Homework 11

CS570 Spring 2025

Due: Apr 25, 2025

1. (**14 points**) Consider the problem Spanning Tree with Bounded Degree (STBD): Given an undirected graph $G = (V, E)$ and integer $k > 0$, determine whether there is a spanning tree where the max vertex degree in the tree is not greater than k. That is, whether there is a subgraph $G'(V, E'), E' \subset E, |E'| = |V| - 1, G'$ is a connected graph and all its vertex degrees are less than or equal to k. Prove that STBD is NP-complete.

   Solution:

   First we need to prove finding the k-spanning-tree is in NP. Given a subgraph G' as certificate, it can be verified in polynomial time whether the degrees of its vertices are all less equal than k and whether it is a valid spanning tree.

   Hence the problem of "Spanning Tree with Bounded Degree"(STBD) $\in$ NP.

   Then we need to prove finding the k-spanning-tree is NP-Hard. We use a Hamiltonian Path to Prove this, which is a known NPC problem.

   To prove that H-Path$\leq$ STBD, We configure the STBD problem as follows: For an input of H-Path Graph G, we input the exact graph into STBD problem and set $K = 2$.

   Claim: G has a H-Path if and only if G has a spanning tree with max degree ≤ 2.

   Proof:
   $\Leftarrow$) If there is a solution to the STBD problem, that means there is a spanning tree that goes through all vertices with degree less or equal to $K = 2$. Since no vertex of it has degree more than two, such a tree is nothing but a path. Since it is spanning, it goes through all the vertices, making it a H-Path in G.
   $\Rightarrow$) Vice-Versa, if there is a H-Path in the graph, this path is simply a tree with max degree less than or equal to 2, and it spans the graph as it contains all the vertices, thus satisfying the STBD problem.

   Rubrics (14 points):

   - 4 pts: Correctly shows in NP
   - 4 pts: Correctly constructs and explains the reduction.
   - 6 pts: Proof in both directions.

2. (**15 points**) Given an undirected graph with positive edge weights, the $BIG-HAM-CYCLE$ problem is to decide if it contains a Hamiltonian cycle C such that the sum of weights of edges in C is at least half of the total sum of weights of edges in the graph. Show that $BIG-HAM-CYCLE$ is NP-complete.

To show the problem is in NP:

(a) Certificate: A sequence of edges

(b) The certifier takes as input an undirected graph (the *BIG-HAM-CYCLE* instance) and the certificate. It verifies that the sequence of edges form a Hamiltonian cycle and that the total weight of the cycle is at least half the total weight of the edges in the graph. Thus *BIG-HAM-CYCLE* is in NP.

To show the problem is NP-Hard: We want to show that *HAM-CYCLE* is polynomial time reducible to *BIG-HAM-CYCLE*.

Given an undirected graph $G = (V, E)$ (instance of *HAM-CYCLE*), fix an **arbitrary vertex** u. For an **arbitrary neighbor** v of this u, set the weight for edge (u, v) to $|E|$ and assign the rest of the edges a weight of 1. This gives us a weighted graph G'. Note that the total weight of the edges in G' becomes $2|E| - 1$.

Claim: G' has a *BIG-HAM-CYCLE if and only if* G has a Hamiltonian cycle **containing the edge** (u, v). (Note that this claim is a little different from the usual approach)

Proof:

(a) Suppose G has a Hamiltonian cycle containing the edge (u, v). By our construction, since this edge has weight $|E|$ in G', this cycle in G' has a weight more than $|E|$, i.e., more than half of total edge-weight. meaning G' has a *BIG-HAM-CYCLE*.

(b) Suppose we have a *BIG-HAM-CYCLE* present in G'. By construction, this cycle must contain the edge (u, v), since otherwise the cycle would not be *big*. Thus, this is a hamiltonian cycle in G containing the edge (u, v).

Now, any hamiltonian cycle of G has to pass through u, thus it has to pass through some neighbor v of u as well. Thus, by repeating the above procedure once for all neighbors of u, we can say G has a Hamiltonian cycle if and only if at least one of the created instances has a *BIG-HAM-CYCLE*. Since this results in at most $|V|$ *calls to the BIG-HAM-CYCLE black box*, the overall construction is in polynomial time.

Thus, *BIG-HAM-CYCLE* is in NP and NP-Hard. Hence it is an NP-Complete problem.

