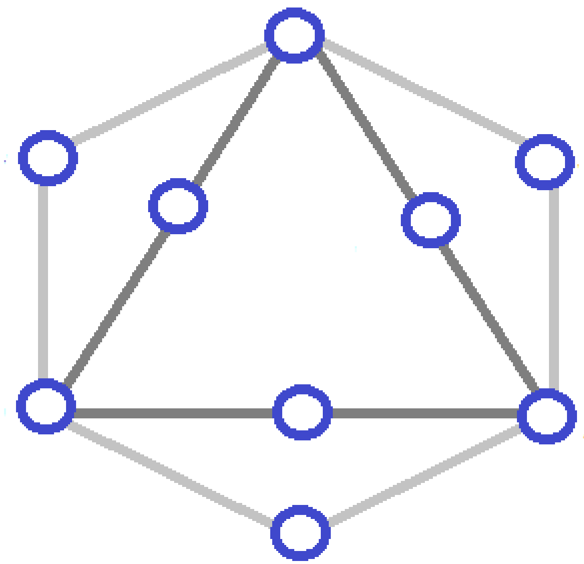
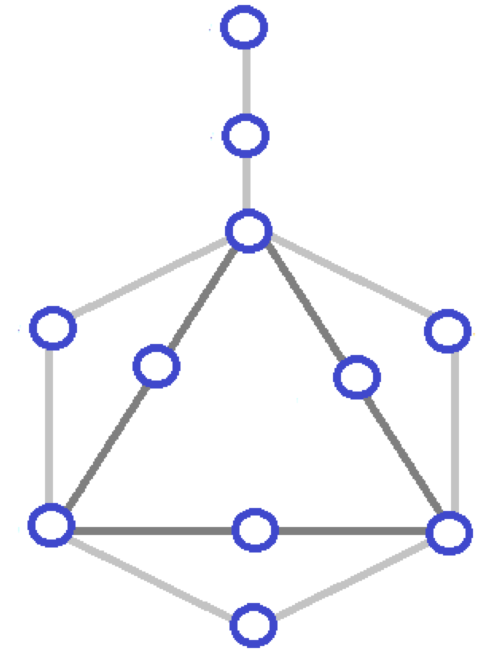
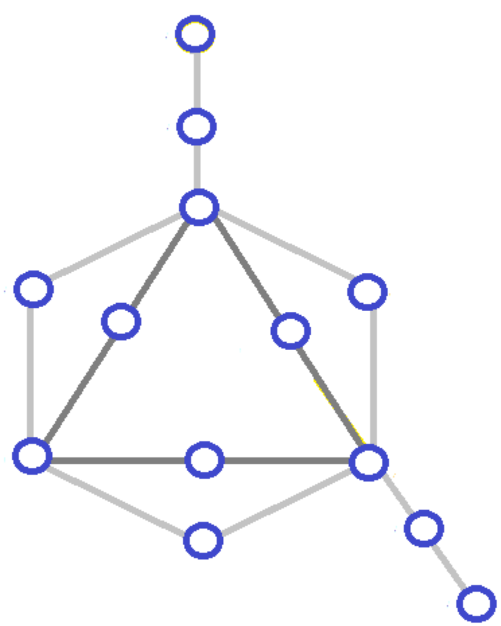
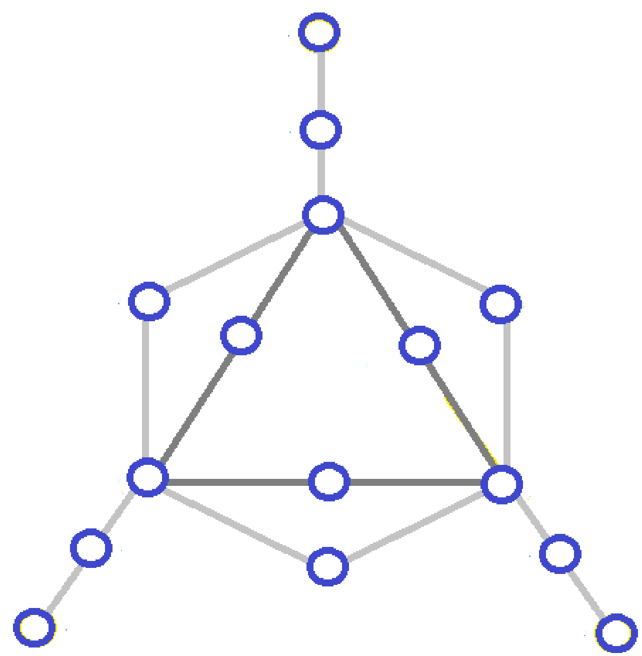


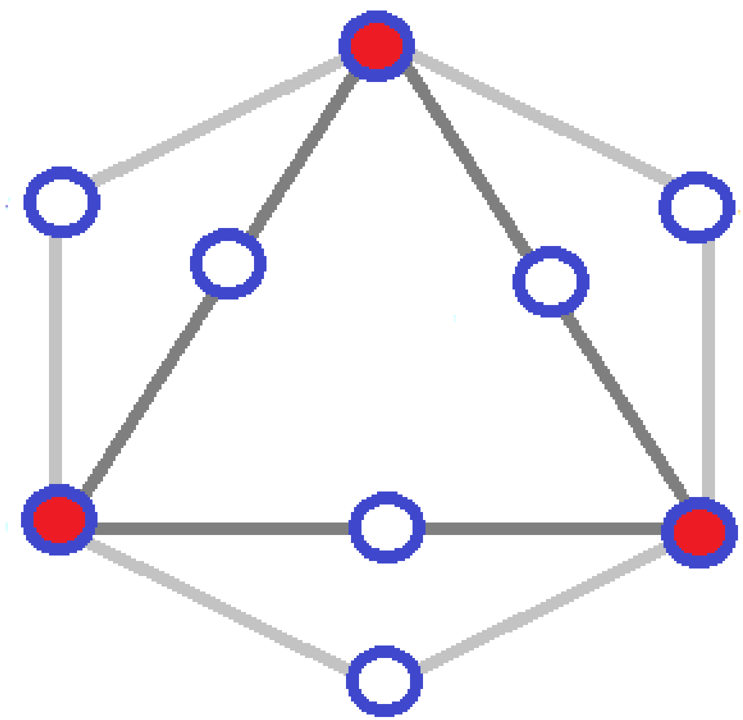
What is the minimum dominating set sizes for these graphs?





