Assignment 5 Solutions

1 NP completeness [10 marks]

Prove that the following problems are **NP**-complete.

(i) CliqueAndIS: Given a graph $G = (V, E)$ and a positive integer k, determine whether there is a clique of size at least k *and* an independent set of size at least k in G.

Solution. First we show that CliqueAndIS is in **NP**. We can design a polynomial-time verifier that requires as certificate the set S of k vertices in the clique, and the set T of k vertices in the independent set. The verifier can check in polynomial time that S is indeed a clique of size k and that T is an independent set of size k.

To show that CliqueAndIS is **NP**-complete, it suffices to show a polynomial-time reduction from a problem that we already know is **NP**-complete. We show that

$$\text{Clique} \leq_{\mathbf{P}} \text{CliqueAndIS}:$$

Let $G = (V, E)$ and k be any input to the Clique problem. Let $G' = (V', E')$ be the graph on $|V'| = |V| + k$ vertices with the new k vertices all being isolated vertices. (So that $E' = E$.) The graph G' constructed in this way always has an independent set of size k (namely, the k newly added vertices) and does not add any new edges, so G' has both a clique *and* an independent set of size k if and only if G has a clique of size k.

(ii) Subgraph: Given a graph $G = (V, E)$ and a graph $H = (V', E')$, determine if H is a subgraph of G—i.e., if there is a mapping π of the vertices in V' to the vertices in V such that for every $u, v \in V'$, $(\pi(u), \pi(v)) \in E$ if and only if $(u, v) \in E'$.

Solution. The problem Subgraph is in **NP** because we can design a verifier that requires as certificate the mapping $\pi : V' \to V$ that indicates where the subgraph H is located within G. This verifier can check in polynomial time that the mapping does correspond to a valid occurrence of H in G.

We show that Subgraph is **NP**-complete by showing that Clique $\leq_{\mathbf{P}}$ Subgraph. For any instance G and k of the Clique problem, we construct the input G, H to the Subgraph problem where $H = (V', E')$ is the clique on k vertices ($|V'| = k$ and $E' = \{(u, v) : u \neq v \in V'\}$). We can construct the graph H in polynomial time, and H is a subgraph of G if and only if G contains a clique of size k, so this is a polynomial-time reduction.

2 More NP completeness [10 marks]

Prove that the following problems are **NP**-complete.

(i) Clubs: Given

- a set of n persons that we associate with the numbers $\{1, 2, \dots, n\}$,
- a collection $\mathcal{C}$ of clubs, where each club is a subset $C \subseteq \{1, 2, \dots, n\}$ that represents the persons who are members of the club, and
- a positive integer k,

determine if there is a set $S \subseteq \{1, 2, \dots, n\}$ of size $|S| \le k$ such that every club contains at least one of the persons in S.

Solution. Clubs is in **NP** because we can design a verifier that requires as certificate a set S of size $|S| \le k$ that includes at least one member of every club; that a provided certificate satisfies this condition can be verified in polynomial time.

We then show that VertexCover $\le_{\mathbf{P}}$ Clubs. Let $G = (V, E)$ and k be the input to the VertexCover problem with $V = \{v_1, v_2, \dots, v_n\}$. Let us build the collection $\mathcal{C}$ of clubs where we create one club $C_e = \{i, j\}$ for each edge $e = (v_i, v_j) \in E$. The collection $\mathcal{C}$ is created in polynomial time, and there is a set S of size $|S| \le k$ that includes at least one member from each club if and only if the original graph G has a vertex cover of size k. (If G has such a vertex cover $\{v_{i_1}, \dots, v_{i_k}\}$ then $S = \{i_1, \dots, i_k\}$ contains a member from each club in $\mathcal{C}$ and, conversely, if $S = \{i_1, \dots, i_k\}$ contains a member from each club then the set $\{v_{i_1}, \dots, v_{i_k}\}$ is a vertex cover of G.)

(ii) EvenSplit: Given n integers $a_1, \dots, a_n$, determine whether there is a set $S \subseteq \{1, 2, \dots, n\}$ for which

$$\sum_{i \in S} a_i = \sum_{i \in \{1,2,\dots,n\} \setminus S} a_i.$$

Solution. The problem EvenSplit is in **NP** because we can design a verifier that asks for the set S as the certificate: in polynomial-time, the verifier can check whether the condition $\sum_{i \in S} a_i = \sum_{i \in \{1,\dots,n\} \setminus S} a_i$ is satisfied.

