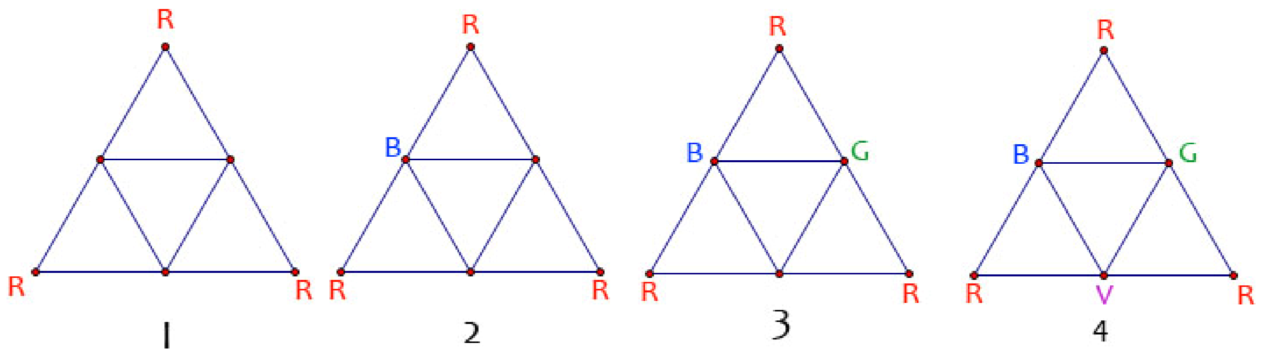
Math 15 – Discrete Structures – Homework 19 Solutions – 5.2

<u>5.2#18</u>: Recall that a coloring of a graph is an assignment of colors to the vertices such that no two vertices of the same color are connected by an edge. The following algorithm attempts to produce a coloring for the vertices of a graph *G* using the fewest number of colors possible.

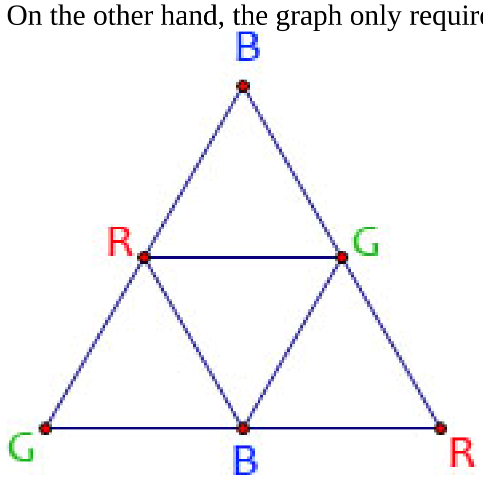
```
$C \leftarrow \emptyset$
while ($G$ has vertices left to color) do
    Pick a new color $x \notin C$.
    $C \leftarrow C \cup \{x\}$
    Assign color $x$ to as many vertices of $G$ as possible,
        such that no edge connects vertices of the same color
```

(a) What kind of algorithm is this?
ANS: This is a greedy algorithm since it tries to add as many vertices of a single color as possible at each stage

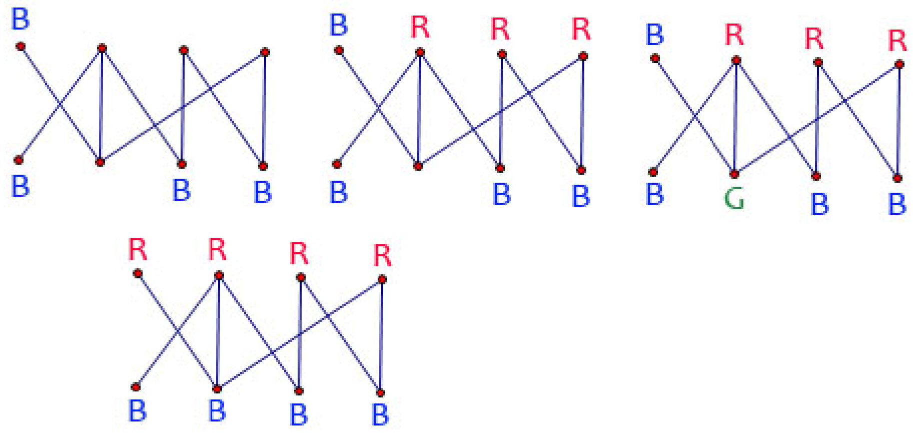
(b) Find a graph for which this algorithm fails to produce a coloring with the fewest possible number of colors.
ANS: Consider the graph of six vertices shown below. The most vertices you can color without an edge connection two vertices of the same color is to color all three vertices of degree 2 the same color, say, red. Then, following the algorithm we can color at most one new vertex at each iteration, requiring a total of 4 colors to complete, as shown here.



