CSCI 120 Introduction to Computation
On bitmaps, colors, graphs, and more (draft)

Saad Mneimneh
Visiting Professor
Hunter College of CUNY

1 A bitmap

Consider the following 4x3 pixel image, which can be created using the Paint software in Microsoft Windows.

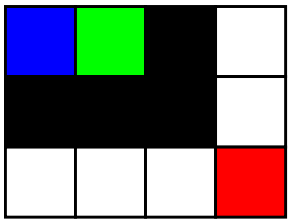

Figure 1: A 4x3 pixel image

When saved as a bitmap, the image is encoded in binary where each individual pixel is represented by three bytes (24 bits). This representation depends on the color of the pixel, and uses the RGB color scheme. In RGB, each color is obtained as a mixture of three colors with different intensities: Red, Green, and Blue. Therefore, each of the three bytes defines the intensity of the corresponding color, a value that will range from 0 to 255 (because 256 patterns are possible with 8 bits). The figure below shows how three color light beams mix on a black surface.

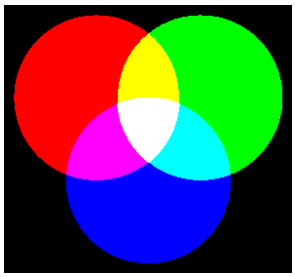

Figure 2: RGB color

Because three light beams with different intensities can produce a range of colors on a black surface (no color), RGB is called additive. In contrast, printers use a subtractive color scheme in which the ink drops subtract colors from white

(the paper). An example of a subtractive color scheme is CYMK (Cyan, Yellow, Magenta, and Black). For instance, Cyan subtracts the red from white so we see cyan (see RGB above). Black is used for economical reasons (colored ink is more expensive), and because it produces better quality black then the one obtained by combining Cyan, Yellow, and Magenta.

Viewing the binary file produced for the above 4x3 image requires a special file reader because most of the software applications that read files assume that files contain text and, therefore, convert bytes to characters using ASCII. A good file reader for viewing binary is *fb*, which can be downloaded from the course website. If we save the image as patch.bmp, then typing **fb -b patch.bmp** at the command prompt will display the following:

```
          0        1        2        3        4        5        6        7
00: 01000010 01001101 01011010 00000000 00000000 00000000 00000000 00000000
08: 00000000 00000000 00110110 00000000 00000000 00000000 00101000 00000000
16: 00000000 00000000 00000100 00000000 00000000 00000000 00000011 00000000
24: 00000000 00000000 00000001 00000000 00011000 00000000 00000000 00000000
32: 00000000 00000000 00100100 00000000 00000000 00000000 00000000 00000000
40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
48: 00000000 00000000 00000000 00000000 00000000 00000000 11111111 11111111
56: 11111111 11111111 11111111 11111111 11111111 11111111 11111111 00000000
64: 00000000 11111111 00000000 00000000 00000000 00000000 00000000 00000000
72: 00000000 00000000 00000000 11111111 11111111 11111111 11111111 00000000
80: 00000000 00000000 11111111 00000000 00000000 00000000 00000000 11111111
88: 11111111 11111111
```

We can also choose to display the bytes in decimal by typing **fb -d patch.bmp** instead:

```
     0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
00: 066 077 090 000 000 000 000 000 000 000 054 000 000 000 040 000
16: 000 000 004 000 000 000 003 000 000 000 001 000 024 000 000 000
32: 000 000 036 000 000 000 000 000 000 000 000 000 000 000 000 000
48: 000 000 000 000 000 000 255 255 255 255 255 255 255 255 255 000
64: 000 255 000 000 000 000 000 000 000 000 000 255 255 255 255 000
80: 000 000 255 000 000 000 000 255 255 255
```

Byte 18 and byte 22 represent the image size. Byte 28 identifies that each pixel is represented by 24 bits (3 bytes, one for each intensity of RGB). We can also identify the following triplets of bytes (right to left bottom up):

```
255 255 255: white
255 255 255: white
255 255 255: white
255 000 000: red

000 000 000: black
000 000 000: black
000 000 000: black
255 255 255: white

000 000 255: blue
000 255 000: green
000 000 000: black
255 255 255: white
```

2 A map

Consider the following map (also downloadable as a bitmap from the course website).

Figure 3: A map

Using the filling tool in Paint, we can color the different regions of the above map (states of USA in this case). To do this, first select a color. Then, by simply clicking the mouse while in the desired region, the entire region will be colored with our choice.

