CS2800 Design and Analysis of Algorithms
Problem Set #10 April 24, 2023

1. Consider the following problem called BoxDepth: Given a set of n axis parallel rectangles (this means that the length and breadth of the rectangle are parallel to the x and y-axes), how big is the largest subset of these rectangles that contain a common point.

   (a) Give a polynomial time reduction from BoxDepth to the problem of computing the largest clique in a graph.

   > **Solution:** Construct a graph $G(V, E)$ where the vertices are the boxes, and there is an edge between u and v if the corresponding boxes intersect. Now, a set of boxes that all intersect amongst themselves forms a clique. Therefore, BoxDepth reduces to finding the largest clique in the graph.

   (b) Describe a polynomial-time algorithm to solve BoxDepth. Any polynomial-time algorithm would suffice. Don't try to optimize the running time.

   > **Solution:** The intersection of a set of axis parallel rectangles is an axis parallel rectangle itself, and its extremities are segments of some rectangles given as input. So, we can try out all possible options for the four sides, and count the number of rectangles that contain this rectangle. A brute force implementation of this idea requires $O(n^4)$ time.

   (c) Explain why this does not prove P = NP.

   > **Solution:** By reducing BoxDepth to clique we have shown that clique is at least as hard as BoxDepth. To prove P = NP, we need a reduction from clique to BoxDepth and then use the poly-time algorithm for BoxDepth.

2. A formula is in disjunctive normal form (DNF) if it consists a disjunction of several terms, where each term is a conjunction of literals. For example $\phi = (x \wedge \bar{y} \wedge z) \vee (\bar{x} \wedge y \wedge \bar{z})$ is a DNF formula.

   (a) Give a polynomial-time algorithm to check if a DNF formula ϕ is satisfiable.

   > **Solution:** If there is a term in the DNF ϕ that does not contain a literal and its negation, then the DNF is satisfiable.

   (b) Consider the following "proof" of P = NP: Given a 3-CNF formula ϕ, convert it to a DNF formula ψ using the distributive law of $\wedge$ and $\vee$ as follows.

   $$(x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y}) = (x \wedge \bar{y}) \vee (y \wedge \bar{x}) \vee (\bar{z} \wedge \bar{x}) \vee (\bar{z} \wedge \bar{y})$$

Now use the algorithm in Part (a) to check for satisfiability. What is wrong in this "proof"?

Solution: The size of the formula can grow exponentially - verify this.

3. Given a graph $G(V, E)$, a *k-coloring* of the graph is an assignment of colors $c : V \to \{1, 2, \ldots, k\}$ such that if $(u, v) \in E$, then $c(u) \neq c(v)$. The 3-Colorability problem is to test if a given graph has a 3-coloring. This problem is known to be NP-complete.

   Suppose that you are given a subroutine, that takes as input a graph G, and decides whether G is 3-colorable or not. Using this subroutine, and assuming that a call to the subroutine takes constant time, design a polynomial time algorithm to compute a 3-coloring $c : V \to \{1, 2, 3\}$, if one exists.

   Solution: The idea is to force a color on a vertex in G, and check if this can be extended to a 3-coloring of the graph. Let 3-color(G) return true iff G can be three colored. Consider three vertices t_1, t_2, t_3 that are all connected to each. In any 3-coloring, they will all have difference colors. Now we proceed as follows: For a vertex u, construct $G_{u,i}$ to be the graph where all the edges of G are present, the triangle between t_1, t_2, t_3 is present, and an edge from t_j to u is present where $j \neq i$. If G is 3-colorable, one of $G_{u,1}$, $G_{u,2}$ or $G_{u,3}$ must be 3-colorable. This must be true due to the following reason: If G is 3-colorable, then consider a function $c : V \cup \{t_1, t_2, t_3\} \to \{1, 2, 3\}$ where $c(u)$ for every $u \in G$ is given by the 3-coloring, and $c(t_i) = i$. This coloring must be a valid 3-coloring in the graph $G_{u,c(u)}$ due to the construction. Check this using the 3-color$(G_{u,i})$ subroutine call. Continue this, and construct the 3-coloring.

4. A Hamiltonian path in an undirected graph $G(V, E)$ is a simple path in G that passes through each vertex in G exactly once. Similarly, a Hamiltonian cycle is a cycle that starts from a vertex $v \in V$, passes through every other vertex exactly once and returns to v. The HamPath and HamCycle problem is to decide if a given input graph G has a Hamitonian path and Hamiltonian cycle respectively.

   (a) Show that HamPath $\leq$ HamCycle.

   Solution: Given $G(V, E)$, create a new graph G' as follows: $V' = V \cup \{s\}$, $E' = E \cup \{(s, u) | u \in V\}$. Suppose that G has a Hamiltonian path, with $v_1, v_2, \ldots, v_n$, then $s, v_1, v_2, \ldots, v_n, s$ is a Hamiltonian cycle in G'. Similarly, suppose that $s, v_1, v_2, \ldots, v_n, s$ is a Hamiltonian cycle in G', the path $v_1, v_2, \ldots, v_n$ must be a Hamiltonian path in G.

   (b) Show that HamCycle $\leq$ HamPath.