Rubrics (10 points):

- 2 pts: Correctly shows *BIG-HAM-CYCLE* $\in$ NP using a proper certificate.
- 4 pts: Correctly constructs and explains the reduction *HAM-CYCLE* $\leq_P$ *BIG-HAM-CYCLE* and discusses how the construction is done in polynomial time.
- 4 pts: Proof in both directions.

3. (**15 points**) In the Bipartite Directed Hamiltonian Cycle (BDHC) problem, we are given a directed graph $G = (V, E)$ that is bipartite, i.e., V can be partitioned as $L \cup R$ s.t. each edge goes from 'L to R' or 'R to L', and we are asked whether there is a simple cycle which visits every vertex exactly once. It is known that the Directed Hamiltonian Cycle is an NP-complete problem. Prove that BDHC is NP-Complete as well.

   Solution:

   (a) To show the problem is in NP:
      - Certificate: A sequence of vertices
      - Certifier: Given a candidate cycle, we just check the vertices along the given sequence one by one to see if every vertex in the network is visited exactly once and if each consecutive vertex pair along the sequence are joined by an edge. As we see, this is doable in polynomial time.

   (b) To prove the problem is also NP-Hard, we reduce Directed Hamiltonian Cycle (DHC) problem to Bipartite Directed Hamiltonian Cycle (BDHC) problem.
   Construction: Given an arbitrary directed graph G, we split each vertex v in G into two vertex v_{in} and v_{out}. Here v_{in} connects all the incoming edges to v in G; v_{out} connects all the outgoing edges from v in G. Moreover, we connect one directed edge from v_{in} to v_{out}. After doing these operations for each vertex in G, we form a new graph G'. Here G' is bipartite graph, because we can put each v_{in} in L and each v_{out} in R without any edges within L or within R.
   Claim: G has DHC if and only if G' has a BDHC.
   Proof:
      i. If there is DHC in G, then there is a BDHC in G'. We can replace each vertex v on the DHC in G into consecutive vertices v_{in} and v_{out}, and (v_{in}, v_{out}) is an edge in G'. Then the new path is a BDHC in G'.
      ii. On the other hand, if there is BDHC in G', then there is a DHC in G. Note that if v_{in} is on the BDHC, v_{out} must be the successive vertex on the BDHC. Then we can merge each vertex pair (v_{in}, v_{out}) on the BDHC in G' into vertex v and form a DHC in G.

   This completes the reduction, concluding that the given problem is NP-complete.

   Rubrics (15 points):

   - 4 pts: Prove in NP.
   - 5 pts: Construction
   - 6 pts: Proof

4. (**16 points**) Suppose we have a variation of the 3-SAT problem called Min-3-SAT, where the literals are never negated. Of course, in this case it is possible to satisfy all clauses by simply setting all variables to true. But, we are additionally given a number k, and are asked to determine whether we can satisfy all clauses while setting at most k variable to be true. Prove that Min-3-SAT is NP-Complete.

   (a) For a truth assignment, we can simply count the number of literals set to true. Then evaluate each clause with the truth assignment. If all clauses equal to true and at most k literals are set to true, then answer yes. So Min-3-SAT $\in$ NP.

   (b) We reduce from vertex cover to Min-3-SAT. For any given instance of the vertex cover problem, we can construct an equivalent Min-3-SAT problem with variables for each vertex of a graph. Each edge (u, v) of the graph can be represented by a clause $(u \vee u \vee v)$ or $(u \vee v \vee v)$ which can be satisfied only by including either u or v among the true variables of the solution. For the constructed Min-3-SAT problem, there is a satisfying assignment within k true variables if and only if there is a vertex cover within k vertices to the corresponding vertex cover problem. Therefore, Min-3-SAT is NP-Hard.

   Thus, this problem is NP-Complete.

