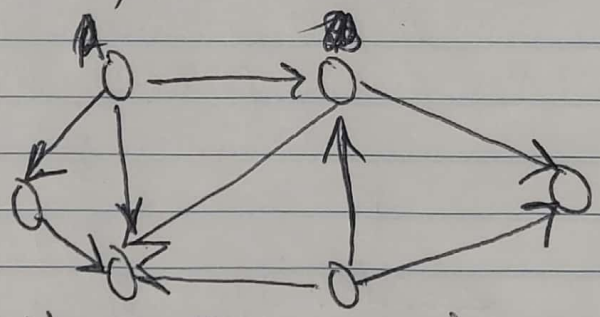
ASSIGNMENT 5

1. Input: You are given a directed graph G, modeling a flight network. There is an edge from airport A to B if there is a direct flight from A to B. The capacity of an edge e is labeled with the corresponding flight capacity cap(e). Each edge has a start time st(e) and a finish time ft(e).
Task:- Given a source and a target airport, our goal is to find the maximum number of passengers that can be sent from the source airport starting at 6AM to the target airport within 5 hours (11AM)

Solution

A → source
C → target

e_1: A → B → C
e_2: A → B → D → C
e_3: A, B → C



⟹ Using Depth first Search

Step I:- I would first of all, find the path and calculate the flow for the path. Would search through the path from the source to the target

Step II: I would then reverse the first step flow and would do a check to determine if the next path can be implemented correctly

Step III: Would then iterate each path & check whether the new flow was implemented and then it stops.

Step IV: - Would then reverse and check if the flow improves in the DFS else it

st ⟶ ft of the capacity edge(e)
start time to finish time; the capacity of the edge c relies on the st (start time)
there should be a condition to check if the start time is equal or greater than ft of the previous edge.
st >= ft

Formal Proof of Correctness

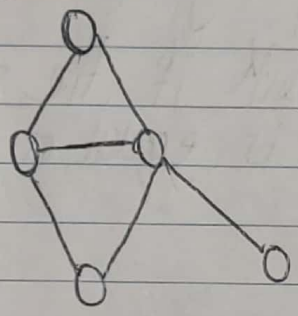
Have a directed graph G, in the graph would want to move from point A to point B A ⟶ B, with the goal to find the maximum number (most capacity)
And to go pass through path B would need to go by the minimized version of G using the maximum flow algorithm
⟹ A ⟶ c ⟶ B
Going from A ⟶ c ⟶ B by taking the most capacity that it gets there correctly by the time constraint; would have the solution unless would go back and check the other best choice and iterate from there (repeat)
With the maximum flow algorithm will get the maximum source, target flow is equal to the capacity of the minimum (s,t) cut.

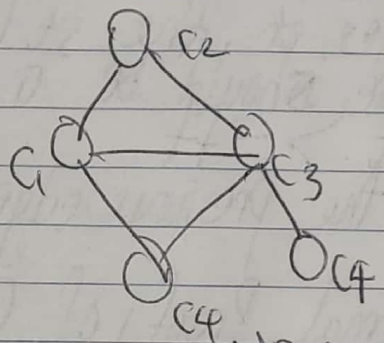
Time Complexity Analysis:-

The time complexity of the algorithm would be
O(v * e)

(2) Let G be a graph with n vertices. The K-colouring problem is to decide whether the vertices of G can be labelled from the set {1, 2, ..., K} such that for every edge (v, w) in the graph, the labels of v and w are different.
Is the (n-4)-colouring problem in P or NP

Solution

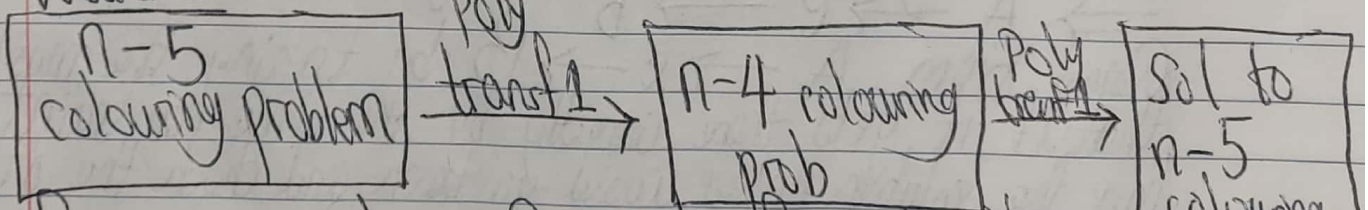




Firstly, I will try to prove n-4 colouring is NP-hard

I would assume the n vertices is bipartite. Would assume somebody proved n-5 colouring problem is NP-hard.