Solution: Construct a new graph $G'(V', E')$ where $G' = V \cup \{s, t, v_1'\}$ where $v_1 \in V$, and $E' = E \cup \{(s, v_1), (t, v_1')\} \cup \{(v_1', u) | (v_1, u) \in E\}$. Suppose there is a Hamiltonian cycle $v_1, v_2, \ldots, v_n, v_1$ in G. Consider the path $s, v_1, v_2, \ldots, v_n, v_1', t$ - this is a Hamiltonian path in G'. Now consider any Hamiltonian path in G', this must either start in s and end in t, or start in t and end in s because they are degree 1 vertices. W.l.o.g, assume that the Hamiltonian path starts in s - then the next vertex must be v_1. Also, the vertex just before t in the Hamiltonian path must be v_1'. Now consider the path $v_1, v_2, \ldots, v_n, v_1'$ in this Hamiltonian path. Every vertex in V is visited exactly once here. Therefore, $v_1, v_2, \ldots, v_n, v_1$ is a Hamiltonian cycle in G.

5. In this question, we will look at some variants of Sat and their complexity.

   (a) Let ϕ be a CNF formula such that each clause contains exactly 3 literals, and each variable occurs in at most 3 clauses. Show that ϕ is satisfiable.

   To do this, the following statement (known as Hall's theorem) may be useful.

   > A bipartite graph $G(L \cup R, E)$ such that $|L| < |R|$ has a matching M that matches every vertex in L if and only if for every $L' \subseteq L$, $|L'| \leq |N(L')|$, where the $N(L')$ is set of neighbors of L'.

   Solution: If $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is the CNF, construct a graph $G(L, R, E)$ where L consists of m vertices $l_1, l_2, \ldots, l_m$ corresponding to the m clauses. The set R consists of $2n$ vertices, one for each literal. Put an edge from l_i to r_j if that literal is present in the clause C_i. If the graph has a perfect matching that matching all the vertices in L, then you can set all those literals as true and the formula becomes satisfiable. We will show that this graph has such a perfect matching using Hall's theorem.

   Let $L' \subseteq L$ be any set and let $N(L')$ be the set of neighbors. Since each clause contains exactly 3 literals, the number of edges in the induced subgraph $G[L \cup N(L')]$ is $3|L'|$. Also, since each literal is in at most 3 clauses, we can also say that the number of edges is at most $3|N(L')|$. Thus $|L'| \leq |N(L')|$ and the graph has a perfect matching. Thus, the formula ϕ is satisfiable.

   (b) Let ϕ be a CNF formula where each clause has at most 3 literals, and each variable appears in at most 3 clauses. Show that this variant of Sat is NP-complete.

   Hint: Replace each occurrence of a variable with a new copy. Add constraints to say that all the copies must take the same truth value.

   Solution: We will reduce from 3-SAT to this version where each clause has exactly three literals. Let ϕ be a 3-SAT formula such that $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Let i^{th} variable occur in k times - replace the j^{th} occurrence with the variable $x_{i,j}$. Now add clauses of the form $(x_{i,1} \Rightarrow x_{i,2}) \wedge (x_{i,2} \Rightarrow x_{i,3}) \wedge \cdots \wedge (x_{i,k-1} \Rightarrow x_{i,k}) \wedge (x_{i,k} \Rightarrow x_{i,1})$. This is equivalent to the set of clauses $(\neg x_{i,1} \vee x_{i,2}) \wedge (\neg x_{i,2} \vee x_{i,3}) \wedge \cdots \wedge (\neg x_{i,k-1} \vee x_{i,k}) \wedge$

$(\neg x_{i,k} \vee x_{i,1})$. Each variable now occurs at most 3 times, and each clause has at most 3 literals and the new formula is satisfiable iff ϕ is satisfiable (verify this statement).

(c) Suppose that ϕ is a CNF formula where each clause has exactly 3 literals, and each variable occurs in at most 4 clauses. Show that this variant of SAT is NP-complete.

Hint: For each clause C_i with two literals, add a new variable x_i' into it. Put additional clauses (with different variables) to force x_i' to take the truth value 0.

Solution: Take the previous part, and let ϕ be a formula where each clause contains at most 3 literals, and each literal appears in at most 3 clauses. Consider a clause $(l_1 \vee l_2)$. Replace this with a clause $(l_1 \vee l_2 \vee x_1')$. Now to add clauses forcing x_1' to be zero - add clauses $(\neg x_1' \vee y_1 \vee z_1)$, $(\neg x_1' \vee y_2 \vee z_2)$, and $(\neg x_1' \vee y_3 \vee z_3)$. To force the y_is and z_is to be zero (thus forcing x_1' to be 0), we add the following clauses for each $j = \{1, 2, 3\}$: $(a_j \vee y_j \vee \neg z_j) \wedge (a_j \vee \neg y_j \vee z_j) \wedge (a_j \vee \neg y_j \vee \neg z_j)$. If we now force the a_js to be zero, then the only way to satisfy these clauses would be for y_j and z_j be both zero. So, we add the clause $(\neg a_1 \vee \neg a_2 \vee \neg a_3)$. Do this for every clause with two literals, and we get a SAT formula with each clause exactly 3 literals and each variable in at most 4 literals.

(d) Given a CNF formula ϕ such that each clause contains an arbitrary number of literals, but each variable occurs in at most 2 clauses, give a polynomial-time algorithm to check if ϕ is satisfiable.

Hint: Sometimes greed is good.

Solution: If a literals occurs exactly once in the formula ϕ, set it to 1, and reduce the formula to a new formula ϕ'. Clearly ϕ is satisfiable iff ϕ' is satisfiable. If a variable appears without negation in two clauses, then too we can do as earlier. Similarly when the variable appears with negation in both clauses.

Now suppose there is a variable x such that $(C \vee x)$ and $(C' \vee \neg x)$ are both present in ϕ. Replace these two clauses with a new clause $(C \vee C')$ and obtain the formula ϕ'. Verify that ϕ is satisfiable iff ϕ' is satisfiable. Keep continuing to check satisfiability.