Name :- Aagam Shah

USC ID :- 5420023430

Algorithm HW 6

Ans 1) Here we need to reduce SAT to Integer Linear Programming in polynomial time.

Any SAT instance has boolean variable and clauses. Our Integer Programming problem will have twice as many variables, one for each variable and its compliment, as well as the following inequality.

SAT $\leq_p$ Integer Linear Program

construction:-

(1) we create constraints as follows:-

(i) $0 \leq x_i \leq 1$

(ii) $0 \leq \neg x_i \leq 1$

(iii) $x_i + \neg x_i = 1$

↳ This can be converted to

$x_i + \neg x_i \leq 1$

and $x_i + \neg x_i \geq 1$

$\therefore \quad 1 \leq x_i + \neg x_i \leq 1$

(iv) For each clause C

$C = \{ x_1, \neg x_1, \ldots x_n \} : x_1 + \neg x_2 + \ldots x_i \geq 1$

claim :- if SAT is satisfiable then Integer linear program has a solution.

Direction 1:- If SAT is satisfiable then Integer Linear Program has a solution.

Proof :- In any SAT satisfiability we consider a TRUE literal to be 1 when we convert it to an ILP. since if expression is satisfied, atleast one literal per clause is true, so the inequality sum is always ≥ 1 making the Integer linear program solution.

example :-

$S: (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4)$

when converted to ILP we have

(i) $x_1 + \neg x_2 + x_3 \geq 1$

(2) $x_2 + x_3 + x_4 \geq 1$

if any one of the SAT literal is set true

(x_1 = True & x_4 = True)

per clause the value is set as 1 and sum ≥ 1.

Direction 2:- If there is a solution to Integer Linear program then SAT is satisfiable.

Proof:- Given a solution of Integer Linear program, all the variable will be 0 or 1. Set the literals having 1 as True and 0 as False. Here due to the constraint no variable and their compliment will be assigned same value, so any legal assignment will also satisfy all the clauses of SAT.

Thus by showing that we can reduce SAT to Integer Linear Program in Polynomial time. Integer Linear Program is NP Hard.

Ans 2) To show that the given problem is NP complete we need to show that it is NP and NP Hard

(1) To show that the problem is NP.

Certificate :- Given a graph $G = (V, E)$, a subset of city of size K. with total number of roads between them larger than equal to M.

certifier :- To verify the certificate we follow these steps

(1) check if total subset has exactly K cities

(2) Iterate through all the pairs of cities in the subset

(i) For each connecting pair of nodes count the no of roads

(3) Check if total no of roads $\geq M$.

This can be done in polynomial time thus the city problem is NP.

(b) Show that city problem is NP Hard.

We reduce the city problem from an known NP problem. Lets consider here the clique problem.

Clique $\leq_p$ City Problem

Clique problem :- Given an undirected graph $G = (V, E)$ and an integer K. determine if there is a clique of size K in G. where a clique is a fully connected subgraph.

construction :-

(1) construct a graph $G' = (V', E')$ where $V' = V$

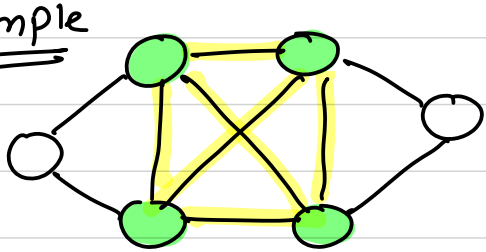
and $E' = E$

(2) Set the no of cities i.e $N = |V|$ ie the no of vertices in G'

(3) Set the value of $K' = K$ where K' is the size of clique.

