Let $L = \{ a^n b^m c^p : n \neq p, \text{ and } n, m, p \geq 0 \}$

1. Design a context-free grammar that generates L.

2. Design a PDA that accepts L.

# Post's Correspondence Program (PCP):

P = {(a, ab), (b, ca), (ca, a), (abc, c)},

MATCH:

(a, ab) (b, ca) (ca, a) (a, ab) (abc, c)

a b ca a abc = abcaaabc

ab ca a ab c = abcaaabc

# PCP over a 1-symbol alphabet {a} is easy:

If P has a pair $(a^k , a^k)$ the answer is yes.

If P has two pairs:

one with more symbols in the first part: $T_1=(a^{r+i} , a^r)$

one with more symbols in the second part: $T_2=(a^s , a^{s+j})$

the answer is yes-

take j copies of $T_1$ and i copies of $T_2$.

Otherwise, the answer is no.

Given: No algorithms exist to solve PCP over an arbitrary alphabet.

Theorem:

No algorithm exist to solve PCP over {0, 1}.

Proof: Assume there is an algorithm which solves PCP over {0,1}. Such an algorithm can be used to solve PCP over an arbitrary alphabet as per the model solutions of assignment #2-

the problem over an arbitrary alphabet can be converted to one over {0,1} by encoding in either unary or binary (your choice).

# The Pumping Lemma for Regular Languages:

If L is a language accepted by a DFA with k states, and $w \in L$, $|w| \geq k$, then $\exists$ x, y, z such that

1. $w = x\,y\,z$,
2. $y \neq \varepsilon$,
3. $|\,x\,y\,| \leq k$, and
4. $x\,y^n\,z$ is in L for all $n \geq 0$.

The pumping lemma is NOT strong enough to work directly to prove that certain languages are not regular.

Let L be a language which has a constant k such that for all $w \in L$, $|w| \geq k$, $\exists$ x, y, z such that

1. $w = x\,y\,z$,
2. $y \neq \varepsilon$,
3. $|\,x\,y\,| \leq k$, and
4. $x\,y^n\,z$ is in L for all $n \geq 0$.

This is necessary but not sufficient for a language to be regular.

Then you CANNOT conclude that L is regular.

Counterexample: See assignment 3.

$L_1 = \{\, u\,u^R\,v : u, v \text{ in } \{a, b\}^+ \}$

Theorem 1:

If Hamilton Cycle has an algorithm which is polynomial time, then so does Hamilton Path.

Theorem 2:

If Hamilton Path has an algorithm which is polynomial time, then so does Hamilton Cycle.

Currently nobody knows a polynomial time algorithm for either problem.

$L = \{ u$ $L_1 = \{ u\, u^R\, v : u, v \text{ in } \{a, b\}^+ \}$

Why does every string w in L of length at least 4 have a string *y* that can be pumped?

Case 1: $|u| = 1$.

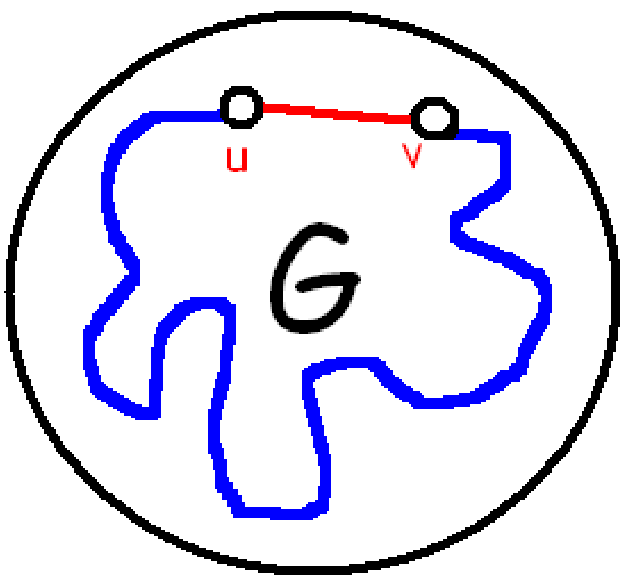
Case 2: $|u| > 1$.

