COL351: Analysis and Design of Algorithms (CSE, IITD, Semester-I-2016-17) Homework-06

Name: Shivank Goel

Entry number: 2014CS10565

- You may use any of the following known NP-complete problems to show that a given problem is NP-complete: 3-SAT, INDEPENDENT-SET, VERTEX-COVER, SET-COVER, HAMILTONIAN-CYCLE, HAMILTONIAN-PATH, SUBSET-SUM, 3-COLORING, CLIQUE.

There are 5 questions for a total of 100 points.

1. (20 points) Consider the following problem:

   F-SAT: Given a boolean formula in CNF form such that (i) each clause has exactly 3 terms and (ii) each variable appears in at most 3 clauses (including in negated form), determine if the formula is satisfiable.

   Answer the following questions with respect to the above problem under the assumptions (i) P = NP, and (ii) P ≠ NP. Give reasons.

   (a) Is F-SAT ∈ NP?

   (b) Is F-SAT NP-complete?

   (c) Is F-SAT NP-hard?

   (d) Is F-SAT ∈ P?

(1) P=NP

(a) Clearly F-SAT belongs to NP since given a assignment of variables we can check in polynomial time if it is satisfying the formula or not.

(b)
- We will also prove it to be NP complete by reducing 3-SAT to F-SAT i.e to prove $3\text{-SAT} \leq_p \text{F-SAT}$.
- For this we have to convert an arbitrary instance of 3-SAT to F-SAT. Consider an arbitrary instance of 3-SAT.
- If a clause of 3-SAT has less than 3 terms then we do the following :-

Case-1) If clause has only 1 term (x_1) then we create 2 dummy (new) variable a_1 & a_2 & add 4 clauses for each such clause with 1 term given by

$(x_1 \vee a_1 \vee a_2)\ (x_1 \vee a_1 \vee \bar{a}_2)\ (x_1 \vee \bar{a}_1 \vee a_2)\ (x_1 \vee \bar{a}_1 \vee \bar{a}_2)$.

Thus ensures that whatever value a_1 & a_2 takes the and of these terms will evaluate to x_1. Proof: If $x_1 = 1$ then clearly each clause is 1 & and is 1. If $x_1 = 0$ then ∃ atleast one clause where a_1 & a_2 are bothe zero (since we have considered all possible ways of disjunction of a_1 & a_2) & hence and of the clauses will be zero. Hence 3-SAT is satisfiable only iff conjunction of these 4 clauses is satisfiable i.e iff $x_1 = 1$. So another way of tackling this could be to just remove singleton clauses & assign them 1 but this way there may be a 1 generated in other clauses as well.

Case-2) If there exist a clause of size 2 in 3-SAT say $(x_1 \vee x_2)$ then we create a new dummy variable say a & make 2 clauses as $(x_1 \vee x_2 \vee a)(\bar{x_1} \vee \bar{x_2} \vee \bar{a})$. Clearly conjunction of these clauses is one iff $x_1 \vee x_2 = 1$ ~~irrespective of a~~ since $\bar{a}$ must be] to satisfy second clause.

- Now we have an instance with all-clauses having exactly 3 terms. But a variable may be present in more than 3 clauses say x_i is present in $C_1, C_2, \longrightarrow C_k$ clauses where $k > 3$.

- We will create k variables $f_1, f_2 \longrightarrow f_k$ corresponding to x_i & they must be replaced in k clauses instead of x_i. But we want all of them to be simultaneous true or simultaneously false since they represent a single variable. Thus we add clauses

$$(f_1 \Rightarrow f_2)(f_2 \Rightarrow f_3) \longrightarrow (f_{k-1} \Rightarrow f_k)(f_k \Rightarrow f_1)$$

Conjuction ∧ of these clauses will be true only when either all f_i's are true or all f_i's are false i.e all are same.

$$(\neg f_1 \vee f_2)(\neg f_2 \vee f_3) \longrightarrow (\neg f_k \vee f_1)$$

& each 2 variable clause will be converted to 2 new clauses. Hence each f_i $i = 1, 2, \ldots k$ will appear in exactly 3 clauses.

