CSC 101 Outline for Final Exam

Spring 2013

1. Sound
   (a) Representation
   (b) Sample rate
   (c) Nyquist Theorem, Nyquist frequency
   (d) Discretization error
   (e) Impulse response function and convolution
2. Matlab
   (a) Generating an array of time values. Example: `t = 0:(1/44100):4 ;`
   (b) Generating a sampled sound of a given frequency
   (c) Adding arrays
   (d) Concatenating arrays
3. Huffman codes (Compression)
   (a) Building a Huffman tree
   (b) Constructing a Huffman code from the tree
   (c) Expected value: Number of bits needed to transmit 1 character
4. Images
   (a) Representation
   (b) 24-bit color
   (c) Color spaces: RGB, HSV, HSI
5. Computer Science Theory
   (a) Apparently difficult problems ($\mathcal{NP}$-hard problems)
      i. Unbounded knapsack problem
      ii. Graph coloring
      iii. Scheduling
   (b) Halting problem

CSC 101 Practice Problems for Final Exam: ANSWERS

1. How is digital sound represented ?

   Answer: Digital sound is represented by a sequence of numbers which represent the value of a periodic function at equally spaced points in time. The number of points in time per second is called the **sample rate**. Typically, CD-quality audio sound uses 44,100 samples / second. "Telephone answering machine quality" sound might use only 8192 samples per second.

   Another issue affecting digital sound quality is the accuracy with which the function values are represented. Audio CDs use 16-bits for each sound sample, allowing for 65536 different possible values for each sample.

2. How many bytes would be needed to store 3 minutes of digital stereo sound, given that each sample is a 16-bit signed number, and the sampling rate is the CD-audio standard of 44100 samples/second ?

   Answer:

   3 minute$\times$60 seconds/minute$\times$44100 samples/second$\times$2 bytes/sample = 15876000 bytes

3. What does Nyquist Theorem tell us about sample rate and the highest frequency that may be represented ?

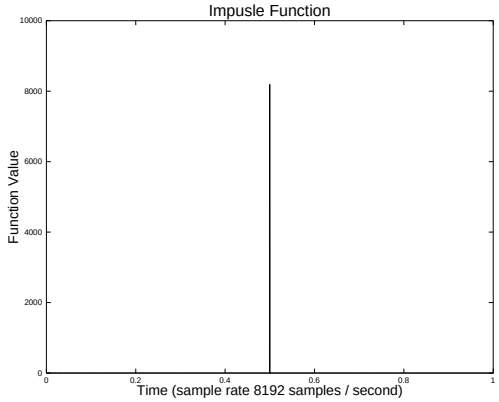
   Answer:

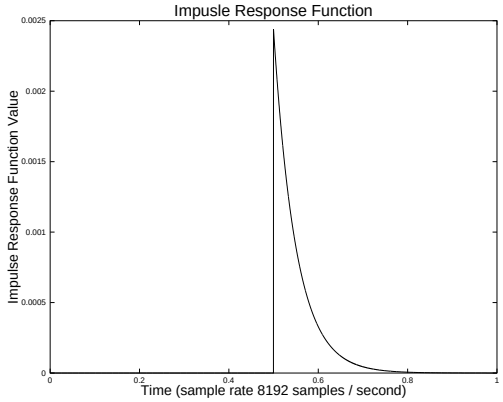
   The highest frequency which is represented in a digital sound is equal to half the sample rate. For example, if the sample rate is 44,100 samples / second, then the highest frequency represented is 22,050 Hz.

4. What is an impulse response function ?

   Answer:

   An impulse response function describes the response of an acoustical (or other) system to a single impulse. The graph below illustrates an impulse function and a sample corresponding impulse response function.





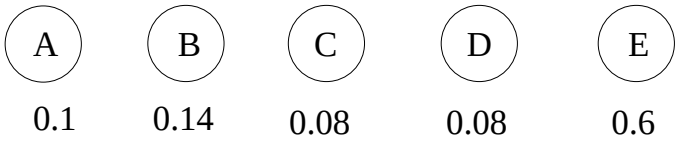
5. What is an impulse response function used for ?