# Hamilton Path

If (G has a Hamilton Cycle) return(yes)

For each missing edge (u, v), if G+(u,v) has a Hamilton Cycle return(yes)

Return(no)




Time:

$O(n^2 * p(n, m+1))$

## What is inefficient about this code?

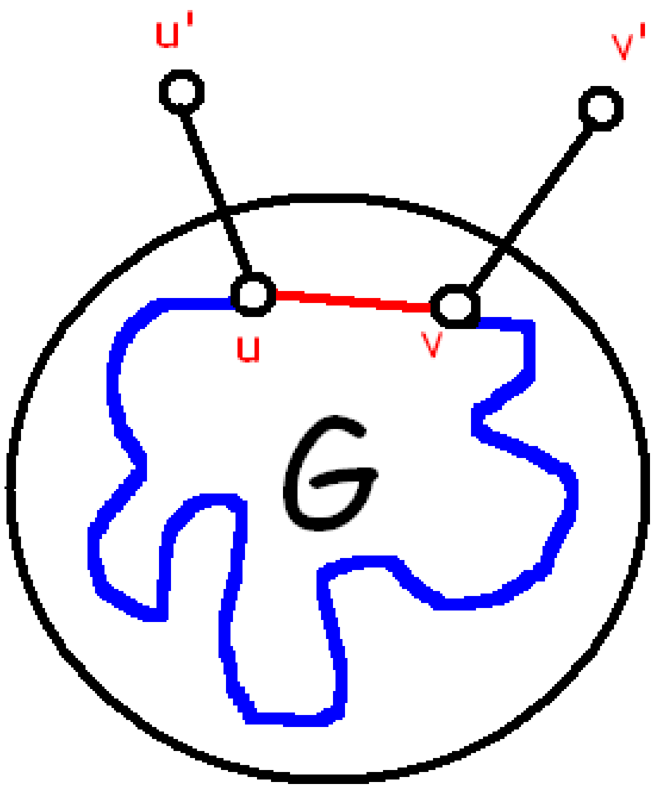
```
for (i=0; i < n; i++)
    for (j=0; j < n; j++)
    {
        tmp= G[i][j]; G[i][j]=1; G[j][i]=1;
        if (hamilton_cycle(n, G)) return(1);
        G[i][j]=tmp; G[j][i]=tmp;
    }
return(0);
```

My solution:

```
if (hamilton_cycle(n, G)) return(1);
for (i=0; i < n; i++)
    for (j=i+1; j < n; j++)
        if (G[i][j] == 0)
        {
            G[i][j]=1; G[j][i]=1;
            found= hamilton_cycle(n, G);
            G[i][j]=0; G[j][i]=0;
            if (found) return(1);
        }
return(0);
```

# Hamilton Cycle:

For each edge (u,v) if G + u′ + v′ + (u, u′) + (v, v′) has a Hamilton Path return (yes)




Return(no)

Time:
$O(n^2\ q(n+2,\ m+2)\ )$

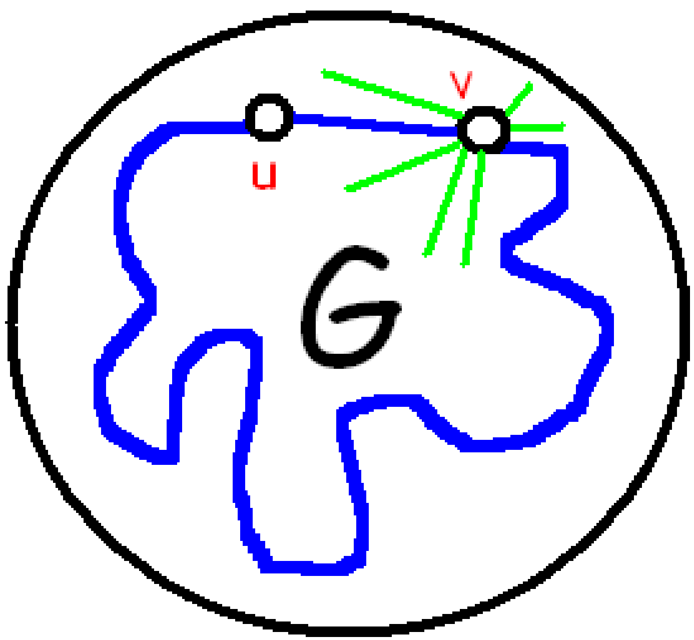
My solution:

```
for (i=0; i < n+2; i++)
    for (j=n; j < n+2; j++) { G[i][j]=0; G[j][i]=0;}

for (i=0; i < n; i++)
  for (j=i+1; j<n; j++)
    if (G[i][j])
    {
       G[i][n]=1; G[n][i]=1; G[j][n+1]=1;G[n+1][j]=1;
       if (hamilton_path(n+2, G)) return(1);
       G[i][n]=0; G[n][i]=0;G[j][n+1]=0;G[n+1][j]=0;
     }
return(0);
```

