CMSC 641 Design & Analysis of Algorithms Spring 2023
Section 01 Homework 8

Faisal Rasheed Khan

VB02734

vb02734@umbc.edu

1 Shipping Coffee

You work for a company that has extensive coffee operations alongside a river. Each day, coffee beans harvested in plantations upriver must be brought to processing plants downriver. The coffee beans are hand-picked, packed in standard containers and brought to the nearest port on the river. Your company has contracts with individual barge captains to pick up the containers and bring them downriver to the ports near the processing plants.

Unfortunately, each captain has docking privileges only at certain ports. It is your job to coordinate which captain will pick up coffee at which ports and where the coffee will be delivered. Furthermore, you have to contend with the following parameters and constraints:

• Your company has n coffee plantations, each one is near a different upriver port.

• Your company has m processing plants, each one is near a different downriver port.

• Your company has a fleet of k barges.

• The barges are slow and can only make one trip from upriver to downriver each day.

• Each barge can dock at multiple upriver ports and pick up coffee containers from multiple plantations and then float down river and drop off the coffee at multiple processing plants. However, there is not enough time for the barge to travel back upriver to pick up more coffee during the remainder of the day.

• Each plantation produces 10 containers of coffee each day.

• Each processing plant can process 75 containers of coffee each day.

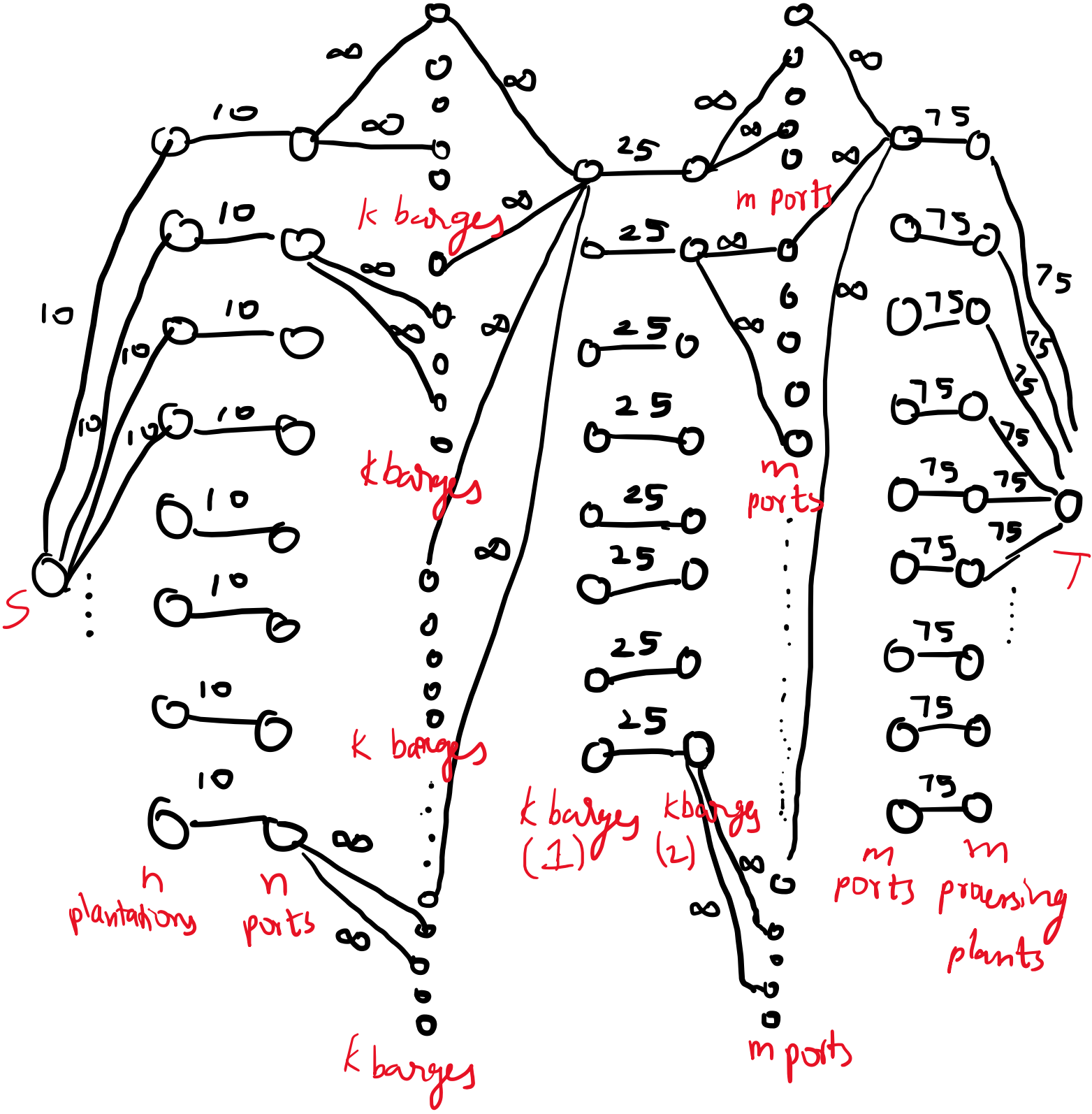
• Each barge can carry 25 containers.

• For each barge captain, you have a list of upriver ports and a list of downriver ports where the captain may dock.

Assignment: Use max flow to produce a solution that maximizes the total number of containers of coffee brought from the plantations to the processing plants without exceeding the capacity of the barges and without delivering more coffee to a processing plant than it can process. Do the following:

1. Using a combination of a diagram, descriptive labels and English sentences, describe how you can transform the problem described above into a flow network.

A.



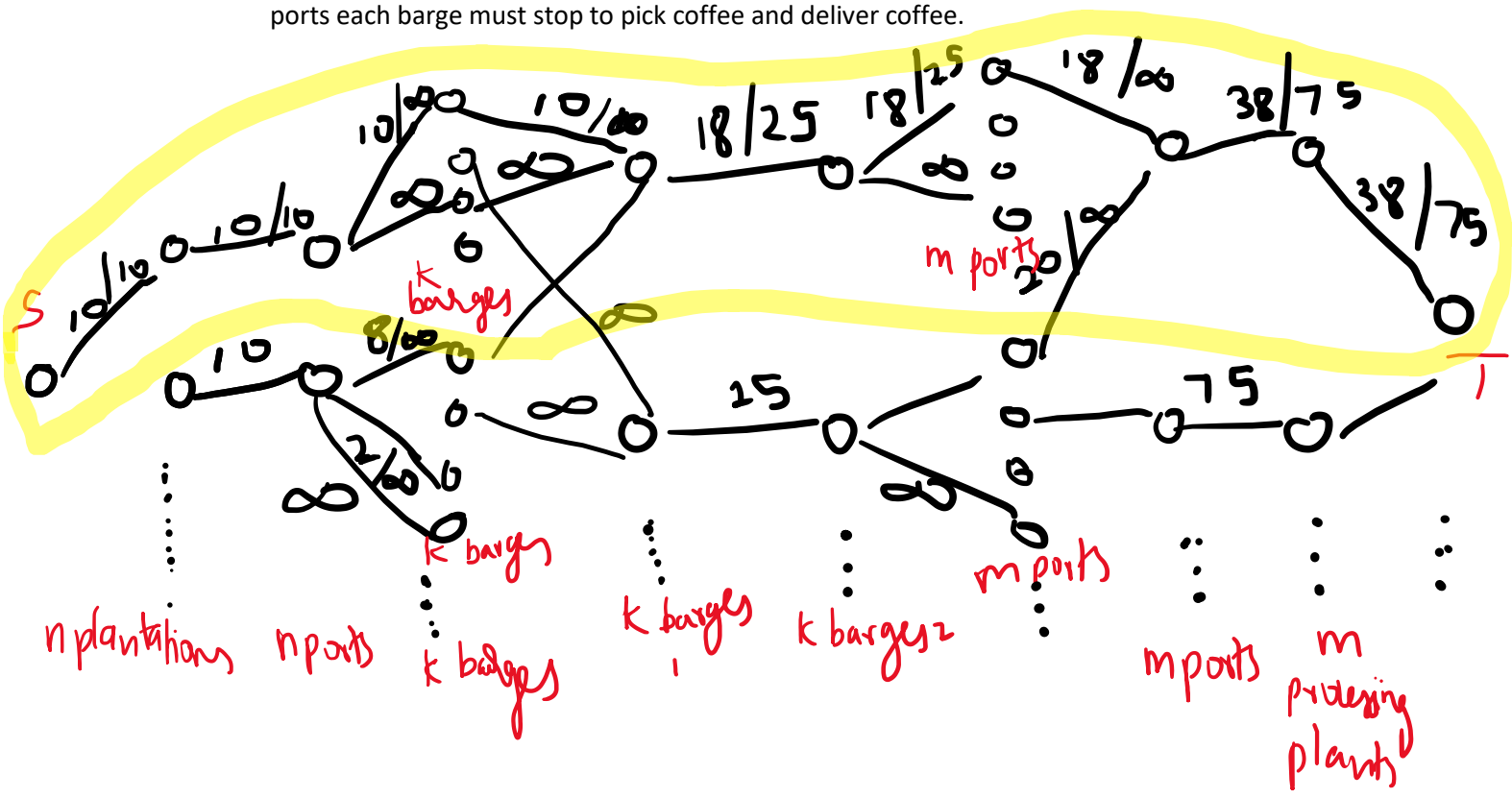
Buliding a flow network:

1 node per plantations n
1 node per ports n
1 node per barges per ports n*k
1 node per barges 1 k
1 node per barges 2 k
1 node per ports per barges m*k
1 node per ports m
1 node per processing plants m
10 plantations per node, so we send capacity of 10 from source S to each plantations n

Each plantation has each port n, so we have capacity 10 from plantation node to port node
Each barges has limited access to the port so we assign capacity ∞ from port to barges who have access to that port.
All the k barges meeting at the barge1 nodes have capacity ∞
Each barge can process only 25 units that's why capacity 25 from barge1 node to barge2 node.
Each barge has limited access to the m ports so we have ∞ capacity from barge2 to k*m port nodes.
And all the meeting of k*m ports nodes to m ports nodes, we have capacity ∞
Each m ports can send 75 capacity to processing plants as processing plants can take up to 75
From processing plants nodes m we have capacity 75 to sink T

2. Argue that your transformation works. That is, the maximum flow of the flow network should tell you which ports each barge must stop at to pick up coffee (and how much to pick up) and which ports each barge must stop at to deliver coffee (and how much to deliver).

A. By Applying Max Flow Min Cut Algorithm, it gives us all the information regarding which ports each barge must stop at to pick up coffee (and how much to pick up) and which ports each barge must stop at to deliver coffee (and how much to deliver).
Max Flow Min Cut Algorithm saturates the necessary edges and ensures that there are no augmenting paths left in the network.
Consider a part of network flow from the above network flow graph to understand which ports each barge must stop to pick coffee and deliver coffee.



Consider the highlighted section of the above network graph, the k barges 1 nodes gives the details regarding which barge must stop to pick up coffee and from that flow we can trace from which port how many units are coming.

Regarding which ports barge must deliver, we can see the flow from k barges 2 to k*m port nodes and we can see how many units its delivering.

We keep capacity constraints at the plantations 10 per node and delivering for processing plants nodes have capacity 75, we limit the capacities such that the conservation flow is maintained.

3. Explain why the solution your algorithm produced is the best possible solution.

A. The solution of my algorithm is the best possible solution because of the conservation flow maintained according to the given constraints. The maximum flow minimum cut theorem, gives the best possible maximum flow which flows from the above graph by saturating the unnecessary edges. Maximum flow is sent from the plantations to the processing plants. My Algorithm will ensure that we will not send flow more than the capacity according to the constraints and this maintains conservation flow and we can not improve the algorithm further while satisfying the constraints according to the problem.

2 Core is NP-Complete

Let G = (V, E) be an undirected graph. We say that C ⊆ V is a core of G if for every vertex u ∈ V − C there exists v ∈ C such that (u, v) ∈ E. That is, every vertex in G is either already in the core, C, or is adjacent to some vertex in C. In this problem, you will show that it is NP-complete to determine whether a graph has a small core. Formally, we define Core by:

Core

INPUT: an undirected graph G = (V, E) and natural number k.

QUESTION: Does G have a core C, where |C| ≤ k ? I.e, does there exist C ⊆ V such that |C| ≤ k and for every u ∈ V −C there exists v ∈ C such that (u, v) ∈ E?

Recall that Vertex Cover was shown to be NP-complete in lecture:

VC

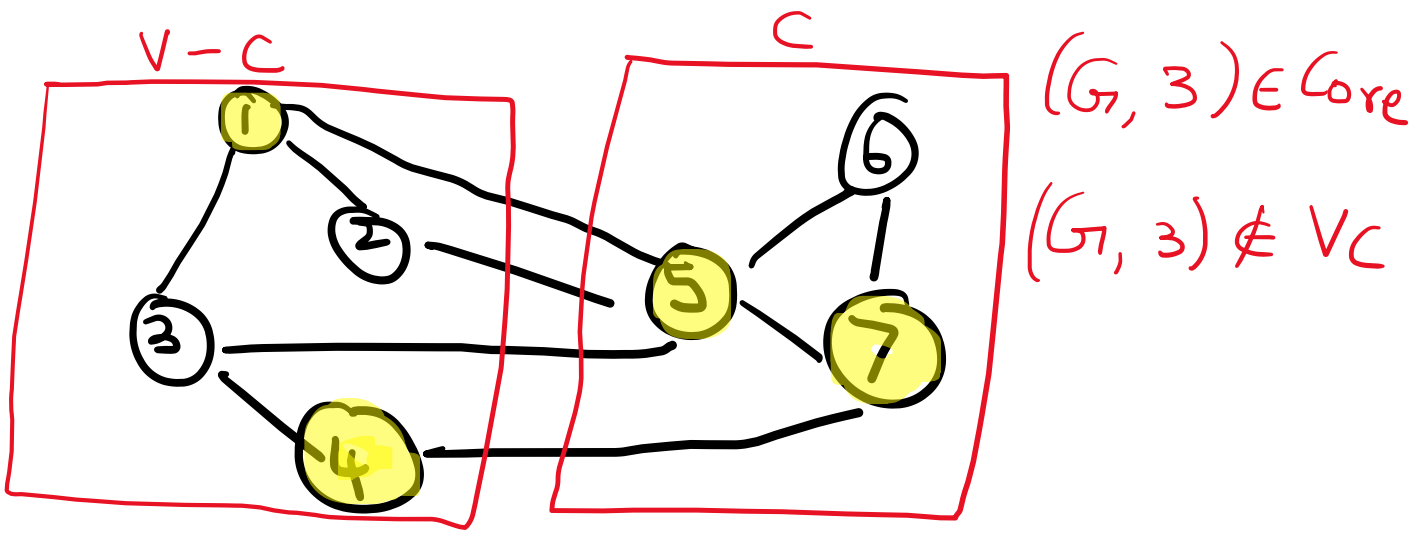
INPUT: an undirected graph G = (V, E) and natural number k.

