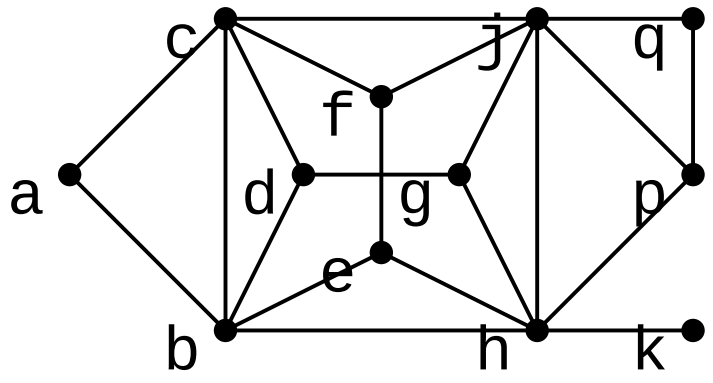
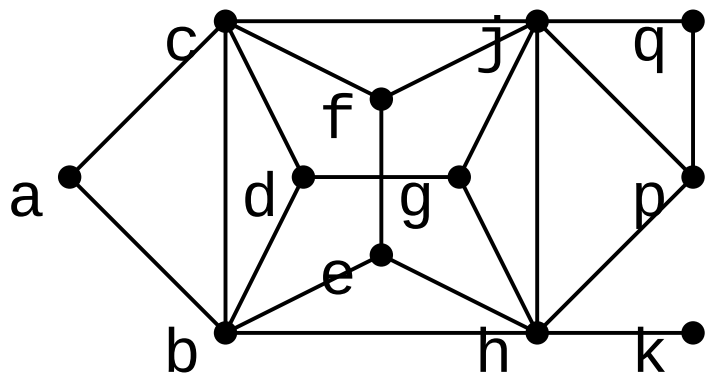


k-independent set (k-IS)

* instance graph G
  query independent set size k?

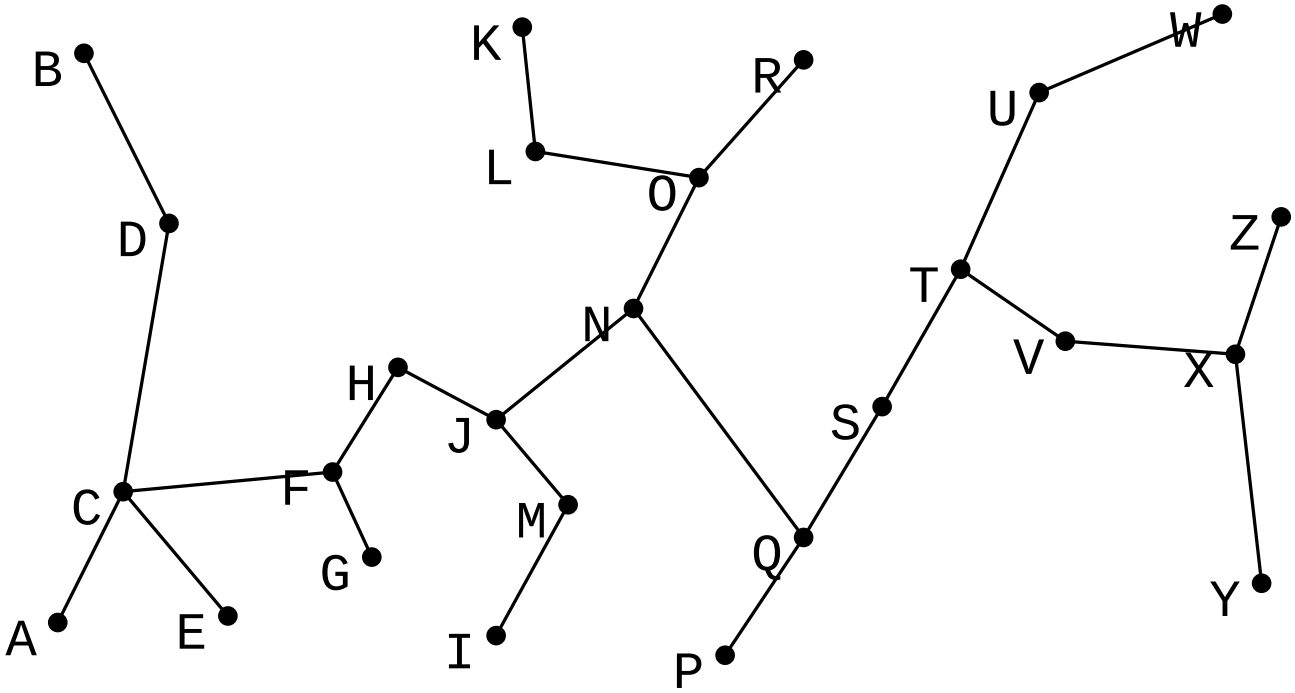
* also called max IS problem (why?)