Claim → 3Sat instance is satisfiable iff F-sat instance is satisfiable.

- It's evident since we plug values of x_i's $\in$ 3Sat to corresponding y_i's in f-sat (Note if x_i appeared in k clauses we have k corresponding y_i's for it) & give arbitrary values to rest, F-sat will be satisfiable.
- Similarly if Fsat is satisfiable we plug value of y_i's $\in$ fsat to corresponding x_i's of 3sat then 3-sat will be satisfiable.

(c) Since all problems in NP are reducible to Fsat ∴ By defn Fsat is NP hard.

(d) Fsat $\in$ P. since P = NP & ∴ P $\subseteq$ NP & we know Fsat $\in$ NP

Case-II $P \neq NP$

(a) Fsat $\in$ NP
(b) Fsat is NP complete } By same arguments as of case-I
(c) Fsat is NP hard
(d) Fsat $\notin$ P since $P \neq NP$ & Fsat $\in$ NPcomplete i.e belongs to hardest problems in NP & it's assumed that they are not solvable in polynomial time since otherwise P = NP.

2. (15 points) Consider the following problem:

LONG-PATH: Given a weighted, directed graph $G = (V, E)$, two vertices $s, t \in V$ and a number W, determine if there is a simple path between s and t such that the sum of weights of edges in this path is $\geq W$.

Recall that a simple path is a path that does not have any vertices repeated. Show that LONG-PATH is NP-Complete.

- Lets denote Longpath by LP. Since given a path we can check in polynomial time if it's a simple path with sum of edge weights $\geq W$ $\therefore$ LP $\in$ NP
- We will denote Hamiltonian Path by HP & show that it's reducible to LP i.e To prove HP $\leq_p$ LP
- Now we give each directed edge in G of HP a weight of 1. We call this graph G'.

Claim: G has a hamiltonian path iff G' has a simple path of length (sum of weights) greater than equal to $|V|-1 = k$ (say)

Pf

- Let G has hamiltonian path. $q_1, q_2, \ldots, q_{|V|}$. Since this visits each vertex exactly once it is a simple path of length = weight $|V|-1$ (= no. of edges.)
- Let G' has a simple path of weight $|V|-1$ $q_1, q_2, \ldots$ Since it's simple path, no vertex is visited twice & since weight of edges is $|V|-1 \Rightarrow$ there exists $|V|-1$ edges in this simple path (since each edge has wt 1) & no edge is repeated. Thus nodes $x = |V|$ & hence it's a hamiltonian path in G. Note G' can't have simple path of length $> |V|-1$ because otherwise we have visited more than $|V|-1$ edges $\Rightarrow$ more than $|V|$ vertices (not poss) or repeated a vertex (not poss).

- Therefore to solve G_1 we can create G' in polynomial time by making weights of each edge = 1 & we will call long path with $|V| = |V| - 1$
- We want to determine s, t to give as input to long-path problem. Since we don't know s & t we try all possible pairs = $O(|V|^2)$ calls to long path problem, that are polynomial number of calls. (Exact calls $|V| C_2$)
- Hence we have reduced $HP \leq_p LP$. Thus LP is NP-Complete since we know HP is NP-complete.

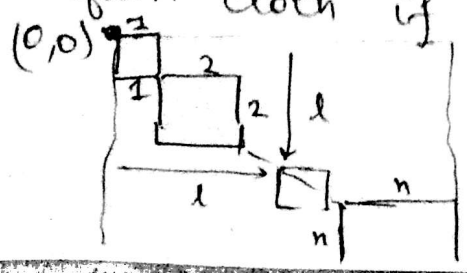
3. (15 points) Consider the following problem:

> CUT: Given a rectangular piece of cloth sheet of dimension $h \times w$, n rectangular items with dimension $h_1 \times w_1, \ldots, h_n \times w_n$ and profits $p_1, \ldots, p_n$, and an integer P, determine whether it is possible to cut these rectangular items from the sheet such that the total profit is exactly P. You are allowed to cut at most one copy of any item from the sheet. You may assume that arbitrary cuts are possible.