QUESTION: Does G have a vertex Cover $V^{|}$, where $|V^{|}|$ ≤ k ? I.e, does there exist $V^{|} \subseteq V$ such that $|V^{|}|$ ≤ k and for every edge (u, v) ∈ E, we have u ∈ $V^{|}$ or v ∈ $V^{|}$?

Show that Core is NP-complete by constructing a $\leq_m^P$-reduction f from VC to Core.

1. Give an example of an undirected graph G and a number k such that (G, k) ∈ Core but (G, k) ∉ VC. You can thus conclude that the identity function does not reduce VC to Core.

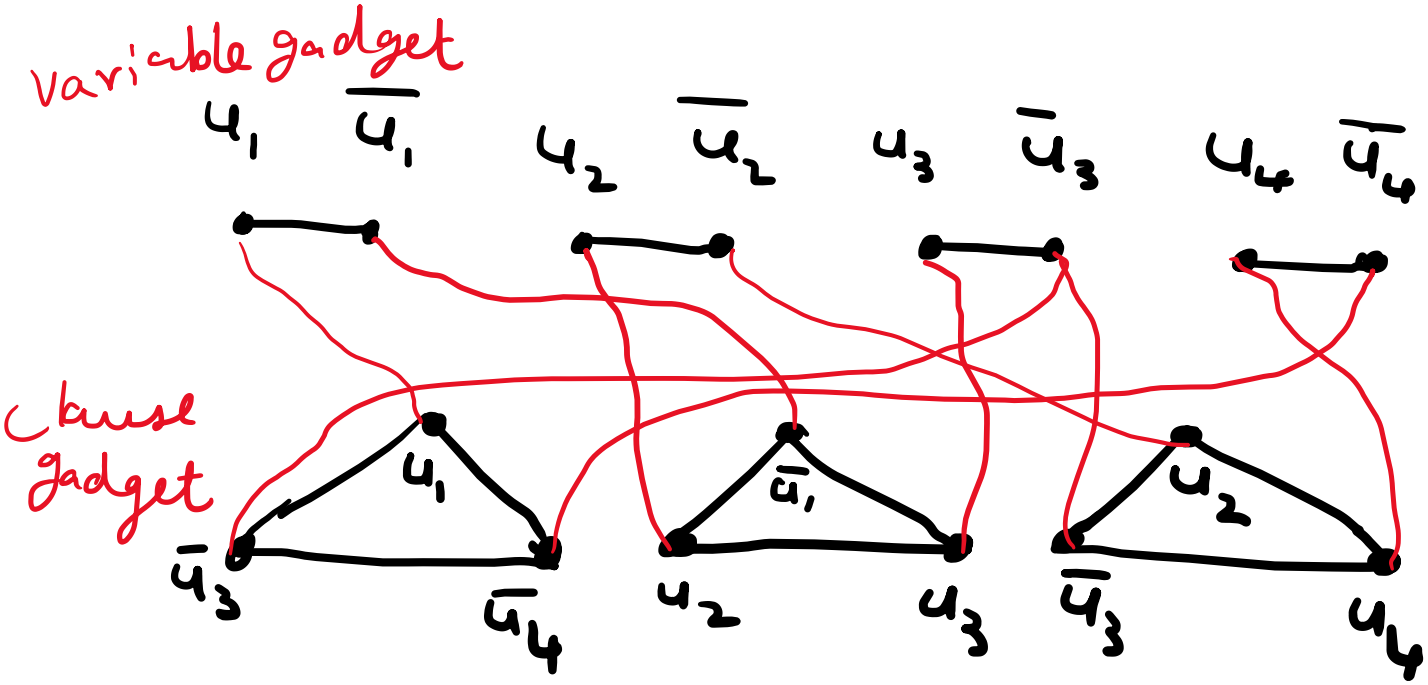
A.



The above graph contains k=3 for core and k=4 for vertex cover.
Therefore, we cannot use identity function to reduce VC to Core.

2. Describe a polynomial-time function f that given input (G1, k1) outputs (G2, k2) where G1 and G2 are undirected graphs and k1 and k2 are natural numbers.
   N.B.: the reduction function f has (G1, k1) as its ONLY input. It has no other information. In particular, it does not know whether G1 has a vertex cover with ≤ k1 vertices.

A. We need to reduce our polynomial time function f from VC to Core.
   First we show Vertex cover runs in polynomial time and it is in NP.
   If we reduce 3 satisfiability to any function then it belongs to NP.
   3SAT$\leq_m^P$Vertex Cover
   And then we show Vertex Cover$\leq_m^P$Core
   VertexCover(VC)={(G,K) | ∃ $V^{|} \subseteq V$, $|V^{|}| \leq k$ and for all (u,v) ∈ E either $u \in V^{|}$ or $v \in V^{|}$.
   Consider boolean function,$\phi = (u_1 \vee \bar{u}_3 \vee \bar{u}_4) \wedge (\bar{u}_1 \vee u_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$



ϕ has n variables and m clauses

G has 2n+3m nodes

K=n+3m

Claim: $\phi \in 3SAT <=> G$ *has a VC with k vertices*

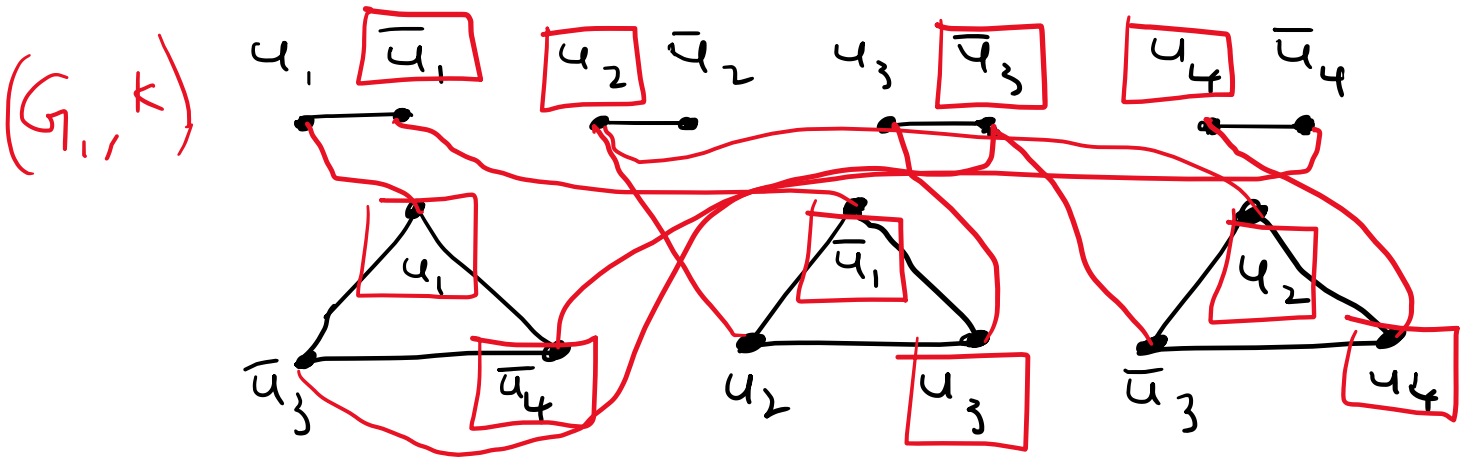
Proof:

Suppose ϕ *is satisfiable*

Consider a satisfying assignment if u_i *is true, pick* u_i *in variable gadget otherwise pick* $\bar{u}_i$

Each clause must have a true literal, pick other 2 nodes in clause gadget.

u_1*=false,* u_2*=true,* u_3*=false,* u_4*,=true*



Suppose G has a vertex cover with <= n+2m nodes

Each variable gadget has an edge that must be covered

Each clause gadget needs 2 vertex to cover 3 edges

Third node must be covered by variable gadget.