Here's an example:

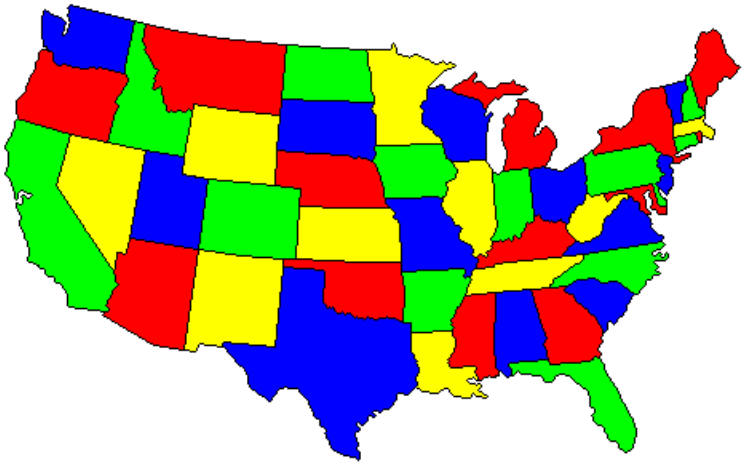

Figure 4: A colored map

Note that when saving the image is JPG, we reduce the quality (but also reduce the file size from about 433 KB to 27 KB).

Figure 5: Saved as JPG

How does the filling tool know when to stop. In other words, how does it detect boundaries? The answers lies in the fact that the Paint software works with a graph representation of the image. Let's look at what a graph is.

3 Graphs

A graph is an abstract notion and consists of a set of nodes and a set of edges connecting pairs of nodes. The nodes are often represented by circles, and edges are often represented by lines (not necessarily straight). Here's an example:

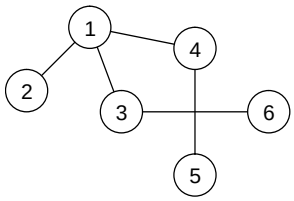

Figure 6: Example graph with 6 nodes and 5 edges. The graph can also be represented by simply stating the set of nodes (also called vertices) V={1, 2, 3, 4, 5, 6} and the set of edges: $E = \{(1,2),(1,3),(1,4),(3,6),(4,5)\}$.

The neighbors of a node in a graph are the nodes that are connected to it by edges. In the above example, node 1 has 3 neighboros $\{2,3,4\}$, and node 3 has 2 neighbors $\{1,6\}$, etc... A graph can be used to model any system where relationship between entities must be expressed. For instance, a graph can model cities where nodes represent places and edges represent roads connecting those places. Another example would be a social network, where nodes represent people and edges represent friendships among them.

The Paint software uses a graph to model the image. In this case, nodes represent pixels, and two nodes are connected by an edge if the corresponding pixels they represent are adjacent. Here's how the bitmap of Section 1 can be modeled as a graph.

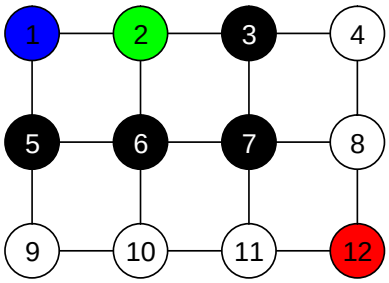

Figure 7: Graph model of bitmap. The graph contains 12 nodes representing the 12 pixels. Each node has also a color attribute corresponding to the color of the pixel it represents. The graph contains 17 edges representing adjacent pixels.

Using the graph model, the filling tool detect boundaries and performs the filling of regions according to the following algorithm (assume we fill with red):

- start at the node (pixel) clicked, let its color be c
- mark the node *visited*
- add it to a queue
- repeat
  - take a node from the queue, color it red
  - for each of its neighbors, if not *visited* and has color c
    - mark it *visited*
    - add it to the queue
- until the queue is empty

Let's look at an example. Say we want to recolor a region red, and we click on pixel (node) 6. Then color c is black.

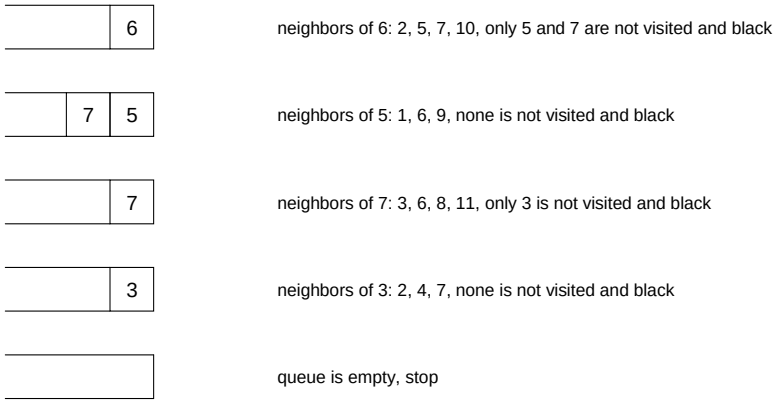

Figure 8: Filling region of pixel 6. Nodes 6, 5, 7, and 3 are colored red.

