SOLUTION OF THE SPECTRA PUZZLE[*]

Robert L. Lamphere
Elizabethtown Community College
Elizabethtown, KY 42701
(270) 769-2371
Robert.Lamphere@KCTCS.EDU

ABSTRACT

In this paper, we solve the SPECTRA Puzzle and express the solution in terms of color-path sets. We can add, invert, and combine these sets by the addition, inverse, and the union operators, respectively. We express the solution by a mathematical formula in which we use color-path sets and its operators. To simplify the C++ implementation of the solution, we define a C++ SPECTRA class. We translate the mathematical formula into C++ code by using the SPECTRA class. The SPECTRA class contains a color-path set and the set operators. The color-path set is implemented as a linked list and the set operators are implemented by extending the C++ operators "+", "--" and "*".

INTRODUCTION

We believe puzzles have pedagogical value. For example, Tower of Hanoi and Spinout are puzzles that illustrate recursion. We have used Tower of Hanoi and Spinout with success in our programming courses. Not only do these puzzles offer a means for students to acquire skill in recursive programming, but they also give the students an opportunity to use their mathematical skills in setting-up and solving the Tower of Hanoi and Spinout recursion equations. For a description and analysis of Tower of Hanoi, see [**1**], and for a description, analysis and a program solution of Spinout see [**2**, **3**]. SPECTRA, on the other hand, is a puzzle whose solution does not depend on recursion. We are going to introduce a set of color-paths and three set operators; an addition operator "+", an inverse operator "--", and a union operator "**c**". The solution will be given as a mathematical formula involving set operators and color-path sets. Even though the formula solves the puzzle, using the formula to get the actual solutions is cumbersome. To help us to compute the solutions from the formula, we define a C++ SPECTRA class in terms of the color-path set and the set operators. When overloaded, the C++ “+”, “--“ and “*” operators will become our addition, inverse and union operators, respectively. We use this SPECTRA class to write a C++ program that will compute all solutions of the puzzle and determine the number of solutions that satisfy the given conditions; for example, to determine the number of solutions that uses a specified number of colors.

The puzzle’s solution contains numerous pedagogical nuggets. It uses the mathematical concept of set and set operators. One of the operators is non-commutative. The SPECTRA class flows naturally from the color-path set and the set operators. The C++ program that computes the solutions follows directly from the mathematical formula and the SPECTRA class.

In the following sections, we will describe the SPECTRA puzzle, develop a mathematical formula that solves the puzzle, define the SPECTRA class, and use that class and formula to write the C++ program that will find all the puzzle solutions.

DESCRIPTION.

The SPECTRA puzzle is from the IQ Company Ltd. SPECTRA consists of 12 mounted discs on rotating arms (see Figure 1). Each disc consists of six wedges with each wedge having a different color. There are six small rotatable arms with two discs on each arm. Two of these small arms are attached at opposite ends of the puzzle’s base. The four remaining small arms are attached to the opposite ends of two larger rotatable arms. The two larger arms are attached to the opposite ends of a still larger rotatable arm that is attached to the puzzle’s base.

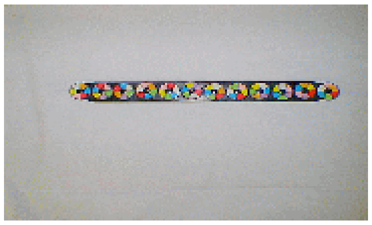
Figure 1. Spectra

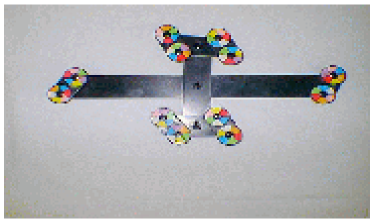
Figure 2. Spectra showing rotatable arms

Eight wedge colors are used in the puzzle and the colors vary from disc to disc. The different colors are represented by the following letters: **B** for blue, **G** for green, **O** for orange, **P** for pink, **R** for red, **U** for purple, **W** for white, and **Y** for yellow.