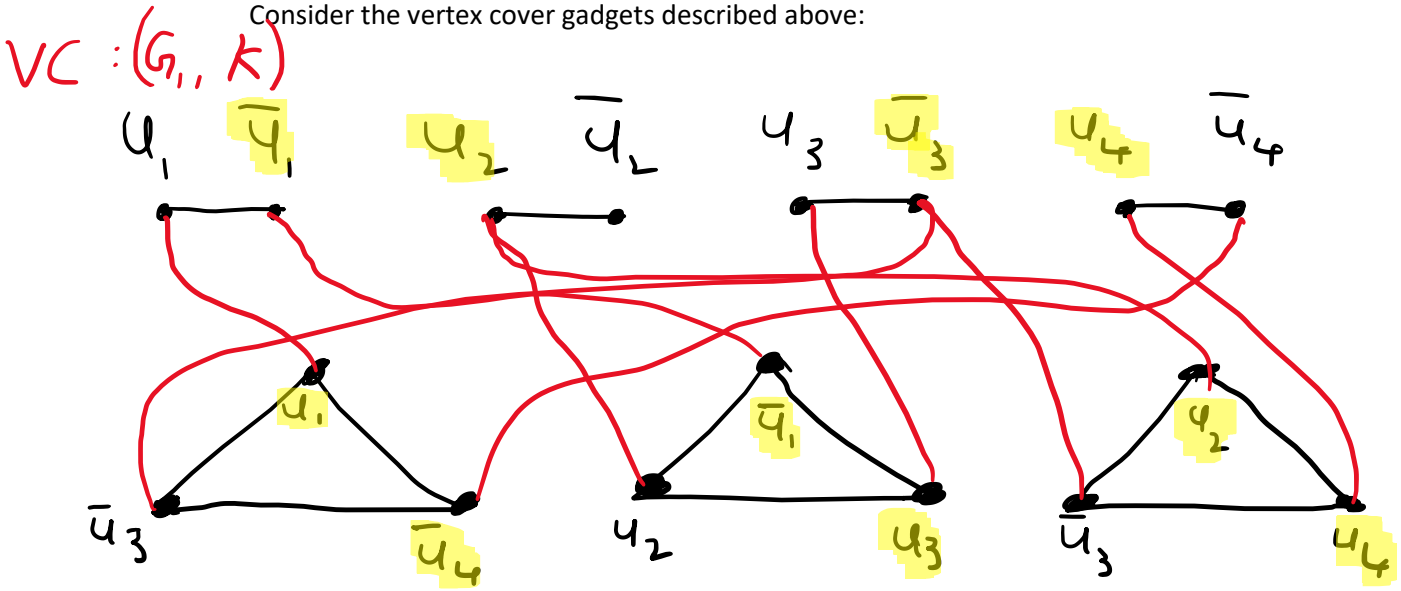
Therefore Each clause has a true literal.

Vertex Cover has been reduced from 3SAT in polynomial time function.

Now we reduce,

Vertex Cover$\leq_m^P$Core

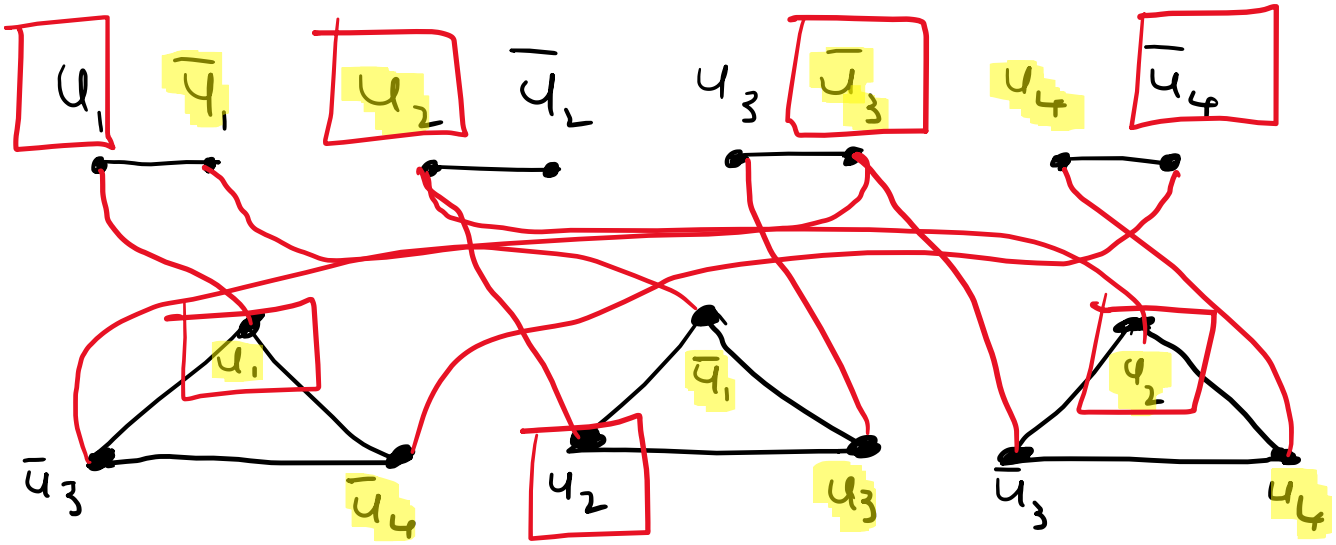
Consider the vertex cover gadgets described above:



Consider the satisfying assignment for Core

u_1*=true,* u_2*=true,* u_3*=false,* u_4*,=false*

Core: (G_2, k)



The above polynomial time reduction function takes (G_1 ,k) as input and outputs (G_2 ,k).
Each variable has one variable true
Each clause gadget has the true literal defined in the variable gadget, The VC ensures V-C edges attached to the C (Core).

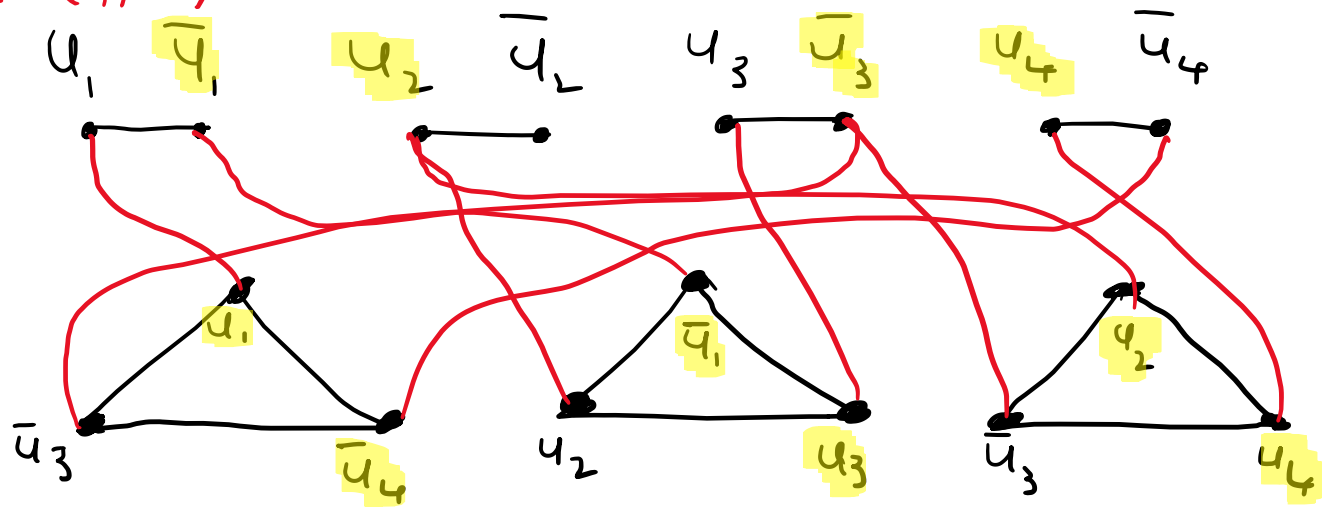
3. Briefly argue that f runs in time polynomial in the size of G.