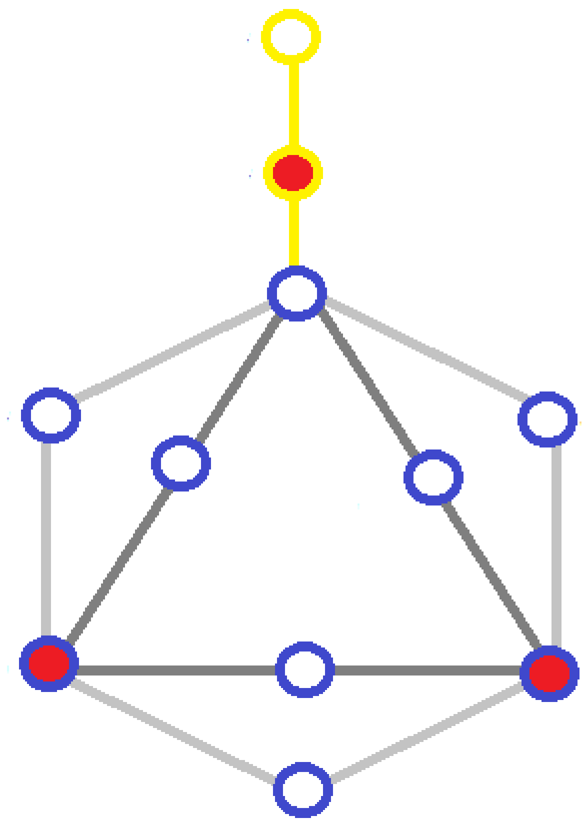


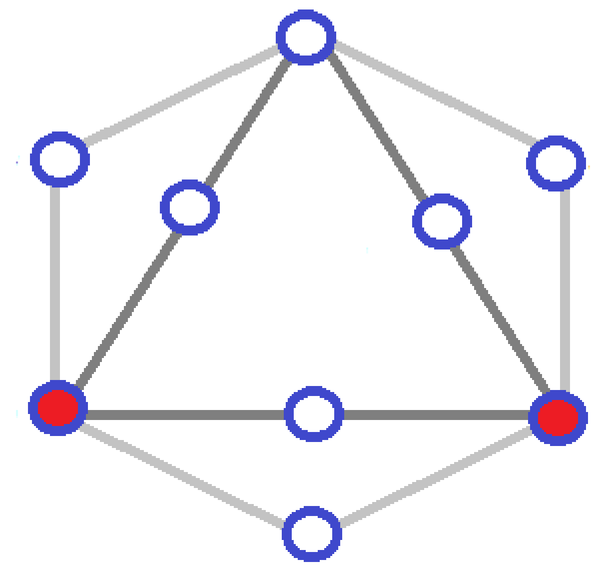
The degree bound indicates maybe two degree 4 vertices could suffice but since the neighbourhoods overlap too much, we need 3.



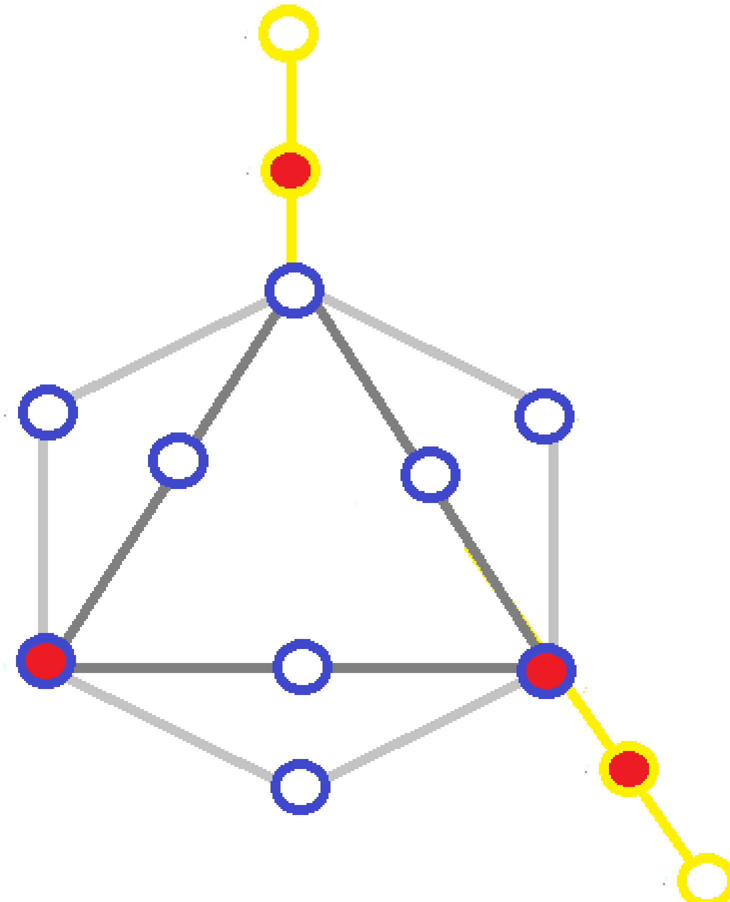
Simple degree bound: if the maximum degree of a vertex in the graphs is Δ then the minimum dominating set size is at least $n/(\Delta +1)$.

But we only need 2 vertices from this subgraph:



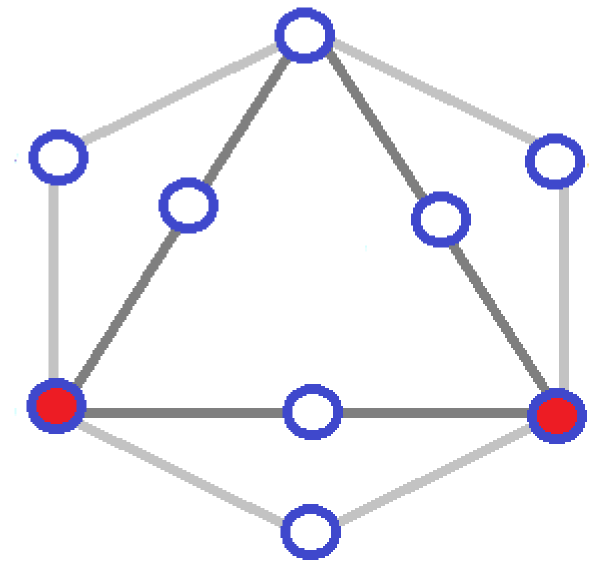


We need 3.

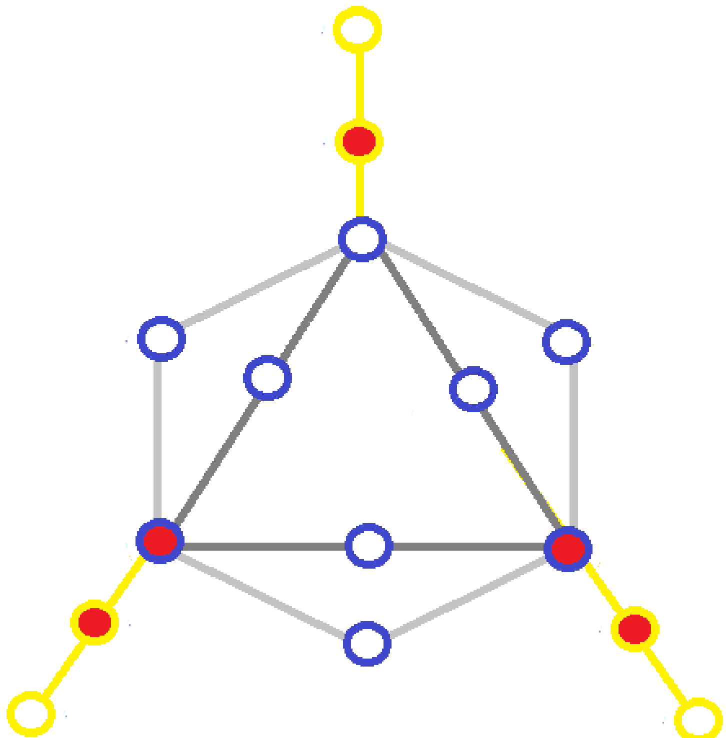


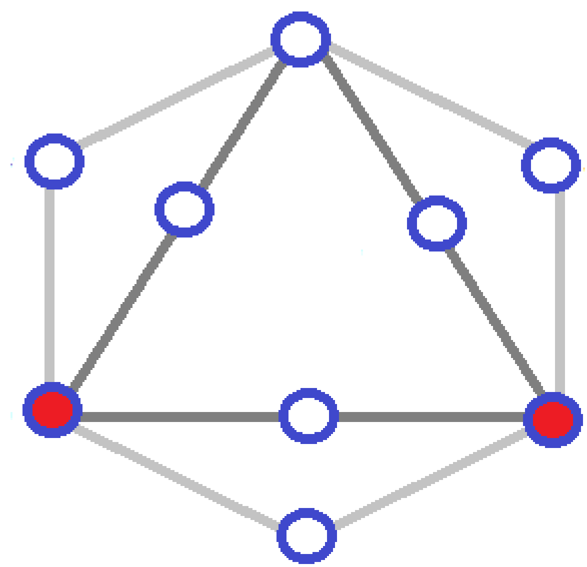
We need 4.