The SPECTRA puzzle is completely solved when the color wedges are lined up so that the colors match-up end-to-end with the same first and last colors. The puzzle is partially solved when the first and last colors are different. There could be many complete and partial solutions. When the complete and partial solutions are known, we can determine the number of complete and partial solutions, the number of solutions containing six colors, seven colors, and eight colors.

When the twelve discs lie in a straight line along the SPECTRA's base, we call it a SPECTRA Configuration. Since the discs are rotatable and the discs themselves are on rotatable arms, many possible SPECTRA configurations exists. (In fact, $2^9 6^{12}$ such configurations exists.) The objective of the puzzle is to find those SPECTRA configurations which partially or completely solve the puzzle.

DEFINITION OF COLOR-PATH SET.

Each wedge of a disc is paired with its diametrical opposite. Since each disc contains six wedges, there are three such pairings. For example, in Figure 3, the green (G) and white (W) wedges form a pair, the pink (P) and blue (B) wedges form a pair, and the red (R) and orange (O) wedges form a pair. These three pairings can be represented by the symbols GW, PB, RO, respectively. These symbols are formed by combining the wedges' color letters; for instance, RO and OR represent the red and orange pairing.

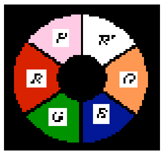

Figure 3.

Before defining Color-Path set, we need to give several definitions.

Orientation Axis Definition. The orientation axis is the straight line that runs down the middle of the SPECTRA's base, or equivalently, the straight line that passes through the center of the two arms which are attached at the ends of the SPECTRA's base (arms A and B in Figure 9).

Disc Orientation Definition. The disc orientation is defined by the wedge pairing that lies on the orientation axis. It is expressed by the wedge pairing symbol with the left wedge's color listed first.

The disc orientation is defined by the wedge pairing that lies on the orientation axis. Since the disc's wedge colors are different from each other, only one wedge pairing can lie on the orientation axis at any one time, and the left wedge's color is listed first and uniquely determines the disc's orientation. For example, the disc in Figure 3 is represented by RO. The red wedge (R) lies to the left of the orange wedge (O) on the orientation axis. Any other position of the disc will not have the red and orange wedges on the disc's orientation axis with the red wedge to the left of the orange wedge. Therefore, the symbol RO uniquely determines the disc's position.

Color-Path Definition. Color-Path is the line up of the color wedges along the orientation axis and it is represented by their color symbols going from left to right.

Examples. The color-path in Figure 4 is BYYP. Another color-path is BYBG, this color-path comes from rotating the right disc counter-clockwise until the blue wedge is on the orientation axis. Figure 4 contains 36 different color-paths.

Each of the $2^9 6^{12}$ SPECTRA Configurations contain a color-path. Each rotatable arm has many different color-paths associated with it, and each disc has six color-paths.

Admissible Color-Path. A color-path with its color wedges lined up along the orientation axis so their color's match-up end-to-end is an admissible color-paths.

Examples. The color-path BYYP in Figure 4 is an admissible color-path. Another admissible color-path is WRRU. The color-path BYBG, however, is not an admissible color-path because the wedge colors do not match-up end-to-end.

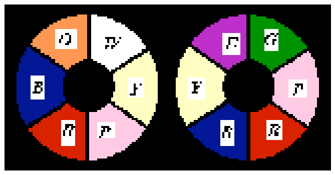

Figure 4.

Color-Path Set Definition. A collection of all possible admissible color-paths for any disc, rotatable arm, or SPECTRA configuration will be called a color-path set. These sets will be represented by [] and their names given in bold.

For example, the color-path set for the arm in Figure 4 is [BYYP, WRRU, OPPY, YBBG].

We can associate a color-path set with each disc. For example, the color-path set for the disc in Figure 3 is [GW, BP, RO, WG, PB, OR]; the orientation axis being the horizontal line through the disc's center. Note that the color-paths GW and WG are not the same. The color-path GW corresponds to the disc being oriented with the green wedge to the left of the white wedge and with both wedges on the orientation axis. The color-path WG has the white wedge to the left of the green wedge and also has both wedges on the orientation axis.

ADDITION OPERATOR "+".