* NP-complete (from k-clique)



how to find max independent set?

```
  * NP-complete  :(
  * restrict instances ?
  * trees?    k-IS in P  :)
    - method 1:     dynamic programming
    - forest has isolated node or leaf (1 nbr)

    - method 2: transform IS-to-LP
        - not necessarily answer-preserving:
          can have non-integer solution :(
        - trees ?
          * integer solution guaranteed :)
          * size of LP polynomial in size of graph,
        - perfect graphs ?
          * integer solution guaranteed :)
          * size of LP can be non-poly in size of graph
```



find max IS?

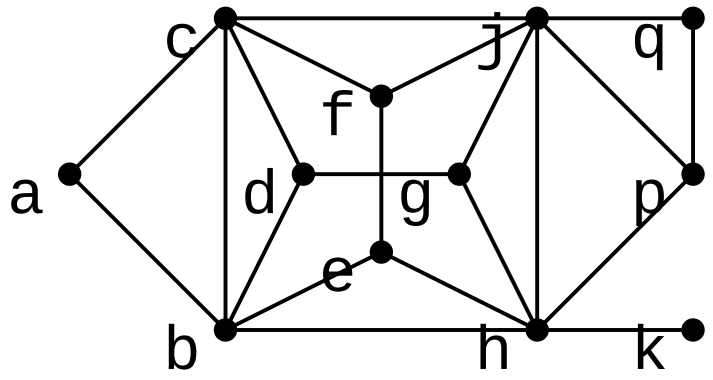
* dynamic programming ?

* leaf: neighborhood has size 1

* lemma: for each leaf x,

  {x} union maxIS(T-x - nbr(x))

  is maxIS(T)



find max IS?

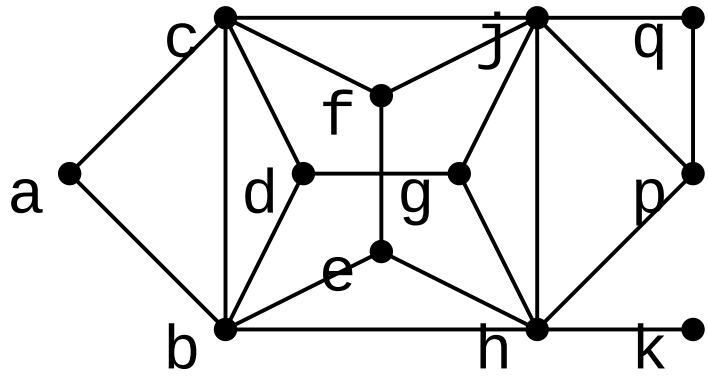
* dynamic programming ?

* simplicial node: neighborhood is clique

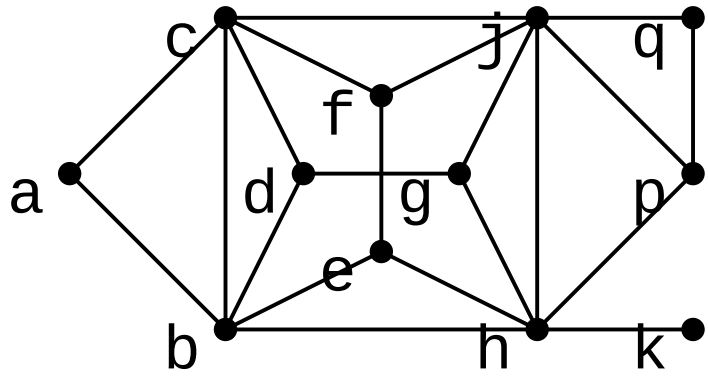
* lemma: for each simplicial node y,

  {y} union maxIS(G -y - nbr(y))