   Rubrics (16 points):

   - 4 pts: Prove in NP.
   - 6 pts: Construction
   - 6 pts: Proof

Ungraded Problems

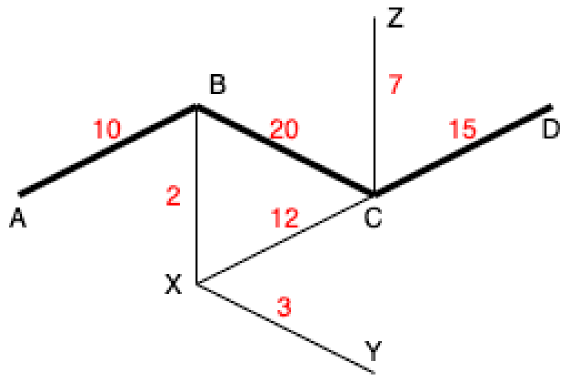
5. The government wants to build a multi-lane highway across the country. The plan is to choose a route and rebuild the roads along this route. We model this problem with a simple weighted undirected graph G with the nodes denoting the cities and the edges capturing the existing road network. The weight of an edge denotes the length of the road connecting the corresponding two cities (assumed positive).

   Let d_{uv} denote the shortest path distance between nodes u and v.

   Let $d(v, P)$ denote the shortest path distance from a node v to the *closest* node on a path P (i.e. $\min_{u \in P} d_{uv}$).

   Finally, we define the *aggregate remoteness* of P as $r(P) = \sum_{v \in V} d(v, P)$.

   In the example shown in the figure below, path $P = ABCD$ is a potential highway. Then, $d(A, P) = d(B, P) = d(C, P) = d(D, P) = 0$, while $d(X, P) = d_{XB} = 2$, $d(Y, P) = d_{YB} = 5$, and, $d(Z, P) = d_{ZC} = 7$. The remoteness of path ABCD is $0 + 0 + 0 + 0 + 2 + 5 + 7 = 14$.



   The government wants a highway with the minimum aggregate remoteness, so that all the cities are somewhat close to the highway. Formally, we state the problem as follows, "Given a graph G, and a non-negative number k, does there exist a path P in G having remoteness $r(P)$ at most k"? Show that this problem REMOTE-PATH is NP-complete.

   REMOTE-PATH $\in$ NP (6 points): A path will be a certificate (thus, poly-size). For each vertex v, $d(v, P)$ can be computed by first running Dijkstra from v and then finding the closest point on P. Adding $d(v, P)$ for all the nodes v gives the total remoteness. Thus, the certifier runs in polynomial time.

   NP-hardness (14 points): We show this with a reduction from Hamiltonian Path (HP). Consider an HP instance graph $G = (V, E)$. Construct a weighted graph H by assigning arbitrary positive weights to all the edges in E. Pass this weighted graph H and $k = 0$ as input to REMOTE-PATH. We show that G has a hamiltonian path if and only if H has a path with zero remoteness. To prove the first direction, suppose P is a hamiltonian path of G. Since, all the nodes lie on P, we have $d(v, P) = 0\ \forall v \in V$, and hence, $r(P) = 0$. Hence, REMOTE-PATH outputs 'Yes'. For the other direction, suppose H has a path with zero remoteness, say P. If there is a node $v \notin P$, then $d(v, P) > 0$ since all the edge-weights are positive. Hence, as $r(P) = 0$, P must contain all the nodes, thus, G indeed has a hamiltonian path. Since assigning the weights is done in $O(|E|)$ time, the reduction is poly-time.

6. For any positive k, the graph k-coloring problem is stated as follows: Determine if the vertices of a given graph G can be colored using k colors such that no two adjacent vertices have the same color. The graph 3-coloring problem is known to be NP-complete. Prove that the 5-coloring problem is NP-complete.

   Solution:

   First, we show that 5-coloring is in NP.

   Certificate: a color solution for the network, i.e., each node has a color.
   Certifier:

   - Check for each edge (u, v), the color of node u is different from the color of node v;
   - Check at most 5 colors are used

   This process takes polynomial time to verify that 5-coloring has a solution. 5-coloring is in NP.
   NP-hard: prove that 3-coloring $\leq_P$ 5-coloring.

   Graph construction:
   Given an arbitrary graph G. Construct G' by adding 2 new nodes u and v to G. Connect u and v to all nodes that existed in G, and to each other. G' can be colored with 5 colors iff G can be colored with 3 colors.

   (a) If there is valid 3-color solution for G, say using colors $1, 2, 3$, we want to show there is a valid 5-coloring solution to G'. We can color G' using five colors by assigning colors to G according to the 3-color solution, and then color node u and v by additional two different colors. In this case, node u and v have different colors from all the other nodes in G', and together with the 3-coloring solution in G, we use at most 5 colors to color G'.

