```
begin
# deal w/ long running cell(s)
runall=true
printstyled(stderr,"runall is $(runall)\n",color=:light_red)
end
```

```
runall is true
```

```
saved graphcol_bt.jl
```

graphcol_2

- the graph is 9-colorable, it would be interesting to know that 8 or less color is enough or not.
- for this purpose, we'll develop (actually i wrote it for a codesignal interview question...) a simple function in julia, that can be used to compute the chromatic number for *very small* graphs: `graphcol_bt`

```
begin
    md"""#### graphcol_2
    * the graph is 9-colorable, it would be interesting to know that 8 or less color is enough or not.
    * for this purpose, we'll develop (actually i wrote it for a [codesignal](https://codesignal.com) interview question...) a simple function in julia, that can be used to compute the chromatic number for *very small* graphs: `graphcol_bt`
    """
end
```

```
  Activating project at `~/Asztal/git/plnotebooks/graphcol_2`
   Resolving package versions...
  No Changes to `~/Asztal/git/plnotebooks/graphcol_2/Project.toml`
    Updating `~/Asztal/git/plnotebooks/graphcol_2/Manifest.toml`
  [8e850b90] ↑ libblastrampoline_jll v5.7.0+0 ⇒ v5.8.0+0
   Resolving package versions...
  No Changes to `~/Asztal/git/plnotebooks/graphcol_2/Project.toml`
  No Changes to `~/Asztal/git/plnotebooks/graphcol_2/Manifest.toml`
   Resolving package versions...
  No Changes to `~/Asztal/git/plnotebooks/graphcol_2/Project.toml`
  No Changes to `~/Asztal/git/plnotebooks/graphcol_2/Manifest.toml`
```

```
graphcol_bt (generic function with 2 methods)
```

```
begin

#--->graphcol_bt

# the backtracking solution
# it is a naive implementation w/o any smartness,
# just administration

function graphcol_bt(G::Vector{Vector{Int}},maxcol::Int) # max number of colors
    # the actual color are in 1..maxcol
    N=length(G)

    forbidden=fill(0,maxcol,N)
    # colors currently forbidden for a particular node
    # forbidden=already reserved by some of its neighbour
    # reserved if >0

    # actual and returned colorings
    color=fill(0,N) # for work with
    color_ret=fill(0,N)

    # modifies the forbidden and color arrays
    function paint(node,c)
        oldc=color[node]
        if oldc>0
            for t in G[node]
                forbidden[oldc,t]-=1
            end
        end

        color[node]=c
        (c==0) && return

        for t in G[node]
            forbidden[c,t]+=1
        end
    end

    found=false
    paint(1,1)

    function trav(node)
        if node>N
            found=true
            color_ret.=color
            return
        end


        for c in 1:maxcol
            (forbidden[c,node]>0) && continue
            paint(node,c)
            trav(node+1)
            found && break
        end

        paint(node,0) # restore the original state
    end # of trav
```

```
    end # of trav

    trav(2)
    (found,color_ret)
end


# a method (variant) that takes a Graph() instance and returns a similar
# object that is returned by Graphs.random_greedy_color
# (imitating by namedtuple)
function graphcol_bt(G::Graph,maxcol::Int) # max number of colors
    GG=[Int[] for n in 1:nv(G)]
    for e in edges(G)
        a,b=src(e),dst(e)
        push!(GG[a],b)
        push!(GG[b],a)
    end
    found,color=graphcol_bt(GG,maxcol)
    if found
```

```
# now deal w/ the original data of project_1
begin
include("../shared/graphcol_1_data.jl")
data=graphcol_1_data()
G=data.G
num_of_students=data.num_of_students
num_of_courses=data.num_of_courses
header=data.header
end
```

```
begin
    n=rand(3:2:9)
    G2=cycle_graph(n)
    @time the_coloring=graphcol_bt(G2,3);println(the_coloring)
    @time failed=graphcol_bt(G2,2);println(failed)
    if runall==true
        # and use graphcol_bt for the original data of project_1
        @time the_coloring=graphcol_bt(G,9);println(the_coloring)
        @time failed=graphcol_bt(G,8);println(failed)
    end
end
```

```
  0.000007 seconds (31 allocations: 2.484 KiB)
(num_colors = 3, colors = [1, 2, 1, 2, 1, 2, 1, 2, 3])
  0.000004 seconds (26 allocations: 2.000 KiB)
(num_colors = -1, colors = nothing)
  9.518313 seconds (131 allocations: 22.523 KiB)
(num_colors = 9, colors = [1, 2, 3, 4, 5, 6, 7, 8, 2, 3  …  8, 7, 9, 2, 4,
6, 7, 2, 1, 5])
  3.392261 seconds (123 allocations: 21.031 KiB)
(num_colors = -1, colors = nothing)
```

Conclusion

- even for this small graph this backtracking solution is very slow, but after executing it we can be sure that fewer than 9 colors (exam dates) is not enough.

```
begin
    md"""
    #### Conclusion
    * even for this small graph this backtracking solution is very slow, but after executing it we can be sure that fewer than 9 colors (exam dates) is not enough.
    """
end
```