Adam Jablonski

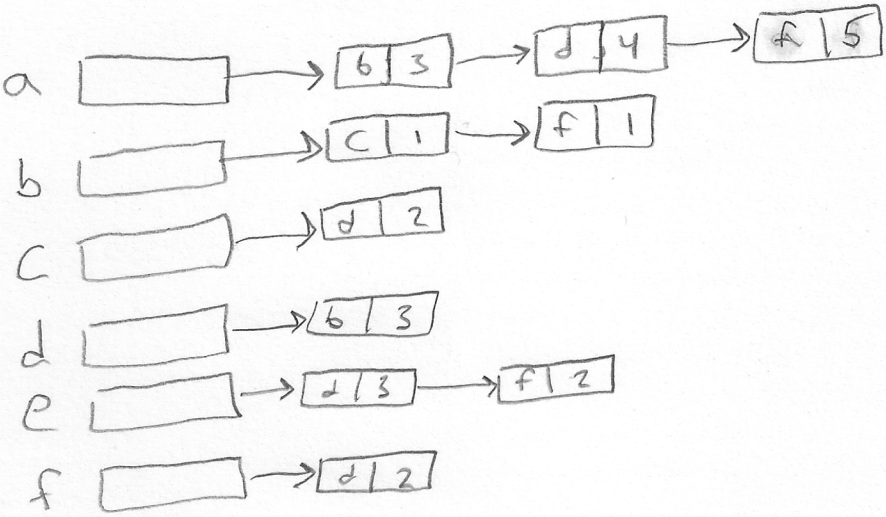
```
Procedure computeEquivalentSets
        Begin
        Set = Initialize ( 1..n )
        Print Equivalent( p, q)

def isAccepting( transitions[state, input] )
        if transitions[0] %2 == 0 :
                return true
        else:
                return false

def equivalent(p, q)
        if p.isAccepting == q.isAccepting:
                if p[1] == 0 and q[1] ==0 and p[0] == q[0]:
                        return True
                elif p[1] == 1 and q[1] == 1 and p[0] == q[0]:
                        return True
                else:
                        return False
        elif p == q  # same state
                return True
        else:
                return false
```

Section 6.1

a)

	a	b	c	d	e	f
a		3		4		5
b			1			1
c				2		
d		3				
e				3		2
f				2		

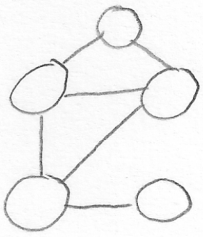
b)



Problem 3

Show $\sum_{i=1}^{n} d_i = 2m$

max degree from a vertex is $n-1$

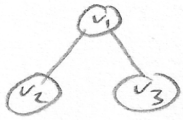


An edge touches two vertices so you double count each edge.

1 edge

$(v_1 = 1\text{edge} + v_2 = 1\text{edge}) = 2\text{edge}$

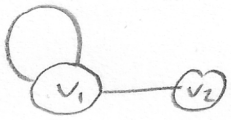
$2 \cdot 1 = 2$ ✓



2 edges

$4\text{edge} = (v_1 = 2\text{edge} + v_2 = 1\text{edge} + v_3 = 1\text{edge}$

$2 \cdot 2 = 4$ ✓



2 edges

$(v_1 = 3\text{edge} + v_2 = 1\text{edge}) = 4\text{edge}$

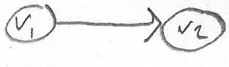
$2 \cdot 2 = 4$ ✓

Problem 4

Show indegree sum = out degree sum

For every out node there is an in node

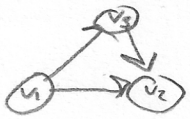
You can't have an inode without an out node



$v_1 = 1$ out

$v_2 = 1$ in

$1 = 1$ ✓

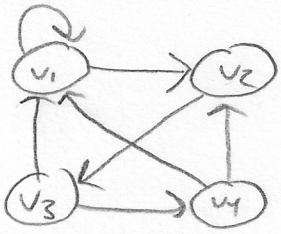


$v_1 = 2$ out : 0 in

$v_2 = 0$ out : 2 in

$v_3 = \frac{1 \text{ out} : 1 \text{ in}}{3 \text{ out} : 3 \text{ in}}$

$3 = 3$ ✓



$v_1 = 2$ out : 3 in

$v_2 = 1$ out : 2 in

$v_3 = 2$ out : 1 in

$v_4 = 2$ out : 1 in

$7 = 7$ ✓

Each edge is both an inNode and an out node