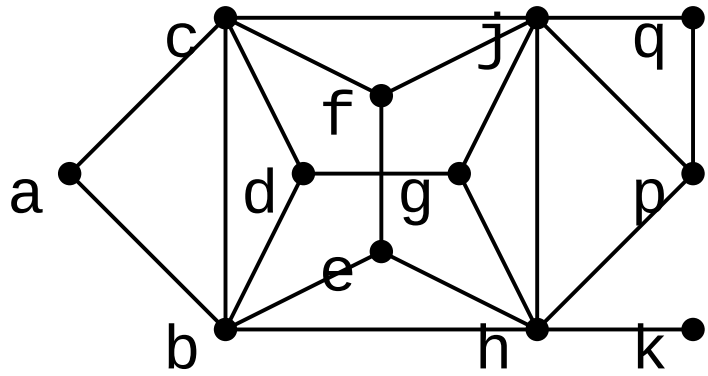
  is maxIS(G)




```
nodes x_1 .. x_12      max'l cliques  y_1 .. y_10
  abc bcd beh cfj  dg  ef ghj  hk hjp jpq
max [1 1 .. 1] X s.t. AX <= [1 1 .. 1]^t
         a b c d e f g h j k p q
    A   |1 1 1                  |
        |  1 1 1                |
        |  1     1     1        |
        |    1     1     1      |
        |      1     1          |
        |        1 1            |
        |            1 1 1      |
        |              1   1    |
        |              1 1   1  |
        |                1   1 1|
```

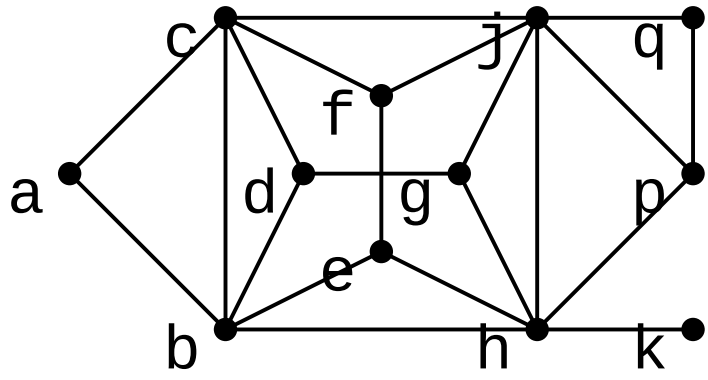



```
P = MixedIntegerLinearProgram()
v = P.new_variable(real=True, nonnegative=False)
a,b,c,d,e,f,g,h,j,k,p,q,z=v['a'],v['b'],v['c'],v['d'],v
P.set_objective(z)
P.add_constraint(z <= a+b+c+d+e+f+g+h+j+k+p+q)
P.add_constraint(a >= 0)
P.add_constraint(b >= 0)
...
P.add_constraint(q >= 0)
P.add_constraint(a+b+c <= 1)
P.add_constraint(b+c+d <= 1)
P.add_constraint(b+e+h <= 1)
P.add_constraint(c+f+j <= 1)
P.add_constraint(d+g   <= 1)
P.add_constraint(e+f   <= 1)
P.add_constraint(g+h+j <= 1)
P.add_constraint(h+k   <= 1)
P.add_constraint(h+j+p <= 1)
P.add_constraint(j+p+q <= 1)
P.solve()
P.get_values(z,a,b,c,d,e,f,g,h,j,k,p,q)
```

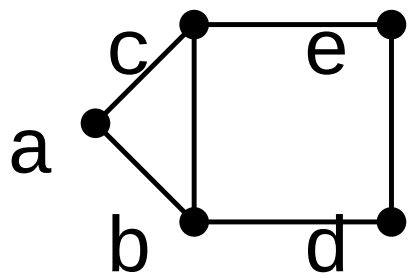


evaluate program at `https://sagecell.sagemath.org/`

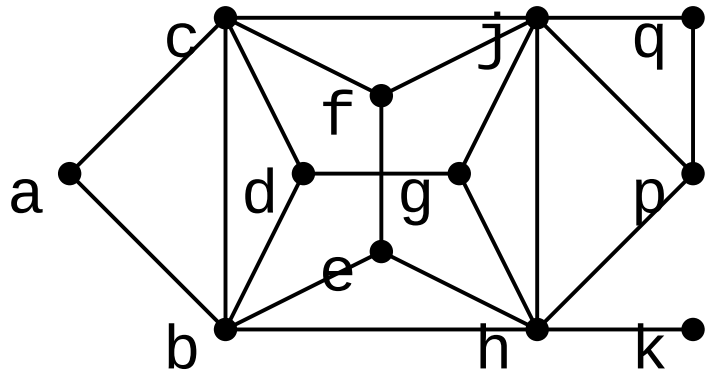
```
solution

[5.0, 1.0,-0.0,-0.0, 1.0, 1.0,-0.0,

     -0.0,-0.0, 1.0, 1.0,-0.0,-0.0]

  z    a    b    c    d    e    f

       g    h    j    k    p    q

max IS = {a, d, e, j, k}

can we easily verify this solution?  dual !
```