   (b) If there is a valid 5-coloring solution for G', we want to show there is a valid 3-coloring solution in G. In G', since node u and v connect to all the other nodes in G and to each other, the 5-coloring solution must assign two different colors to node u and v, say colors 4 and 5. This leaves the remaining graph as G and only 3 remaining colors in our set. Since we do have a valid 5-coloring solution, the remaining 3 colors successfully form a valid 3-color solution for G.

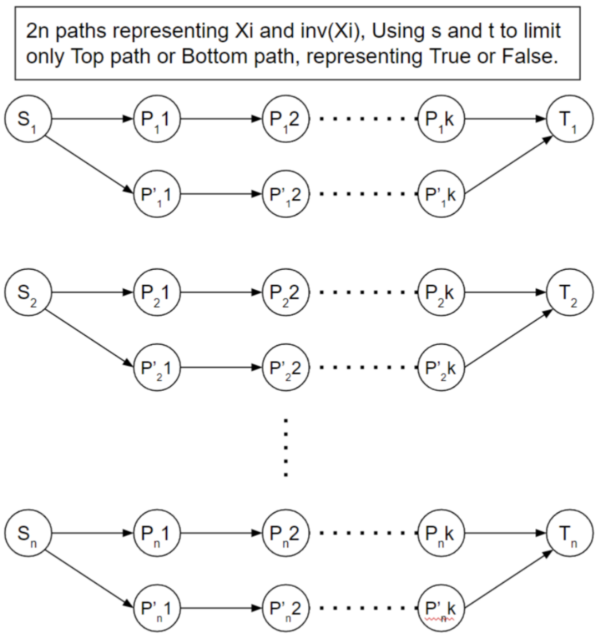
   Thus, 3-coloring $\leq_P$ 5-coloring.

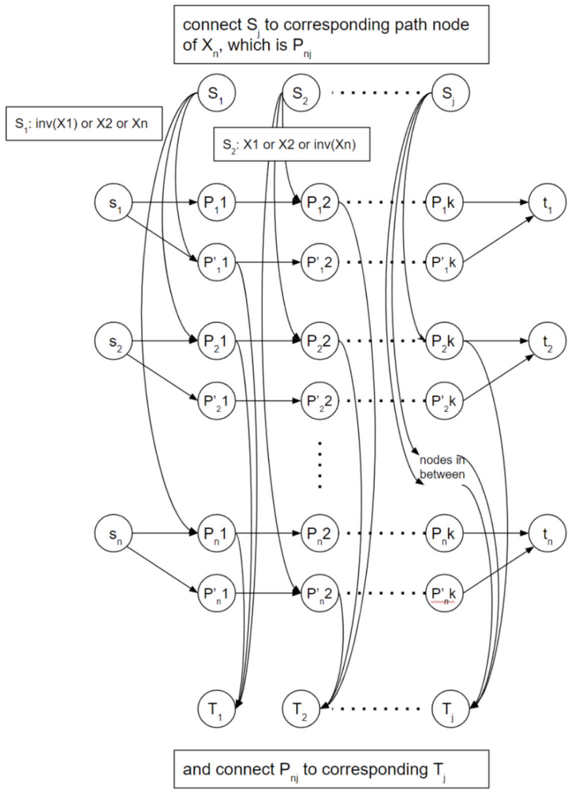
7. You are given a directed graph $G = (V, E)$ with weights on its edges $e \in E$. The weights can be negative or positive. The Zero-Weight-Cycle Problem is to decide if there is a simple cycle in G so that the sum of the edge weights on this cycle is exactly 0. Prove that this problem is NP-complete. Hint: For NP-hardness, reduce from the Subset sum problem that can be stated as "Given the set of positive numbers $S = \{w_1, ..., w_n\}$ and target sum $W > 0$, is there is a subset of S that adds up to exactly W ?".

