Problem 1, CS21 Set 6, Matt Lim

To show

$$\text{SUBGRAPH ISOMORPHISM} = \{(G, H) : G \text{ contains a subgraph isomorphic to } H\}$$

is NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. To prove the first part, we can simply give a certificate and a polynomial time verifier. The certificate is a subgraph of G and an isomorphism. The verifier will check all the edges of the subgraph to see if they correctly correspond with the isomorphism. This can clearly be done in polynomial time. To prove the second part, we show that CLIQUE is polynomial time reducible to SUBGRAPH ISOMORPHISM. Note that, instead of using the version of CLIQUE used in the lecture notes, we will use the following equivalent definition:

$$\text{CLIQUE} = \{(G, k) : G \text{ has a subgraph isomorphic to a } k \text{ clique}\}$$

This definition holds since if a graph G has a clique of size $> k$, then it has a clique of size k, which means graph G clearly has a subgraph isomorphic to a k clique. The following details the reduction:

Our reduction function f will map (G, k) to (G, H). Note that if k is greater than the number of vertices in G, we simply stop and map to no. If not, then, using k, we will generate a k clique. This step is polynomial because k is less than the size of the input, and to generate a k clique we just put all possible edges between k vertices which is of order k^2. We will let this generated k clique be our H. As for G, we just copy it over (we use the same graph) which is clearly polynomial. So we have described our reduction function and shown that it is polynomial.

Now we will show that this reduction function f maps yes to yes and no to no. First we will show that it maps yes to yes. So, given that we have a yes instance (G, k) of CLIQUE, we want to show that this means we have a yes instance (G, H) of SUBGRAPH ISOMORPHISM. So, if we have a yes instance (G, k) of CLIQUE, that means that G has a subgraph isomorphic to a k clique. Then, since we let H be this k clique, it follows that G contains a subgraph isomorphic to H. So yes maps to yes.

Now we will show that this reduction function f maps no to no. To prove this, it suffices to show that if our generated (G, H) is in SUBGRAPH ISOMORPHISM, then its equivalent (G, k) is in CLIQUE. So, let (G, H) be in SUBGRAPH ISOMORPHISM. By our reduction function, we know that H is a k clique. And since we have that G has a subgraph isomorphic to H, we also have that G has a subgraph isomorphic to a k clique, making (G, k) be in CLIQUE. So no maps to no.

We proved that our reduction function is polynomial above. So finally, we can conclude that SUBGRAPH ISOMORPHISM is NP-complete.

Problem 2, CS21 Set 6, Matt Lim

First we will establish some things. Let $C = \{S_1, S_2, S_3, \dots, S_n\}$ be a collection of sets and define $U = \cup_i S_i$. We say that $T \subseteq C$ is a *cover* if every $u \in U$ occurs in at least one $S_i \in T$. To show

$$\text{SET COVER} = \{(C, k) : \text{there is a cover } T \subseteq C \text{ with } |T| \leq k\}$$

is NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. To prove the first part, we can simply give a certificate and a polynomial time verifier. The certificate is a cover $T \subseteq C$ with $|T| \leq k$. The verifier will check that $T \subseteq C$ by iterating through the elements of T and checking if they are in C, which is clearly in polynomial time, and will check that $|T| \leq k$ by iterating through the elements of T and counting them, then checking that the size is less than or equal to k (an operation that is also clearly polynomial). The verifier will also check that the union of T is equal to U to make sure that T is a cover, another polynomial operation. To prove the second part, we show that VERTEX COVER (VC), which is defined below as

$$\text{VC} = \{(G, k) : G \text{ has a VC of size} \leq k\}$$

is polynomial time reducible to SET COVER. The following details the reduction:

Our reduction function f will map (G, k) to (C, k). We will do this in the following way. For each vertex $v_i \in G$, we will generate a set S_i to put in C. This set will be composed of all the edges that v_i is connected to. So, for example, if v_j is connected to v_k and v_n $(j \neq k \neq n)$, $S_j = \{(v_j, v_k), (v_j, v_n)\}$. Then $U = E$, where E is all of the edges in G. k will not change in this reduction function.