# Hamilton Cycle: A better solution

For each neighbour of v if $G + u' + v' + (u, u') + (v, v')$ has a Hamilton Path return (yes)




Return(no)

Time:

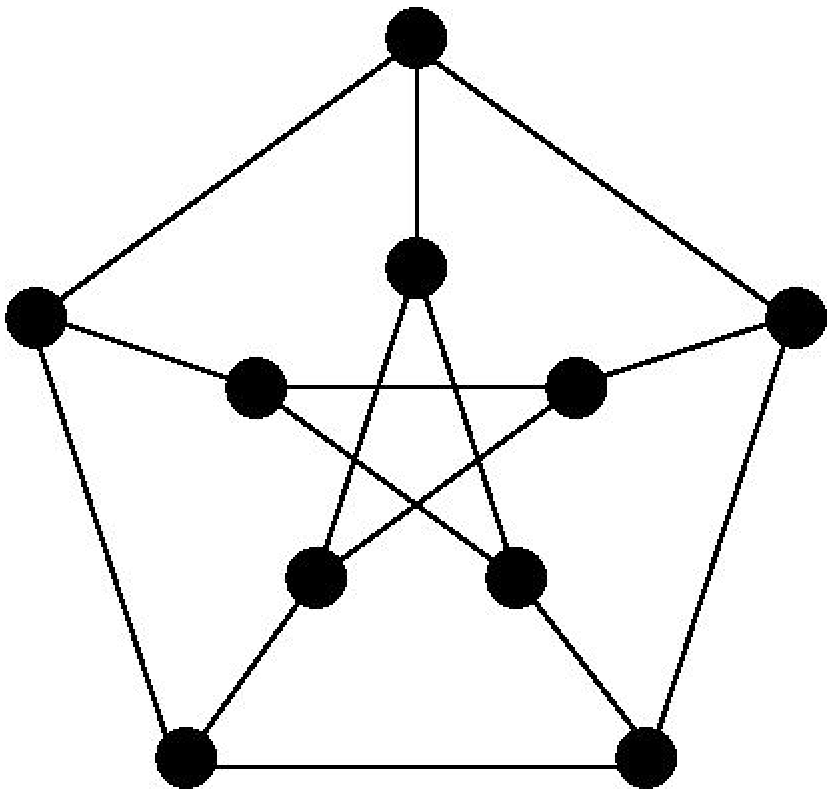
$O(n * q(n+2, m+2))$

## Some algorithms which do not work:

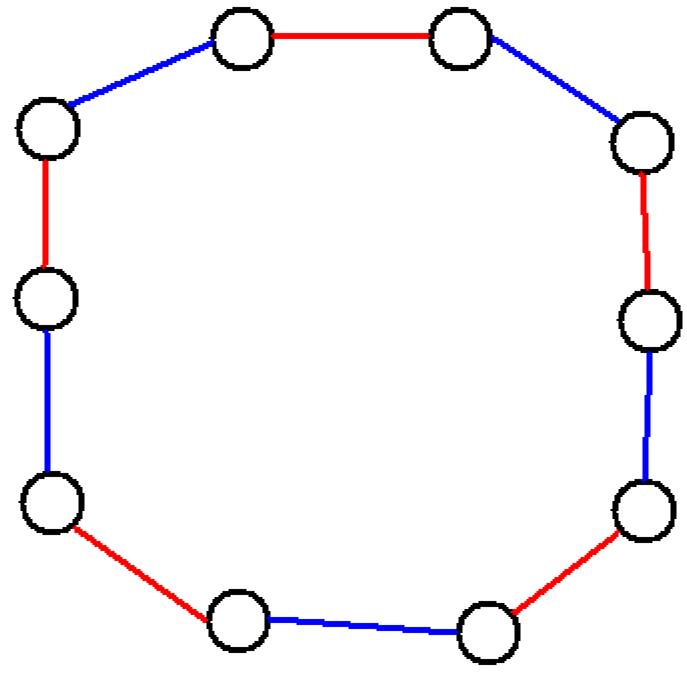
If G-v has a Hamilton path for all vertices v this does NOT mean G has a Hamilton cycle.

