1 HW6, Problem 1

1. `Clique`

   - Decision version of `Clique`

     Given a graph $G = (V, E)$ and a target value k, does the graph have a clique (complete subgraph) on (at least) k vertices?

   - `Clique(D)` is in $\mathcal{NP}$

     A candidate solution t to this problem consists of a subset S of vertices. Given an instance (G, k) of `Clique(D)` and a candidate solution t, we can quickly confirm that (G, k) is a **yes** instance of `Clique(D)` —that is, G has a clique on k vertices— as follows: check that S contains k vertices; for every pair of vertices in S, confirm that there exists an edge between them. This requires, e.g., $O(n^2)$ time.

   - Reduction from `Independent Set(D)` to `Clique(D)`:

     Given an instance $(G = (V, E), k)$ of `IS(D)`, construct an instance $(G' = (V, E'), k)$ of `Clique(D)` where E' contains precisely the edges that do not appear in G. The transformation from G to its complement G' requires polynomial time.

   - Proof of equivalence of instances.

     Clearly an independent set S in G forms a clique in G', and vice versa. Equivalence of the instances is now straightforward.

2. `Subgraph Isomorphism`

   - Decision version of `Subgraph Isomorphism`

     Given a graph $G = (V, E)$ and another graph H, does G have a subgraph isomorphic to H?

   - `Subgraph Isomorphism(D)` is in $\mathcal{NP}$

     A candidate solution t to this problem consists of a subgraph $G' = (V', E')$ of G, where $V' \subseteq V, E' \subseteq E$ and a mapping f from the vertices in H to the vertices in G'. Given an instance (G, H) of `Subgraph Isomorphism(D)` and a candidate solution t, we can quickly confirm that (G, H) is a **yes** instance of the problem —that is, G contains a subgraph isomorphic to H— as follows: rename the vertices and edges in G' using the mapping f, and confirm that G' is identical to H.

   - Reduction from `Clique(D)` to `Subgraph Isomorphism(D)`:

     Given an instance (G, k) of `Clique(D)`, construct the input (G, H) to `Subgraph Isomorphism(D)`, where H is a clique on k vertices. The transformation requires polynomial time.

- Proof of equivalence of instances.

  If G contains a clique of k vertices, then H is a subgraph of G. Conversely, if H is a subgraph of G, then G has a clique on k vertices.

3. `Dense Subgraph`

   - Decision version of `Dense Subgraph`

     Given a graph $G = (V, E)$ and two integers a and b, does G have a subgraph on a vertices with at least b edges among them?

   - `Dense Subgraph(D)` is in $\mathcal{NP}$

     A candidate solution t to this problem consists of a subset S of vertices in G. Given an instance (G, a, b) of `Dense Subgraph(D)` and a candidate solution t, we can quickly confirm that (G, a, b) is a **yes** instance of the problem —that is, G contains a subgraph on a vertices with at least b edges among them— as follows: confirm S contains a vertices and ensure there are at least b edges among the a vertices (e.g., in time $O(m)$).

   - Reduction from `Clique(D)` to `Dense Subgraph(D)`

     Given an instance (G, k) of `Clique(D)`, construct the instance (G, a, b) of `Dense Subgraph(D)` where $a = k$, $b = \binom{k}{2}$. This requires $O(1)$ time.

   - Proof of equivalence of instances.

     A clique on k vertices is the complete graph on k vertices, hence has $\binom{k}{2}$ edges. Equivalence of the instances is now straightforward.

2 HW6, Problem 2

- Decision version of (undirected) `Dominating Set`

  Given an undirected graph $G = (V, E)$ and a target value k, does the graph have a dominating set of size at most k?

- `Dominating Set(D)` is in $\mathcal{NP}$

  A candidate solution t to this problem consists of a subset S of vertices. Given an instance (G, k) of `Dominating Set(D)` and a candidate solution t, we can quickly confirm that (G, k) is a **yes** instance of the problem —that is, G has a dominating set of size at most k— as follows: confirm that S contains k vertices; for every vertex $v \in V$, confirm that either $v \in S$ or there exists an edge (u, v) such that $u \in S$. This requires time $O(n + m)$.

- Reduction from `Vertex Cover(D)` to `Dominating Set(D)`:

  We will transform an arbitrary instance of `VC(D)` into an instance of `Dominating Set(D)`. Given a pair $(G = (V, E), k)$, which is the input to `VC(D)`, we will transform it into the input $(G' = (V', E'), k)$ of `Dominating Set(D)` using the hint: for every edge $e = (u, v) \in E$, introduce a node $x_e \in V'$, and edges (u, x_e) and (x_e, v).

  Then V' consists of all the vertices in V and the new vertices x_e, and E' consists of all the edges in E and the new edges as above. Clearly the transformation requires polynomial time.

- Proof of equivalence of instances.

  ⇒ Suppose G has a vertex cover S of size k. Then S is a dominating set of size k in G'. For suppose a vertex $u \in G'$ is not in S and none of its immediate neighbors is in S.

  - If $u \in V$, then none of the edges incident to u is covered by S in G.
  - If $u = x_e$ for some $e \in E$, then e is not covered in G.