4 Planar graphs and the four color theorem

A graph is called planar if its edges can be drawn in the plane (on paper) without crossing (recall edges can be curves not necessarily straight lines). For instance, the graph in Figure 6 is planar because edge (3,6) can go around node 5 from below. Alternatively, node 6 may be moved below node 5 and, therefore, edge (3,6) will not cross edge (4,5). The following two graphs are also planar, try to redraw them to avoid all edge crossings.

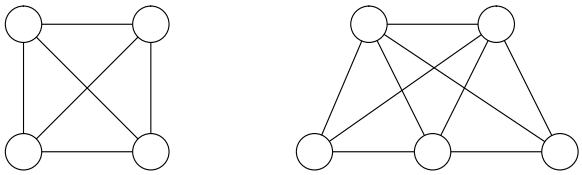

Figure 9: Two planar graphs

An interesting question is the following: If edges can curve and we can redraw graphs, aren't all graphs planar? Of course, the answer is no; otherwise, the definition is useless. Here's an example known as the utility problem: Consider three houses and three utilities: electricity, water, and gaz. We would like to connect every house to all three utilities, as shown below. Try to do it while avoiding all edge crossings (hint: you can't).

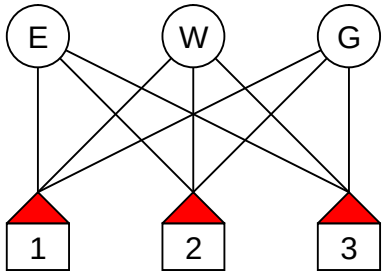

Figure 10: The utility problem (6 nodes, 9 edges). This graph is known as $K_{3,3}$.

So why are planar graphs important? This dates back to 1852 when Francis Guthrie in England asked if it is possible to color maps using four colors only in such a way that neighboring countries (or states) get different colors. But how is that relevant to graphs? A map can be represented by a planar graph. Each country (or state) is represented by a node, and two nodes are connected by an edge if their corresponding countries (or states) share a boundary. The result is a planar graph.

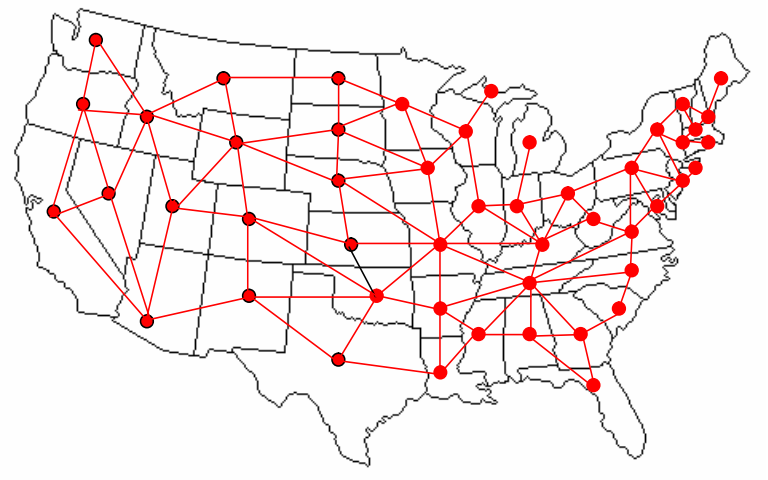

Figure 11: Planar graph representation of map

Note that this planar representation is not possible if a country is geographically disconnected, which could happen if, for instance, a country like USA occupies another country like Iraq, so both countries become one. Here's a fictitious example:

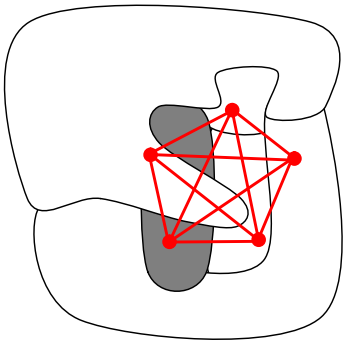

Figure 12: A map with five countries. A disconnected country is shown in gray. Each country is a neighbor of the other four. The resulting graph is not planar (check). This graph is known as K_5.

Therefore, we only need to color the nodes of a planar graph in such a way that neighboring nodes get different colors. For decades, the problem of coloring a planar graph using only four colors was kicked around by mathematicians as a simple but elusive puzzle, until the difficulties in obtaining a proof became apparent in the 1870s. An erroneous proof was published by Kempe in 1879, and the problem was regarded as solved for a good decade before the error was discovered. After the collapse of Kempe's proof, for more than a century many mathematicians tried in vain to solve this question. In 1976, Appel and Haken gave a proof, but their proof used computers heavily to check an enormous number of cases (more than 1000 hours of CPU time). Even today, the use of computers could not be eliminated from the proof, and we still don't have a "pure" mathematical proof of what is now known as the four color theorem.