But we only need 2 vertices from this subgraph:



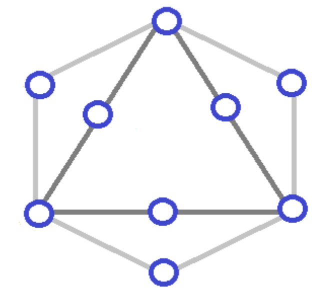
But we only need 2 vertices from this subgraph:

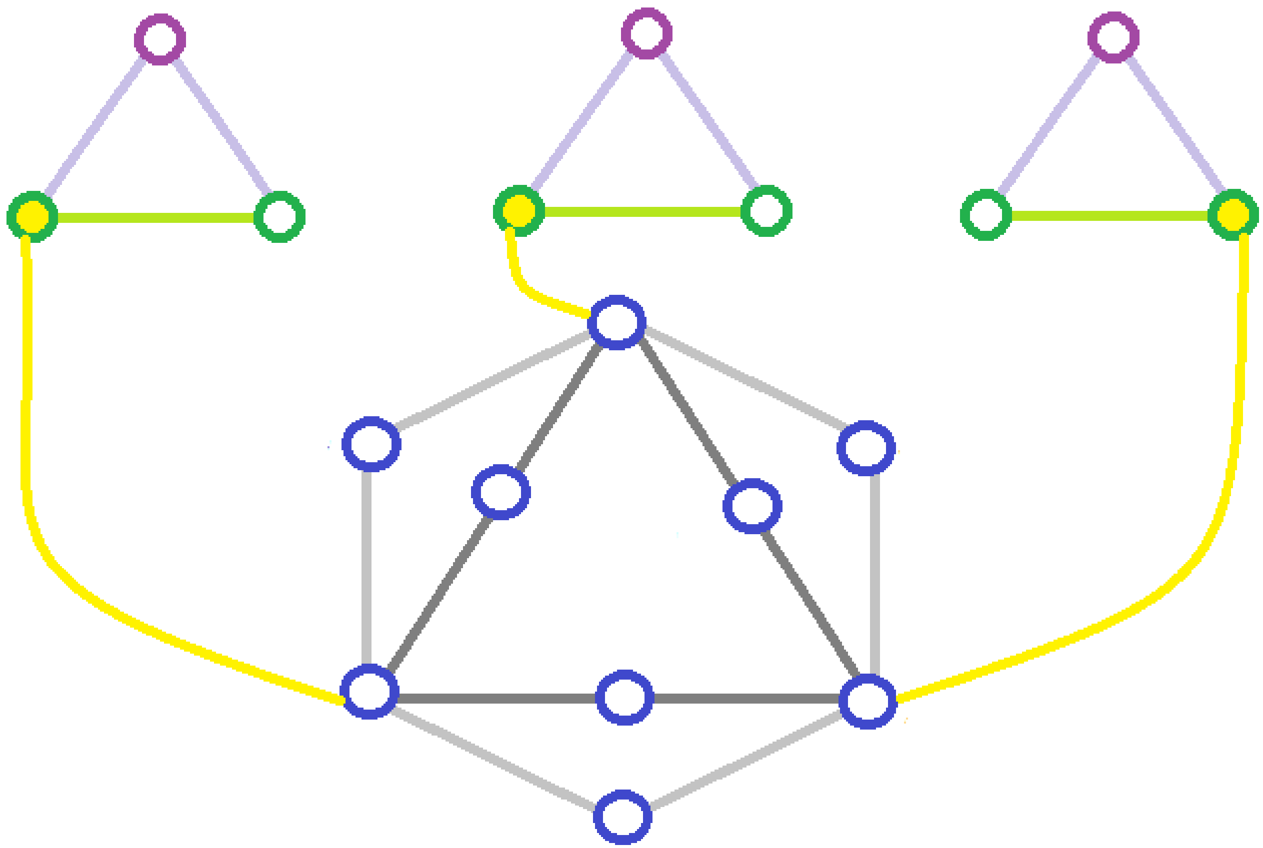


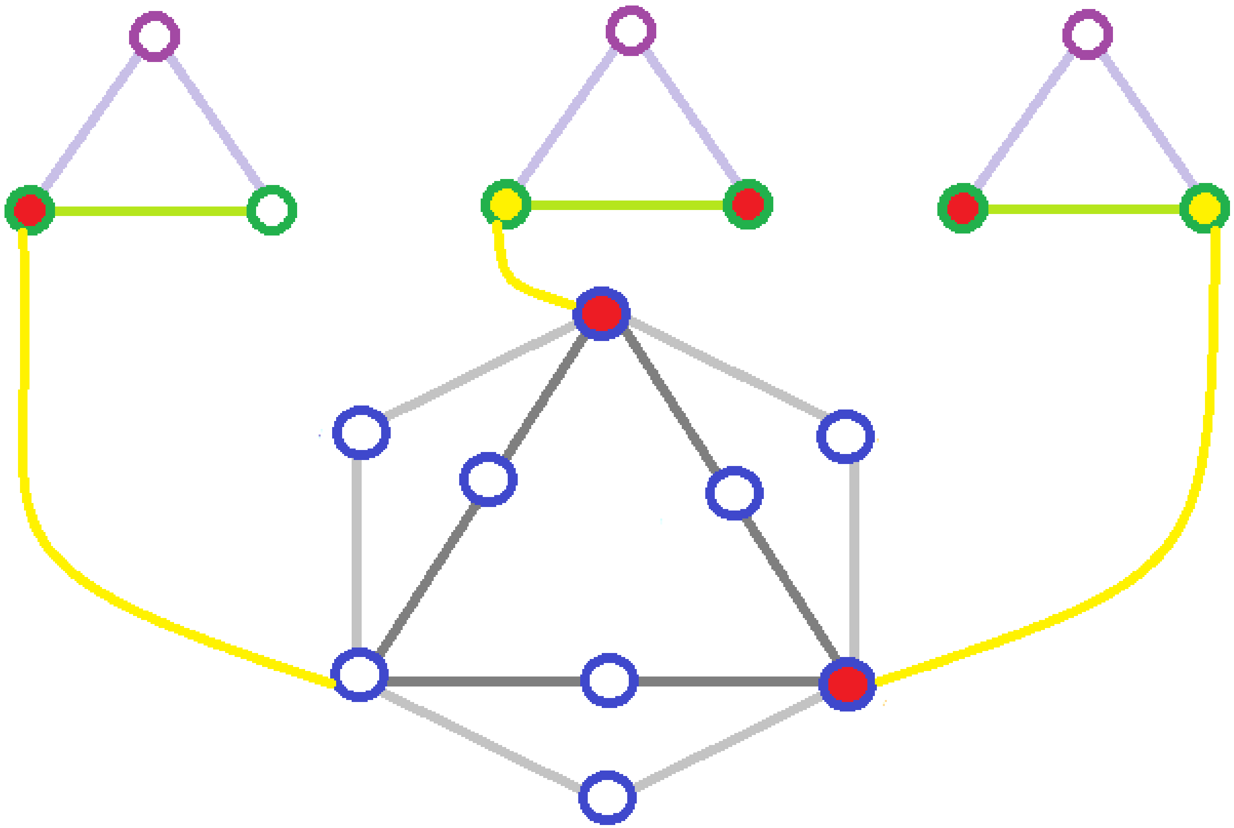


