centered submatrix corresponding to quadruple $(x_1, y_1, x_2, y_2)$ looks like below:

![Centered Submatrix Diagram](image.png)

Sum of elements in the centered submatrix can be divided into four quadrant I, II, III, IV, as shown above. Now we create a graph $G$ which would have 5 layers $\frac{n}{2}$ of nodes. Edges would be only between consecutive layers. First and fifth layer corresponds to choice of $x_1$. Three layers in between first and third layer correspond to choices of $y_1$, $x_2$ and $y_2$. Weight of edge between consecutive layers would be the negative of weight of corresponding. It can be summarized as below.

![Graph Diagram](image.png)

Now it is clear that value of maximum centered submatrix is negative of value of shortest path from some $x_i$ in first layer to same $x_i$ in last layer. Thus Maximum Centered Submatrix can be reduced to APSP by subcubic reductions.

**Exercise 5 (10 Bonus points)** Let $G = (V,E)$ be a directed weighted graph with edge weights in $\{-\lfloor n^c \rfloor, \ldots, \lfloor n^c \rfloor\}$. The **Betweenness Centrality** of a given node $v \in V$ is the number of pairs $s, t$ such that $v$ lies on a shortest path from $s$ to $t$:

$$BC(v) = |\{(s,t) \mid s, t \in V \setminus \{v\}, s \neq t: d(s,t) = d(s,v) + d(v,t)\}|$$
Show that computing $BC(v)$ is equivalent to APSP under subcubic reductions.

**Hint:** Modify the reductions between Radius and APSP shown in the lecture.

**Solution:**

First we reduce negative triangle to Betweenness Centrality.

Let $(G = (V, E), w)$ be the input instance of Negative Triangle. In particular, $n = 2^k + 1$ is the number of nodes of $G$. WLOG assume that weight of any negative triangle is atmost -2 (multiply weights by 2 to achieve this).

We construct a weighted directed graph $(G', w')$ as follows. Graph $G'$ contains four sets of nodes $I, J, K,$ and $L$ (layers). Each layer contains a copy of each node $v \in V$. Let $v_I$ be the copy of $v$ in $I$, and define analogously $v_J, v_K$ and $v_L$. Let $Q = \Theta(M)$ be a sufficiently large integer. For each edge $uv \in E$, we add to $G'$ the edges $u_Iv_J, u_Jv_K,$ and $u_Kv_L$, and assign to those edges weight $2Q + w(uv)$.

We add to $G'$ a dummy node $b$, and edges $v_Ib$ and $bv_L$ for any $v \in V$, of weight $3Q - 1$ and $3Q$, respectively.

We also add to $G'$ two sets of nodes $Z = \{z_0, \ldots, z_k\}$ and $O = \{o_0, \ldots, o_k\}$. For any $v \in V$, we add the following edges of weight $3Q - 1$ to $G'$. Let $v_0, v_1, \ldots, v_k$ be a binary representation of $v$ (interpreted as an integer between 0 and $n - 1 = 2^k - 1$). For each $j = 0, \ldots, k$, we add edges $v_Iz_j$ and $o_jv_L$ if $v_j = 0$, and edges $v_Io_j$ and $z_jv_L$ otherwise.

We also add edges $o_jz_j$ and $z_jo_j$ of weight $3Q - 1$ for $j = 0, \ldots, k$.

Observe that $k = O(\log n)$, hence there are $O(n \log n)$ edges of the latter type.

On $(G', w')$ we compute $BC(b)$, and output YES to the input Negative Triangle instance iff $BC(b) < n$.

Let us prove its correctness. The only paths passing through $b$ are of the form $s_I, b, t_L$ and have weight $6Q - 1$. For $s \neq t$, there must exist a node $w \in Z \cup O$ such that $s_I, w, t_L$ is a path of cost $6Q - 2$. Therefore, the only pairs of nodes that can contribute to $BC(b)$ are of the form $(s_I, s_L)$. The shortest path of type $s_I, v_J, w_K, s_L$ has weight at most $6Q - 2$ if $s$ belongs to a negative triangle, and at least $6Q$ otherwise. Therefore $BC_{s_I, s_L}(b) = 1$ if $s$ does not belong to any negative triangle, and $BC_{s_I, s_L}(b) = 0$ otherwise. The correctness follows.

For the other direction, we can easily reduce Betweenness Centrality to APSP.