We now show that SubsetSum $\le_{\mathbf{P}}$ EvenSplit. Given an instance of the SubsetSum problem with positive integers $w_1, \dots, w_n$ and a target value T, define $N = \sum_{i=1}^{n} w_i$. We generate the input $a_1, \dots, a_{n+1}$ with $a_i = w_i$ for each $1 \le i \le n$ and $a_{n+1} = N - 2T$. The sum of all the elements satisfies $\sum_{i=1}^{n+1} a_i = N + (N - 2T) = 2N - 2T$ so the answer to the EvenSplit problem is `Yes` if and only if there is a subset $S' \subseteq \{1, 2, \dots, n+1\}$ such that

$$\sum_{i \in S'} a_i = \sum_{i \notin S'} a_i = N - T.$$

If there is a set $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i = T$, then the set $S' = S \cup \{n+1\}$ satisfies $\sum_{i \in S'} a_i = T + (N - 2T) = N - T$ so `Yes` instances to SubsetSum get transformed into `Yes` instances to EvenSplit.

Conversely, if there exists $S' \subseteq \{1, 2, \ldots, n+1\}$ for which $\sum_{i \in S'} a_i = N - T$, we can assume without generality that $S' \ni n+1$ (if not, simly exchange the sets S' and $\{1, 2, \ldots, n+1\} \setminus S'$). Then $S = S' \setminus \{n+1\}$ satisfies $\sum_{i \in S} w_i = \sum_{i \in S'} a_i - a_{n+1} = N - T - (N - 2T) = T$ so the result of our transformation is a `Yes` instance to EvenSplit only when the original instance to SubsetSum is also a `Yes` instance.

3 And even more NP-completeness... or not? [10 marks]

In the Clique3 problem, we are given a graph $G = (V, E)$ with maximum degree 3 and a positive integer k; we must determine if G has a clique of size at least k or not. (A graph G has *maximum degree* d if every vertex in G is incident to at most d edges.)

(i) Prove that Clique3 $\in$ **NP**.

Solution. We can design a verifier that requires as certificate the set S of k vertices that form a clique in G. It can check in polynomial time whether G contains an edge between every two vertices in S, so this is a polynomial-time verifier.

(ii) Here's a claimed proof that Clique3 is **NP**-complete. Explain why the argument is incorrect.

> We showed in part (i) that Clique3 is in **NP**. We know from lectures that Clique is **NP**-complete. All that remains is to show that there is a polynomial-time reduction from Clique3 to Clique. Let F be the (trivial) algorithm that takes in a graph G with vertices of degree at most 3 and a parameter k, and leaves both as-is. The algorithm F runs in polynomial time and gives a transformation from inputs of the Clique3 problem to inputs of the Clique problem, and the answer to these inputs is always identical. Therefore, this is a valid polynomial-time reduction and Clique3 is **NP**-complete.

Solution. To show that Clique3 is **NP**-complete, it suffices to show that Clique $\leq_{\mathbf{P}}$ Clique3—but the argument instead shows that Clique3 $\leq_{\mathbf{P}}$ Clique, which implies nothing about the **NP**-completeness of Clique3.

(iii) In the VertexCover3 problem, we are given a graph $G = (V, E)$ with maximum degree 3 and a positive integer k; we must determine if G has a vertex cover of size at most k or not. It is known that VertexCover3 is **NP**-complete, and for this question we may use this fact without proof.

Here's another claimed proof that Clique3 is **NP**-complete. Explain why the argument is incorrect.

> We already showed in part (i) that Clique3 is in **NP**. We complete the proof that it is **NP**-complete by giving a polynomial-time reduction from VertexCover3 to Clique3. Let F be the algorithm that transforms the input (G, k) into the input $(G, n - k)$. The algorithm F has polynomial-time complexity. And $C \subseteq V$ is a vertex cover in G if and only if $V \setminus C$ is a clique in G, so G has a vertex cover of size $\leq k$ if and only if it has a clique of size $\geq n - k$ and therefore our transformation gives polynomial-time reduction from VertexCover3 to Clique3.

Solution. The set $C \subseteq V$ is a vertex cover in G if and only if $V \setminus C$ is an *independent set* in G, not a clique. So if we replace "clique" with "independent set" throughout the proof, we would indeed have a valid proof that the problem INDEPSET3 of determining whether a graph of maximum degree 3 has an independent set of size at least k or not is **NP**-complete, but this result says nothing about CLIQUE3 itself.