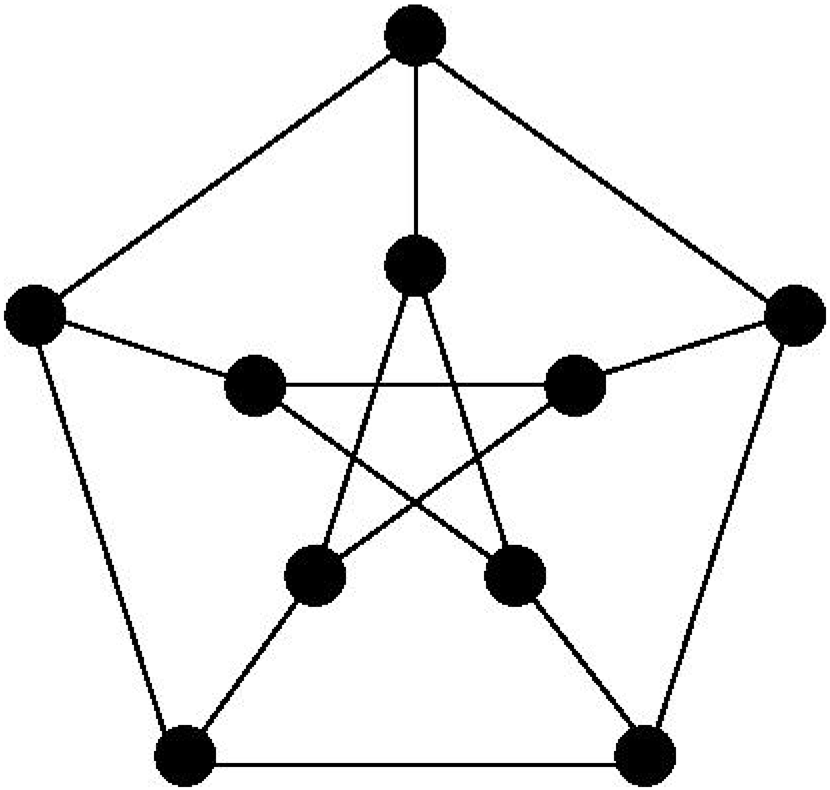
If G-e has a Hamilton path for all edges e this does NOT mean G has a Hamilton cycle.