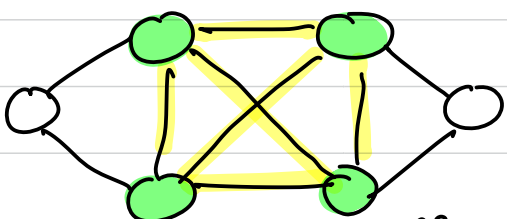
(4) Set the value of $M = \frac{K'(K'-1)}{2}$ representing the no of roads in the graph.

example



$G = (V, E)$

$K = 4$



since G and G' have same graph

$G' = (V', E') = (V, E)$

$N = |V|$ = no of vertices

$K' = K = 4$

$M' = \frac{K(K-1)}{2} = \frac{4(4-1)}{2}$

$= \frac{4 \times 3}{2}$

$= 6$

Claim:- If G has a clique of size K. if and only if G' has a subset of cities of size K having total no of roads $\geq M$.

Proof $\Rightarrow$ If G has a clique of size K. then there exists a subset of K' vertices in G' such that each pair of vertices is atleast connected by an edge. The total no of roads between them $M' =$ total no of edges in clique $M' = \frac{K(K-1)}{2}$

There fore, G' has a subset of cities of size K with total number of roads between them $\geq M$.

Proof $\Leftarrow$ If G' has a subset of cities of size K' with total no of edges between them $\geq M'$. then this subset must have $\frac{K(K-1)}{2}$ edges.

This would imply that each pair of cities in the subset is connected by a road. This mean that the corresponding vertices in G that form a clique of size k.

Since we can reduce Clique to city problem in polynomial time. It is NP-Hard.

∴ Thus, the city problem is NP as well as NP Hard, thus becoming NP complete.

Ans3) Proove that SAT' is NP complete

(i) Show that SAT' is NP

certificate :- Given a SAT in which CNF formula with m clauses and n variable and assignment to the values of the variable's $(x_1, x_2 \ldots x_n)$

certifier :- Given a certificate, we can check whether exacty 'm-2' clauses are satisfied by simply substituting the values of the variable's in each clauses and evaluating the boolean expression. This can be done in polynomial time. if exacty 'm-2' clauses are satisfied then we accept the certificate or else we reject them.

Since, we can verify the certificate in polynomial time. SAT' is NP

(2) Show that SAT' is NP Hard.

To show that SAT' is NP Hard, we need to prove that any known NP problem can be reduced to SAT' in polynomial time. We can do this by showing SAT is reducible to SAT' in polynomial time.

$$SAT \leq_p SAT'$$

* construction

Given an instance of SAT problem we can construct an instance of SAT' problem.

(1) create 2new variable y and z

(2) Add 4 new clauses in the CNF formula.

$$(y) \wedge (\neg y) \wedge (z) \wedge (\neg z)$$

and make it a new formulae as F'

(3) Now F' will have 'm+4' clauses and 'n+2' variables.

Claim:- We need to show F is satisfiable with m clauses. if and only if F' is satisfiable with $(m'-2)$ clauses.
↳ $(m' = m+4)$

Proof :-

Direction1:- F is satisfiable when we have m clauses then F' is satisfiable when we have $(m'-2)$ clauses where $(m' = m+4)$ clauses.

Proof :- If F is satisfiable, there exists an arrangement of the values to the variables. such that all m clauses in F are satisfied. We can extend this by setting 'y' and 'z' as true, which will satisfy y and z but not to $\neg y$ and $\neg z$. consider any such sequence of y and z. and always 2 clauses will be true out of 4 newly added clause. Thus (exactly) 'm'-2' clauses $(m' = m-4)$ ie $(m'-2 = m+2)$ clauses will be satisfied. and SAT' will output = True.

Direction 2 :- If F' is satisfiable for exactly $(m'-2)$ clauses ie $(m' = m+4;\ m'-2 = m+4-2 = m+2)$ clauses

then F is satisfiable for m clauses.

Proof:- if F' is satisfiable for (m'-2) clauses then there are 2 cases.

① case 1:- consider out of the total m' clauses. m'-2 clauses are satisfied. here all the m clauses of original problem are satisfied and thus we can say F is also satisfied for m clauses.

eg:- consider.