(iv) Prove that CLIQUE3 $\in \mathbf{P}$.

Solution. Every vertex in a clique of size k is connected to at least $k-1$ other vertices, so a graph with maximum degree at most 3 cannot have any clique of size greater than 4. We can therefore answer `No` immediately on any input G, k with $k > 4$. For values of $k \leq 4$, we can enumerate all $\binom{n}{4} = O(n^4)$ sets of k vertices in the graph and check whether they form a clique. The resulting algorithm runs in polynomial time, so CLIQUE3 is in $\mathbf{P}$.

4 Almost acyclic graphs [10 marks]

In the AlmostDAG problem, we are given a directed graph $G = (V, E)$ and a positive integer k; we must determine if we it is possible to remove at most k edges from E to obtain a directed acyclic graph.

Prove that AlmostDAG is **NP**-complete.

Hint. You should consider using a reduction from VertexCover. See Piazza for a more detailed hint, if required.

Solution. AlmostDAG is in **NP** because we can define a verifier that asks for the set of k edges to remove to obtain a DAG as the certificate. As we saw in lecture 13, we can use DFS to check in polynomial time whether the directed graph G' obtained by removing the identified edges is acyclic or not. (Technically, in the lecture notes the algorithm only is specified when G' is *connected*: to extend the algorithm for arbitrary graph, run the same test on all the connected components of G'.)

We now show that VertexCover $\leq_{\mathbf{P}}$ AlmostDAG using the reduction provided in the hint. Given an instance G, k to the VertexCover problem with $G = (V, E)$ and $V = \{v_1, \ldots, v_n\}$, we produce an instance G', k to the AlmostDAG problem where the graph $G' = (V', E')$ is defined by the set of $2n$ vertices $V' = \{v_1, \ldots, v_n, v'_1, \ldots, v'_n\}$ and the edges are defined by

$$E' = \{(v_i, v'_i) : 1 \leq i \leq n\} \cup \{(v'_i, v_j), (v'_j, v_i) : (v_i, v_j) \in E\}.$$

We want to show that G has a vertex cover of size at most k if and only if we can remove k edges from G' to make it acyclic.

($\Rightarrow$). We first claim that if G contains a vertex cover of size at most k, then G', k is a `Yes` instance to AlmostDAG. Let $C \subseteq V$ be a vertex cover of G of size $|C| \leq k$. Define

$$F = \{(v_i, v'_i) : v_i \in C\}.$$

Then $|F| \leq k$ and we claim that the graph $G'' = (V', E' \setminus F)$ is acyclic. Indeed, a cycle T in G' must use some edge of the form (v'_i, v_j); the corresponding edge (v_i, v_j) must be in the graph G which means that at least one of v_i or v_j must be in C. But if $v_i \in C$, T cannot be a cycle because the only edge to v'_i is $(v_i, v'_i) \in F$. Similarly, if $v_j \in C$ then T cannot be a cycle because the only outgoing edge from v_j is $(v_j, v'_j) \in F$. Therefore, G'' must be acyclic, as claimed.

($\Leftarrow$). We now claim that if the constructed input G', k is a `Yes` instance to AlmostDAG, then G contains a vertex cover of size at most k. Let $F \subseteq E'$ be a set of $|F| \leq k$ edges for which $G'' = (V', E' \setminus F)$ is acyclic. If F contains any edge of the form (v'_i, v_j), then we can replace it with the edge (v_i, v'_i) instead and the graph G'' obtained by removing the edges in F is still acyclic because any cycle in G' that uses the edge (v'_i, v_j) must also use the edge (v_i, v'_i) since it is the only incoming edge to the vertex v'_i. After performing all these replacements, we obtain a set of edges in the form

$$F = \{(v_i, v'_i) : v_i \in C\}$$

for some set $C \subseteq V$ of size $|C| \leq k$. Lastly, we now claim that C is a vertex cover in G. Assume for contradiction that it is not; then there is some edge (v_k, v_ℓ) such that neither v_k nor v_ℓ are in C. But if that's the case, in the graph $G'' = (V', E' \setminus F)$ we have a cycle

$$v_k \to v'_k \to v_\ell \to v'_\ell \to v_k,$$

contradicting the fact that G'' is acyclic.

