OZOBOT BASIC TRAINING LESSON 3

HOW MANY CODES CAN OZOBOT READ?

What students will learn

- What are the guidelines for using static and flash codes?
- How many combinations are there for static codes with 2, 3 and 4 colors?
- Find a solution for an example of the "Traveling Salesman Problem"

Topics

- Math: logic
- Math: multiplication and addition
- Math: combinatorics, graph theory and optimization
- Robotics: communicating with the bot
- Computer science: visual coding

Maze challenge

Traveling Salesman Problem.

Real-life connection

Examples of applications of combinatorics, graph theory and optimization.

Age

Grades 3-10

Ozobot skill level

Beginner

STEM topics

- Advanced math concepts: combinatorics, graph theory and optimization
- Computer science: use visual codes to program the robot
- Inter-disciplinary concept: use programming and robotics to find a math problem solution
- Connection to real-life applications

Common Core Standards

CCSS.MATH.PRACTICE.MP1 Make sense of problems and persevere in solving them.
CCSS.MATH.PRACTICE.MP2 Reason abstractly and quantitatively.
CCSS.MATH.PRACTICE.MP4 Model with mathematics.
CCSS.MATH.PRACTICE.MP5 Use appropriate tools strategically.
CCSS.MATH.PRACTICE.MP6 Attend to precision.
CCSS.MATH.PRACTICE.MP7 Look for and make use of structure.

CCSS.MATH.CONTENT.3.OA.A.1 Represent and solve problems involving multiplication and division.
CCSS.MATH.CONTENT.3.OA.D.8 Solve problems involving the four operations, and identify and explain patterns in arithmetic.
CCSS.MATH.CONTENT.3.MD.D.8 Geometric measurement: recognize perimeter.
CCSS.MATH.CONTENT.4.OA.A.3 Use the four operations with whole numbers to solve problems.
CCSS.MATH.CONTENT.5.OA.A.1 Write and interpret numerical expressions.
CCSS.MATH.CONTENT.6.EE.A.2 Write, read, and evaluate expressions in which letters stand for numbers.
CCSS.MATH.CONTENT.7.EE.B.3 Solve real-life and mathematical problems using numerical and algebraic expressions and equations.

High School: Modeling

Materials

- Ozobots (about 1 per group of 3 students), charged
- Digital tablet (iOS or Android operating system), charged and screen brightness set to 100% (one tablet for each group of students)
- Ozobot app (download for free on iOS app store or google play, go to settings and select: "keep tablet awake")
- Markers in colors black, red, light blue and light green (we recommend you use Ozobot Markers. Alternatively, choose Sharpie's wide chisel tip or Crayola classic markers), one set per group
- Printouts #1-3 and #7. If you are doing the optional chapter 4, you will also need printouts #4 and 5. If you are doing the optional chapter 5, you will also need printout #6. If you are using the worksheet to guide students through the maze exercise, you will also need printout #8.
- Printout of OzoCodes reference chart (www.ozobot.com/gamezone/colorlanguage), one per group

- Optional: printout of lesson PDF, one per group if students are learning self-guided

Estimated duration

1 or several class sessions, depending on the age of students and inclusions of the optional chapters.

LESSON

1. Calibration on the tablet

Just like on paper, Ozobot can follow lines and read codes on a tablet as well. There are some differences though. Ozobot does not need to turn on the lights underneath since there is so much light coming from the tablet already. Make sure that the brightness of your tablet is set to as bright as possible so Ozobot can see well enough.

Also, the calibration routine is different. On your tablet, do the following:

Open the Ozobot app, go to home if you are not there already, and press the "Ozobot Tuneup" button. Make sure the screen brightness is set to 100%. On the Tune Up page, press the "Calibrate Sensors" button. Follow the 3 steps shown on the bottom of the screen.

You will have to calibrate this way every time you start playing on a tablet. This also means that you will have to calibrate every time you switch from playing on paper to playing on a tablet. And the reverse is also true: you will have to calibrate on paper (see lesson 1 on how to do that) every time you switch from playing on a tablet to playing on paper.

If your Ozobot still behaves strangely, try moving away from any bright lights. Ozobot's sensors are very sensitive and too much light from the surroundings will confuse Ozobot.

Ok, now we are ready to go!

2. Let's use codes, all of them!

Ozobot can read a lot of codes, on paper and on tablets. There are two different kinds of codes that Ozobot can read:

1. Flash codes: these codes only work on a digital device and they are the ones used in the Ozobot app in the OzoDraw Challenge. These codes are the round ones and flash a sequence of colors very quickly. Ozobot stops, reads them and executes the commands.