   Answer: The convolution (a type of weighted average) of a sound with an impulse response function is often used to simulate echos and reverberation. Most modern popular music includes sounds (especially vocals) processed with extensive simulated reverberation (e.g. listen to vocals by Enya).

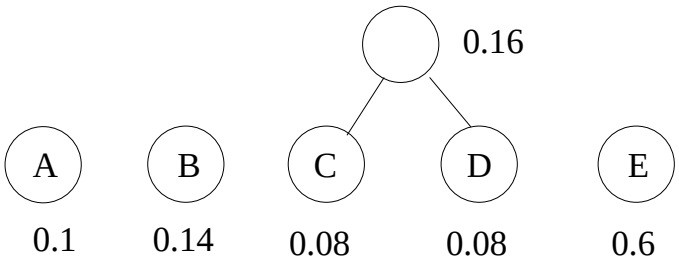
6. Given a 5-symbol alphabet $\{A, B, C, D, E\}$, with the following probabilities, construct a Huffman tree.

Letter	A	B	C	D	E
Probability	0.1	0.14	0.08	0.08	0.6

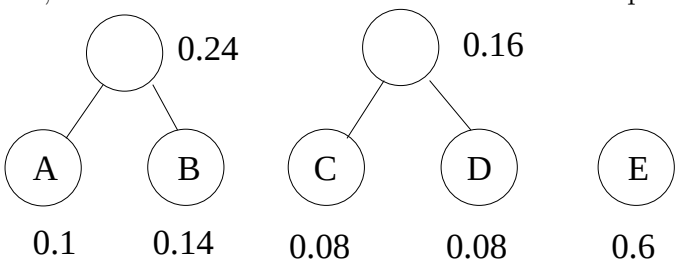
   Answer: Start by creating one leaf node for each letter. and label it with its probability:



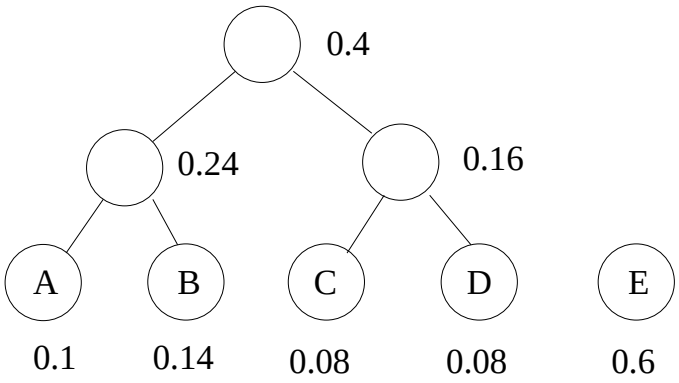
   Select two nodes with two lowest probabilities. Create a new node, and make the two selected nodes the children of the new node. Label the new node with the sum of the probabilities of the two child nodes.



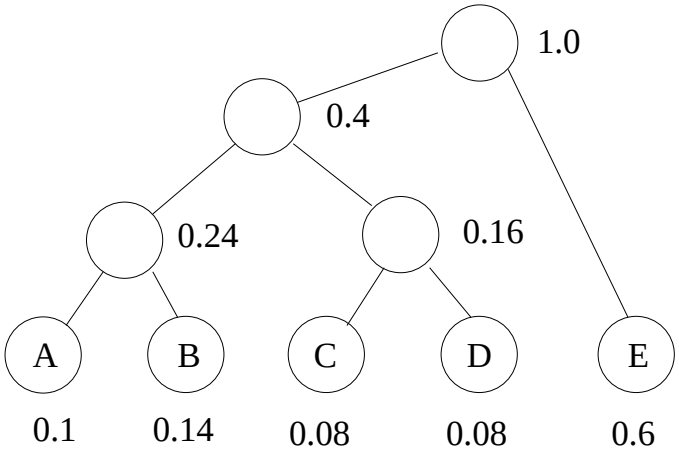
   Again, select two nodes with the lowest probability. When selecting the two nodes, only select from the remaining nodes that have no ancestors. In the diagram above, we select from the four probabilities: $0.1, 0.14, 0.16, 0.6$. The two smallest probabilities are 0.1 and 0.14, corresponding to "A" and "B". Create a new node and link the chosen nodes. As before, label the new node with the sum of the two probabilities.