Would be



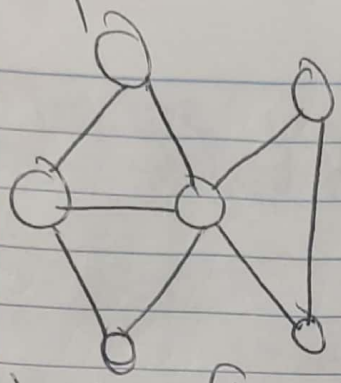
Given an instance G graph of n-5 colouring problem, I will create an instance H of n-5 colouring problem

Algorithms:-

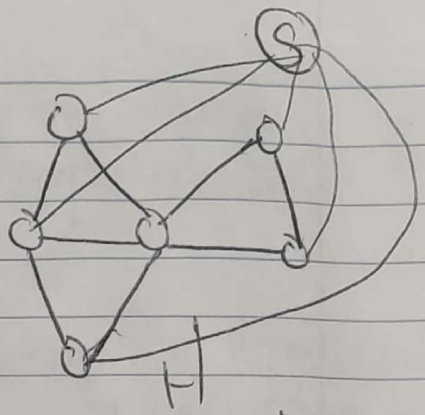
Step 1:- I would take a copy of G and add a new vertex S.

Step 2:- Would add the edges S to all other vertices to create H

bipartite



transfer-1 →

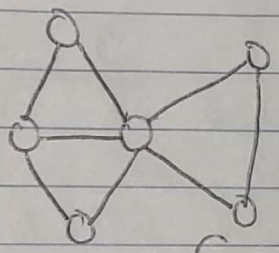


G H

Equivalence

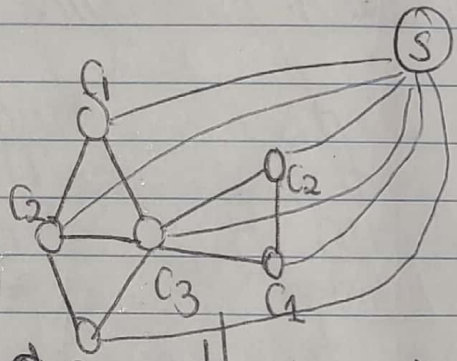
If there is a colouring of n-5 G, I can find a n-4 colouring of H in poly time.

- Would take the given n-4 colouring and add the n-5 colouring to s.



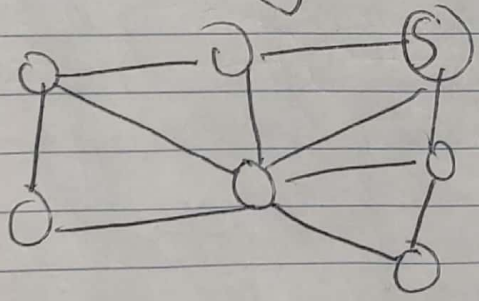
G

from n-4 colouring to n-5 colouring →

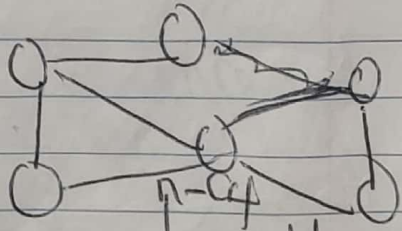


H

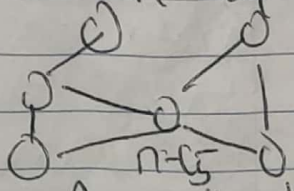
I can find n-5 colouring of G in poly time if there is a colouring of n-4 colouring of H.