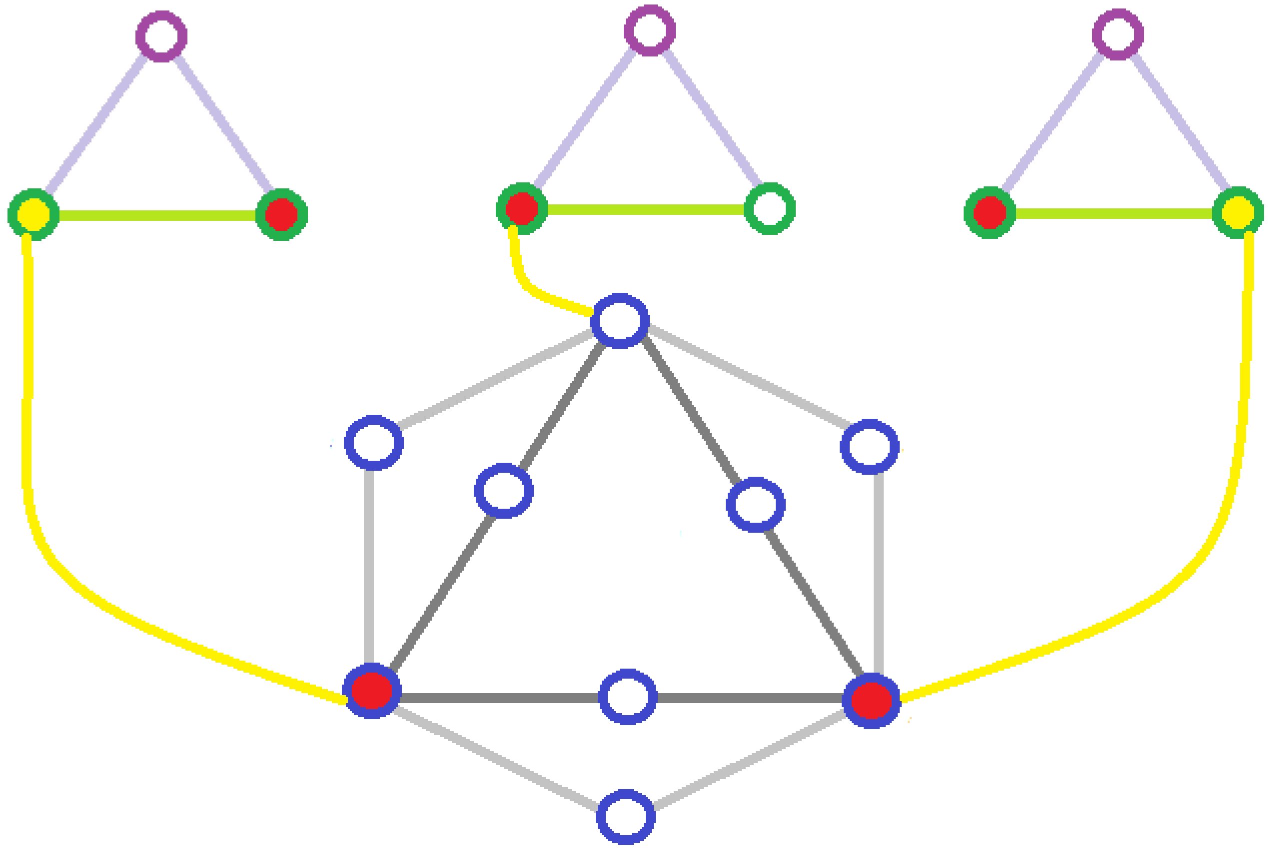
We need 5.

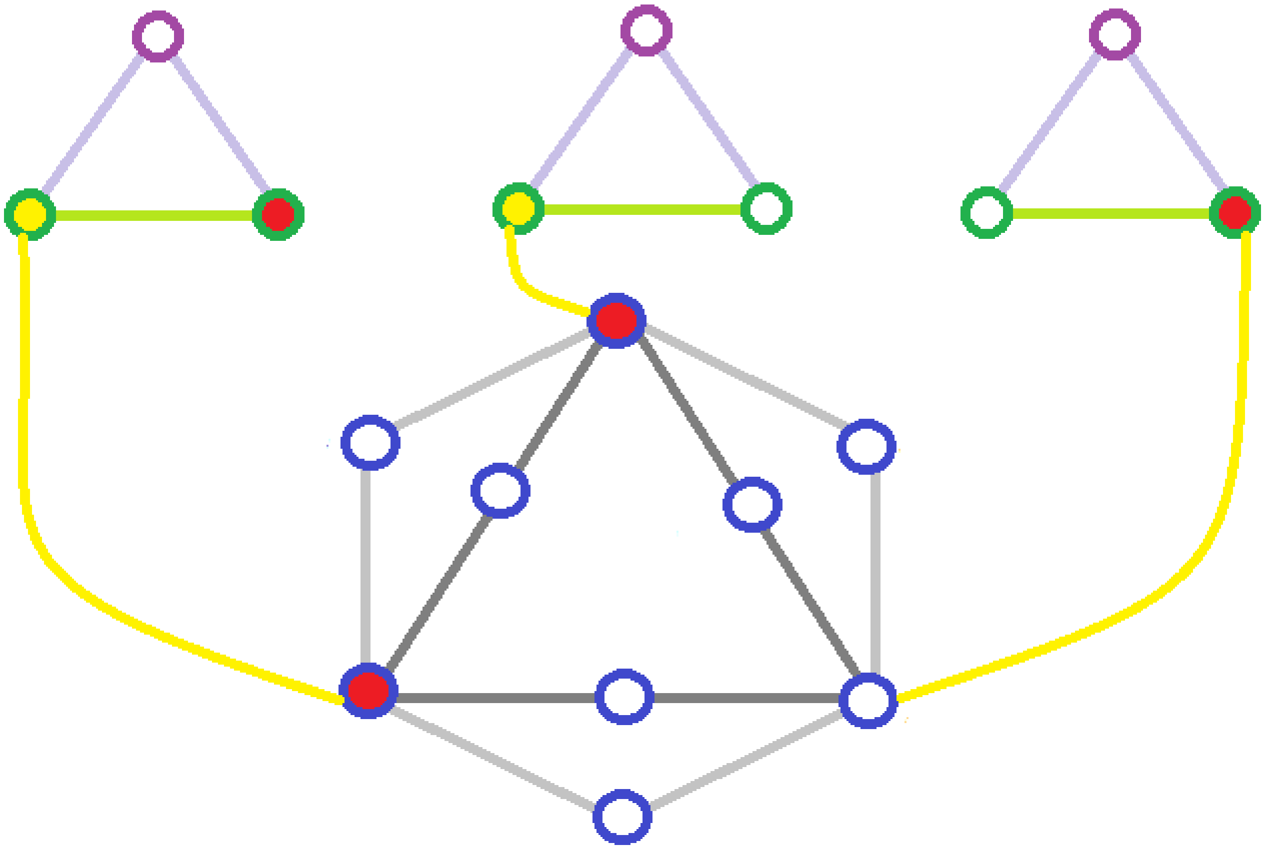
If at least one yellow vertex is red, then we can find a dominating set using on 2 vertices from the subgraph:



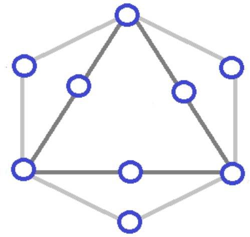


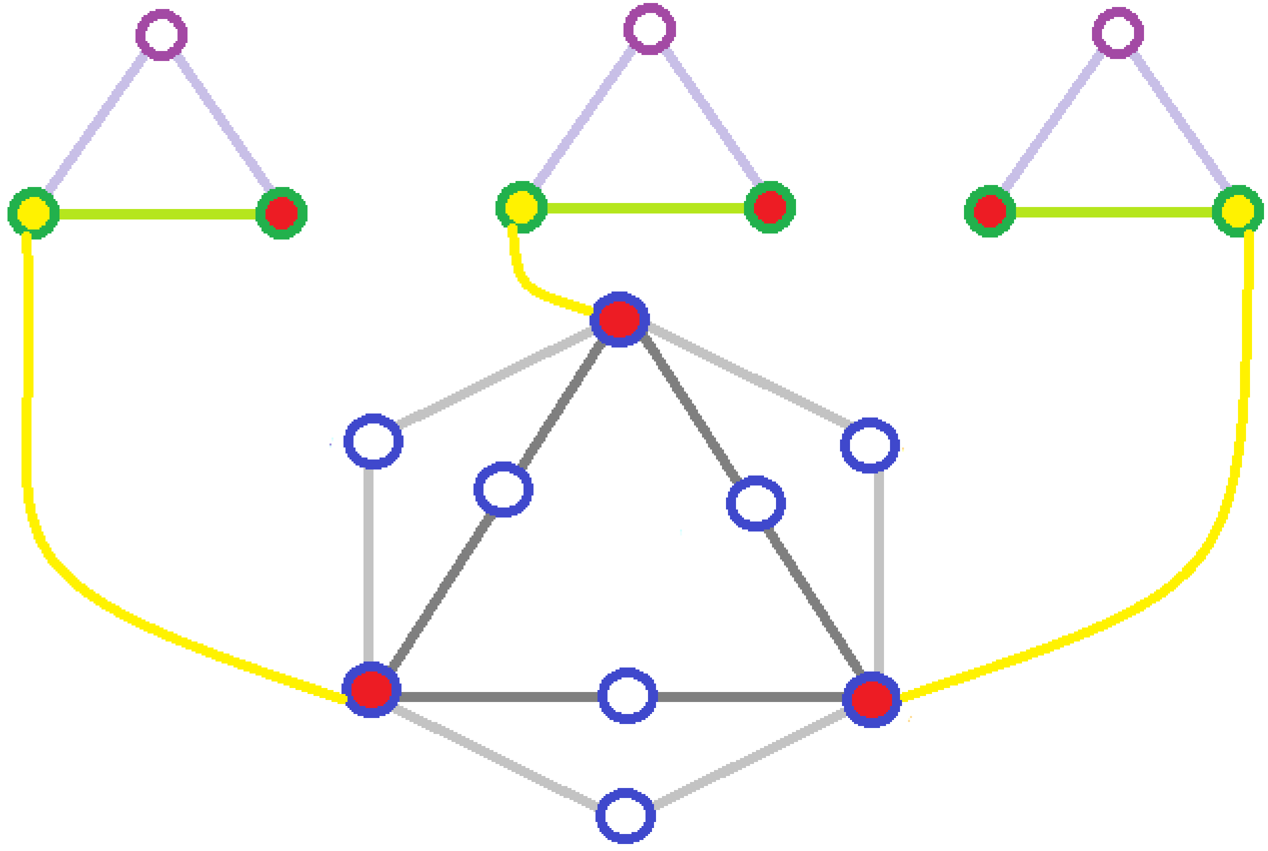






If at no yellow vertices are used,
we need 6 vertices instead of 5,
we need 3 from the subgraph:





Class NP

A decision problem (yes/no question) is in the *class NP* if it has a nondeterministic polynomial time algorithm. Informally, such an algorithm:

1. Guesses a solution (nondeterministically).

2. Checks deterministically in polynomial time that the answer is correct.

Or equivalently, when the answer is "yes", there is a certificate (a solution meeting the criteria) that can be verified in polynomial time (deterministically).

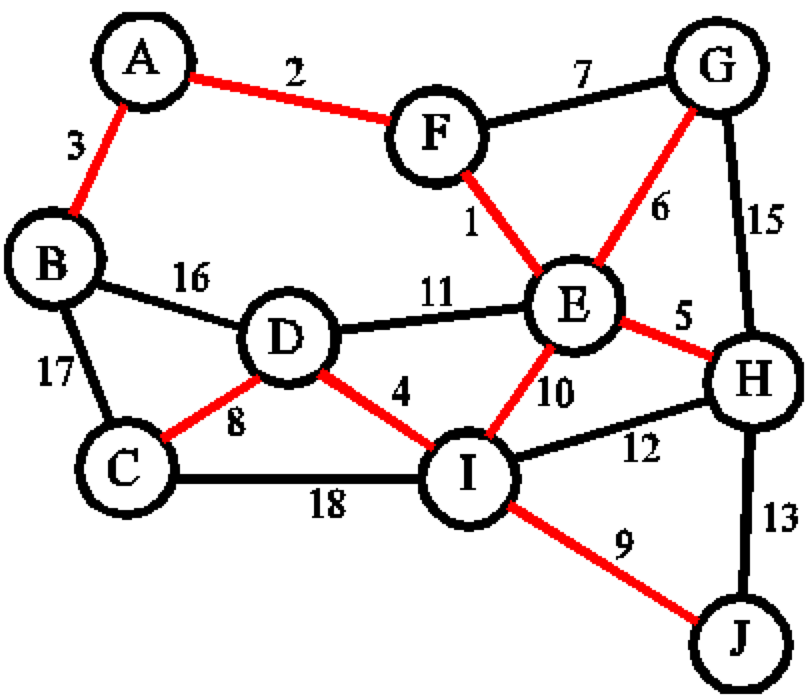
Example problem which is in P and NP:

Minimum Weight Spanning Tree (CSC 225).

Input: Graph G, integer k.

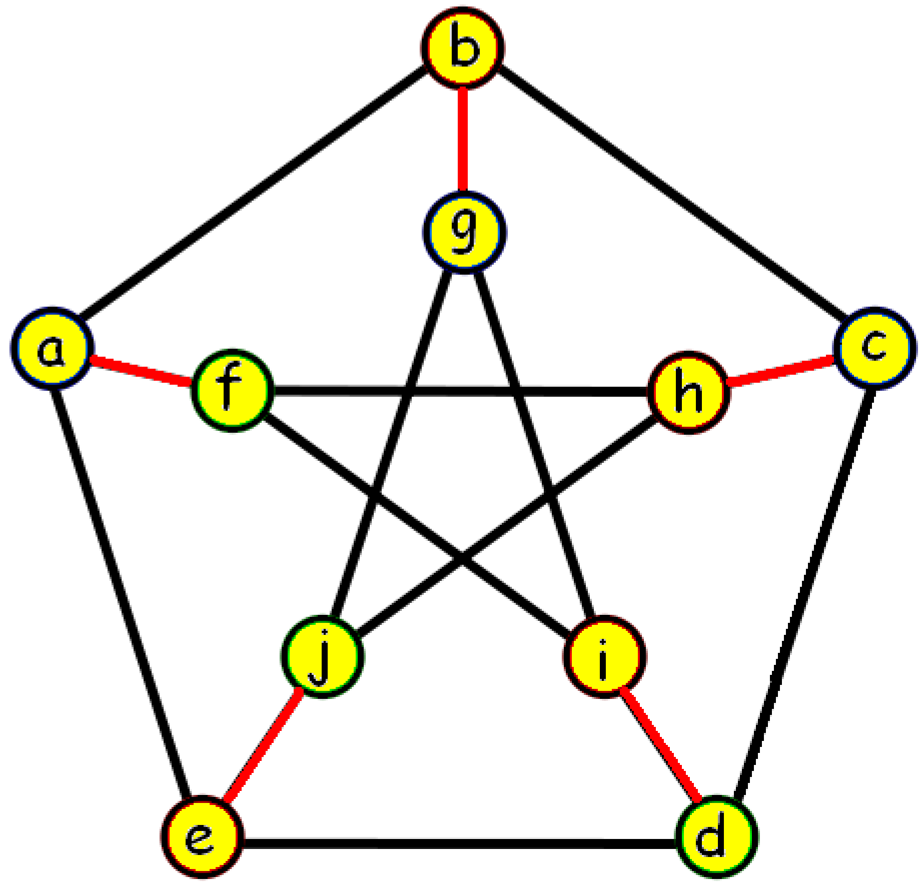
Question: Does G have a spanning tree of weight at most *k*?

If you are provided with a tree with weight at most k as part of the solution, the answer can be verified in O(n) time.



Matching is in NP: Given a graph G and integer k, does G have a k-edge matching?

Matching: disjoint edges.



Certificate, k=5:

(a,f) (b,g) (c,h)

(d,i)(e,j)

Hamilton Cycle is in NP: Input graph G.