   Select two nodes with the lowest probability among the remaining three probabilities $0.24, 0.16,$ and 0.6. Select the nodes labeled 0.24 and 0.16 and merge as before.

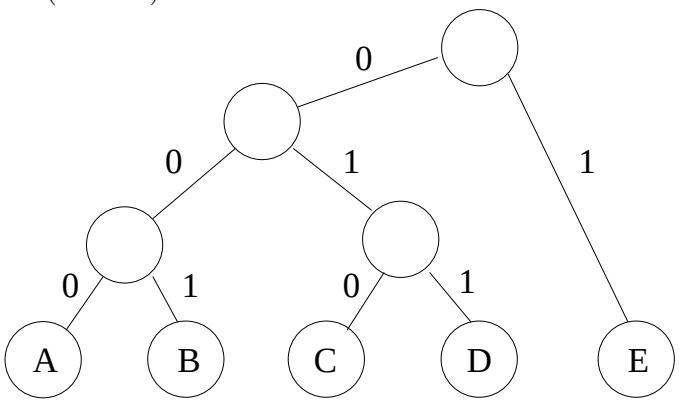


Finally, merge the final two nodes:



7. Construct a Huffman code using your tree in problem 6

Answer: To construct a code, label the edges of the tree with 0 (left branch) and 1 (right branch). The code consists of the labels along the path from the root (top) node down to the leaf (bottom) nodes.



The table below gives the codes:

A	B	C	D	E
000	001	010	011	1

8. Compute the expected number of bits needed to transmit one symbol.

   Answer:

   Let x denote the number of bits used to send one symbol. We compute the expected value as follows:

$$E(x) = 0.1 \times 3 \ + \ 0.14 \times 3 \ + \ 0.08 \times 3 \ + 0.08 \times 3 \ + \ 0.6 \times 1 = 1.8$$

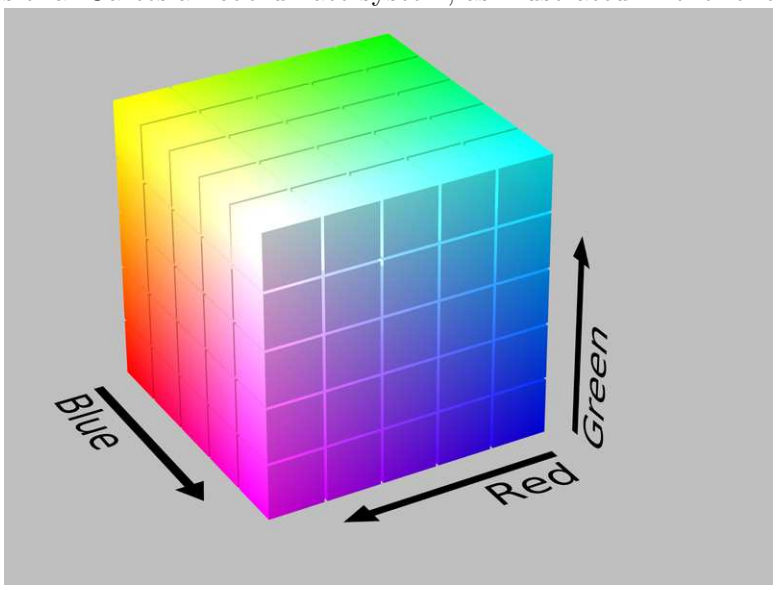
9. What is compression ratio ? Assuming it takes 3 bits to represent one symbol using a fixed length code (in problem 6), what is the compression ratio for your Huffman code ?