delete s →



↓ would rename n-C4 to C5



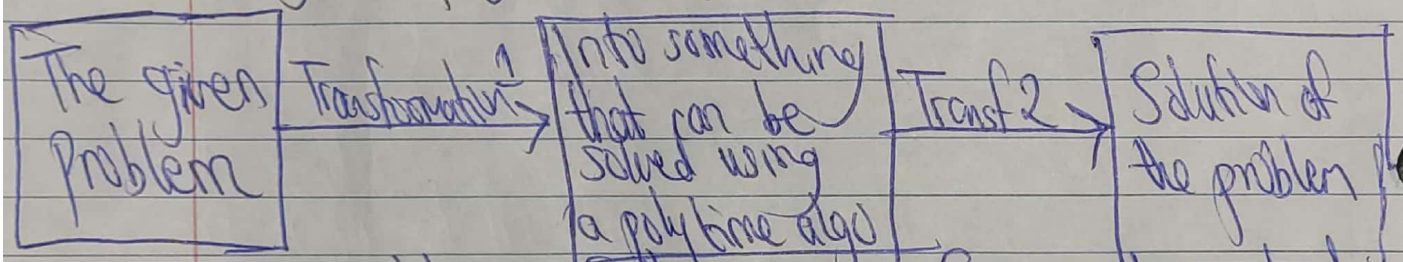
A n-4 colouring of G.

③ Let G be a graph of n vertices representing a set of gamers. There is an edge between two nodes if the corresponding gamers are friends. You want to partition the gamers into two disjoint groups such that no two gamers in the same group are friends?

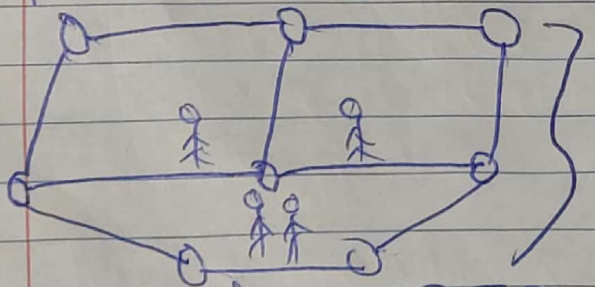
Solution

A problem is in P if there is a polynomial time algorithm for it.

I would prove the problem is in P by designing an algorithm and showing the time complexity and using a poly time algorithm as a black box to solve the problem



This problem of the gamers, G can be solved using bipartite. Would assume the gamers to be bipartite



This is not a correct solution because there are two gamers in the same group.

- Would use DFS

I → Using DFS, I would choose any random vertex and then set the value to one. And if there is another vertex in the DFS, we will have two options
- either the vertex color is not 1 and we are going to set it 0 and then begin a DFS on that vertex
- or the vertex value is 1 will then return false

Step II:- After the DFS is complete and we couldn't find any values that have the same vertex as the DFS has, will then return True

Partition the games, would assume the games to be a S of games where the subset A and B such that $A \cup B = S$, $A \cap B = \emptyset$

$$\sum_{a \in A} a = \sum_{b \in B} b$$

This problem can be solved using P. The given problem is in complexity class P

And the running time of the problem is $O(V+E)$ $\Longrightarrow$ to be in P

(4) You are given a graph G, and the problem is to test, whether there is a cycle of length at most 1000 or not.

Solution

I can prove this problem by using the Hamiltonian cycle Problem as in NP because it visits every vertex once.

Yes. Because if you give me a cycle and claim to be a Hamiltonian cycle, then I can take the following steps to verify in polynomial time.

Step 1:- I can count the num of vertices on the path if the cycle is a cubic. Would check there is less than or exactly 3 vertices in the cycle/graph. Then the problem would still be in NP because it is not a correct solution and is unlikely that we could find an effective algorithm

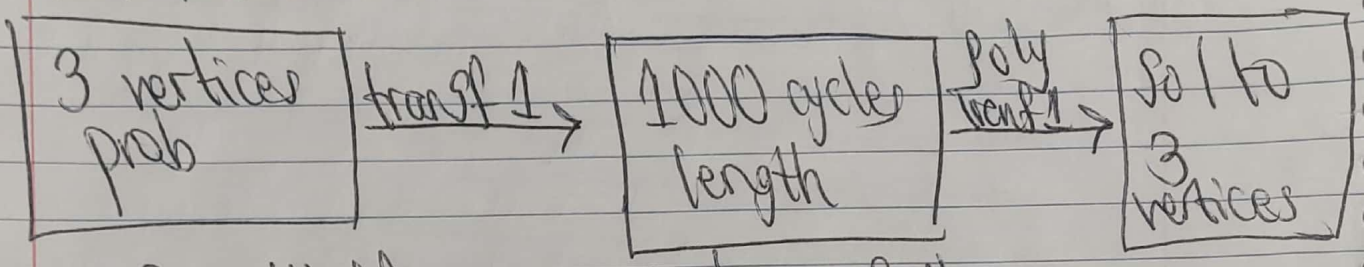
Step 2:- I would check whether any vertices is repeated in the cycle
- if repeated then the solution is incorrect