Show that CUT is NP-complete.

CUT $\in$ NP since given a set of items we can check if the sum of their profits is P & that it is possible to cut them from the cloth in polynomial time.

- To prove SUBSET SUM $\leq_p$ CUT. Let $A = \{a_1, a_2, \ldots, a_n\}$ be an arbitrary instance of subset sum where we have to check if $\exists$ a subset $A' \subseteq A$ s.t $\sum_{a_i \in A'} a_i = S$.
- We will construct an instance CUT problem as follows.
  - → Consider n cloths with profits $p_1, p_2, \ldots, p_n$ s.t $p_i = a_i \quad \forall\, i \in \{1, 2, \ldots, n\}$
  - → Set $P = S$
  - → Set the dimensions of ith cloth as $(i \times i)$ & Dimension of rectangular piece of cloth to be $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ i.e a square of $\frac{n(n+1)}{2} \times \frac{n(n+1)}{2}$
- Note that given any subset P' of n clothes it's always possible to cut them from above square cloth. Infact all the n clothes can be easily cut from cloth if we cut ith cloth as follows.



$l = \sum_{k=1}^{i-1} (k-1)$. Move l units right & down from (0,0) & cut a cloth of $(i \times i)$.

Claim: Subset Sum is satisfied iff ∃ soln to CUT problem

Proof: ⇒ Let subset sum is satisfied s.t $u_1, u_2, \ldots u_k \in$ are s.t $\sum_{i=1}^{k} u_i = S$. Then we pick corresponding cloths with profits $p_1, \ldots p_k$ & we will have satisfying sum of profits $\sum_{i=1}^{k} p_i = P$. Also these clothes can be easily cut from larger cloth by method mentioned before.

⇐ Let there exist a possible cut $p_1, \ldots p_k$ s.t $\sum_{i=1}^{k} p_i = P$ & they are cuttable from cloth.

~~Note: the claim~~ Then if we pick corresponding numbers from set A say $u_1, \ldots u_k$ they will satisfy $\sum_{i=1}^{k} u_i = S$ since we had

Set $a_i = p_i \; \forall \; i$ & $P = S$.

Thus CUT Problem is NP-complete.

4. (25 points) Consider the following problem:

SPAN-TREE: Given an undirected graph $G = (V, E)$, determine if there is a spanning tree of G that has at most 10 leaves.

Either show that this problem is NP-complete or give a polynomial time algorithm (along with correctness proof and running time).

Span Tree is NP complete.

Given a G & Spanning Tree we can check in polynomial time if it's a spanning tree of G and has ≤ 10 leaves.

To Prove

Undirected Hamiltonian Path $\leq_p$ Span-tree

- Consider an undirected graph G.
- Claim: G has Hamiltonian path iff G has a spanning tree with 2 leaves.
- Let G has Hamiltonian path $a_1, a_2, \longrightarrow a_n$

  Clearly $a_1 - a_2 \longrightarrow a_n$ is a spanning tree with 2 leaves a_1 & a_n.
- Let G has spaming tree with 2 leaves say T.

  → Since T has only 2 leaves ⇒ by defn there are only 2 vertices of degree 1

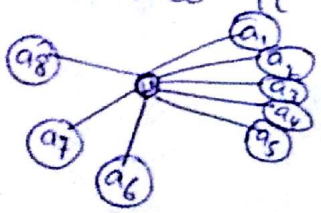
  → Claim: all other vertices of T has degree 2.

  Suppose for contradiction it has degree 3 then each branch from it will end up at a leaf giving us atleast 3 leaves which is a contradiction.

  Thus T has all nodes with degree 2 except leaf nodes which give us a straight chain type tree covering all nodes and will be a hamiltonian path for G.

