HW - due 12/5 / Mon midnight

ch 16.4

1.) a graphic matroid   MSIP

$G = (V, E)$
an undirected graph

if $X \in I$, $Y \in I$, $|X| > |Y|$

then $\underline{\exists a \in X - Y}$ $\{a\} \cup Y \in I$ ← 작은것

some element must exist in the set $X - Y$

Exchange property
교환특성

$M_G = (S, I)$ where

$S = E$ a subset of E called $X \in I$ iff X is acyclic
$I \subseteq 2^S$

↓

prove that this definition ①② 3 16.4

My Answer

when $G = (V, E)$ is an undirected graph $M_G = (S, I)$ is a matroid

① $S = E$ is an finite set since the nodes of graph G is finite

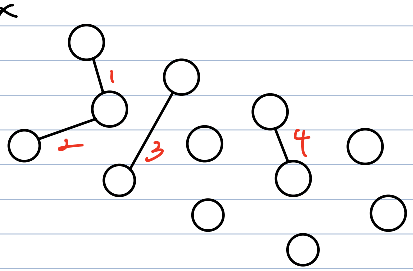
② if $X \in I$ then all subsets of $X \in I$ → Hereditary

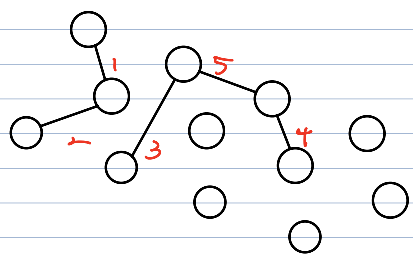
③ Exchange property

let's say $G_A = (V, A)$, $G_B = (V, B)$, G_A and G_B are forest of graph G
$|B| > |A|$ (G_B has more nodes than G_A) they are both Acyclic.

forest G_A has $|V| - |A|$ tree and G_B has $|V| - |B|$ trees. The numbers of G_B trees are smaller than trees of G_A

Ex





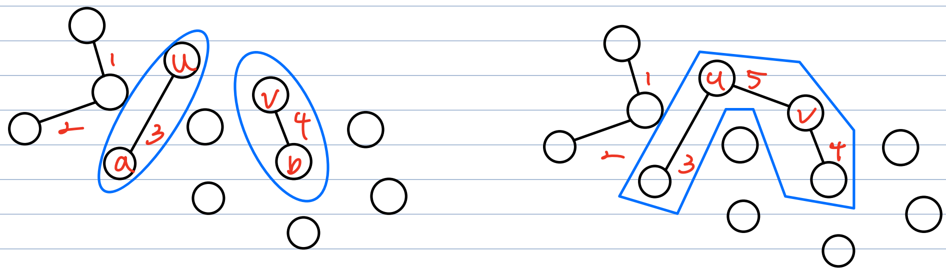
G_A G_B

A: 4
3 trees

B = 5
2 trees

G_B has a lesser trees than G_A so G_B must has some tree which connects 2 trees of G_A

4.



G_A $\quad\quad\quad\quad$ G_B

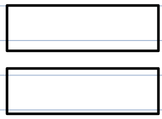
$G_A \cup \{(u,v)\} \in G_B$ so the exchange property fulfills.

this is only true when G_B and G_A are both acyclic. when they are cycle from both sets. then the tree # could be the same even when $|B| > |A|$

ch 22.3 $\boxed{\text{Stack}}$

2. DFS (G,S)

Adj ( )



Give an implementation of DFS (G,S) that uses a stack explicitly. Your code should consist of 2 parts:

(1) initialization

(2) an iterative parts that explicitly uses push and pop operations.

dfs.py 1 ×

dfs.py > DFSGraph > dfsVisit

```python
from pythonds.graphs import Graph

class DFSGraph(Graph):
    def __init__(self):
        super().__init__()
        self.time = 0

    def dfs(self):
        for aVertex in self:
            aVertex.setColor('white')
            aVertex.setPred(-1)

        for aVertex in self:
            if aVertex.getColor() == 'white':
                self.dfsVisit(aVertex)

    def dfsVisit(self, startVertex):
        self.time += 1
        startVertex.setDiscovery(self.time)
        startVertex.setColor('gray')
        for nextVertex in startVertex.getConnections():
            if nextVertex.getColor() == 'white':
                nextVertex.setPred(startVertex)
                self.dfsVisit(nextVertex)
        startVertex.setColor('black')
        self.time += 1
        startVertex.setFinish(self.time)
```