A. Our G has 2n+3m nodes and Vertex Cover has n+2m nodes and we proved with the satisfiability that VC runs in polynomial time .
Our Core Graph has n+m nodes which is less than the Vertex Cover nodes, as this is NP Problem we will be verifying the solution so we will be considering what the answer will be, so n+m runs in a polynomial time.

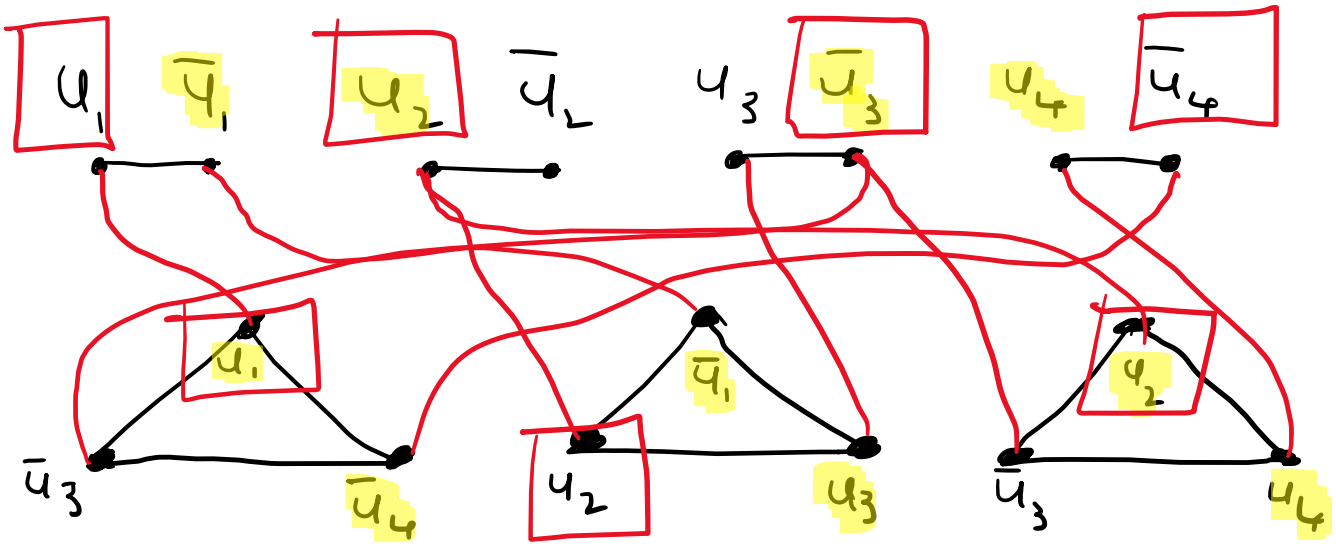
4. Prove that if (G1, k1) ∈ VC then (G2, k2) ∈ Core where (G2, k2) = f((G1, k1)) for the function f you described in part 1.

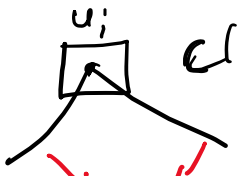
A. graph G = (V, E) and natural number k, G have a core C, where |C| ≤ k C ⊆ V such that |C| ≤ k and for every u ∈ V −C there exists v ∈ C such that (u, v) ∈ E
Claim: Vertex Cover$\leq_m^P$Core
Proof:
ϕ has n variables and m clauses
G has 2n+3m nodes
K=n+m
Consider boolean function,$\phi=(u_1 \vee \bar{u}_3 \vee \bar{u}_4) \wedge (\bar{u}_1 \vee u_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$
Consider the satisfying assignment for Core (G_2 ,k)
u_1=true, u_2=true, u_3=false, u_4,=false

VC : (G_1, k)



Cover : (G_2, k)





atleast one edge from V-C to C in Cover (G_2, k)

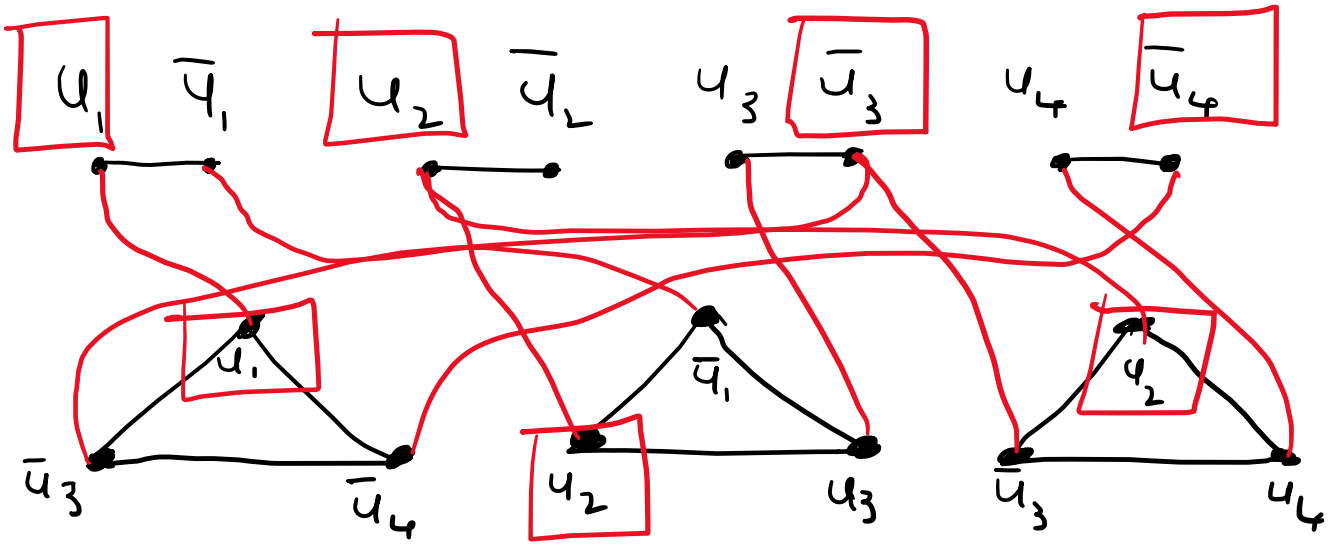
u_i → this vertex cover also ensures edge b/w V-C vertex to C

The above clause gadget & variable gadget satisfies

5. Prove that if (G2, k2) ∈ Core then (G1, k1) ∈ VC where (G2, k2) = f((G1, k1)) for the function f you described in part 1.

A.