- If G has no NON DEGREE 1 vertex i.e it has all vertices of degree 1 then G can either be o or o—o which clearly have spanning tree with ≤ 10 leaves.

- Construct G' from G in the following way.
Pick a arbitrary NON DEGREE 1 vertex say $v \in G$. & add 8 new nodes to it $a_1, a_2, \ldots, a_8$ as follows.



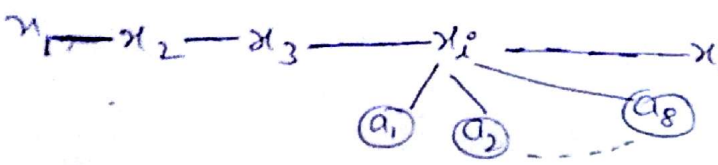
Let $G \equiv (V, E)$
then $G' = (V', E')$ where
$E' = \{V \cup (v, a_i)\} \; \forall \; i \in 1, 2, \ldots 8$
$V' = \{V \cup a_i\} \; \forall \; i \in 1, 2, \ldots 8$

Claim: G has hamiltonian path iff G' has a spanning tree of at most 10 leaves.

Proof:

⇒ Let G has Hamiltonian path $x_1, x_2, \ldots x_n$ $n \equiv$ no. of vertices $\equiv |V|$

Case-I $v \notin \{x_1, x_2\}$ i.e not an end point of hamiltonian path
Let $x_i = v$, then we can construct following spanning tree in G' with 10 leaves.



} corresponding spanning tree in G' with 10 leaves.

Case-II $v \in \{x_1, x_2\}$ ie end point of hamiltonian path then we can construct a spanning tree in G' of 9 leaves.

⇐ Let G' has a spanning tree of atmost 10 leaves say T

- Since all a_i's $i \in 1, 2, \ldots 8$ must be spanned all of them will be present as leaves in T since they were only connected with just one vertex in G' itself & had degree 1. Thus T has atleast 8 leaves.

Note: All nodes of degree 1 in G will remain untouched in G' & hence no. of leaves in T = 8 + no. of nodes in G of degree 1.

Case-I T' has 8 leaves ⇒ G had only 1 node v & trivially had a hamiltonian path.

Case-II T' has 9 or 10 leaves. Removing a_i's. the remaining tree T' span all nodes of G and has 1 or 2 leaves. 1 leaf again means single node in G. If T' has 2 leaves then we get a spanning tree of G, T' that has 2 leaves ∴ hence ∃ a hamiltonian path in G.

~~Therefore, Span-Tree is NP-complete.~~

(Note T has > 10 leaves ⇒ G had > 2 vertices of degree 1 & can't be spanned by a single hamiltonian path.)

Hence Span-Tree is NP-complete.

5. (25 points) A cycle cover of a given directed graph $G = (V, E)$ is a set of vertex disjoint cycles that cover all the vertices. Consider the following problem:

CYCLE-COVER: Given a directed graph $G = (V, E)$, determine if there is a cycle cover of G.

Either show that this problem is NP-complete or give a polynomial time algorithm (along with correctness proof and running time).

The problem is not NP complete

- Given $G = (V, E)$ we construct $G' = (V', E')$ where

V' = for each vertex $v_i \in G$ we create 2 vertices l_i & r_i in V' for $\forall i \in \{1, 2, \longrightarrow n\}$ $n \equiv$ no. of vertices in G.

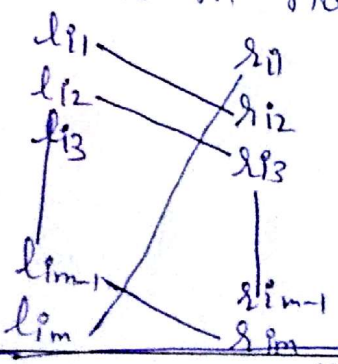
E' = Let $V_L \equiv l_i$'s & $V_R \equiv r_i$'s. for each edge (v_i, v_j) in G we add 2 edges in E' that are (l_i, r_j) & (r_i, l_j)

- Clearly G' is a bipartite graph. (Undirected).
- There is vertex disjoint cycle cover of G iff there exist a maximum matching in $G'(V_L, V_R, E')$.