CNF :- [(c1) ∧ (c2) ∧ ... (cm)] ∧ (y) ∧ (¬y) ∧ (z) ∧ (¬z)

satisfied ↗ ⟷ m clauses.

② case 2:- consider out of the original m clause only 'm-2' are satisfied and the other newly introduced 4 are satisfied in F' making m'-2 ie (m+2) clauses satisfied.

eg:- CNF :- [(c1) ∧ (c2) ... (cm-1) ∧ (cm)] ∧ (y) ∧ (¬y) ∧ (z) ∧ (¬z)

⟷ satisfied.

This case cant occur since in the newly added

4 clauses we cant have y and $\neg y$ or z and $\neg z$. Thus we cant have this case 2. case 2 is not possible.

∴. F must be satisfied when F' is satisfied.

∴. Since we can reduce SAT to SAT' in polynomial time. SAT' is NP Hard.

∴. Since SAT' is NP and also NP-Hard. We can say SAT' is NP-complete.

Ans 4) Showing Longest Path problem is NP complete.

(i) Show that Longest Path Problem is NP.

Certificate:- Given a graph $G = (V, E)$ and number K and a Path P

Certifier:- We will check if all the vertices in the path are distinct and consecutive vertices have an edge amongst them. and the length of path is at least K. If this is satisfied we will accept the certificate or else reject it. This can be verified in polynomial time.

(ii) Show that Longest Path is NP Hard.

TO show that longest path is NP Hard we need to reduce it from a known NP problem. ie we will reduce Hamiltonian

Path to Longest Path problem in polynomial time

Hamiltonian Path $\leq_p$ Longest Path

construction

(1) Create a graph $G' = (V', E')$ where $V' = V$ and $E' = E$

claim:- G has a Hamiltonian Path. if and only if G' has a Longest Path of length $K = |V| - 1$.

Direction:- if G has a Hamiltonian path then G' has a Longest Path of length $= |V| - 1$.

Proof:- Suppose G has a Hamiltonian Path. A Hamiltonian path is a simple path that visits every vertex once. Thus Hamiltonian Path in G will be of length $= |V| - 1$, which means we will have a longest simple path

of length = $|V| - 1$.

Direction 2:- If G' has a longest simple path of length = $|V| - 1$ then G has a Hamiltonian Path.

Proof:- Suppose G' has a longest simple path of length = $(|V| - 1)$. since G' and G have the same vertex and edge set. this path also exists in G. As there are $|V|$ vertices in total and the path has length $= (|V| - 1)$, it must visit every vertex only once. Therefore G will have a Hamiltonian path.

Thus we can say Longest Path problem is NP Hard

Thus Longest Path problem is NP-complete. since we have shown its NP as well as NP Hard.

Ans 5) If we are given a polynomial time algorithm that states if the graph has a hamiltonian cycle or not then we can modify it to find out the cycle and the sequence of verticies that form the cycle.

Algorithm:-

(1) Consider we have an Algorithm HCD which when given a graph $G = (V, E)$ outputs if we have a hamiltonian cycle or not in polynomial time.

(2) Check if the input Graph $G = (V, E)$ has atleast 2 verticies, otherwise return the single vertex or empty set if the graph is empty.

(3) Pick an arbitary starting vertex: V_{start} from the graph

(4) Initialize an empty set, $H[]$ to store the sequence of verticies that form the cycle.

(5) For each vertex V_i adjecent to V_{start} :-

(i) Remove the edge (V_{start}, V_i) from the graph temporarily

(ii) call Algorithm HCD. on the modified graph $G' = (V, E - \{(V_{start}, V_i)\})$.

(iii) If the HCD algorithm returns "Yes" for the input then there exists an Hamiltonian cycle in G'. which is subgraph of G. set $G = G'$

(iv) If the HCD algorithm returns "NO", then every Hamiltonian cycle in G .contains e. and we put back e into graph G.