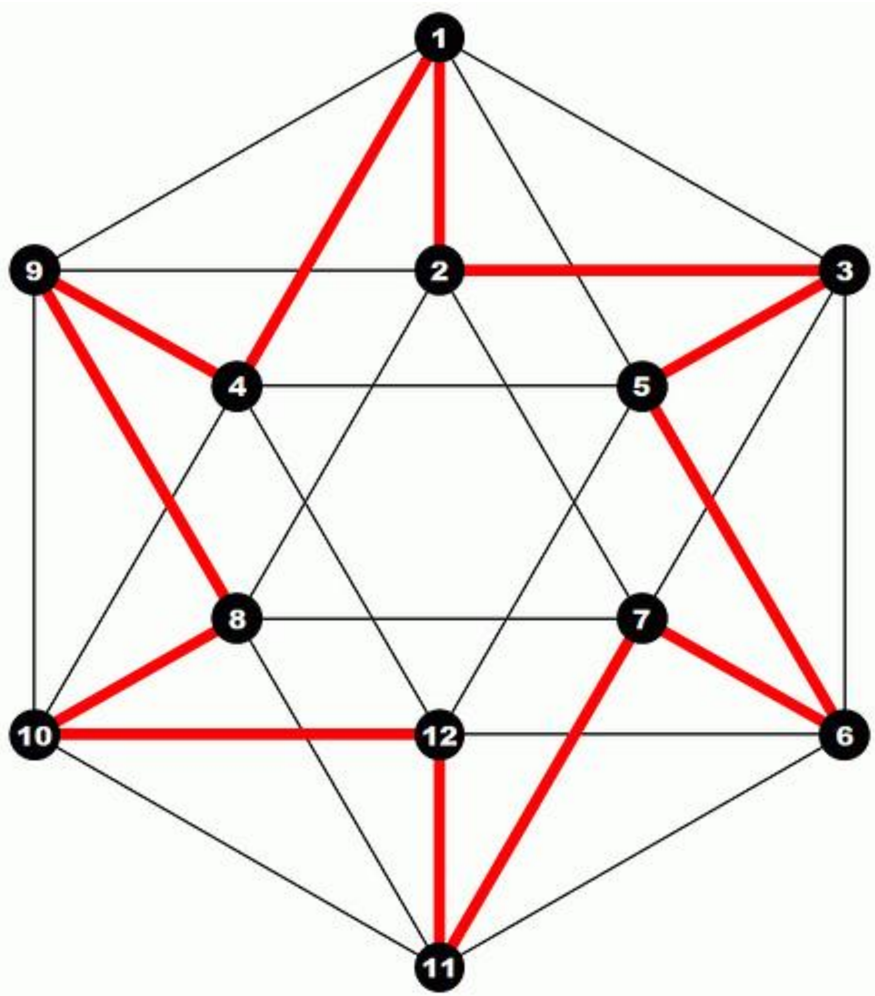
Does G have a Ham. cycle?

Certificate:

1, 2, 3, 5,

6, 7, 11, 12,

10, 8, 9, 4



Picture from: http://mathoverflow.net/faq

Does P= NP? the Clay Mathematics Institute has offered a $1 million US prize for the first correct proof.

Some problems in NP not known to be in P:

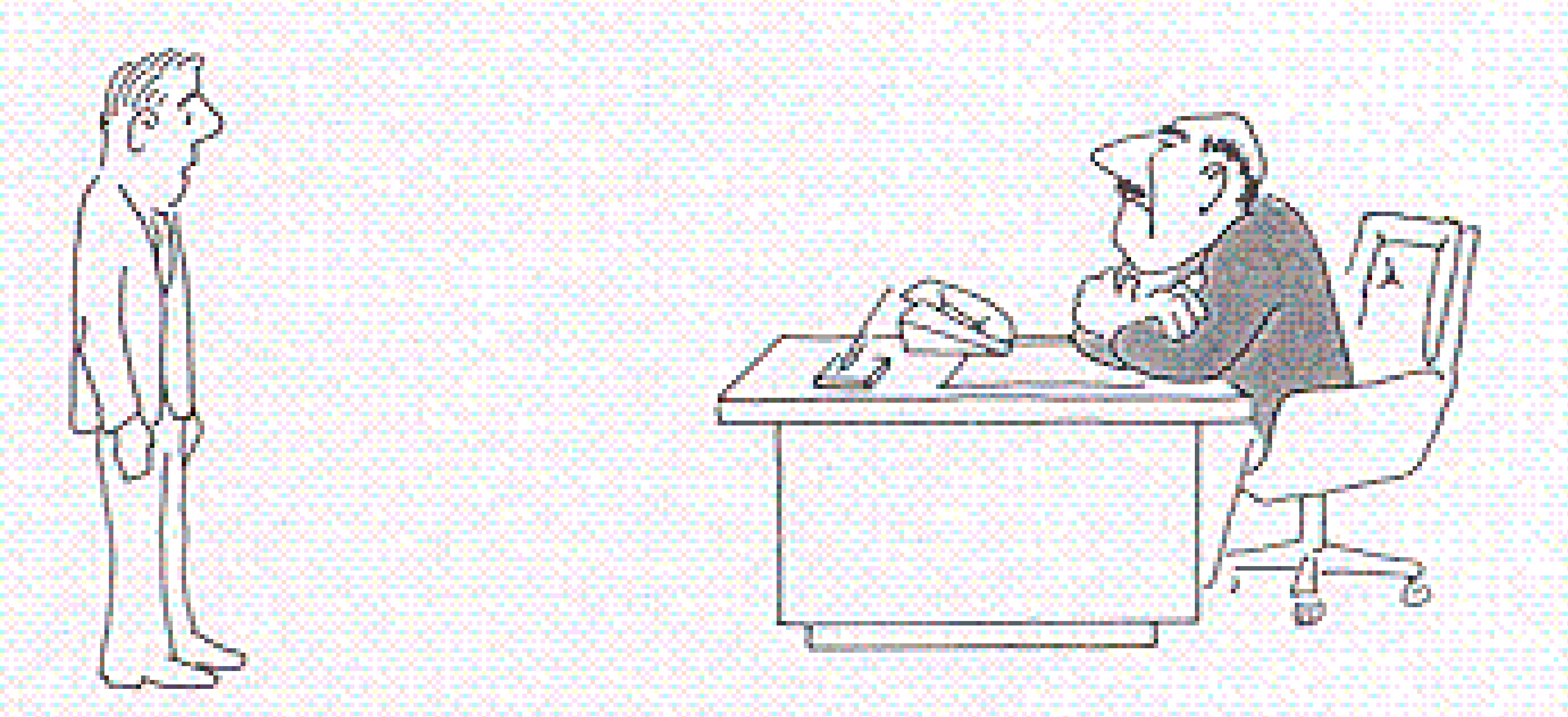
Hamilton Path/Cycle

Independent Set

Satisfiability

Note: Matching is in P. Learn more in a graph algorithms class.