→ Let there is a cycle cover of G say $C = \{C_1, C_2, \longrightarrow C_k\}$

Let C_i is a cycle given by $v_{i_1} \to v_{i_2} \to \longrightarrow v_{i_m} \to v_{i_1}$

In G' we have a corresponding subgraph with perfect matching of nodes $l_{i_1}, l_{i_2} \longrightarrow l_{i_m}$ & $r_{i_1}, r_{i_2} \longrightarrow r_{i_m}$ corresponding to above m nodes as follows. Hence doing this for each C_i we'll find a max matching of G', Since C_i's cover all vertices of G max matching will be matching between all vertices of G'. →continue



Algorithm

→ From G create G' as described above
→ Find maximum matching in G'
→ If max matching is $= |V|$ ie no. of vertices of G then return True else return False.

Time Complexity

→ Creating G' $O(V+E)$
→ Max matching $O(VE)$ by ford fulkerson
→ $O(VE+V+E) = O(VE)$

← Let there is a maximum Matching in G'.

- If (l_i, r_j) edge $\in$ max matching $\Rightarrow$ (v_i, v_j) was directed edge in G
- ~~If (l_j, r_i) edge $\in$ max matching $\Rightarrow$ (v_j, v_i) was directed edge in G.~~
- Since all l_i are matched with same r_j, this implies all $v_i \in G$ has an outgoing edge.
- Since all l_i are mapped to different r_j's this implies all $v_i \in G$ has an outgoing edge to a different vertex, (i.e No 2 v_i's have outgoing edge hence all the vertices of G has an incoming edge too. (by P.P)
- Consider graph G'' as a subgraph of G corresponding to i.e containing only edges as given by maximum matching of G'. In G'' as proved above each vertex will have exactly one incoming edge & one outgoing edge. G'' will have all vertices of G.

Claim: G'' will be a cycle cover of G.

Proof: G'' will have atleast 1 cycle since say v_{i_1} has outgoing edge v_{i_2} then v_{i_2} must have an outgoing edge to either v_{i_1} or a new vertex v_{i_3}. Continuing this way v_{i_k} has either an outgoing edge to $v_{i_{k+1}}$ (new vertex) or ~~one of the vertices $v_{i_1} \rightarrow v_{i_k}$. In latter case we have found the cycle. Otherwise we continue till v_{i_n} (n = total nodes) which must have an~~

~~edge to one of the $v_{i_1}, v_{i_2} \longrightarrow v_{i_{k-1}}$ & we will get a cycle~~

or v_{ik} will have an edge to v_{i_1}. Note v_{ik} can't have an edge to $v_{i_2}, \longrightarrow v_{i_{k-1}}$ since these vertices already have some incoming edge from a vertex other than v_{ik} & different vertices have outgoing edges to different vertices i.e no two vertex have an outgoing edge to same vertex. If v_{ik} has edge to v_{i_1} then we get a cycle $(v_{i_1} \to v_{i_2} \longrightarrow v_{ik} \to v_{i_1})$ else we continue till v_n ($n \equiv$ no. of vertices in G). Now since every vertex has an outgoing edge v_n must have 1 outgoing edge & that too would be towards v_1 (By above argument). Hence there always exist a cycle.

- Now, remove cycle $v_{i_1}, v_{i_2} \longrightarrow v_{ik}$ from G''. Remaining vertices again have exactly 1 outgoing edge & 1 incoming edge & different vertices have outgoing edges to different vertices. So we find another cycle $v_{ik+1}, v_{ik+2} \longrightarrow v_{im}$ & we remove it from G''. — ⊛

- Repeating ⊛ we will get vertices of G divided into disjoint cycles.
- Note: Cycles formed from G'' will be disjoint is obvious since otherwise some vertex will have more than 1 outgoing or incoming edge.