On the other hand, the graph only requires 3 colors, as shown below:



Alternately, consider the bipartite graph shown here (thanks to Joe Moeller for this one.):

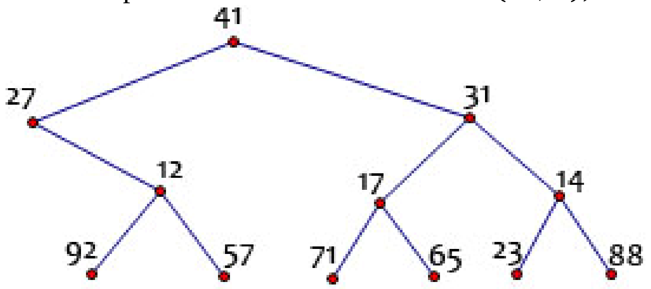


Comparing the first and second rows, you see how the algorithm can choose 4 vertices in the first iteration in several different ways. If it goes with the first row's choice, you end up needing 3 colors, where in the second row, you see only two colors are required.

5.2#20: Let $\mathcal{T}$ be a binary tree whose nodes are elements of some set U. The following algorithm searches $\mathcal{T}$ for a target value $t \in U$ and returns true if and only if t is a node in $\mathcal{T}$,

```
Function Search($t \in U, \mathcal{T} \in \{\text{binary trees}\}$)
   if $\mathcal{T}$ is empty then
      return false
   else
      if $t$ = the root of $\mathcal{T}$ then
         return true
      else
         return ($\text{Search}(t, \text{left subtree of } \mathcal{T}) \vee \text{Search}(t, \text{right subtree of } \mathcal{T})$)
```

(a) Write a top-down evaluation of Search$(17, \mathcal{T})$, where $\mathcal{T}$ is the tree shown below:



$$\begin{aligned}
\text{Search}(17,\mathcal{T}) &= \text{Search}(17,\mathcal{T}_{27}) \vee \text{Search}(17,\mathcal{T}_{31})\\
&= \text{Search}(17,\emptyset) \vee \text{Search}(17,\mathcal{T}_{12}) \vee \text{Search}(17,\mathcal{T}_{17}) \vee \text{Search}(17,\mathcal{T}_{14})\\
&= \text{false} \vee \text{Search}(17,\mathcal{T}_{92}) \vee \text{Search}(17,\mathcal{T}_{57}) \vee \text{true} \vee \text{Search}(17,\mathcal{T}_{23}) \vee \text{Search}(17,\mathcal{T}_{88})\\
&= \text{false} \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset) \vee\\
&\qquad \text{true} \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset) \vee \text{Search}(17,\emptyset)\\
&= \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{true} \vee \text{false} \vee \text{false} \vee \text{false} \vee \text{false}\\
&= \text{true}
\end{aligned}$$

(b) This is a divide and conquer type algorithm – it searches the tree by searching its two subtrees.

<u>5.2#24</u>: Write a divide and conquer algorithm that computes the sum of all elements of a finite set $K = \{k_1, k_2, \dots, k_n\}$ of integers.

SOLN

```
function DivConqSum(X)
    if X = $\{x\}$ then return $x$
    else
        $X_1 \leftarrow \{k_1, k_2, \dots, k_{\lfloor \frac{n}{2} \rfloor}\}$
        $X_2 \leftarrow \{k_{\lfloor \frac{n}{2} \rfloor + 1}, \dots, k_n\}$
        return DivConqSum($X_1$)+ DivConqSum($X_2$)
```