2. Static codes: these work on paper and on a digital devices and are built using a sequence of short color segments.

Let's look at the static codes some more. Open the Ozobot app, go to home if you are not there already, and go to OzoDraw Freedraw. The codes that can be used are all in the bar on the bottom right of the screen. When you first get to this screen, you can see all the codes that control Ozobot's speed. There are a lot of them, and you can scroll the window with the codes to see all of them. If you want to see other codes, tap the speed button and another category comes up. The codes in the bar will change and you can look through them. There are 5 different categories. Look through all of them to see what kind of codes Ozobot can read.

Now let's try to use them. We have to make sure Ozobot can read the codes, so there are a few guidelines that we have to follow. Take a look at handout #1, it's a cheat sheet on how to use static codes. Let's go through it:

1. Ozobot can only understand static codes if they are placed on a black line, so draw some lines in black.

2. Drag codes from the code bar onto a black line. Tap once to toggle between flash and static codes. We want to use static codes for now. Make sure to align the codes with the line, otherwise Ozobot won't be able to read them. If you need to reposition them, just drag the codes to the new location. You might also have to rotate them, and rotation can be a bit tricky. There are two ways to do it:
   1) Place your thumb onto the code and your index finger next to it. Keep your thumb in place and move your index finger around the code to rotate.
   2) Place index and middle finger on either side of the code. Move both fingers in a circular motion around the center of the code to rotate.

3. Don't place codes too close to intersections or curves. If Ozobot has trouble reading a code, try dragging it away from intersections or curves.

4+5. Some of the codes are to be used on line-ends (2-color codes), and the rest need a black line before and after the code to work properly.

6. For some codes, it matters if Ozobot reads them from left to right, or from right to left. Try out the snail dose/ nitro boost code for example.

7. If you want to use flash codes instead of static ones, tap on the code once. Tap another time and it's a static code again.

8. If you don't need a code anymore, drag that code off screen and it disappears.

With these guidelines in mind, try out some of the codes in OzoDraw!

3. How many 2-color static codes can Ozobot read?

You can see that there are different color codes: some are short and have only 2 colors, most of them have 3 colors, but there are also 4-color codes. How many codes can Ozobot read?

Let's start with a 2-color code. What colors are available for us to use? For codes, Ozobot is programmed to read the colors Red (R), Green (G), Blue (B) and Black (X). Now, since static codes are meant to be used on a black line, we cannot use black as the first or last color in a code, otherwise Ozobot would not see any difference between the line and the black color code at the beginning or end of a code. So, black is out for 2-color codes. That leaves us with R, G and B.

That means, for the first color in the code, we have 3 choices: R, G and B. See handout #2. The three options are there, place Ozobot on the line on top and let Ozobot decide which color to take as the first code color! But wait! Your Ozobot has been calibrated for play on a digital screen and now you will use Ozobot on paper. So you will have to calibrate on paper before placing Ozobot on the line. Use the calibration dot on the handout to do so.

Note: If you are unsure of how to calibrate Ozobot on paper, please take a look at the "Ozobot Tips Sheet" available for download from the Ozobot STREAM website. Or review Ozobot Basic Training Lesson 1.

Did Ozobot choose red?

If so, then what color can the second color be? It cannot be red as well. Ozobot would not detect a color change and would think it was just red once. This means that, if we have red as the first color, then the second color can only be green or blue.

Did Ozobot choose green?

If so, then what color can the second color be? It cannot be green as well. Ozobot would not detect a color change and would think it was just green once. This means that, if we have green as the first color, then the second color can only be red or blue.

Did Ozobot choose blue?

If so, then what color can the second color be? It cannot be blue as well. Ozobot would not detect a color change and would think it was just blue once. This means that, if we have blue as the first color, then the second color can only be red or green.

Take a look at handout #3. Color in the options for the second color for each of the choices for first color.

Not let Ozobot choose a 2-color combination. What did Ozobot choose? Repeat this a couple of times.

How many choices are there total? We can write down all the options we have:
R G
R B
G R
G B
B R
B G
6 options total! But we didn't have to write them all down. We know we have 3 options for the first color and 2 options for each first color choice. That makes 3*2 = 6 total choices!

Relation to lesson 2: what are the probabilities of choosing each of the codes? (answer 1/6)

4. How many 3-color static codes can Ozobot read?

This chapter is optional. If skipped, just know that there are 21 choices for 3-color codes and 60 choices for 4-color codes and move on to chapter 6.