The four color theorem: A planar graph can be colored with only four colors such that neighboring nodes get different colors (thanks to mathematicians and computers).

5 Trees and binary search

Starting at a given node v in a graph, if it is possible to visit other nodes by traveling along edges at most once per edge, and come back to node v, the graph is said to have a cycle. A graph with no cycles is called a *tree* (for a reason that will become apparent shortly). For instance, the graph in Figure 6 is a tree because it has no cycles. The following figure shows a graph with a cycle and a graph with no cycles (a tree).

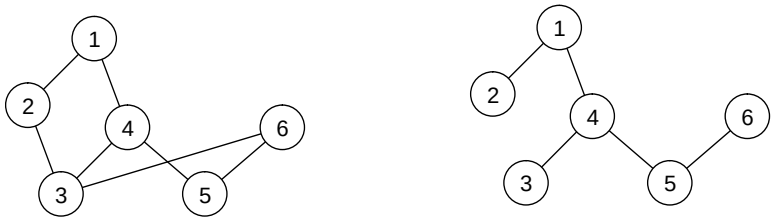

Figure 13: The first graph has cycles, e.g. starting at node 1, we can visit node 2 by traveling along edge (1,2), then node 3 by traveling along edge (2,3), then node 4 by traveling along edge (3,4), then back to node 1 by traveling along edge (4,1). The first graph contains other cycles. Try to find a cycle that goes through all the nodes. The second graph has no cycles, it is a tree.

A rooted tree is a tree with one node designated as the root. Mathematicians and computer scientists draw rooted trees upside down like in the following figure with parent/child relationship.

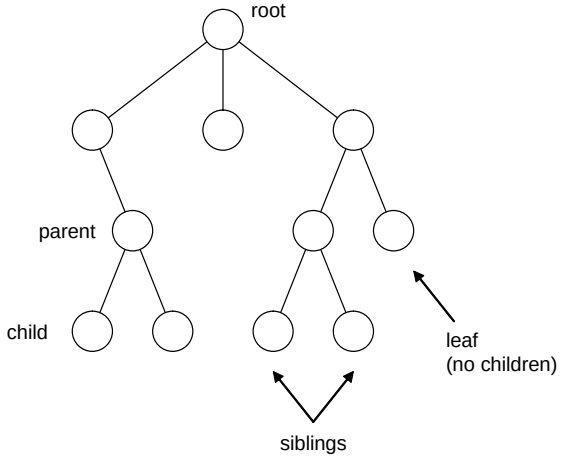

Figure 14: A rooted tree showing the root, parent/child relationship, siblings relationship, and leaves.

A binary tree is a rooted tree in which every node has at most two children. Binary trees are useful for searching. Consider the following problem: we are given a list of numbers and we would like to search the list for a specific number and report whether that number exists or not. Obviously, one algorithm is to examine all numbers in the list one at a time, and stop whenever one of them is equal to the number we are searching for. While this algorithm works, we have to check the entire list in the worst case; for instance, when the given number is not in the list. A better approach is to construct a binary tree representation of the list as follows: Let the first number in the list be the root. Then insert the rest of the numbers by starting at the root and either going left or right depending on whether the number is smaller or larger, respectively. In other words, each number is placed as a new leaf in a binary tree by comparing it to intermediate nodes on the path from the root to that leaf. Here's an example for the list: 4 7 2 1 5 8 3 6.

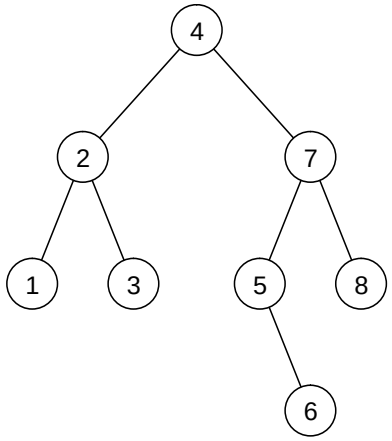

Figure 15: A binary tree for the numbers 4 7 2 1 5 8 3 6.

Using the tree model above, searching for 5 would proceed in the following way: starting at the root, we determine that 5 is larger than 4, so we go right, then we determine that 5 is less than 7, so we go left, then we find 5. Searching for 9 would proceed in the following way: starting at the root, we determine that 9 is larger than 4, so we go right, then we determine that 9 is larger than 7, so we go right, then we determine that 9 is larger than 8, but since 8 is a leaf (we can't go right), we know that 9 does not exist. This search algorithm is called binary search and the tree constructed in this way is called a binary search tree. Obviously, the number of times we have to check a number is bounded by the height of the tree, which is usually much less than the length of the entire list.