Addition Definition. Let **A** and **B** be color-path sets. Then **A** + **B** is the color-path set formed by concatenating the color-paths of **A** to those of **B**'s only if the right most color letter of **A**'s color-path matches the left most color letter of **B**'s color-path.

Example. Referring to Figure 4, we have for the left disc, **A** = [BY, YB, OP, PO, RW, WR], and for the right disc, **B** = [YP, PY, BG, GB, RU, UR]. Then,

(1) **A** + **B** = [BYYP, YBBG, OPPY, WRRU].

The Y in **A**'s color-path BY is matched with the Y in **B**'s color-path YP yielding the first color-path, BYYP, in (1). The **A**'s color-paths YB, OP, and WR are similarly matched with **B**'s color-paths BG, PY, and RU yielding the other three color-paths in (1).

Computing **B** + **A**, we have

(2) **B** + **A** = [YPPO, PYYB, GBBY, URRW].

Now comparing (1) and (2), we find **A + B**   **B + A**. Hence, the addition operator "+" is not commutative.

Theorem 1. Suppose arms A and B are adjacent and **A** and **B** be their color-path sets. Then **A + B** yields all the admissible continuous color-paths that run along arms A and B.

Proof. Follows from the definition of addition.

Given three color-path sets, A, B and C, we may form the two expressions A + (B + C) and (A + B) + C. As with addition of the integers, there is an associative law for the addition operator "+."

Theorem 2. Let **A**, **B**, and **C** be color-path sets. Then,

A + (**B** + **C**) = (**A** + **B**) + **C**. (associative law)

Proof. Let *b..c* be any color-path of set **B.** Pick any color-path from **C**, say *c..d,* such that, the ending color of path *b..c* is the same as the beginning color of path *c..d*. Then, by definition of the addition operator, (**B** + **C**) yields the color-path, *b..cc..d*. Now pick any color-path from A, say *a..b***,** such that, the ending color of path *a..b* and the beginning color of path *b..cc..d* are equal. Then, **A** + (**B** + **C**) yields the color-path *a..bb..cc..d*. Similarly, the operation (**A** + **B**) produces the color-path, *a..bb..c;* therefore, (**A** + **B**) + **C** yields the color-path, *a..bb..cc..d.* , the same color-path that comes from **A** + (**B** + **C**). Hence,

A + (**B** + **C**) = (**A** + **B**) + **C.**

INVERSE OPERATOR "--".

Inverse Definition. Let **A** be a color-path set. Then --**A** is the color-path set containing the color-paths found in A with the color-path-color symbols written in inverted order.

Example. Let **A** = [BYYPPG, RGGYYR, BWWOOG]. Then

–**A** = [GPPYYB, RYYGGR, GOOWWB].

Theorem3. Let **A** be the color-path set for arm A. Let AN be arm A rotated 180° and **AN** be its color-path set. Then **AN** = --**A**.

Proof. When arm A is rotated 180°, its admissible color-paths remain admissible color-paths. Thus, **A** and **AN** contain the same number of color-paths. Since arm AN is arm A rotated 180°, the color wedges that make up the color-paths of arm AN are those of arm A in reverse order. Therefore, to get the color-paths of **AN**, reverse the color symbols in each **A**'s color-path.

Hence, it follows from the definition of the inverse operator that **AN** = --**A**.

From the definition of "+" and "--" we have the following

Theorem 4. Let arms A and B be adjacent and attached to arm C. Let arms A, B, and C color-path sets be **A, B**, and **C**, respectively. Let **C** = **A** + **B**. Then

--**C** = **--B** + **--A**.

Proof. From Theorem 1, we have that **C** is the set of all admissible color-paths on arm C. Now rotating arm C 180° results in rotating arms A and B 180° and putting the rotated-arm B to the left of the rotated-arm A. Thus by Theorem 1, **--B** + **--A** is the color-path set of all the admissible color-paths from rotated-arm B to rotated-arm A, i.e., all the color-paths that runs along arm C after arm C is rotated 180°. The wedge colors in the admissible color-paths that run along the rotated-arm C are the same wedge colors in the color-paths along arm C (before being rotated), except the wedge colors are inverted. This conclusion follows from the simple observations that any valid color-path on any arm is still a valid color-path after that arm is rotated 180°, and rotating any arm 180° results in inverting that arm's wedge-colors.