Now we will show that this reduction function f maps yes to yes and no to no. First we will show it maps yes to yes. So, given that we have a yes instance (G, k) of VC, we want to show that this means we have a yes instance (C, k) of SET COVER. So, assume we have a yes instance (G, k) of VC. That means that G has a subset of $\leq k$ vertices that covers all the edges. Then, given our reduction function f, that means that there is a subset T of $\leq k$ sets (of C) whose union contains all the edges of G. That is, T equals E which equals U. So we have that there exists a T that is a cover and that is a subset of C whose size is less than or equal to k. So yes maps to yes.

Now we will show that this reduction function f maps no to no. To prove this, it suffices to show that if our generated (C, k) is in SET COVER, then its equivalent (G, k) is in VC. So, let (C, k) be in SET COVER. We have that C contains $\leq k$ sets that contain all the edges of G. And we can map each of these sets (as well as all the other sets) to a single vertex in G, where that vertex contains all the edges in the set that maps to it. So we have that there are $\leq k$ vertices in G that cover all the edges in G. So no maps to no.

Now we must only prove that our reduction function f is polynomial time computable. Let n be the number of vertices in G. Then mapping the edges of each vertex to the set C will be bounded by Bn^2, where B is the maximum number of edges between two vertices and n is the number of vertices. And we just copy the k, which is also polynomial. So we have that the reduction function is polynomial.

Finally, we can conclude that SET COVER is NP-complete.

Problem 3, CS21 Set 6, Matt Lim

To show that MIN BISECTION:

$$\{(G, k) : G \text{ is a connected graph having a bisection with at most } k \text{ edges crossing it}\}$$

is NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. To prove the first part, we can simply give a certificate and a polynomial time verifier. The certificate is a bisection with at most k edges crossing it, in the form of the 2 bisected partitions (Given as the set of vertices in one partition, the set of vertices in the other partition, and the set of all edges). The verifier will iterate through all the edges, adding 1 to a counter (that starts at 0) every time it encounters an edge with one vertex in one partition and the other vertex in the other partition. It will then check that this counter is at most k. It will also check that the vertices in both partitions are unique and that both partitions contain the same number of vertices. This verifier clearly runs in polynomial time. To prove the second part, we show that

$$\text{MAX CUT} = \{(G = (V, E), k) : \text{there is a cut } S \subset V \text{ with at least } k \text{ edges crossing it}\}$$

is polynomial time reducible to MIN BISECTION. The following details the reduction.

Our reduction function f will map (G, k) to (G', k'). This will be done in the following way. First, let $n = |V|$, the number of vertices. We will generate G_1 from G by adding n vertices to G (without adding any new edges), making the size of G_1 $2n$. Then we will generate G_2 from G_1 by flipping all the edges (get rid of existing edges, add all other possible ones). Let us clarify that last step. So, let B be the max number of edges between 2 vertices in G_1. Then, if $(u, v) \in G_1$ had 3 edges, then $(u', v') \in G_2$ will have $B - 3$ edges. Now we have that G_1 has a bisection with at least k edges crossing it. This is because, when generating G_1, we can add the new n vertices such that there exist an equal number of vertices on both sides of the max cut. We also have that G_2 has a bisection with at most $Bn^2 - k$ edges crossing it. This is true because the amount of crossed bisection edges that we add is bounded above by Bn^2, and the amount of crossed bisection edges that we remove is bounded below by k. Now we will generate G_3 from G_2 by adding a $2n$ clique on top of G_2. This gives us that G_3 has a bisection with at most $n^2 + Bn^2 - k$ edges. This is clearly true since the amount of crossed bisection edges we add is bounded above by n^2. Now we will let $G' = G_3$ and $k' = n^2 + Bn^2 - k$.