(6) Repeat step 5 until we are left with $|V|$ edges. Since after each step we are left with subgraph having Hamiltonian cycle. At termination we are left with edges having Hamiltonian cycle.

(7) Run BFS on this and add all the vertex to $H()$ and return the $H()$ as the output.

Since Hamiltonian cycle decision Alg is polynomial time and we call this algorithm once for each vertex the overall time complexity of the new Algorithm is Polynomial.

Ans 6)

(i) To show that the Algorithm will terminate in finite time.

The Greedy Algorithm iteratively moves vertices from one side of the cut to another side, with the main aim of increasing the number of edges in the cut.

Analysis on the given greedy algorithm

The size of vertex |V| is finite. Thus there is a finite number of possible vertex configuration for the set C. In each iteration we either move the vertex from C to V/C or from V/C to C This means that in each iteration, the algorithm is changing the configuration of C.

Since we are only moving vertices that increase the no of edges in the cut. the cut size strictly increases with each

iteration. However the max possible no of edges in the cut is limited by the total no of edges in the graph G denoted by E.

As the cut size increases, there must come a point when no vertex can be moved further increases the no of edges in the cut. At this point the algorithm will not find any vertex that satisfy the condition in step 2 and thus will terminate

To summarize, the Algorithm must terminate because there are finite number of possible vertex configurations for the set C. and the maximum cut size is limited by the total number of edges in Graph.

(2) Show that when Algorithm stops the size of the cut is atleast half of optimum.

construction :-

(1) Let us add edge weight = 1 on all edges of the graph G on which we need to find the maximum cut. ie $w(u,v) = 1$.

(2) consider the node A being in C and B in V/C

Let $W = \sum_{e \in E} w_e$ be the sum of weights of all the edges of the graph G. we extend our notation for 2 nodes u and v. we use w_{uv} to denote we if there is an edge e joining u and v. and 0 otherwise.

∴ For any node $u \in A$, we must have.

$$\sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$$

Since otherwise u should have moved to other side of the partition and (A,B) would not be optimal. Suppose whenwe add up these inequality for all nodes $u \in A$. any edge that is present in A will appear 2 times in LHS inequality. while any edge of A having one edge in LHS and one in RHS appear only once.

$$\therefore \quad 2 \sum_{(u,v) \in A} w_{uv} \leq \sum_{u \in A, v \in B} w_{uv} = w(A,B)$$

we apply the same for set B.

$$2 \sum_{(u,v) \in B} w_{uv} \leq \sum_{\substack{u \in B \\ v \in A}} w_{uv} = w(A,B)$$

Adding both inequality and dividing by 2 we get.

$$\sum_{(u,v) \in A} w_{uv} + \sum_{(u,v) \in B} w_{uv} \leq w(A,B)$$

The LHS inequality count for all the that does not cross from A to B. When we add $w(A, B)$ to both sides we get.

$$\sum_{(u,v) \in A} w_{uv} + \sum_{(v,v) \in B} w_{uv} + w(A, B) \leq 2\, w(A, B)$$

Here the highlighted part = w i.e the total of all edge weights

$$\therefore \quad W \leq 2\, w(A, B)$$

$$\therefore \quad w(A, B) \geq \frac{1}{2} W$$

As we know the optimal solution is limited by the total edge weight of the graph $= W$, thus we can say.

$$\therefore \quad w(A, B) \geq \frac{1}{2}\, w(A^*, B^*)$$ where $w(A^*, B^*)$ | 2 OPTimal solution.

Thus when the Algorithm terminates our solution is at least half of optimal.

Ans 7) **Algorithm**

(1) consider a Graph (Planar graph) $G = (V|E)$ having V vertices and E edges.

(2) color the graph G using 4 coloring in polynomial time using any coloring algorithm

(3) This will split the vertices V into 4 groups C_1, C_2, C_3 and C_4

(4) Main idea here to use the vertices in the not most frequenty used color group to be in our vertex cover.