Hence, by Theorem 3,

--**C** = **--B** + **--A**.

UNION OPERATOR "**c**".

Union Definition. Let **A** and **B** be color-path sets. Then **A c B** is the color-path set containing **A**'s color-paths and **B**'s color-paths.

Example. Let **A** = [BYYPPG, RGGYYR, GRRWWP] and **B** = [GYYB, PUUW]. Then **A c B** = [BYYPPG, RGGYYR, GRRWWP, GYYB, PUUW].

Theorem 5. If **A**, **B**, and **C** are color-path sets and **C** = **A c B**, then **--C** = **--A c --B.**

Proof. Let c be a color-path in --**C**. From Theorem 2, we have that c is an inverted color-path of **C**. It follows from the inverse definition, that c is either in **--A** or **--B** and therefore c is in **--A c --B**. Similarly, if d is a color-path of **C**, then d is in either **A** or **B**. Let e be the inverted d, i.e., d with its colors inverted . Then, e is in both **--A c --B** and, by the inverse definition, in **--C.** Hence,

--C = **--A c --B**.

CALCULATING ADMISSIBLE COLOR-PATHS.

The following example illustrates how the addition and inverse operators are used to find all the admissible color-paths for a given rotatable arm C with two smaller rotatable arms A and B attached to it. There are several cases (see Figures 5, 6, 7 and 8) we need to consider. The first case is the configuration of Figure 5.

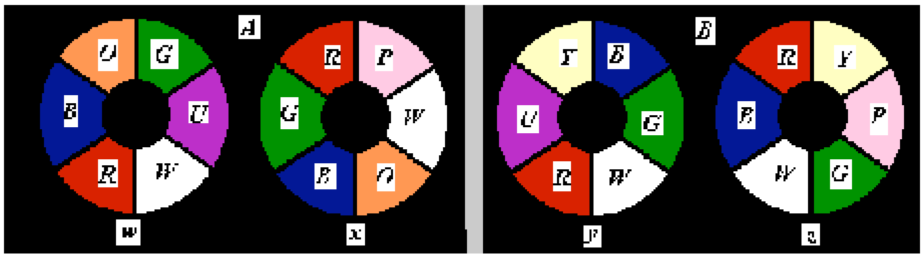

Figure 5. Arm C.

The rotatable arm A is made-up of discs w and x (w is the left disc and x the right one) and rotatable arm B is made-up of discs y and z.

We have, **w** = [GR, UB, OW, RG, BU, WO] and **x** = [OR, GW, BP, RO, WG, PB]. Then by Theorem 1, the admissible color- paths on arm A is given by

A = **w** + **x** = [GRRO, UBBP, OWWG, RGGW, WOOR], and by Theorem 2,

--**A** = **x** + **w** = [ORRG, PBBU, GWWO, WGGR, ROOW].

And similarly let y = [BR, UG, YW, RB, GU, WY] and z = [GR, YW, PB, RG, WY, BP]. Then

B = **y** + **z** = [BRRG, UGGR, YWWY, RBBP, WYYW] and

--**B** = **z** + **y** = [GRRB, RGGU, YWWY, PBBR, WYYW].

Thus by Theorem 1, the admissible color- paths on arm C of Figure 5 is

A + **B** = [WOORRBBP, RGGWWYYW].

We will use the color-path sets **A**, --**A**, **B**, and --**B** to compute other color-path sets that are associated with the different configurations (Figures 6, 7 and 8).