   Answer: A compression ratio is the ratio of the number of bits representing the compressed data to the number of bits representing the uncompressed data.[1] We can use the ratio of the expected values to compute the compression ratio.

$$\frac{1.8}{3} = 0.60$$

10. Describe the difference between the color spaces: RGB, HSV, HSI

    Answer: The RGB color space represents a color by three numbers r, g, b, representing the relative amounts of red, green, and blue respectively. Typically, one byte (8 bits) is allocated to each of the numbers r, g, and b. Using unsigned base two representation of integers, each number is between 0 and 255 (inclusive). This arrangement is often referred to as "24-bit color". The three numbers r, g, b can be thought of as variables in a 3-dimensional Cartesian coordinate system, as illustrated in the following diagram:

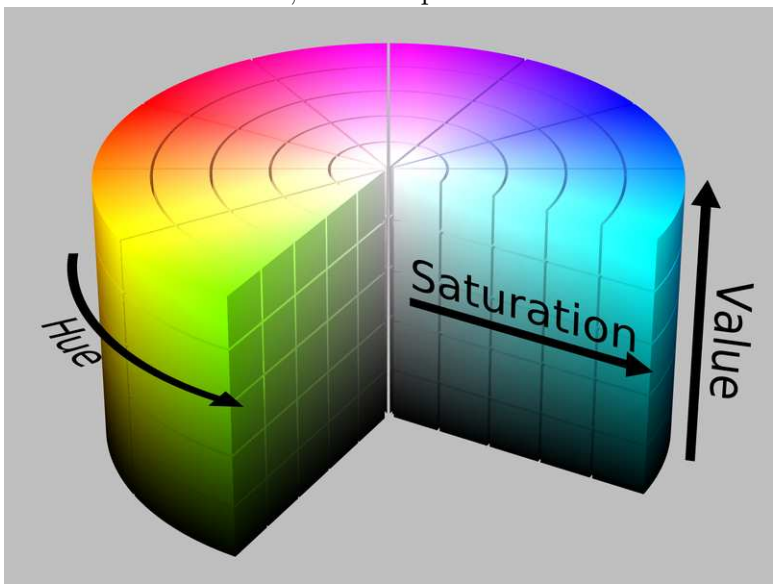


[1] Some authors will use the reciprocal of this quantity.

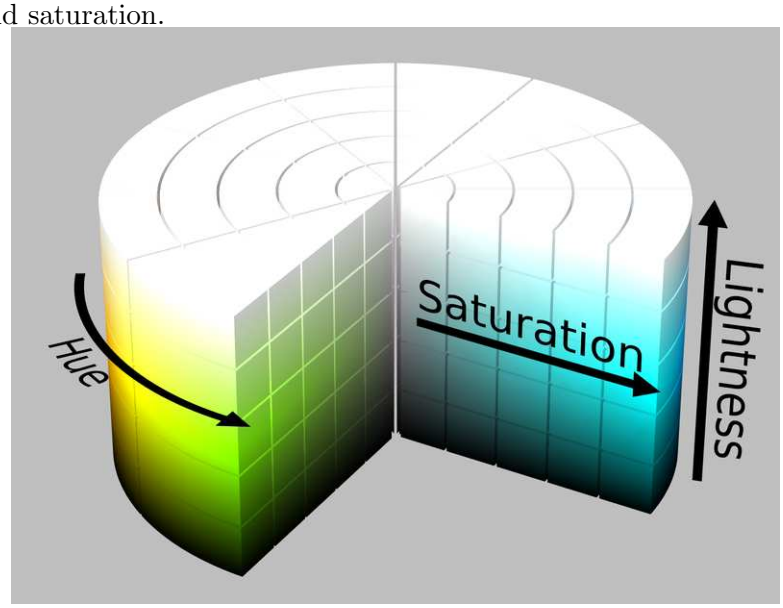
The HSV color space represents a color by three numbers h, s, v, These three numbers represent the hue, saturation, and value of the color. HSV represents colors in a cylindrical coordinate system as shown in the diagram below. The hues are arranged in a circular pattern starting with red, and moving through the spectrum to blue, then purple, and back to red again. Think of the number h (the hue) as an angle between 0 and 360° around circumference of the HSV cylinder.

The saturation parameter s refers to the purity of the color. For the moment, imagine that we fix the hue at blue. At the the center of the cylinder (where the saturation is 0), the color is gray. At the edge of the cylinder (where the saturation is at a maximum) the color is a pure hue.