# Theorem: The Petersen graph has no Hamilton cycles.

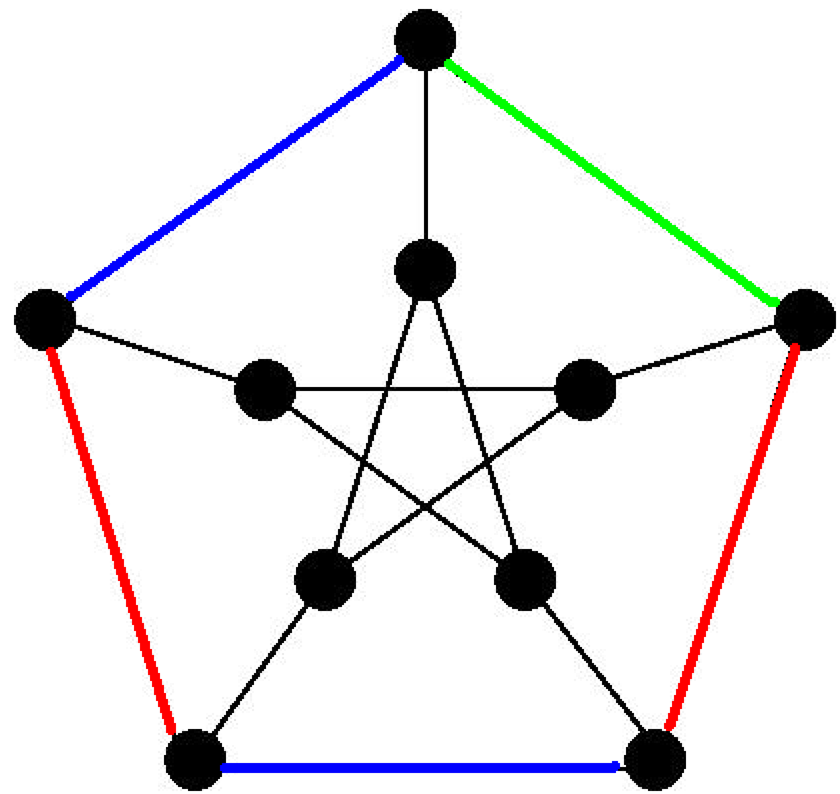


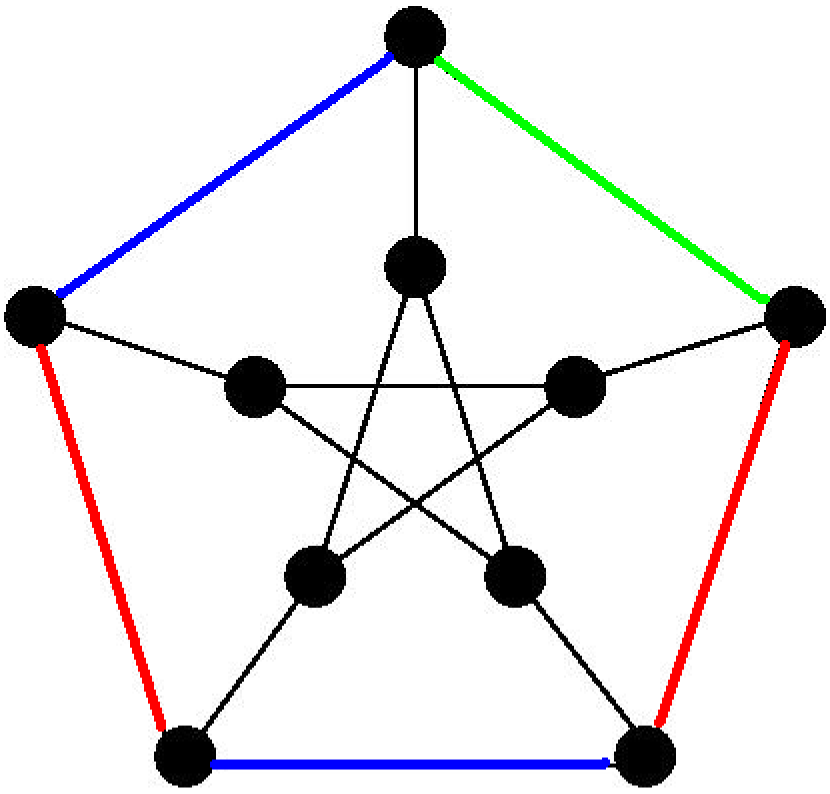
If it did, the edges would be 3-edge colourable.