The admissible color-paths for arm C after rotating arm B (see Figure 6) is given by

A + --**B** = [UBBPPBBR, OWWGGRRB, WOORRGGU, RGGWWYYW],

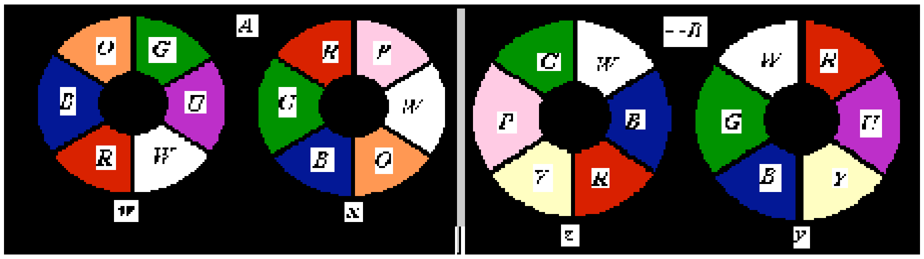

Figure 6. Arm C with B-arm rotated

and after rotating arm A, with arm B in its original position, the admissible color-paths for arm C (see Figure 7) is then given by

--**A** + **B** = [PBBUUGGR, WGGRRBBP, ROOWWYYW],

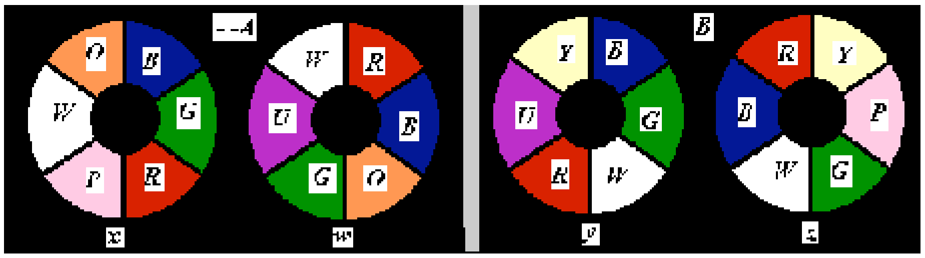

Figure 7. Arm C with A-arm rotated

and

-- **A** + -- **B** = [ORRGGRRB, WGGRRGGU, ROOWWYYW] gives the admissible color-paths for arm C when both arms A and B are rotated (see Figure 8).

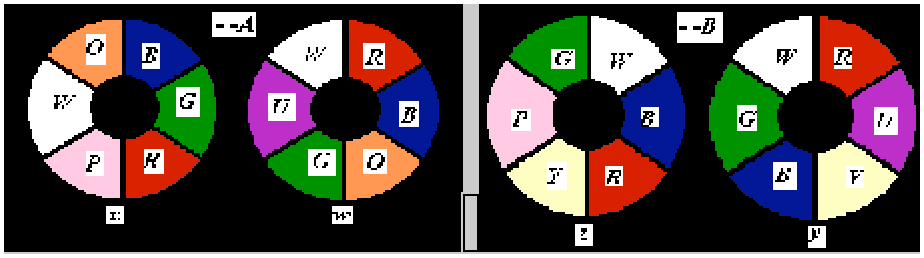

Figure 8. Arm C with A and B arms rotated

Let **C** be a color-path set. Now use the union operator, **c**, to combine all the above color-path sets into one color-path set, namely,

(3) **C** = (**A** + **B**) **c**(**A** + --**B**)**c**(--**A** + **B**)**c**(--**A** + --**B**).

Then by Theorem 1, **C** contains all the admissible color-paths that exists on arm C.

C contains all the admissible color-paths of arm C, but the color-paths contain no direct informationon how the rotatable arms A and B are oriented. For an arm, any color-path of that arm's color-path set cannot be used to determine how the arms attached to it are configured.

Now rotate arm C. The rotation puts arm B to the left of arm A. From Theorems 1 and 3, the color-path sets,

B + **A**,

--**B** + **A**,

B + --**A**, and

--**B** + --**A**,

contain all the admissible color-paths that exists on C's rotated arm. Now combine these sets into a single color-path set **E,** where

E = (B + A)**c**(--B + A)**c**(B + --A)**c**(--B + --A).

E contains all the admissible color-paths that run along C's rotated arm.

We could use Theorems 4 and 5 on equation (3) to find E, viz., **E** = --**C**.

SOLUTION.