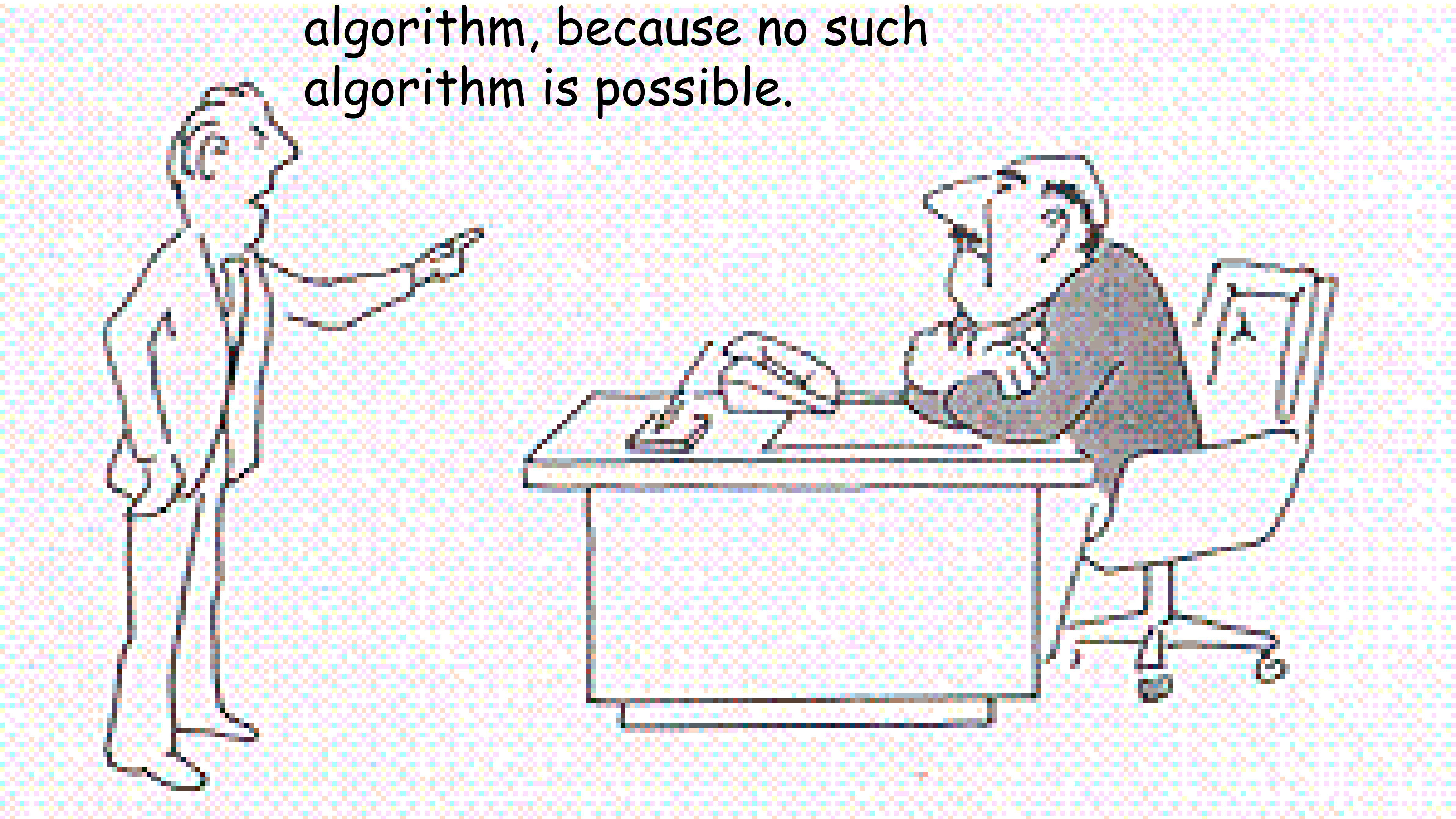
NP-completeness

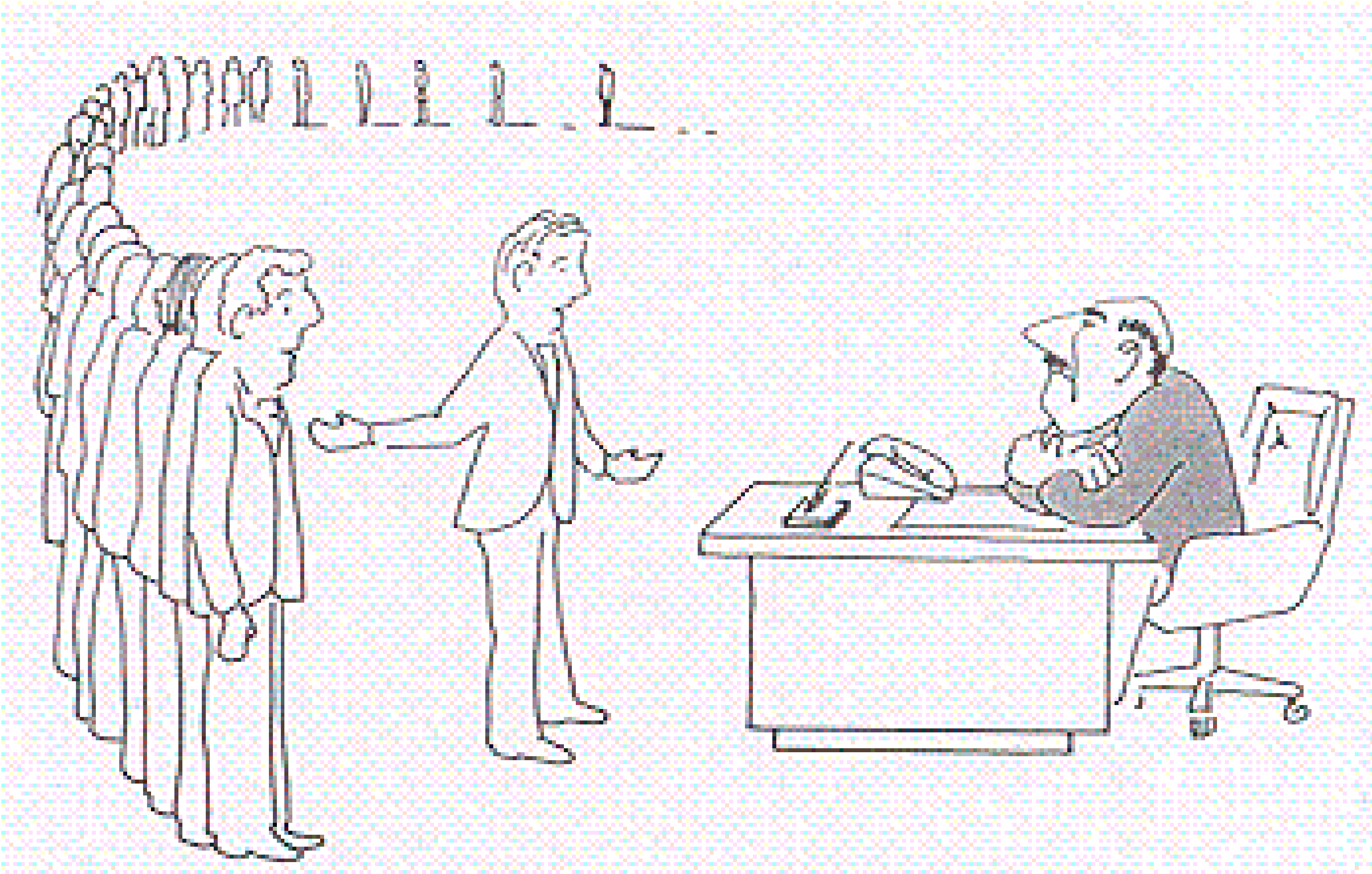


I can't find an efficient algorithm,

I guess I'm just too dumb.

I can't find an efficient algorithm, because no such algorithm is possible.





I can't find an efficient algorithm, but neither can all these famous people.