Step 3:- Would check if every pair of consecutive vertices on the path/cycle is connected by an edge - if not, then it is not a valid solution.
Or you could solve the problem by using DFS; by checking the vertex if yes would then calculate the cycle.
Step 2-3 can be checked in polynomial time.
Therefore, the Hamiltonian cycle problem is in NP.

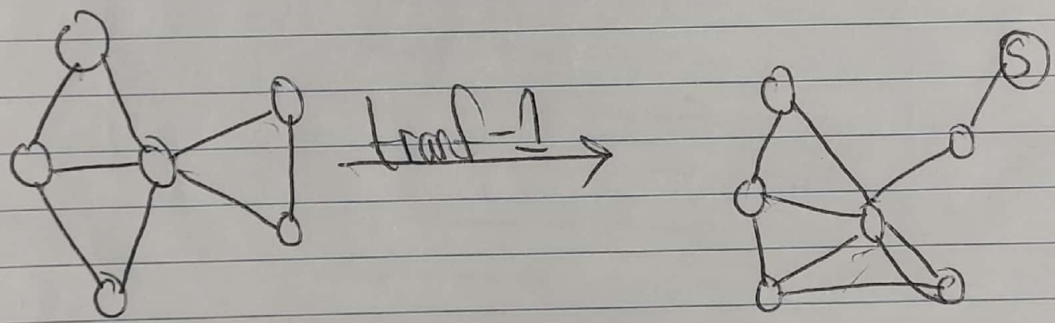
NP

④ Formal Proof

I would try to prove that the cycle of length is NP-hard. Would assume someone had already proved the cycle to be in a cubic plane of three vertices. is NP-hard.

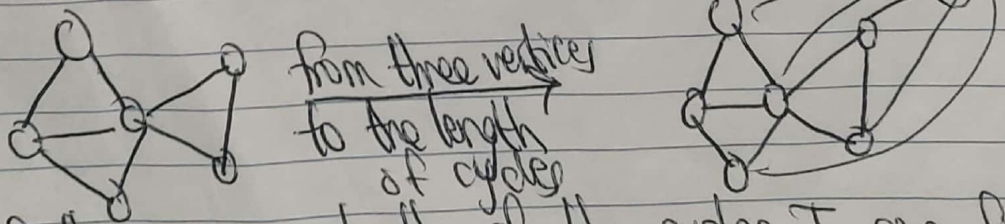


Transf 1:- Would give an instance of the Hamiltonian cycle problem as in NP. Would create an instance of H to be the 1000 cycles at most or not.

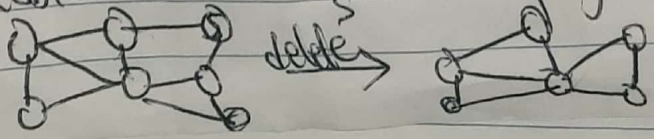


Transf 2 (Equivalence)

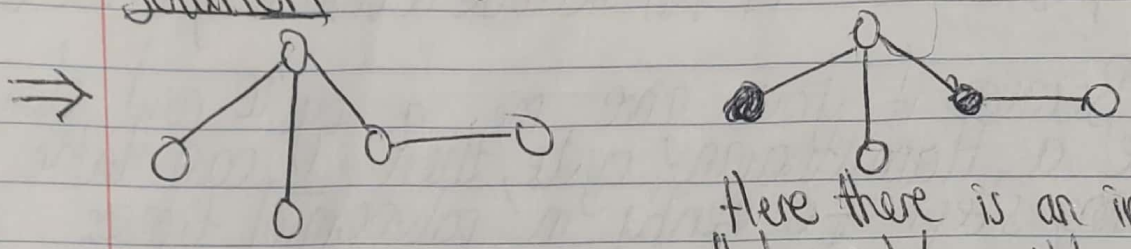
If there is a cubic plane of G, then I can find the length of the cycles of H in a poly time - take the vertices and add the length of the cycle to S



if there is a length of the cycles, I can find the cubic vertices of G in poly time