Now we will show that this reduction function f maps yes to yes and no to no. First we will show it maps yes to yes. So, given that we have a yes instance (G, k) of MAX CUT, we want to show that this means we have a yes instance (G', k') of MIN BISECTION. So, assume (G, k) is a yes instance of MAX CUT. Recall that our reduction function f mapped (G, k) to (G', k'). G' has $2n$ vertices, an even number of vertices, so it is possible for it to have a bisection. And we showed while describing the reduction function that G' in fact does have a bisection with at most $k' = n^2 + Bn^2 - k$ edges crossing it. G' is also connected, since we added the $2n$ clique. So we have that G' is a connected graph having a bisection with at most k' edges crossing it. So (G', k') is a yes instance of MIN BISECTION. So we have that yes maps to yes.

Now we will show that this reduction function f maps no to no. To prove this, it suffices to show that if our generated (G', k') is in MIN BISECTION, then its equivalent (G, k) is in MAX CUT. So, let (G', k') be in MIN BISECTION, where G' and k' are as described in the reduction function. Now let us reconstruct G from G', basically by reversing f. We remove the $2n$ clique, flip the edges, and get rid of n vertices that are not connected to anything. Right before that last step, we have our graph G_1 that has $2n$ vertices, with a bisection of at least k edges crossing it. And since the n vertices that we remove do not affect the edge count at all, removing them leaves us with a graph where there exists a cut with at least k edges crossing it. So (G, k) is in MAX CUT and we have that no maps to no.

Now we must only prove that our reduction function f is polynomial time computable. To do this, we will consider each step. First, we add n vertices. This is clearly polynomial. Then, we flip all the edges. So, we get rid of a constant number of edges (those programmed into the graph) and add a number of edges bounded by Bn^2, both of which are polynomial. Finally, we add a $2n$ clique, which adds a number of edges bounded by n^4, which is polynomial. So we have that the reduction function is polynomial.

Finally, we can conclude that MIN BISECTION is NP-complete.

Problem 4, CS21 Set 6, Matt Lim

(a) To show that

$$\text{PARTITION} = \{(S = \{a_1, a_2, \cdots, a_n\}) : S \text{ is partitionable}\}$$

is NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. To prove the first part, we can simply give a certificate and a polynomial time verifier. The certificate is a set S of positive integers that is partitionable, given in the form of two subsets T and $S - T$. The verifier will iterate through T, checking if all the elements are positive and summing them up. The iterator will then iterate through $S - T$, checking if all the elements are positive and summing them up. It will then compare the sums and see if they are equal. All these operations are polynomial, To prove the second part, we show that

$$\text{SUBSET SUM} = \{(S = \{a_1, a_2, \cdots, a_n\}, B) : \text{ there is a subset of } S \text{ that sums to } B\}$$

is polynomial time reducible to PARTITION. The following details the reduction.

Our reduction function f will map (S, B) to S'. It will do this by having $S' = S \cup \{P, 2B\}$, where $P = \sum_{i=1}^{n} a_i$.

Now we will show that this reduction function f maps yes to yes and no to no. First we will show it maps yes to yes. So, given that (S, B) is a yes instance of SUBSET SUM, we want to show that this means that we have a yes instance (S') of PARTITION. So, assume that (S, B) is a yes instance of SUBSET SUM. Then we have that there is a subset of S whose elements sum to B. Now consider the S' our reduction function gives us, $S' = S \cup \{P, 2B\}$. We have that there exists a subset of S that sums to B. We also have that there exists a different subset (that shares no elements with the previously mentioned subset that sums to B) of S that sums to $P - B$. So, if we let T be the subset of S' that includes the subset that sums to B unioned with P, and let $S' - T$ be the subset of S' that includes the subset that sums to $P - B$ unioned with $2B$, we have that $\sum_{a \in T} a = \sum_{a \in S' - T} a = P + B$. So we have that S' is in PARTITION, and thus that yes maps to yes.