NP-complete Problems

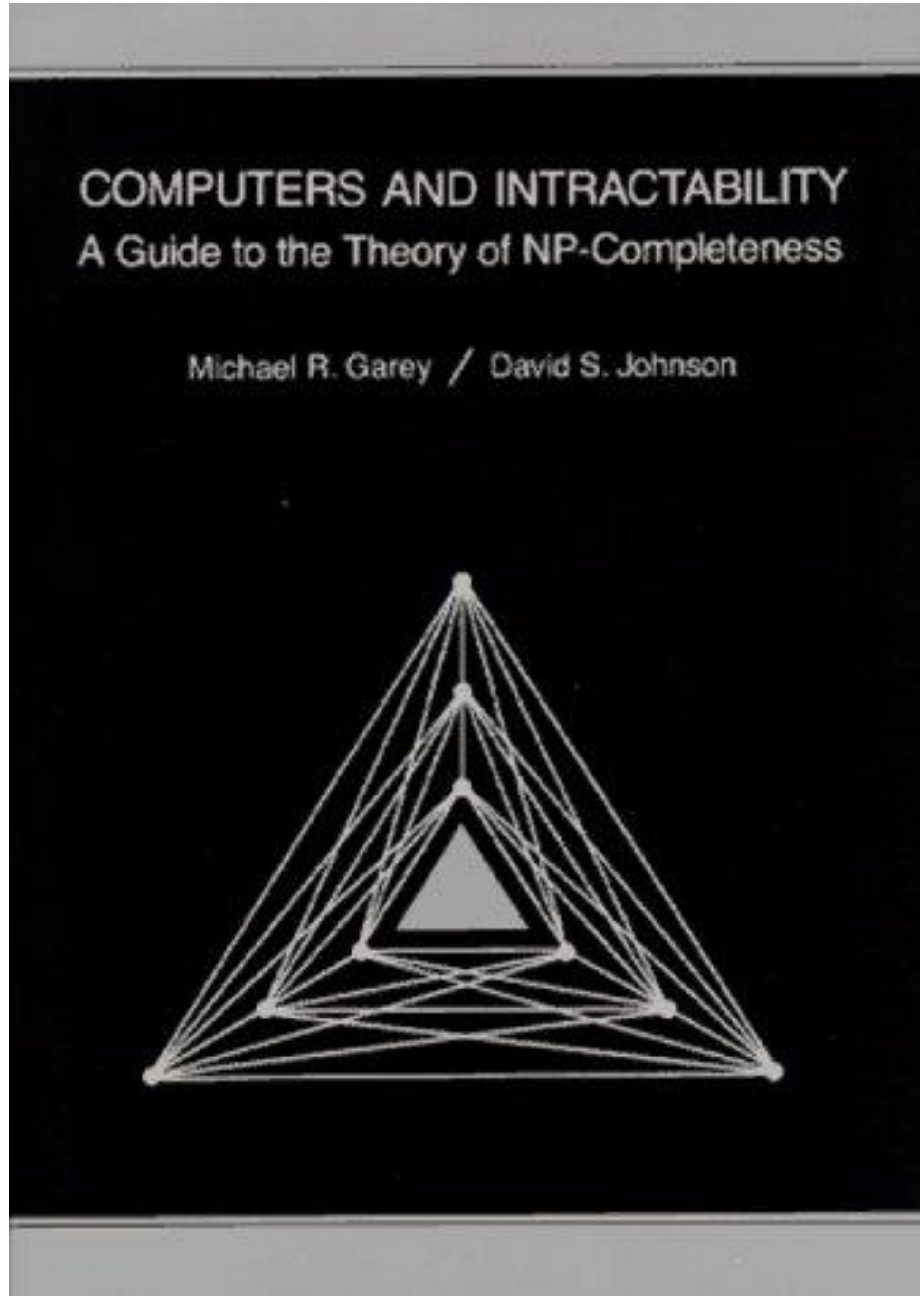
The class of problems in NP which are the "hardest" are called the *NP-complete* problems.

A problem Q in NP is NP-complete if the existence of a polynomial time algorithm for Q implies the existence of a polynomial time algorithm for all problems in NP.

Steve Cook in 1971 proved that SAT is NP-complete. Proof: will be given in our last class. Other problems: use reductions.

Bible for NP-completeness:

M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completness*, W. H. Freeman, 1st ed. (1979).



SAT (Satisfiability)

Variables: $u_1, u_2, u_3, \ldots u_k$.

A **literal** is a variable u_i or the negation of a variable $\neg u_i$.

If u is set to *true* then ¬ u is *false* and if u is set to *false* then ¬ u is *true*.

A **clause** is a set of literals. A clause is *true* if at least one of the literals in the clause is *true*.

The input to SAT is a collection of clauses.

This SAT problem has solution
u_1=T, u_2=F, u_3= T, u_4=F

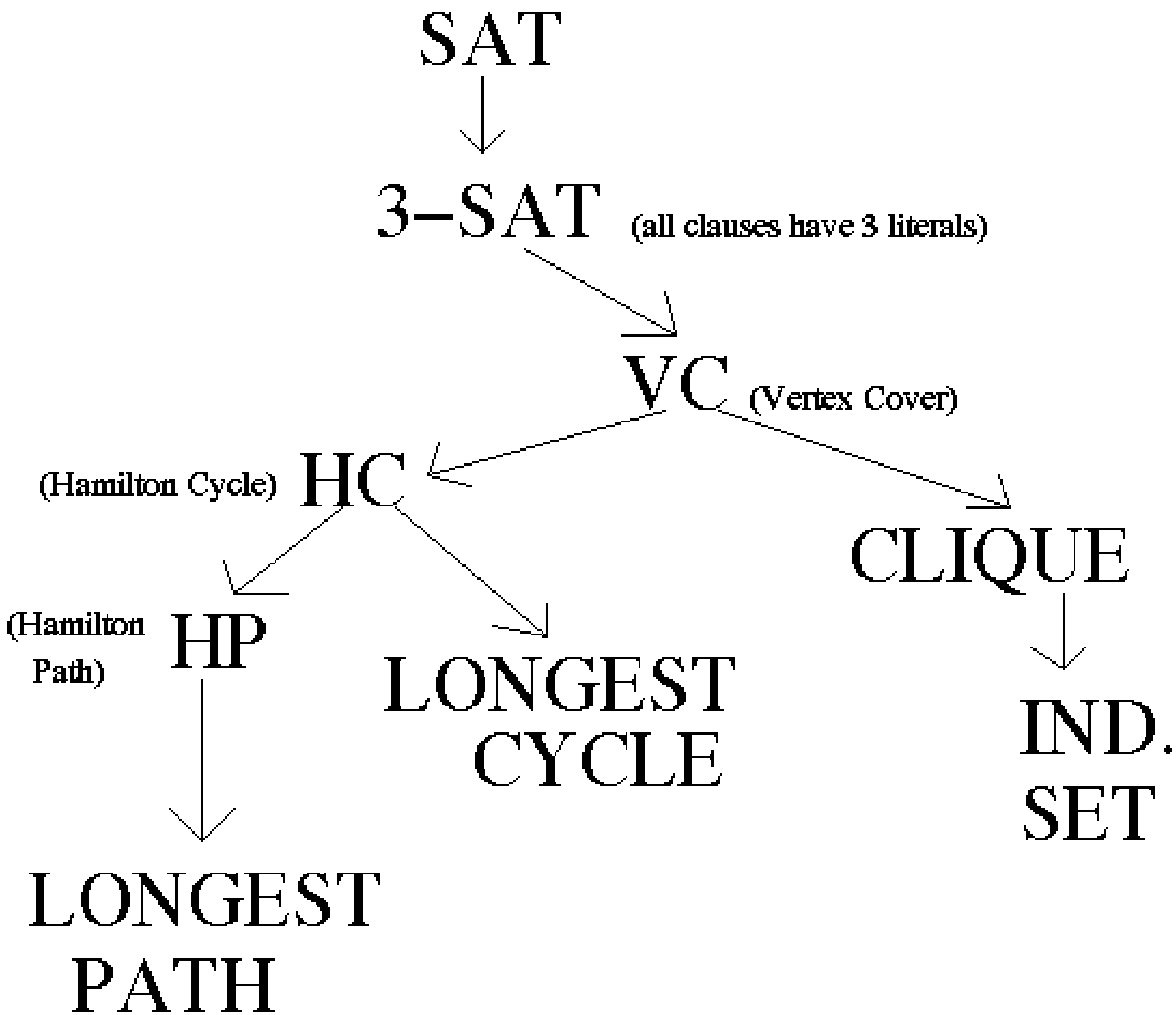
(u_1 OR u_2 OR u_4) AND ($\neg$ u_2 OR u_4) AND
($\neg$ u_1 OR u_3) AND ($\neg$ u_4 OR $\neg$ u_1)

Does this SAT problem have a solution?

(u_1 OR u_2) AND ($\neg$ u_2 OR u_3) AND

($\neg$ u_3 OR $\neg$ u_1) AND ($\neg$ u_2 OR $\neg$ u_3) AND

(u_3 OR $\neg$ u_1)



History of NP–completeness Reductions

3-SAT- each clause must contain exactly 3 variables (assignment- at most 3).

Given: SAT is NP-complete.

Theorem: **3-SAT is NP-Complete.**

Theorem: If we can solve the dominating set problem in polynomial time, then we can solve 3-SAT in polynomial time.

Or equivalently: Given that 3-SAT is an NP-complete problem, we prove that Dominating Set is NP-complete.

3-SAT Problem:

(x1 or x1 or x2) AND (¬x1 or ¬ x2 or ¬ x2) AND (¬x1 or x2 or x3)