The SPECTRA puzzle with its different arms labeled is shown in Figure 9. Referring to that figure, we note that arm G is made up of arms C and D; arm H is made up of arms E and F; and arm M is made up of arms G and H. The arms A, B, and M are attached to the puzzle's base. (Arms A and B in figure 9 are not the same arms in Figure 5.) All the arms are rotatable.

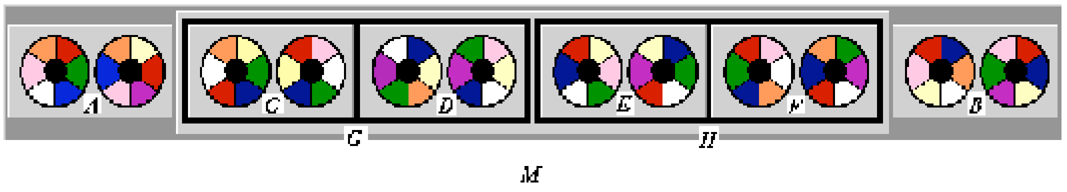

Figure 9. SPECTRA PUZZLE with arms labeled.

Let **A**, **B**, **C**, **D**, **E**, and **F** be arms A, B, C, D, E and F color-path sets, respectively.

For the calculations below, we will assume that the color-path sets **A**, **B**, **C**, **D**, **E**, and **F** are known. They are easily calculated because their corresponding arms contain only two discs.

Let **G**, **H** and **M** be the color-path sets for arms G, H and M, respectively. Then applying Theorems 1 and 3 to arms G and H, we find

$$(6)\qquad \begin{aligned}\mathbf{G} &= (\mathbf{C}+\mathbf{D})\mathbf{c}(\mathbf{C}+--\mathbf{D})\mathbf{c}(--\mathbf{C}+\mathbf{D})\mathbf{c}(--\mathbf{C}+--\mathbf{D}),\text{ and}\\ \mathbf{H} &= (\mathbf{E}+\mathbf{F})\mathbf{c}(\mathbf{E}+--\mathbf{F})\mathbf{c}(--\mathbf{E}+\mathbf{F})\mathbf{c}(--\mathbf{E}+--\mathbf{F}),\end{aligned}$$

and, since arm M is comprised of arms G and H,

$$(7)\qquad \mathbf{M} = (\mathbf{G}+\mathbf{H})\mathbf{c}(\mathbf{G}+--\mathbf{H})\mathbf{c}(--\mathbf{G}+\mathbf{H})\mathbf{c}(--\mathbf{G}+--\mathbf{H}).$$

M contains all the admissible color-paths that run along arm M.

We can now compute all the solutions of the SPECTRA puzzle from the three color-path sets, **A**, **B** and **M**. Applying Theorems 1, 2 and 3 to arms A, M and B, we find

$$(8)\qquad \begin{aligned}\mathbf{S} = {}&(\mathbf{A}+\mathbf{M}+\mathbf{B})\,\mathbf{c}(\mathbf{A}+\mathbf{M}+--\mathbf{B})\mathbf{c}\,(\mathbf{A}+--\mathbf{M}+\mathbf{B})\,\mathbf{c}(\mathbf{A}+--\mathbf{M}+--\mathbf{B})\mathbf{c}\\ &(--\mathbf{A}+\mathbf{M}+\mathbf{B})\,\mathbf{c}(--\mathbf{A}+\mathbf{M}+–\mathbf{B})\mathbf{c}\,(--\mathbf{A}+--\mathbf{M}+\mathbf{B})\,\mathbf{c}(--\mathbf{A}+--\mathbf{M}+--\mathbf{B}),\end{aligned}$$

where, **S** contains all the partial and complete solutions to the SPECTRA puzzle, i.e, all the admissible color-paths that run along the length of the puzzle's base. Recall that complete solutions are the ones whose color-paths have equal beginning and ending colors, and partial solutions do not.

Even though (8) solves the puzzle, using it to get the actual solutions is cumbersome. We require help in the form of a C++ computer program to compute the solutions efficiently. To simplify the C++ implementation of (8), we define a C++ SPECTRA class. This SPECTRA class implements color-path sets as an abstract data type that supports the addition, inverse and union set operations.