Now we will show that this reduction function f maps no to no. To prove this, it suffices to show that if our generated (S') is in PARTITION, then its equivalent (S, B) is in SUBSET SUM. So, let $S' = S \cup \{P, 2B\}$ be in PARTITION. We have that $P + 2B > P = \sum_{i=1}^{n} a_i$. This tells us that P and $2B$ cannot both be in one partition of S', since the rest of the elements sum up to something less than it. So, WLOG we will say that P is in $T \subseteq S'$ and $2B$ is in $S' - T$. We also have that $\sum_{a \in T} a = \sum_{a \in S' - T} a = P + B$. Then we can see that there must be some subset of elements in S that add up to B, because the sum of the elements in T must equal $P + B$, and the number P is already in that set. So we have that (S, B) is in SUBSET SUM and thus that no maps to no.

Now we must only prove that our reduction function f is polynomial time computable. To do this, we must simply consider the steps of the function. All we did was make a new set that was the union of the original set S and two new numbers. This process is clearly polynomial. So we have that our reduction function is polynomial.

Finally, we can conclude that PARTITION is NP-complete.

(b) To show that

$$\begin{aligned}\text{KNAPSACK} &= \{(c_1, c_2, \ldots, c_n, v_1, v_2, \ldots, v_n, V, C) : \exists\, T \subseteq \{1, 2, \ldots, n\} \\ &\quad \text{for which } \sum_{t \in T} v_t \geq V \text{ and } \sum_{t \in T} c_t \leq C\}\end{aligned}$$

is NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. To prove the first part, we can simply give a certificate and a polynomial time verifier. The certificate is a set of items from the knapsack (T in the definition of KNAPSACK), along with numbers V and C, whose total value is at least V without exceeding a budget C. So basically, it is T (along with V, C) as described in the definition of KNAPSACK. The verifier will sum the value of the items and ensure that it is greater than or equal to V and will also sum the total cost of the items and ensure that it does not exceed C. It will also verify that all the items (indices) are in the knapsack (valid indices). All these verifications are clearly polynomial. To prove the second part, we show that

$$\text{SUBSET SUM} = \{(S = \{a_1, a_2, \cdots, a_n\}, B) : \text{ there is a subset of } S \text{ that sums to } B\}$$

is polynomial time reducible to KNAPSACK. The following details the reduction.

Our reduction function f will map (S, B) to $(c_1, c_2, \cdots, c_n, v_1, v_2, \cdots, v_n, V, C)$ in the following way. It will map $C = V = B$ and map $c_i = v_i = s_i$.

Now we will show that this reduction function f maps yes to yes and no to no. First we will show that it maps yes to yes. So, given that (S, B) is a yes instance of SUBSET SUM, we want to show that this means that we have a yes instance $(c_1, c_2, \cdots, c_n, v_1, v_2, \cdots, v_n, V, C)$ of KNAPSACK. So, assume that (S, B) is a yes instance of SUBSET SUM. Then we have that there is a subset of S whose elements sum to B. This means that there exists some subsets of cs and vs, some $T \subseteq \{1, 2, \cdots, n\}$, such that $\sum_{t \in T} v_t = B \geq V$ and $\sum_{t \in T} c_t = B \leq C$, since we made $C = V = B$. So we have that yes maps to yes.

Now we will show that this reduction function f maps no to no. To prove this, it suffices to show that if our generated $(c_1, c_2, \cdots, c_n, v_1, v_2, \cdots, v_n, V, C)$ is in KNAPSACK, then its equivalent (S, B) is in SUBSET SUM. So, let $(c_1, c_2, \cdots, c_n, v_1, v_2, \cdots, v_n, V, C)$ be in KNAPSACK. Let $C = V = B$ and $s_i = v_i = c_i$. Then, since some subset of vs sum to B (and some subset of cs sum to B) then some subset of ss sum to B. So we have that (S, B), where S is the set made up of all the ss, is in SUBSET SUM. Thus we have that no maps to no.

Now we must only prove that our reduction function f is polynomial time computable. This is clearly the case, since we just map a set S to 2 new sets containing the same exact elements, then map $C = V = B$. Both processes are clearly polynomial. Thus we have that our reduction function is polynomial.

Finally, we can conclude that KNAPSACK is NP-complete.