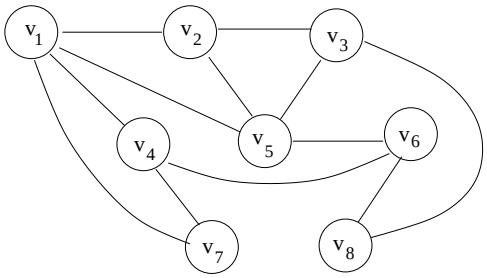
The value parameter v refers to the amount of color present. For example, refer to the diagram below and imagine a vertical line at maximum saturation (at the outside of the cylinder) and located at the blue hue. When v is zero, the corresponding color is black. When v is at a maximum, we see a pure blue.



The HSL color space represents a color by three numbers h, s, ℓ, These three numbers represent the hue, saturation, and lightness of the color. HSL represents colors in a cylindrical coordinate system as shown in the diagram below. The hues and saturation have the same interpretation as in the HSV color space. The difference is that the vertical axis represents lightness instead of value. When the lightness is 0, the color is black. When the lightness is at a maximum, the color is white regardless of the values of hue and saturation.

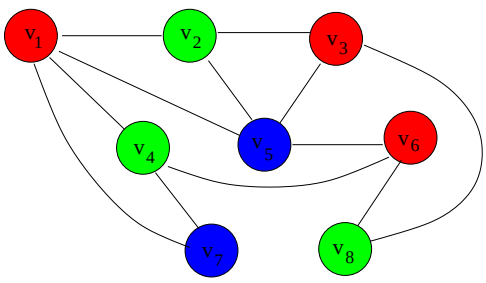


11. Can the following graph be colored with three colors ? Can it be colored with four colors ?

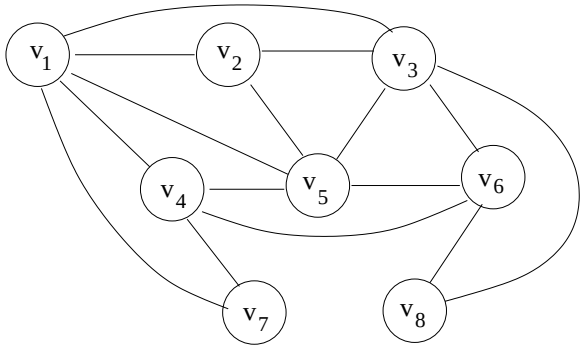


Answer: The **graph coloring problem**: We are given a graph and our problem is to assign a color to each node in the graph but with the requirement that if two nodes are connected by an edge, those two directly connected nodes can not be the same color.

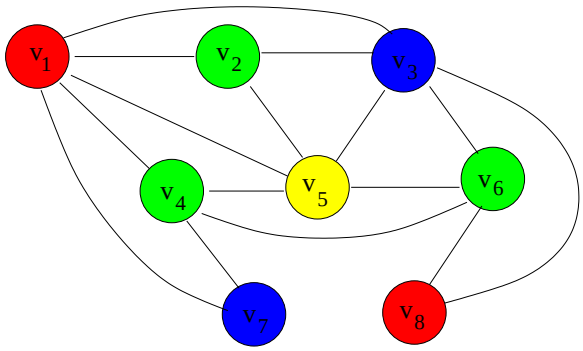
The graph in this example can be colored with three colors:



12. Can the following graph be colored with three colors ? Can it be colored with four colors ?



Answer: It can not be colored with three colors, but can be colored with four colors. I you look at the edges incident on nodes $v_1, v_2, v_3,$ and v_5, you will notice that every node is connected to every other node. This situation is called a *clique* in a graph, in this case these nodes form a 4-clique. No two of the four nodes can be colored using the same color (since there is an edge between every pair of nodes in the clique). Therefore, this graph will need at least four colors. It can be colored using four colors as follows:



The minimum number of colors required to color a graph is called the **chromatic number** of a graph.