(5) Let the most freaventy colored group be C_1, Thus all vertices in Vertex cover = $C_2 \cup C_3 \cup C_4$ as our vertex cover.

Since we are using the polynomial time algorithm to color our graph. thus our solution is also solvable in polynomial time.

Assumption used :- The minimum vertex cover is of atleast size $\frac{|V|}{2}$.

Approximation Ratio

- considering the fact that most frequenty used color will be of size $\frac{|V|}{4}$

If this is not true then all colors will be of size $< \frac{|V|}{4}$ and as a result, total will be less than $|V|$.

This implies that the vertex cover is of size atmost $3\frac{|V|}{4}$.

Considering our assumption that lower bound is $\frac{|V|}{2}$.

$\therefore$ Approximation Ratio $= \frac{3|V|/4}{|V|/2} = \frac{3}{2}$

Proof of Approximation (without using our assumption).

- we formulate the vertex cover as LP.

∴ minimize $\left(\sum_{v \in V} x_v\right)$

subject to

$x_u + x_v \geq 1 \quad \forall (u,v) \in E$

$0 \leq x_v \leq 1 \quad \forall v \in V$

we can consider the solution where LP has the values of x_u to be one of the following 3 values - $\{0, \frac{1}{2}, 1\}$

Lets call those 3 set of values $(0, \frac{1}{2}, 1)$ be denoted by 3 variables $V_0, V_{\frac{1}{2}}, V_1$. In normal vertex cover we set the value of vertex cover as $V_{\frac{1}{2}}$ or V_1. Since we

have 4 color graph. we can easily discard the vertex belonging to the most frequent class from $V_{1/2}$. without the loss of genarality. the most frequenty used class being C_1.

$$\text{Vertex cover} = V_1 + \bar{V}_{1/2}$$

where

$$\bar{V}_{1/2} = V_{1/2} - C_1$$

After solving the LP, let the optimal value of LP be $ALG = \frac{1}{2} V_{1/2} + V_1$. This is the lower bound for our vertex cover in optimal case.

If OPT is the optimal vertex cover then.

$$|OPT| \geq ALG.$$

considering the size of $|OPT| = |V_1| + |\bar{V}_{1/2}|$

Let $V'_{1/2}$ denote the vertices in $V_{1/2}$

belonging to color group i where i have values as $(1,2,3,4)$ and here $i=1$, is the most frequenty used group.

$$\therefore \quad |V^{1}_{1/2}| \geq \frac{|V_{1/2}|}{4}$$

As a result.

$$|\overline{V}_{1/2}| \leq 3\frac{|V_{1/2}|}{4}$$

$$\therefore \quad |VC| \leq |V_1| + \frac{3}{4}|V_{1/2}|$$

$$\leq |V_1| + \frac{3}{4}\left(2(ALG - |V_1|)\right)$$

$$\leq \frac{3}{2} OPT$$

$$|VC| \leq \frac{3}{2} OPT$$

$\therefore$ The Approximation Ratio of $3/2$.

Ans 8) We need to show that this approximation algorithm doesn't have constant approximation ratio.

Lets consider the optimal solution be s^* and our solution be S.

consider the example of knapsack having capacity = W.

Item	weight	value	value / weight $[v_i/w_i]$
1	1	3	3
2	W	W	1

Since our algorithm will pick the items in decresing order of (v_i/w_i) and thus our algorithm picks the items in that order till we don't run out of space in our knapsack. for the above example our algorithm will pick only 1 item because item 2 wont be able to

for 1/n. Knapsack once item 1 is selected. So the total value returned by our algorithm = 3

Now, optimal solution for the above consist of only item 2 of weight W. which would give us the value of W.

∴ The approximation algorithm ratio = $\frac{3}{W}$

This ratio becomes smaller and unboundedly small as W increases and will reach to arbitarily zero.

Thus we can say that approximate algorithm is pretty bad. that it doesn't provide constant approximation.