5. Problem (Battleneck Path)
Input:- An edge-weighted undirected graph G and an integer K.
Question:- Does there exist a path such that the sum of the edge weights is at least K
Solution

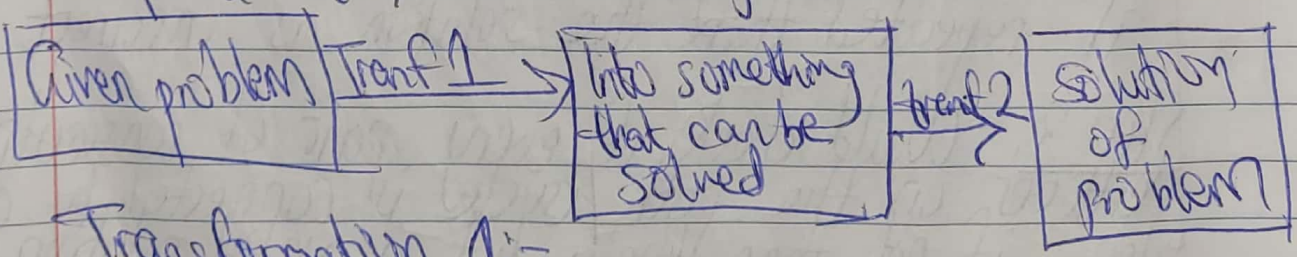


Here there is an integer K of sum of edge that exists possible in the path

Graph $G=(V,E)$ and an integer K

Step 1:- Show that the problem is in NP
Step 2:- Show that the problem is NP-hard
The first step can be done by showing a given solution can be verified in polynomial time
Step 2:- can be done by reducing a problem

Graph $G=(V,E)$ and integer K



Transformation 1:-
Given an instance G of IS (Independent set) problem, I will create an instance H of the bottleneck path problem
Step:- I would add the weight of G to the problem and it would become H of the instance

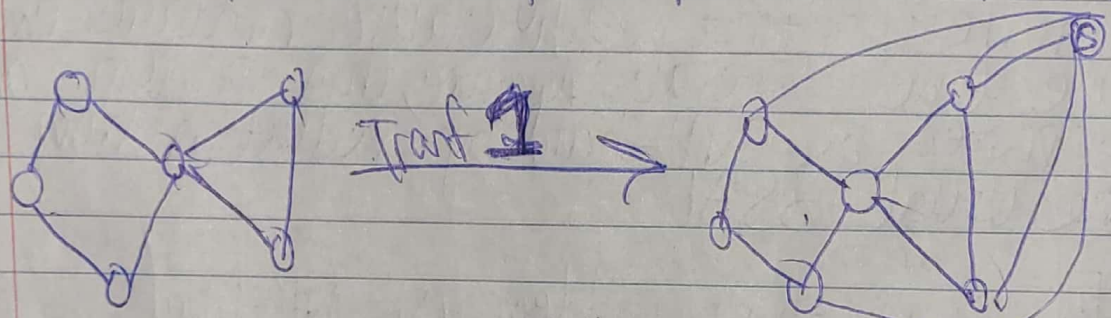
Transformation 2:-

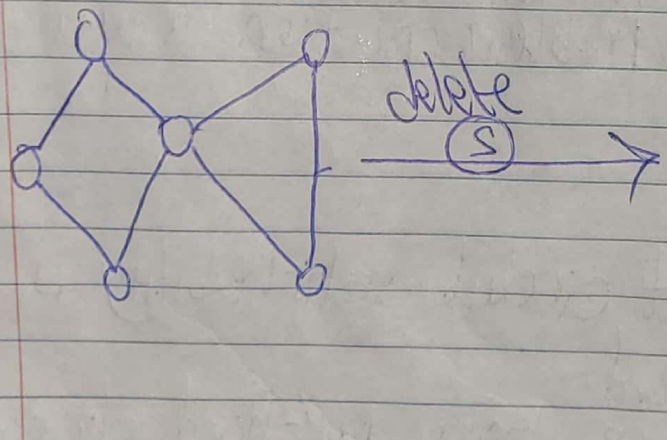
tranf-2 → If there is IS of G1 (independent set of G) then I would find it in the polynomial algorithm time by adding the weights.

delete (S) → delete the weight

and then for each pair of vertices in the independent set, will then compute the sum of the edge weights in the paths between both pairs.