   Solution: Zero-weight-cycle is in NP because we can exhibit a cycle in G, and it can be checked that the sum of the edge weights on this cycle are equal to 0.
   We now show that subset sum $\leq$ Zero-weight-cycle. We construct an instance of the Zero-weight-cycle in which the graph has nodes $0, 1, 2, ..., n$, and an edge (i, j) for all pairs $i < j$. The weight of the edge (i, j) is equal to w_j. Finally, there are edges $(j, 0)$ of weight $-W$ for all $j = 1, \ldots, n$.
   We claim that there is a subset that adds up to exactly W if and only if G has a zero-weight-cycle. If there is such a subset S, then we define a cycle that starts at 0, goes through the nodes whose indices are in S, and then returns to 0 on the edge $(n, 0)$. The weight of $-W$ on the edge $(n, 0)$ precisely cancels the sum of the other edge weights. Conversely, all cycles in G must use the edge $(n, 0)$, and so if there is a zero-weight-cycle, then the other edges must exactly cancel $-W$, in other words, their indices must form a set that adds up to exactly W.

8. The Directed Disjoint Paths Problem is defined as follows. We are given a directed graph G and k pairs of nodes $(s_1, t_1), (s_2, t_2), ..., (s_k, t_k)$. The problem is to decide whether there exist node-disjoint paths $P_1, P_2, ..., P_k$ so that P_i goes from s_i to t_i. Show that Directed Disjoint Paths is NP-complete.

Solution: The problem is in NP, since we can exhibit a set of disjoint paths P_i, and it can be checked in polynomial time that they are paths in G, connect the corresponding nodes, and are disjoint.
Now we show 3-SAT $\leq_p$ Directed Disjoint Paths. Consider a 3-SAT problem given by a set of clauses $C_1, ..., C_k$, each of length 3, over a set of variables $X = x_1, ..., x_n$. To create the corresponding instance of the Directed Disjoint Paths problem, we will have $2n$ directed paths, each of length k, one path P_i corresponding to variable x_i and one path P'_i corresponding to $\overline{x_i}$. We add n source-sink pairs corresponding to the n variables, and connect source s_i to the first node on paths P_i and P'_i and connect the last nodes in paths P_i and P'_i to sink t_i. Note that there are two directed paths connecting s_i to t_i : the path (s_i, P_i, t_i), and the path (s_i, P'_i, t_i). We will think of selecting the first of these paths as setting the variable x_i to false (as the variable through the copies of $\overline{x_i}$ are left unused), and selecting the second path will correspond to setting the variable x_i to true.


2n paths representing Xi and inv(Xi), Using s and t to limit only Top path or Bottom path, representing True or False.
S1 P1 1 P1 2 P1 k T1 P'1 1 P'1 2 P'1 k
S2 P2 1 P2 2 P2 k T2 P'2 1 P'2 2 P'2 k
Sn Pn 1 Pn 2 Pn k Tn P'n 1 P'n 2 P'n k



Now we will add k additional source sink pairs, one corresponding to each clause C_j.Let S_j and T_j be the source sink pair corresponding to clause C_j. We will claim that there is a path from S_j to T_j disjoint from the path selected to connect the $s_j - t_j$ source-sink pairs if and only if clause C_j is satisfied by the corresponding assignment. Assume clause C_j contains the literal t_{j1}, t_{j2} and t_{j3}. Now we have a path P_i or P'_i corresponding to each of these variables or negated variables. The paths have n nodes each, let v_{j1}, v_{j2} and v_{j3} denote the j-th node on the 3 corresponding paths. We add the edges (S_j, v_{jl}) and (v_{jl}, T_j) for each of $l = 1, 2, 3$.
Now we claim that the resulting directed graph has node disjoint paths connecting the source-sink pairs $s_i - t_i$ and $S_j - T_j$ for $i = 1, 2, ..., n$ and $j = 1, ..., k$ if and only if the 3-SAT instance is satisfiable. One direction is easy to see: if the 3-SAT instance is satisfable, then select the paths connecting s_i to t_i corresponding to the satisfying assignment, as suggested above. Then the source-sink pair S_j and

T_j can be connected through the path using the true variable in the clause.
Finally, we need to show that if the disjoint paths exist, then the 3-SAT formula has a satisfying assignment. Note that the paths P_i and P_j are disjoint, and the graph has no edges connecting different paths. The only edges outside these paths in the graph are edges entering one of the sinks, or leaving a source. As a result the only paths in the graph connecting a $s_i - t_i$ pair are the two paths (s_i, P_i, t_i) and (s_i, P'_i, t_i), and the only paths in G connecting $S_j - T_j$ pairs are the three possible paths through each of the 3 variable nodes in C. Hence, sets of disjoint paths connecting the source-sink pairs,correspond to satisfying assignments.