  In both cases, we conclude that S is not a vertex cover, thus contradicting our initial assumption.

  ⇐ Suppose S' is a dominating set of size k in G'. Construct a vertex cover S in G as follows: for every $v \in S' \cap V$, include v in S. For every $x_e \in S'$, include one of the endpoints of $e \in E$ in S.

  Then the vertices in S form a vertex cover of size k in G. For suppose there is an edge $e = (u, v)$ not covered by S. Then $u, v \notin S'$, and $x_e \notin S'$ (otherwise, u or v would appear in S). Then S' is not a dominating set in G': vertex x_e is not in S', and none of its two immediate neighbors u,v are in S' either.

3 HW6, Problem 3

- Decision version of `node-disjoint paths`

  Given a directed graph $G = (V, E)$, a collection of paths $P = \{P_1, \ldots, P_c\}$ and a target value k, are there at least k node-disjoint paths in P?

- `Node-disjoint paths(D)` is in $\mathcal{NP}$

  A candidate solution t to this problem consists of k subsets $S_1, \ldots, S_k$ of vertices. Given an instance $((G, P), k)$ of `Node-disjoint paths(D)` and a candidate solution t, we can quickly confirm that $((G, P), k)$ is a **yes** instance of the problem —that is, G has k node-disjoint paths— as follows: first, confirm that each path S_j appears in the collection P; then for every vertex $i \in V$, ensure that it does not appear more than once in all of $S_1, \ldots, S_k$. Both of these steps can be implemented in polynomial time.

- Reduction from `Independent Set(D)` to `Node-disjoint paths(D)`:

  We will transform an arbitrary instance of `Independent Set(D)` into an instance of `Node-disjoint paths(D)`. Given a pair $(G = (V, E), k)$, which is the input to `IS(D)`, we will transform it into the input $(G' = (V', E'), P_1, \ldots, P_c, k)$ of `Node-disjoint paths(D)` using the hint: for every edge $e \in E$, introduce a node $e \in V'$; make G' the complete directed graph on V'.

  We now proceed to defining the collection of paths in G'. For every vertex $i \in S$, let $Inc(i)$ be the set of its incident edges; then $Inc(i)$ is a set of vertices in G'. Further, the vertices in $Inc(i)$ form a path in G' (the order of the vertices in $Inc(i)$ does not matter since G' is the complete graph on V'). We now define $c = n$, so that there are n possible paths in G', with path P_i defined by the set of vertices $Inc(i)$. Thus each path in G' corresponds to a vertex in G.

  Clearly the transformation can be completed in polynomial time.

- Proof of equivalence of instances.

  $\Rightarrow$ Suppose G has an independent set $S = \{v_1, \ldots, v_k\}$ of size k.
  We claim that the k paths defined by $Inc(v_1), \ldots, Inc(v_k)$ in G' are node-disjoint. For suppose a vertex $e \in V'$ appears in two paths, say, the paths corresponding to vertices i and j in S. Then $e \in Inc(i)$ and $e \in Inc(j)$, that is, e joins i and j. Hence S is not an independent set.

  $\Leftarrow$ Suppose there are k node-disjoint paths in G' from the collection of paths $P_1, \ldots, P_n$. We claim that the vertices in G corresponding to these paths form an independent set. For suppose P_i and P_j are among the k node-disjoint paths in G' but there is an edge $e = (i, j)$ between vertices i and j in G. Then $e \in Inc(i)$ and $e \in Inc(j)$. Since $P_i = Inc(i)$ and $P_j = Inc(j)$, vertex e appears in both paths. Thus P_i and P_j are not node-disjoint.

4 HW6, Problem 4

1. We will reduce this problem to `Max Flow`, thus proving that it can be solved in polynomial time.

   Given a directed $G = (V, E)$ and two vertices $s, t \in V$, construct a flow network $G' = (V, E, c)$, where s is the source, t is the sink and $c_e = 1$ for every edge $e \in E$. The transformation requires polynomial time.

   We now claim that there are k simple edge-disjoint paths in G if and only if the value of the max flow in G' is at least k.

   ⇒ Suppose G has k simple edge-disjoint paths from s to t. Then we can send 1 unit of flow across each of these paths in G', obtaining a flow of value k in in G'.

   ⇐ Suppose the value of the max flow in G' is k. Since the capacities in the network are integers, by the integrality theorem, there is a max flow where every flow value is integer. Since every edge has a capacity of 1, every flow value in this integral max flow vector is either 0 or 1. By capacity constraints, for the max flow to have value k, there are k edges out of the source that each carry a flow of value 1. By flow conservation, each of these edges starts a path that must end at t. If the path is simple, it is one of the paths we are looking for; else if there is a cycle in the path, decrease the flow along the cycle to be 0: we now have a simple s-t path along which the flow is 1.

   Note that the k simple paths constructed in this way are indeed edge-disjoint: for suppose two paths starting at s, ending at t and each carrying a flow of value 1 shared an edge that does not leave s or enter t. Then the flow on this edge should be at least 2, violating capacity constraints.