∴ the Bottleneck Path Problem is NP-hard





to the bottleneck problem edge weighted undirected graph G1

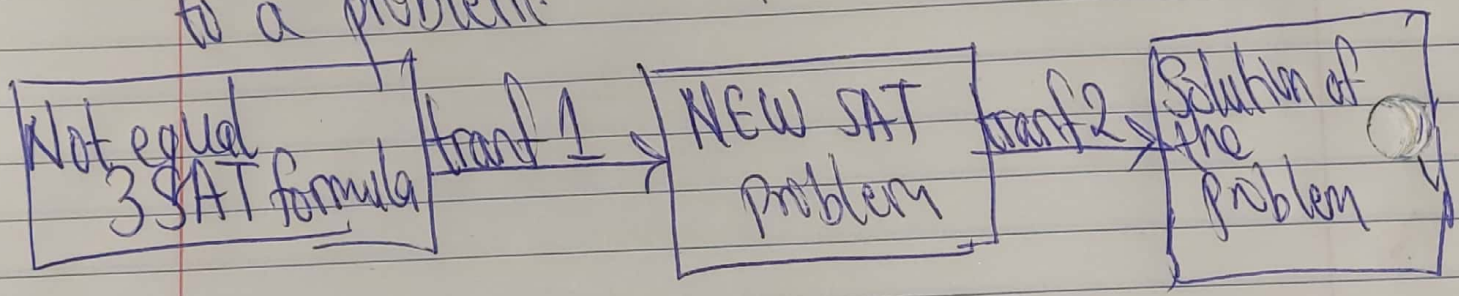
(6.) Problem (NEW-SAT)

Input: You are given a 3-SAT formula. A true value is considered as (+1) and a false value is considered (-1)

Question: Does there exist a truth-value assignment for each variable such that the sum of the three variables of each clause is between -1 and 1?

Solution

To prove this problem is NP-hard would take NP-hard problem and then reduce it to a problem.



3-SAT formula $= (v_1 \vee v_2 \vee \bar{v}_3) \wedge (\bar{v}_2 \vee \bar{v}_3 \vee \bar{v}_1) \wedge \dots$

$(v_1 \vee \bar{v}_2 \vee v_3) \wedge (\bar{v}_2 + v_1 + \bar{v}_4) \wedge (v_1 \vee v_4 \vee v_1)$

1 -1 1 -1 1 1 -1 -1 -1

$\bar{3} \; -1$

the 3SAT problem is going to be NP and that's because of the CNF formula checks whether the solution satisfies the problem

Transf 2:-
Would then deduce/reduce the problem:
- We are going to reduce the problem by NotAllEqual 3SAT.

Given the instance of NotAllEqual3SAT we will describe the New Sat so that the solution agree to the problem P.

$P' = (v_1 \vee v_2 \vee v_3) \wedge (\bar{v}_2 \vee v_1 \vee v_3) \wedge (v_1 \vee v_2 \vee v_3)$ (v1vv
1 0 1 — 0 1 1 — 1 0 0 — 0 1 0

So for each clause we would change the inverse of the variable from 0 to -1.
for P would give at least one negated variable and then the output is going to be between -1 and 1.

$P' = (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_2 v$
1 -1 1 — -1 1 1 — 1 -1 1 — -1 1

IF P' has a solution set then P has a solution set
We are going to change the negative values from -1 to 0 and that way, we are going to get our P

$P = (v_1 \vee v_2 \vee v_3) \wedge (v_2 \vee v_1 \vee v_3) \wedge (v_1 \vee v_1 \vee v_3) \wedge (v_1 \vee v_2 v$
1 0 1 — 0 0 1 — 0 1 0

and is not equal 3SAT

So therefore, the NEW SAT problem is NP-complete if not equal,

1. $1 + (-1) + 1 = 1$
2. $1 + (-1) - 1 = -1$

and is not in range

both cases 1 & 2 are polytime then the NEW SAT is polytime