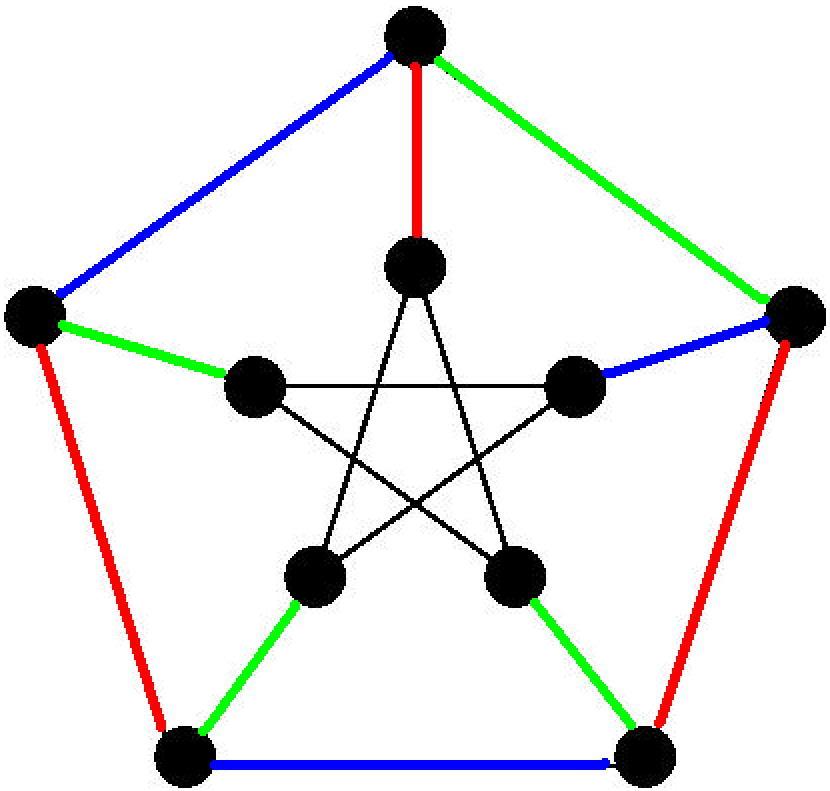


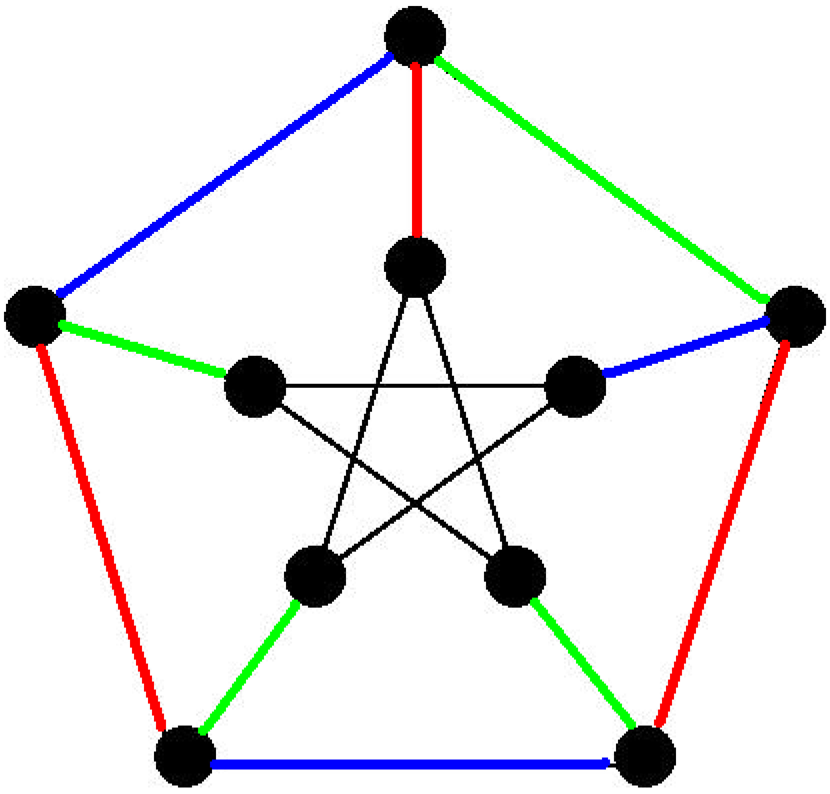
On outside cycle: one colour used once and others used twice:



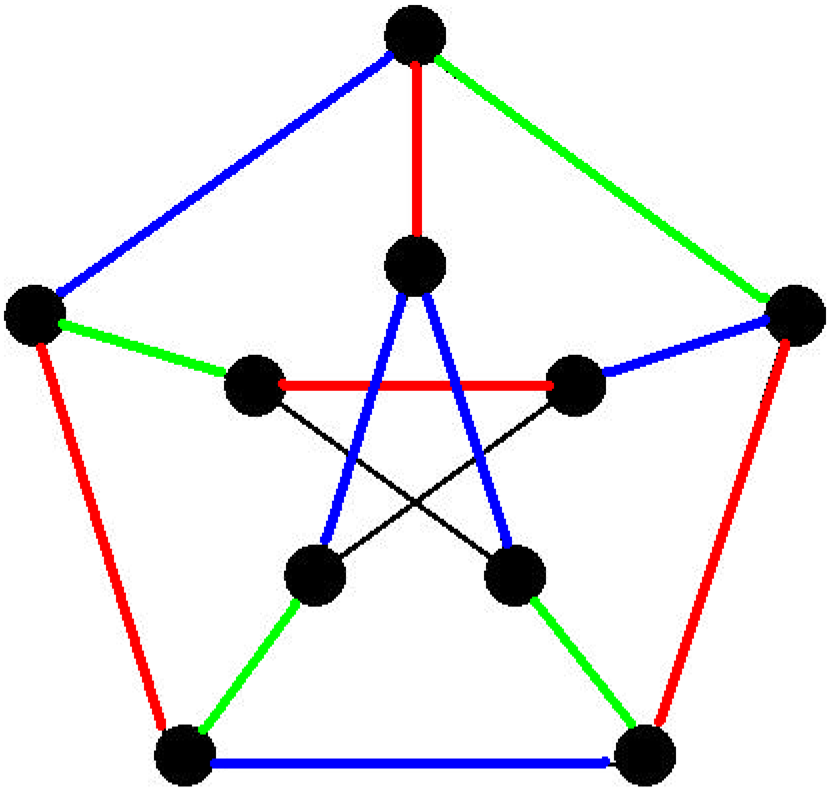


Colours of spoke edges are forced:

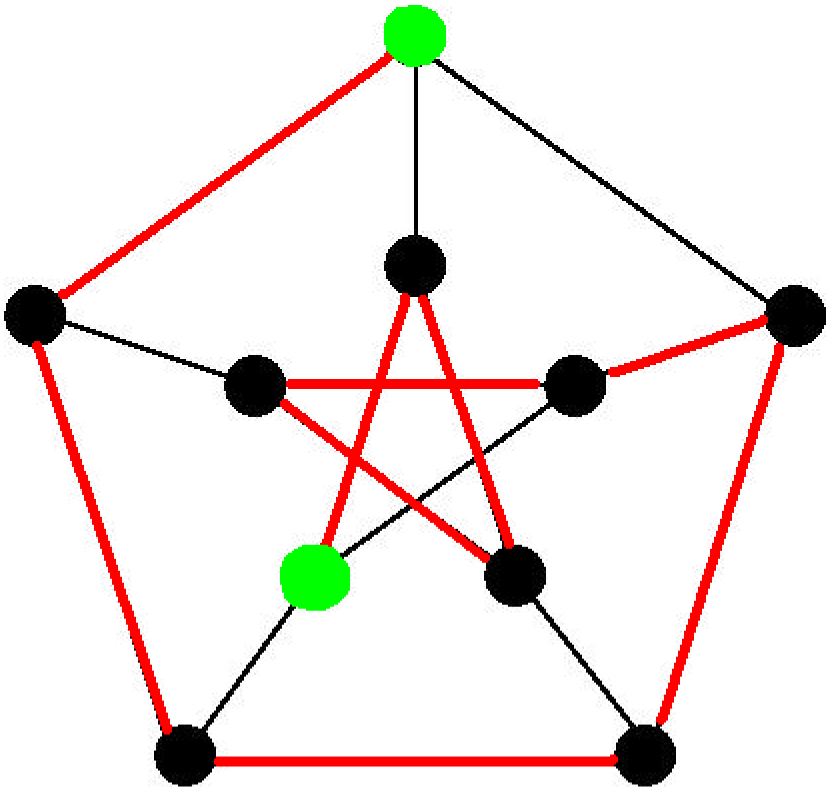




More forcings then stuck:



# G-v has a Hamilton Path for all v.



# G-e has a Hamilton Path for all e.