5 Programming question [10 marks]

In the CONSTRAINEDAPSP problem, we are given a directed weighted graph $G = (V, E)$ with positive edge lengths $w : E \to \mathbb{R}^{>0}$ and a subset $S \subseteq V$ of vertices; for each pair of vertices $u, v \in V$, we must determine the length $L(u, v)$ of the shortest path from u to v *that visits at least one of the vertices in S along the path*. When no such path exists for a given pair u, v, the answer is ∞.

(i) Design and analyze an algorithm that solves the CONSTRAINEDAPSP problem. You should aim for an algorithm with time complexity $O(n^3)$ or better. Ideally, your algorithm will have time complexity $o(n^3)$ when $|S| = o(n)$. If that's the case, provide the time complexity analysis of the algorithm in terms of both $|S|$ and n.

Solution. We will give an algorithm with time complexity $O(|S|n^2)$. We will first compute $L(s, v)$ for each $s \in S, v \in V$ using Dijkstra's algorithm and then reverse the edges of the graph to find $L(v, s)$ for each $v \in V, s \in S$. The solution for $u, v \in V$ will then be $\min_{s \in S} L(u, s) + L(s, v)$.

Algorithm 1: CAPSP($G = (V, E), S, w$)

$E' \leftarrow \emptyset$
$G' \leftarrow (V, E')$
$L \leftarrow |V| \times |V|$ array of ∞

for *each* $(u, v) \in E$ **do**
  Add (v, u) to E'

for *each* $s \in S$ **do**
  Run DIJKSTRA(G, w, s) to fill in $L(s, \cdot)$
  Run DIJKSTRA(G', w, s) to fill in $L(\cdot, s)$

for *each* $u, v \in V$ *where* $u \notin S$ *or* $v \notin S$ **do**
  for $s \in S$ **do**
    $L(u, v) \leftarrow \min(L(u, v), L(u, s) + L(s, v))$

Theorem 1. CAPSP *solves the* CONSTRAINEDAPSP *problem and has time complexity* $\Theta(|S|n^2)$ *where* $n = |V|$.

Proof. **Correctness:** Let $d(u, v)$ be the length of the shortest path $u \to v$ in G and $d'(u, v)$ be the same for G', so that $d(u, v) = d'(v, u)$. By the guarantee of Dijkstra's algorithm we know that for all $s \in S, v \in V, L(s, v) = d(s, v)$ and $L(v, s) = d'(s, v) = d(v, s)$.

Let $u, v \in V$ and suppose that for some $s \in S$, $u, p_1, \ldots, p_k, s, q_1, \ldots, q_\ell, v$ is a shortest path from u to v containing at least one vertex from S. Then $d(u, s) + d(s, v) = L(u, s) + L(s, v)$ is the length of this path since $u, p_1, \ldots, p_k, s$ must be a shortest path $u \to s$ in G and $s, q_1, \ldots, q_\ell, v$ must be a shortest path $s \to v$ in G. So in the final loop we will set $L(u, v) = L(u, s) + L(s, v)$.

Time Complexity: Reversing the edges to construction G' can be done in time $O(n+m)$ where $n = |V|, m = |E|$. The most efficient implementation of Dijsktra's algorithm (with a Fibonacci heap) has time $\Theta(m + n \log n)$, and it is run $2|S|$ times for a total of $\Theta(|S|(m + n \log n))$. Thus since $m \leq n^2$ we have total time complexity

$$\Theta((m+n) + |S|(m + n\log n)) = \Theta(|S|(n^2 + n\log n)) = \Theta(|S|n^2)\,. \qquad \square$$

Another, less efficient, solution is to use Floyd-Warshall on a transformation of G, where we copy all the vertices $V \setminus S$ along with all their edges, and then add the edges $S \to V'$. Define $G' = (V', E')$ by $V' = V \cup \{v' : v \in V \setminus S\}$ and $E' = E \cup \{(s, v') : s \in S, (s, v) \in E\} \cup \{(u', v') : (u, v) \in E\}$. Run Floyd-Warshall on the graph G' to get distances D. Then for all $u, v \in V$ we have $L(u, v) = D(u, v')$.

Informally, this solution works since for any path $u \to s \to v'$ in G' there is a path $u \to s \to v$ in G and vice versa. Any path $u \to v'$ in G' must have a vertex from S since there are no edges (u, v') in G'.

The time complexity of this algorithm is $\Theta(n^3)$ since the graph G' is of size $|V'| \leq 2|V| = 2n, |E'| \leq 2|E| = 2n$ and can therefore be constructed in time $O(n+m)$. Floyd-Warshall runs in time $\Theta(|V'|^3) = \Theta(n^3)$.