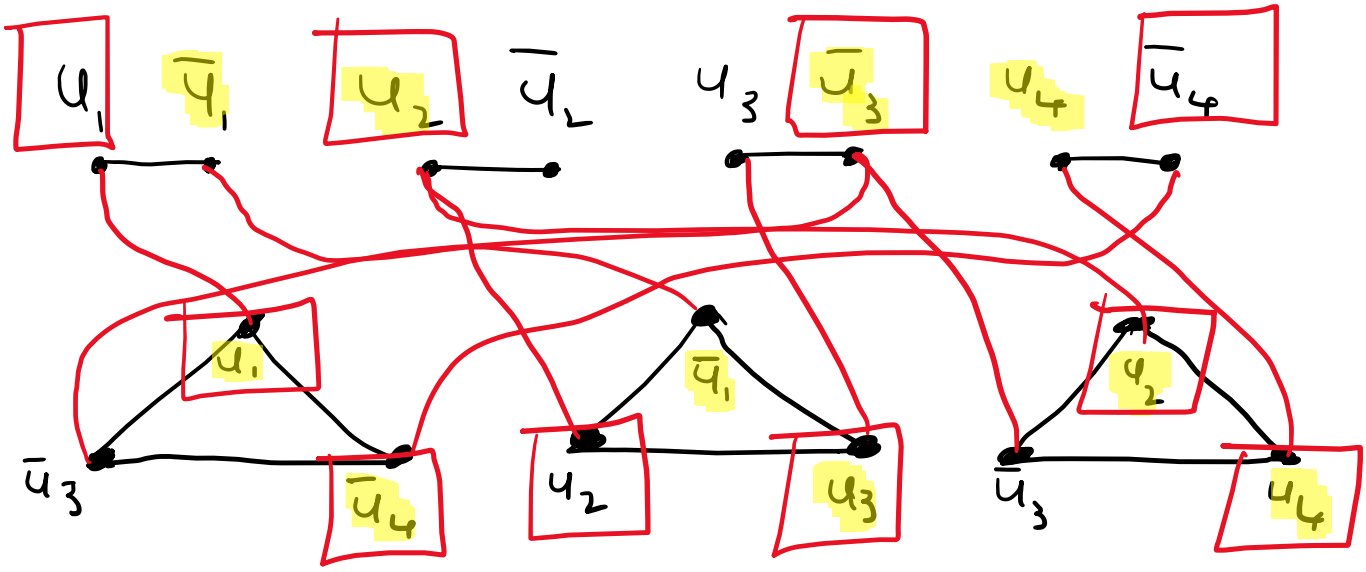
Core (G_1, k)



Suppose G has a vertex cover with <= n+m nodes
Each variable gadget has an edge that must be covered
Each clause gadget needs 1 vertex to cover 2 edges
Therefore Each clause has a true literal.

To prove VC from Core we just need to cover one vertex in the clause gadget such that the Vertex Cover satisfies. So just ensuring each clause gadget has 2 vertex to cover 3 edges and the third edge will be covered by variable gadget.

VC (G_1, k)



3 Hamiltonian Cycle vs Hamiltonian Path

The Hamiltonian Cycle (HC) and the Hamiltonian Path (HP) problems are similar, but different:
Hamiltonian Cycle (HC)

Input: an undirected graph G = (V, E)

Decide: Is there a cycle in G that visits every vertex in V exactly once?

Hamiltonian Path (HP)

Input: an undirected graph G = (V, E) and two vertices s ∈ V and t ∈ V , where s ≠ t.

Decide: Is there a path from s to t in G that visits every vertex in V exactly once?

Construct a $\leq_m^P$-reduction f from HC to HP. Then, prove your function f is indeed a polynomial time many-one reduction. Use the following steps:

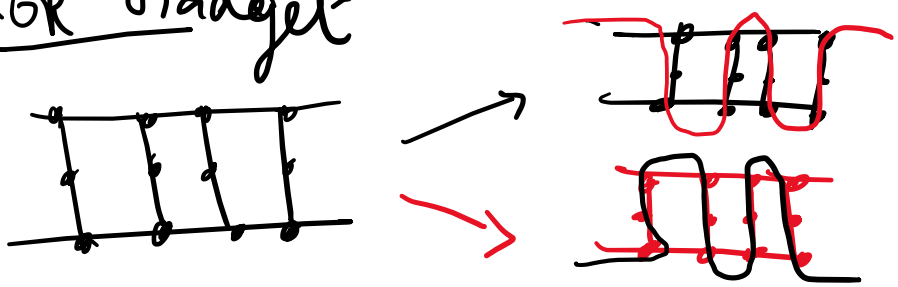
1. Give a high-level description of a polynomial-time function f that given an undirected graph G, outputs a graph $G^{|}$ and two vertices s and t.
   N.B.: the reduction function f has G as its ONLY input. It has no other information. In particular, it does not know whether G has a Hamiltonian Cycle. Furthermore, f must output $G^{|}$, s and t in the case where G has a Hamiltonian Cycle and when G does not have a Hamiltonian Cycle. Specifically, do not start your description with "Let (u, v) be an edge in a Hamiltonian Cycle of G."

A.

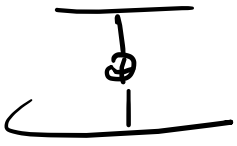
Hamiltonian Cycle(HC) = G(V, E) is an undirected graph s.t there exists a cycle in G that includes every vertex of V exactly once.

HC ∈ NP because we can guess & verify

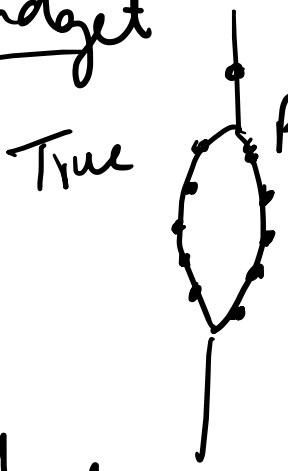
XOR Gadget



Short hand:

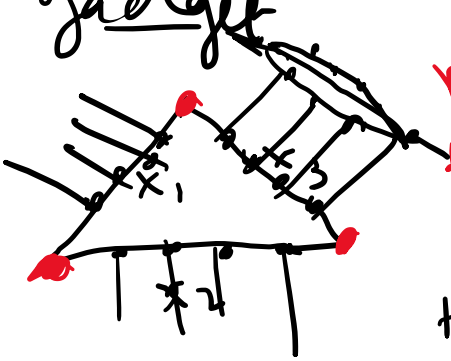


Variable gadget



only covers one side, not covered vertex should be covered by variable clause

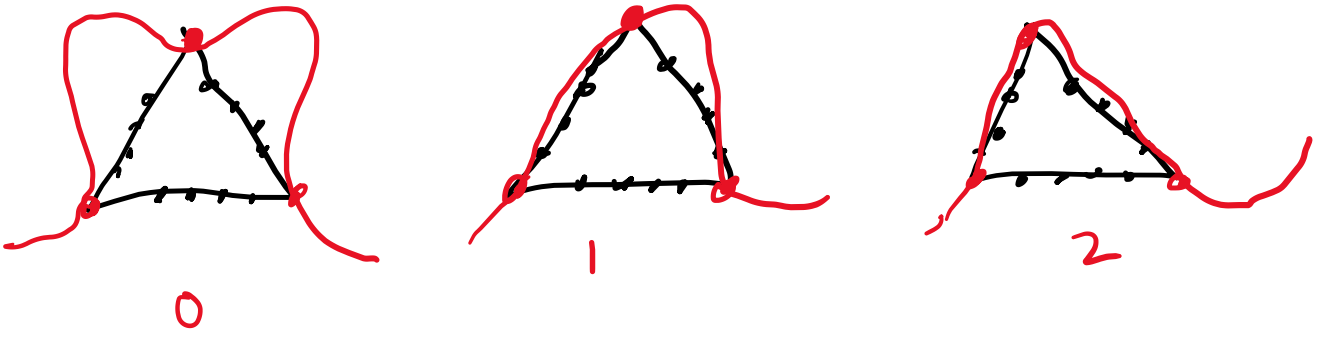
Clause gadget



red ones are connected to all red corners

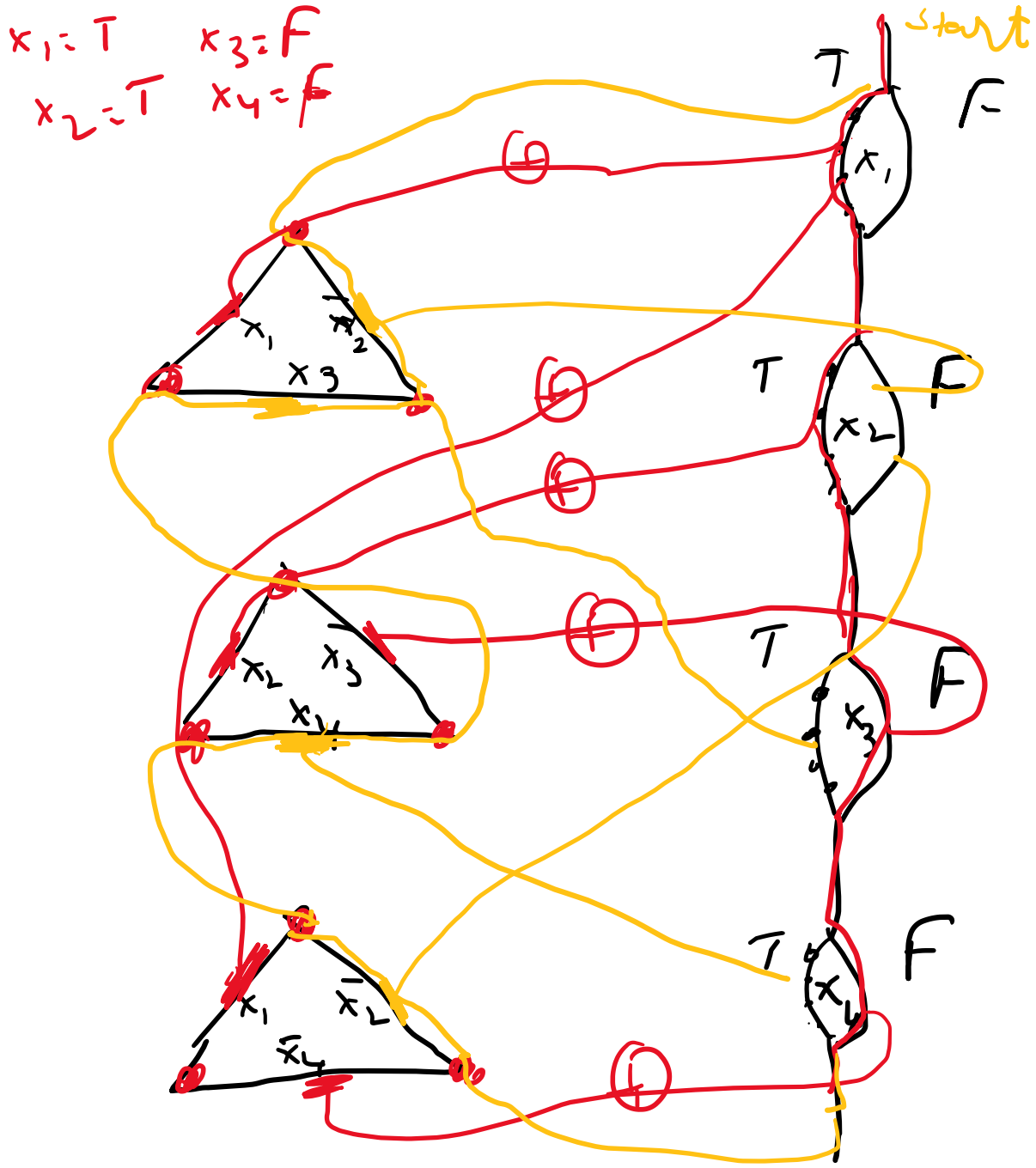
black ones are connected to XOR Gadget

Traverse:



Consider a Boolean function, ϕ

$$\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee \bar{x}_3 \vee x_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4)$$



Claim: All vertices visited exactly once
variable gadgets take true/false,
if true it covers one side of clause
if we don't visit one side in variable gadget, clause gadget takes care.

Proof: Every clause is satisfied so I visit correct edges.

with xor gadgets we visit one side or the other, not both.
For clause we pick max 2 sides not 3
Any H.C will have edge to one side of clause gadget picked up by variable gadget
variable picks one side (i.e T/F) not both.
we come down the true side we make visible true & if false side we make false
every gadget has one side that is taking care of variable gadget.
Therefore we reduced Hamiltonian cycle.

Hamiltonian Path:

an undirected graph G = (V, E) and two vertices s ∈ V and t ∈ V , where s ≠ t.

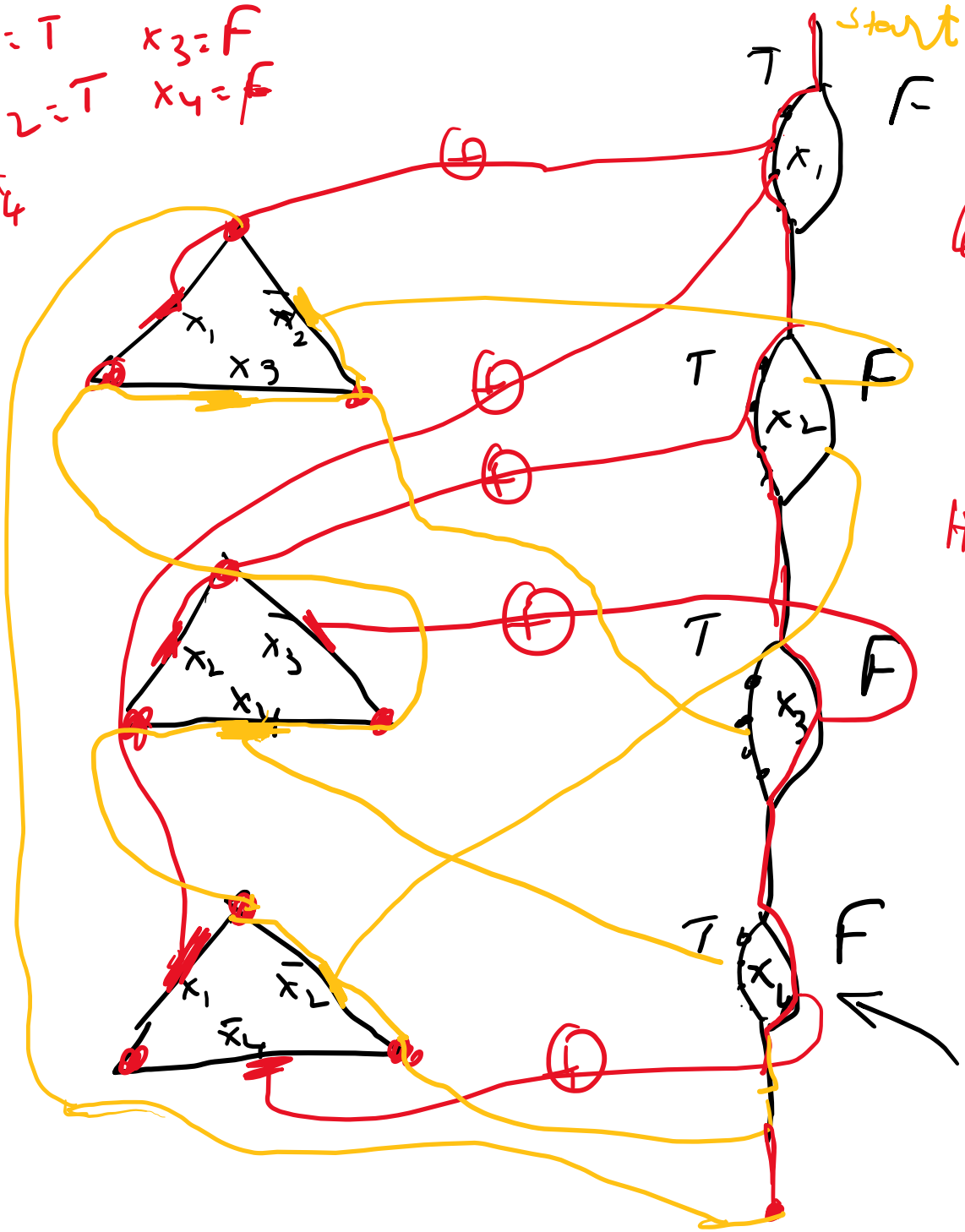
a path from s to t in G that visits every vertex in V exactly once

Considers similar as Hamiltonian cycle
except source vertex & end vertex is specified.