SPECTRA CLASS.

The solution of the SPECTRA puzzle, as given by (8), can be easily implemented as a C++ program given a SPECTRA class. This approach to implementing the solution is a natural approach to pursue. The SPECTRA class implements color-path sets as an abstract data type that supports the addition, inverse and union set operations. A C++ variable declared as a SPECTRA type will be a color-path set. The C++ operators "+", "--", and "*" will function as the addition, inverse and union color-path set operators. We have successfully implemented our solution using this approach. We provide below the SPECTRA class we used to implement our solution and a small description of each of the class methods.

```
struct ColorNode
{
  char Colors[26];
  ColorNode *NextNode;
};

typedef ColorNode* ColorPath;
```

Color-Path Node Definition

Color-path set becomes a linked list in our SPECTRA class. Each color-path in the color-path set is a node in that linked list. The definition of the color-path node is given in the shaded box above. The node consists of a pointer (*NextNode) to the next color-path node and an array (Colors) of the SPECTRA color letters representing the color-path. The linked list - ColorPath - is in the private section of the SPECTRA class.

The SPECTRA class is sufficient to solve the SPECTRA puzzle. Later we will add other methods which will determine the number of partial and complete solutions the puzzle has, and the number of solutions containing six, seven and eight colors.

```
class SPECTRA
{
 private:

 ColorPath ColorPathSet; // Color list. Color path is a
                         // linked list.
 public:
  SPECTRA();  // default constructor.
  SPECTRA(ColorPath);  // default constructor.

  void Add(char ColorList[]);// insert a color-path into
                             // SPECTRA's linked list
  void Display(); // Print all the color-paths in the
                  // SPECTRA's linked list
  // Add color-paths
  SPECTRA SPECTRA::operator+(const SPECTRA a);
  SPECTRA SPECTRA::operator--(); // Reverse color-paths
  // Union operator
```

Class definition for SPECTRA

The following descriptions of the SPECTRA methods are sufficient for anyone who wants to implement the SPECTRA class.

Description of the SPECTRA Methods. All methods use the class variable ColorPath, which is the linked list that represents the color-path set. A linked list node contains one color-path.

The constructors **SPECTRA()** and **SPECTRA(ColorPath a)** initialize the linked list to NULL and to the color-path set **a**, respectively. The color-path set **a** should be duplicated and the duplicated copy should be assigned to the class variable ColorPathSet.

The method **Add** takes for input **ColorList**, which is a color-path, and inserts it as the first element in **ColorPathSet**. **ColorList** is a string of wedge color letters.

The method **Display** prints all the color-paths in the linked list **ColorPathSet**. Each color-path is inclosed in brackets ([]).

The addition method ("**+**") corresponds to the color-path set's addition operator. The color-path set addition operator "**+**" extends the C++ addition operator. The method takes two SPECTRA variables as input and adds their linked lists (ColorPathSets) together. Their sum becomes a linked list in a new SPECTRA type variable. The new variable is returned. The input variables are not modified.

The inverse method ("**--**") corresponds to the color-path set's inverse operator. The color-path set inverse operator "**--**" extends the C++ decrement operator. The inverse method takes as input one SPECTRA variable and reverses the color-path's color letters for each color-path node in the linked list. The inverse becomes a linked list in a new SPECTRA variable. The new variable is returned. The input variable is not modified.

The union method ("*") corresponds to the color-path union set operator - **c**. The color-path set operator "**c**" extends the C++ multiplication operator "*". The method takes as input two SPECTRA variables and combines their linked lists. The union method simply concatenates the two input SPECTRA variable's linked lists. The resulting linked list becomes the linked list in a new SPECTRA variable and that variable is returned. The input variables are not modified.

SPECTRA PROGRAM.