13. How is the graph coloring problem related to scheduling ?

Answer:

A scheduling problem may be described as follows:

You want to schedule n classes: $C_1, C_2, ..., C_n$. You have m time slots available: $T_1, T_2, ..., T_m$. In addition, you have k rooms: $R_1, R_2, ..., R_m$. But, some pairs of classes must not be held at the same time. Our scheduling problem includes a list L of pairs of classes which may not be held at the same time.[2]

[2]For example, two required major classes must not be held at the same time in the same semester. Some graduating seniors will need both classes to graduate on time.

Also, two classes must not be scheduled at the same time if the same faculty member is teaching both.

In addition, lab classes (e.g., CSC 101) must not be scheduled at the same time as classes which are required for graduate students because a graduate student TA must be able to attend the lab without missing his own class.

Ok, so how does that relate to graph coloring ?

Suppose for the moment that we have as many rooms as we could ever possibly need. Under those circumstances, the scheduling problem is equivalent to the graph coloring problem. The classes $C_1, C_2, ..., C_n$ can be represented by the nodes $v_1, v_2, ..., v_n$ of a graph. Whenever two classes C_i and C_j can not be held at the same time, then connect the nodes v_i and v_j by an edge. The available time slots can then be treated as the "colors" in a graph coloring problem.

Any two classes which can not be held at the same time will correspond directly to two vertices which can not be the same color.

14. Describe the unbounded knapsack problem.

Answer: The unbounded knapsack problem consists of a set of n numbers:

$$\{x_1, x_2, ..., x_n\}$$

and a total T. The problem is to select any of the x's as many times (zero or more times) as you would like so that the sum adds up to T. Mathematically, the problem is to find non-negative integers $a_1, a_2, ...a_n$ such that

$$a_1x_1 \ + \ a_2x_2 \ + \ a_3x_3 \ + \ \ ... \ \ + a_nx_n \ \ = \ \ T$$

You solved several instances of the unbounded knapsack problem in the Makey-makey lab.

The "0-1 knapsack problem" is similar, except that each of the a's can only be a zero or a one.

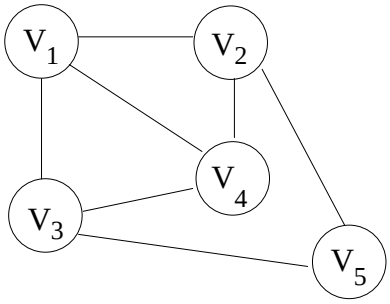
15. What characteristic(s) make the $\mathcal{NP}$-hard problems difficult ?

An essential characteristic of the $\mathcal{NP}$-hard problems is that there is no mathematical or logical rule that allows you to make a selection that is guaranteed to make progress towards a solution. There is a certain amount of guessing involved in these problems. A good analogy is to think of the problem as if it were a maze. Whenever a guess is made, it may be a wrong choice; i.e., that choice leads to a dead-end instead of a solution.

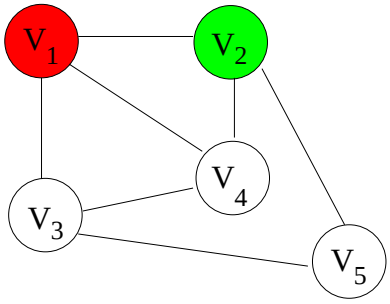
For example, let's consider an easy problem: sorting N numbers. If we scan our list of numbers and put the largest number at the end, we have made progress towards our solution without any guesswork. The largest number is in the correct position. We can complete the sort by choosing the next largest number among the remaining $N-1$ numbers. We see steady progress towards a solution, with no back-tracking or second guessing.

In contrast, consider the graph coloring problem. You select a node to color, and then you select a color. At the time you select the color, you have no guarantee that your color choice is not a mistake. I.e., after you have put the color on the node, there may be no way to complete the graph coloring with the remaining color choices.

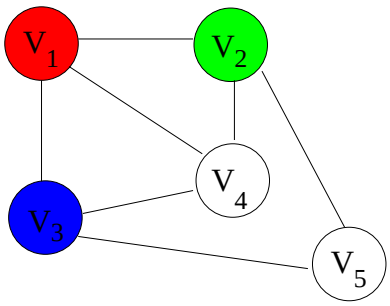
For example, consider the following graph, and we'll try to color it using three colors.