$x_1 = T \quad x_3 = F$

$x_2 = T \quad x_4 = F$

x_4



let x_4 be the end vertex of Hamiltonian path

End vertex

Claim: All vertices visited exactly once (with start & end vertex)
variable gadgets take true/false
if true it covers one side of clause
if we don't visit one side in variable gadget, clause gadget takes care.

Proof: Every clause is satisfied so I visit correct edges.
with xor gadgets we visit one side or the other, not both.
For clause we pick max 2 sides not 3
Any H.C will have edge to one side of clause gadget picked up by variable gadget
variable picks one side (i.e T/F) not both.
we come down the true side we make visible true & if false side we make false
every gadget has one side that is taking care of variable gadget.
Therefore we reduced Hamiltonian Path

2. Briefly argue that f runs in time polynomial in the size of G.

A. We visit each node exactly once and there for we consider one edge per node we discard remaining edges.
Therefore the above function takes polynomial time with the help of XOR Gadgets.

3. Prove that for all undirected graphs G and ($G^{|}$, s, t) = f(G) if G has a Hamiltonian Cycle, then there is a Hamiltonian Path from s to t in $G^{|}$.

A. Above we have considered reducing the Hamiltonian path from Hamiltonian cycle. Hamiltonian path is same as Hamiltonian cycle except end nodes should be source and vertex of it. HC consists of all vertexes so there also exists vertex start and end of Hamiltonian path.
The XOR Gadgets helps us in visiting each vertex exactly once, The XORs other path traverse of vertex is handled by clause gadget.
Therefore, G and ($G^{|}$, s, t) = f(G)

4. Prove that for all undirected graphs G and ($G^{|}$, s, t) = f(G) if there is a Hamiltonian Path from s to t in $G^{|}$ then G has a Hamiltonian Cycle.

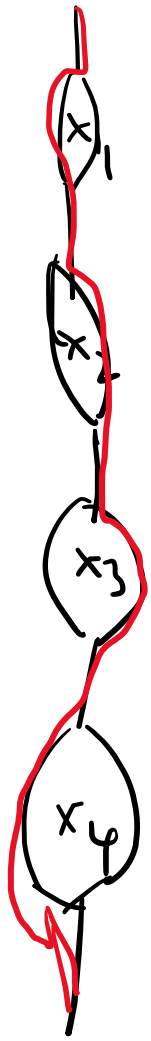
A.

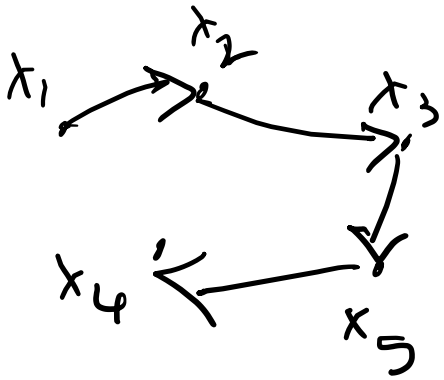
G & (G', S, t) ∈ HP then
G has Hamiltonian cycle
This is true as we have HP then we also have HC
∴ For satisfying Assignment we go either True / False & remaining path which may be T or F will be covered by clause gadget.

H·P

start x_1

End x_4



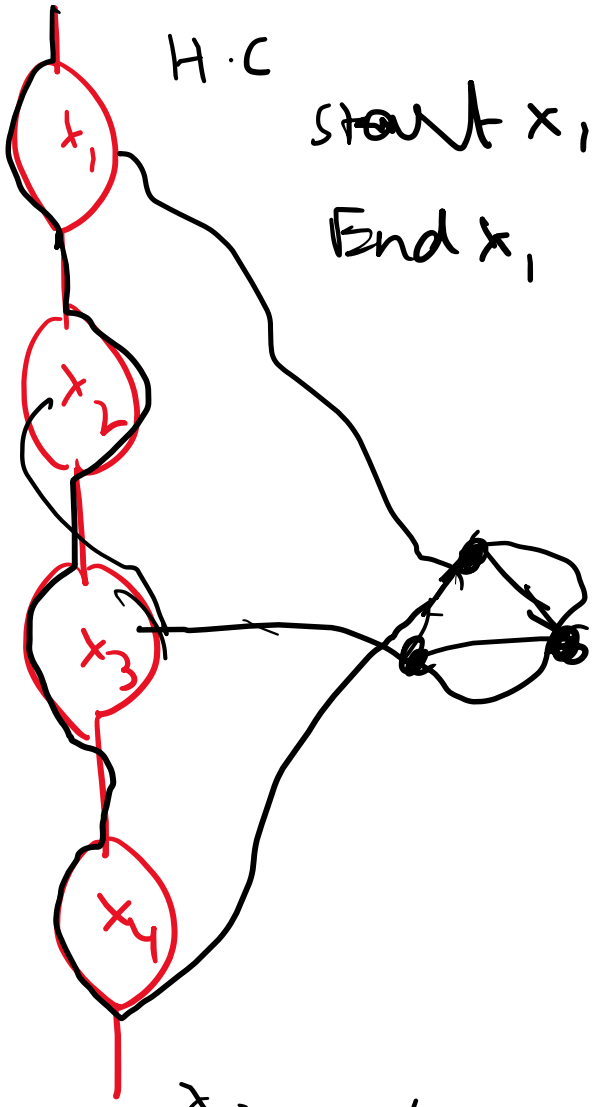


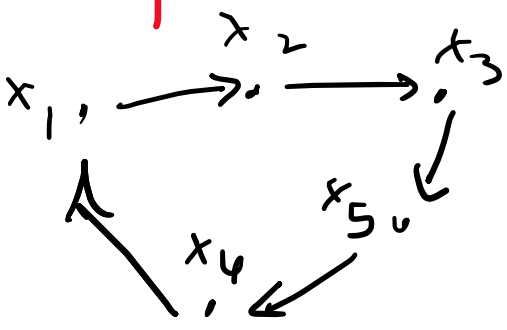
Path

H·C

start x_1

End x_1





cycle

References:

https://archive.org/details/introduction-to-algorithms-by-thomas-h.-cormen-charles-e.-leiserson-ronald.pdf/page/427/mode/2up

https://archive.org/details/AlgorithmDesign1stEditionByJonKleinbergAndEvaTardos2005PDF/page/n201/mode/2up

12.1-Network-Flow-post.pdf - Google Drive

12.2-Max-Flow-Min-Cut-post.pdf - Google Drive

13-Network-Flow-Applications-post.pdf - Google Drive

15.3 Edmonds-Karp with Proofs (optional).pdf - Google Drive

16-NP-Completeness-post.pdf - Google Drive

17.1-SAT-vs-Coloring-post.pdf - Google Drive

18.3-Hamiltonian-Cycle-post.pdf - Google Drive