In this section, we translate the SPECTRA puzzle solution given by equation (8) into a C++ program using the SPECTRA class. The program's logic parallels the derivation steps we used to derive (8). Following the derivation steps leading to equation (8), and using Figure 9's labeling, we find the following program's solution algorithm:

```
(1) Initialize the SPECTRA variables A, B, C, D, E, and F to the arms A,
    B, C, D, E and F color-path sets, respectively.
(2) Compute all the color-paths that run along arms G and H (equation (6)).
(3) Compute all the color-paths that runs along arm M (equation (7)).
(4) Compute all the complete and partial solutions of the SPECTRA puzzle
    (equation (8)).
(5) Print the SPECTRA puzzle's complete and partial solutions.
```

SPECTRA Program Solution Algorithm

Converting the above solution algorithm into a C++ program yields the following C++ program.

```
void main ()
{
   SPECTRA   A, B, C, D, E, F, G, H, M, S;
    // We need to initialize A, B, C, D, E and F to the
    // admissible color-paths for each rotatable arm,
    // respectively. We give the initialization for C
and
    // the other initializations are not shown.
   C.Add("WGGR");
   C.Add("RYYW");
   C.Add("GWWY");
   C.Add("YRRG");
   C.Add("OUUP");

    // compute all admissible color-paths for arm G
   G = ( C + D )*( C + --D )*(--C + D)*(--C + --D);
    // compute all admissible color-paths for arm H
   H = ( E + F )*( E + --F )*(--E + F)*(--E + --F);
    // Now, compute all admissible color-paths for arm M
   M = ( G + H )*( G + --H )*(--G + H)*(--G + --H);
    // Finally, compute all the solutions, i.e,
    // admissible color-paths from one end of the
    // SPECTRA's base to the other end, of the puzzle.
   S = ( A + M + B )*( A + M + --B )*(A + --M + B)*
       (A + --M + --B)*( --A + M + B )*( --A + M + --B )*
       (--A + --M + B)*(--A + --M + --B);
   S.Display() // Print all the solutions
}
```

SPECTRA Program

The above SPECTRA program outputs 85 solutions. The partial solution of six colors, [WRRBBRRGGWWOOWWGGRRYYBBG]; the two complete solutions, [URRWWYYWWGGRRYYWWGGBBOOU] and [OPPYYWWGGBBPPBBUUGGRRBBO], containing seven colors and eight colors, respectively, are three of the 85 solutions.

Extension. We can extend the SPECTRA class by adding to it the methods listed in the shaded box below. These additional methods compute the number of partial and complete solutions, the number of complete solutions, the number of solutions in which 6-colors are used, and the number of solutions in which 7 and 8 colors are used.

```
// The following methods are used to count the complete
// and partial solutions, to count the number of solutions
// that uses 6-colors, 7-colors, and 8-colors.
int Count(); // return the number of complete and partial
             // solutions (nodes)
int BEcount(); // return the number of complete solutions
int EightColorsUsed(); // return the number of solutions
                       // (nodes) that uses 8 colors
int SevenColorsUsed(); // return the number of solutions
                       // (nodes) that uses 7 colors
int SixColorsUsed();   // return the number of solutions
                       // (nodes) that uses 6 colors
```

New Spectra Class methods

After adding these new methods to the SPECTRA class, place the following code

```
cout << “\nThere are “ << S.count()
     << “ solutions, of which “ << S.BEcount()
     << “ are complete solutions. Additionally, “
     << “there are \n”
     << S.SixColorsUsed()
     << “ solutions using 6 colors, “
     << S.SevenColorsUsed()
     << “ solutions using 7 colors, and “
     << EightColorsUsed()
     << “\n solutions using 8 colors”;
```

into the SPECTRA program after the S.Display() instruction. The above code results in the following output:

> *There are 85 solutions, of which 12 are complete solutions. Additionally, there are 16 solutions using 6 colors, 52 solutions using 7 colors, and 17 solutions using 8 colors.*

ACKNOWLEDGMENT. Many thanks to Ms. Martiza Nall for her generous help in creating the figures shown here.

REFERENCES

1. Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1989.
2. Robert L. Lamphere, “Spinout Puzzle and Recursion”, The Journal of Computing in Small Colleges, Vol. 15, n.2, (January 2000) 232 - 241.
3. Robert L. Lamphere, “A Recurrence Relation in the Spinout Puzzle”, The College Mathematics Journal, 27(1996) 286 - 289.