Suppose we have made choices for v_1 and v_2 as follows:

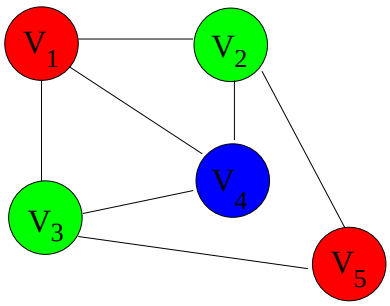


Now we choose a color for v_3. Let's choose "blue" (see the diagram below). But, now when we try to color v_4 we notice that it can be neither red, nor green, nor blue.



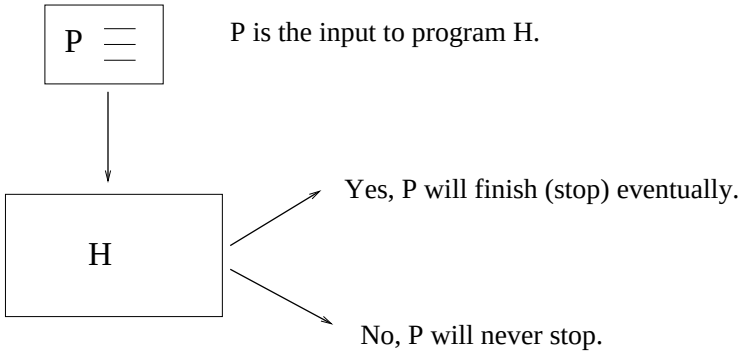
Choosing blue for node v_3 was a mistake. Keep in mind this is a simple example. In a graph containing a large number of nodes (e.g., hundreds), a mistake might not be discovered until several moves later.

If we choose green for node v_3 we see the graph can be colored using just three colors.



CSC 101 Additional Notes

The **halting problem** is as follows. You are given a computer program, let's call it P. You want to give the instructions comprising P to another program I'll call H. The program H is supposed to examine the instructions in program P and decided whether P will eventually finish running if it is started.[3] The situation can be illustrated as follows:



THEOREM: Program H does not exist.

The mathematical proof of the theorem above is beyond the scope of CSC 101, however the theorem has been proven since the mid 1930's and is 100% accepted by the mainstream computer science community.

Matlab Notes

Recall the ":" operator for generating an array of uniformly spaced numbers. For example(s):

```
> 1:7
ans =

   1   2   3   4   5   6   7

> 10:0.2:11
ans =

    10.000    10.200    10.400    10.600    10.800    11.000
```

Recall the syntax for writing an array and for the concatenation of two arrays. For example:

```
> x = [ 2 3 5 7 11 13 ];
> y = [ 17 19 23 ] ;
> z = [ x, y ]
z =

    2    3    5    7   11   13   17   19   23
```

[3] It is possible for a program to loop forever.

Recall that we can use parenthesis to select elements from an array. We can also use an array to index another array:

```
> z(4)
ans =  7

> z( [ 4 6 ] )
ans =

    7   13

> ix = 9:-1:1
ix =

   9   8   7   6   5   4   3   2   1

> z(ix)
ans =

   23   19   17   13   11    7    5    3    2
```

Recall the syntax for adding two arrays:

```
> u = [ 2 3 5 7 ] ;
> v = [ 11 13 17 19 ] ;
> w = u + v
w =

   13   16   22   26
```

Recall the syntax for multiplying two arrays:

```
> uv = u .* v
uv =

    22    39    85   133
```

In Matlab, the operator "*" refers to matrix multiplication (row times column in the context of linear algrebra). To perform element-by-element multiplication, the correct operator is ".*".

Recall using built-in functions for generating a sound. For example, to create 3 seconds of a sampled sound (sine wave) at 220 Hz with a sample rate of 44,100, we can use the following commands:

```
> t = 0 : (1/44100) : 3 ;
> y = sin( 2.0 .* pi .* 220.0 .* t ) ;
```

I can see a plot of the first 800 samples of the sine wave using the command:

```
> plot( y( 1:800) ) ;
```