2. We will reduce this problem to the problem of finding k **edge**-disjoint paths in a directed graph.

   Given a directed graph $G = (V, E)$ and two special vertices s, t, construct a flow network $G' = (V, E, c)$ where s is the source, t is the sink, all edges have capacity 1 and all vertices (except for s, t) have capacity 1.

   We will show that G has k **node**-disjoint s-t paths if and only if the max flow in G' is at least k.

   ⇒ Suppose G has k node-disjoint paths from s to t. Then we can send 1 unit of flow across each of these paths in G'. Clearly the resulting flow in G' satisfies all node / edge capacity constraints, and flow conservation constraints, and its value is k.

$\Leftarrow$ Suppose the value of the max flow in G' is k.

Let G'' be the flow network where every node u is replaced by two new uncapacitated nodes u_{in} and u_{out}, and an edge (u_{in}, u_{out}) with capacity 1, such that

- *all* incoming edges to u now enter u_{in}; and
- *all* outgoing edges from u now leave u_{out}.

Note that this is a regular flow network where only edges have capacities.

From problem 8ii in HW5, we know that G' has a max flow of value k if and only if G'' has a max flow of value k. Now from part 1 of this problem, since the max flow in G'' is k, there are k edge-disjoint s-t paths in G''. For every such path, contract all vertices u_{in} and u_{out} into a single vertex u. We claim that the k resulting paths are **node**-disjoint s-t paths in G.

For suppose P_1 and P_2 are two of the k edge-disjoint s-t paths in G'' but, after contraction of the vertices, the resulting paths in G share a node u. Since in G' all incoming edges to u enter u_{in}, and all outgoing edges from u leave u_{out}, it must be that edge (u_{in}, u_{out}) appeared in both P_1 and P_2, contradicting the fact that P_1 and P_2 are edge-disjoint.

5 HW6, Problem 5

1. Min-cost flow

$$\begin{aligned}
\min_{f_{ij}\geq 0} \quad & \sum_{f_{ij}} a_{ij} f_{ij} & \\
\text{subject to} \quad & \sum_{(i,j)\in E} f_{ij} - \sum_{(j,i)\in E} f_{ji} = s_i & \text{, for all } i \in V \\
& f_{ij} \leq c_{ij} & \text{, for all } (i,j) \in E
\end{aligned}$$

2. Assignment problem

   For every pair $(i,j) \in A$, define the binary *variable*

$$x_{ij} = \begin{cases} 1, & \text{if person } i \text{ is assigned to job } j \\ 0, & \text{otherwise} \end{cases}$$

   Constraints: every person must have one job and every job must be assigned to one person.

$$\begin{aligned}
\max_{x_{ij}} \quad & \sum_{(i,j)\in A} a_{ij} x_{ij} & \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} x_{ij} = 1 & \text{for all } i \in P \\
& \sum_{i:(i,j)\in A} x_{ij} = 1 & \text{for all } j \in J \\
& x_{ij} \in \{0,1\} & \text{for all } (i,j) \in A
\end{aligned}$$

3. Uncapacitated facility location

   For every facility $i \in F$ and every client $j \in D$, define the binary *variable*

$$y_{ij} = \begin{cases} 1, & \text{if client } j \text{ is assigned to facility } i \\ 0, & \text{otherwise} \end{cases}$$

   Also, for every facility $i \in F$ define the binary *variable*

$$x_i = \begin{cases} 1, & \text{if facility } i \text{ is open} \\ 0, & \text{otherwise} \end{cases}$$

Constraints: every client must be assigned to one facility; if a client is a assigned to a facility, then that facility must be open.

$$\begin{array}{lll} \min\limits_{x_i, y_{ij}} & \sum\limits_{i \in F} f_i x_i + \sum\limits_{i \in F, j \in D} c_{ij} y_{ij} & \\ \text{subject to} & \sum\limits_{i \in F} y_{ij} = 1 & \text{for all } j \in D \\ & y_{ij} \leq x_i & \text{for all } j \in D, \text{ for all } i \in F \\ & y_{ij} \in \{0, 1\} & \text{for all } i \in F, j \in D \\ & x_i \in \{0, 1\} & \text{for all } i \in F \end{array}$$

4. Bin packing

Let m be the total number of possible sizes for the items. Enumerate every *valid* configuration $C^j = (t_1^j, t_2^j, \ldots, t_m^j)$ —that is, every configuration C^j such that

$$\sum_{1 \leq i \leq m} t_i^j s_i \leq 1.$$

Assume there are N valid configurations. For every valid configuration define the integer *variable*

$$x_j = \text{number of bins packed according to configuration } C^j.$$

Therefore, our program will have N variables.

Constraints: all items must be placed in bins.

Suppose there are a_i items of size s_i in our input.

$$\begin{array}{lll} \min\limits_{x_j} & \sum\limits_{1 \leq j \leq N} x_j & \\ \text{subject to} & \sum\limits_{1 \leq j \leq N} t_i^j x_j \geq a_i & \text{for all } 1 \leq i \leq m \\ & x_j \in Z_+ & \text{for all } 1 \leq j \leq N \end{array}$$