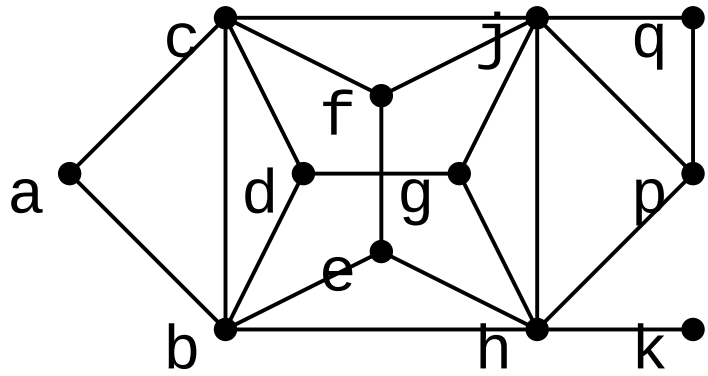

```
primal:   max C^t X s.t.    AX <= B

dual:     min B^t Y s.t. Y^t A >= C^t

dual (max form):

          max -B^t Y s.t. Y^t A >= C^t
```




```
primal       max   [1 1 1 1 1] X
             st    |1 1 1 . .| X  <= |1|
                   |. 1 . 1 .|       |1|
                   |. . 1 . 1|       |1|
                   |. . . 1 1|       |1|

all-integer solution?
* {x_j} (weighted node set) hits each clique <=1 time
* union{x_j} forms an independent set

dual         min     [1 1 1 1] Y
            st Y^t |1 1 1 . .|    >= [1 1 1 1 1]
                   |. 1 . 1 .|
                   |. . 1 . 1|
                   |. . . 1 1|
all-integer solution?
* {y_k} (weighted clq set) hits each node >=1 time
* union{y_k} forms a clique cover

all-integer solution? duality theorem says
  exist same size independent set, clique cover :)
```

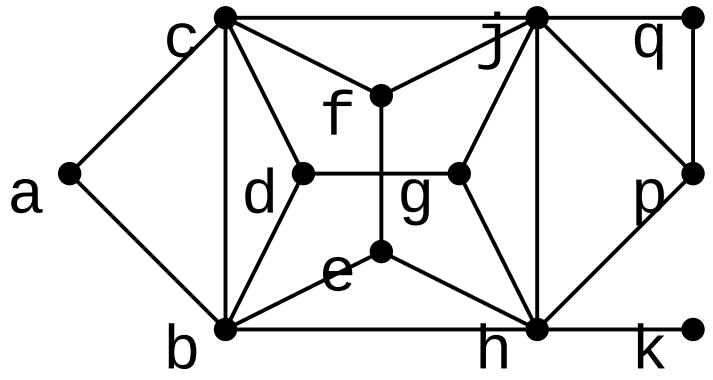



```
D = MixedIntegerLinearProgram()
v = D.new_variable(real=True, nonnegative=False)
a,b,c,d,e,f,g,h,j,k,z=v['a'],v['b'],v['c'],v['d'],v['e'
D.set_objective(z)
D.add_constraint(z <=-a-b-c-d-e-f-g-h-j-k)
D.add_constraint(a >= 0)
    ..
D.add_constraint(k >= 0)
D.add_constraint(a     >= 1)
D.add_constraint(a+b+c >= 1)
D.add_constraint(a+b+d >= 1)
D.add_constraint(b+e   >= 1)
D.add_constraint(c+f   >= 1)
D.add_constraint(d+f   >= 1)
D.add_constraint(e+g   >= 1)
D.add_constraint(c+g+h+j >= 1)
D.add_constraint(d+g+j+k >= 1)
D.add_constraint(h     >= 1)
D.add_constraint(j+k   >= 1)
D.add_constraint(k     >= 1)
..
```



evaluate dual program https://sagecell.sagemath.org/

```
dual solution

[-5.0,1.0,-0.0,-0.0,-0.0,1.0,1.0,-0.0,1.0,-0.0,1.0]
      *                   *   *        *        *
  z   a    b   c    d    e   f    g   h    j   k

min clique cover = {abc, dg, ef, hk, jpq}

we have found

  * IS hitting every maximal clique,    size 5

  * set of cliques hitting every node,  size 5

    - so every IS set has size <= 5

   clique cover easily verifies IS-is-max :)
```



for this graph,

how did we know that IS-to-LP solution

would be all-integer?

answer:

graph is in a class called *perfect*

perfect graphs have this property:

solution to IS-to-LP always all-int :)