There are certainly more options for 3-color codes than there are for 2-color ones, but how many of them exactly?
The options for the first color are R, G or B, just like with 2-color codes. Nothing changes for the first color, so you can take handout #2 again and let Ozobot decide.

The second color choice is different from before though. We have another option now: black. Since the second color does not have any contact with the black line we are placing the code on, we can use it now. And no matter what color Ozobot chose first, we can always use black as the second color since it is different from the first.

Take handout #4 and color in all the choices for the second color. Let Ozobot decide again which two colors to chose.

Let's look at the third color. We cannot use black, since it is the last color of the code. So we are left with R, G and B again. Let's assume for a moment that Ozobot picked R as the second color. Then, similarly as before, G and B are our only choices for the third color. Notice that the first color choice doesn't matter when we are picking the third color. They can be the same color and, since they are not next to each other, Ozobot will be able to read them both just fine.

But what happens if the second color was black? In that case, we can actually choose either one of the 3 colors R, G and B. This is another big difference from 2-color codes.

Using handout #5, fill in all the options for the third color. We now have all combinations!

We are ready to answer the big question now: how many 3-color codes can we assemble? We probably don't want to write them out since there are many of them, so let's try to calculate how many:

- There are 3 first color choices
- For each of these, there are 3 second color choices
- For each second color choice, there are either 2 or 3 choices

Now, just look at red as the first color choice. 2 of the second color choices have 2 options for the 3rd color, and one has three options. In a formula, we can write this as:

$$2*2 + 1*3 = 4 + 3 = 7$$

Since this is the case for all 3 first color choices, the total number of choices is:

$$3 * 7 = 21$$

5. How many 4-color static codes can Ozobot read?

This is optional and may be more suitable for grades 4 and up. If this is skipped, just know that there are 60 choices and continue with chapter 6 below.

There are even more options for 4-color static codes, so we won't even graph all of them. Fortunately, the first and second color options are the same as for 3-color codes.
Use handout #6 to fill in a all choices for red as the first color. Note that black is available not only as the second but also as the third color.

So how many combinations of colors do we have for 4-color codes? Let's calculate the number of options if we choose red as the first color:

- There are (2*2 + 1*3) = 7 options each if we choose green or blue as the second color, and
- There are (3*2) = 6 options if we choose black as the second color, which makes
- A total of (2*7 + 1*6) = 20 options

These are the options if we have red as the first color. Similarly, there are 20 options each for green or blue as the first color. This makes

3*20 = 60

choices for 4-color static codes.

6. How many static codes can Ozobot read?

Let's recap, we have:

6	2-color codes, and
21	3-color codes, and
60	4-color codes

which makes a total of 87 codes!

Not all of the codes are programmed. That means Ozobot will do absolutely nothing after reading those codes. They are reserved for use in the future. You can find all available codes in the OzoCodes reference chart.

Maze Exercise

What we have done so far - looking at how many and what kind of combinations can be formed is part of combinatorics, a branch of mathematics. It is closely related to graph theory, another branch of mathematics, and we are going to use both in the maze exercise.

Take a look at handout #7. This is a graph, hence graph theory. A graph consists of vertices (the edges) and nodes (either a line end or intersection of vertices). This graph represents the "Traveling salesman" exercise, which dates back to 1930 and is one of the most studied problems in graph theory.

You can see 5 different cities on the nodes of the graph. (Note that the map is an abstraction and not geographically correct). A salesman (i.e. Ozobot) has to visit each city once and only once. At the same time, the salesman wants to go on the shortest path. You can see numbers next to some of the vertices. Those indicate the length of that line between the nearest intersections or corners.

Task: Ozobot is the salesman that has to visit all cities on the map. But Ozobot can visit each city once and only once. Also, Ozobot is supposed to take the shortest route to accomplish this. Your task is to figure out in what order Ozobot is supposed to visit the cities. Once you find this out, use directional codes (Go Straight, Go Left, Go Right) to guide Ozobot through the map. You will not have to use all the code slots on the map – just black out the ones you don't need. Finally, put Ozobot on the map and check if he's going on the correct route. Have fun!

Note to teacher: Younger students may need help to complete this task. The following instructions guide them through all the steps they need to take. Depending on their age, you may want to use part or all of the instructions and all, part or none of the worksheet on printout #8.

Let's now try to figure out which way is the best way to go. We first have to find the distances between the cities. Notice that the length numbers do not appear everywhere on the map. So we have to find out the length of each of the edges.

Mark the missing lengths on the handout. Remember to find rectangles and use the fact that opposite sides of rectangles have the same length.

Now we can compute the distances between all of the cities. You can do this on your own or use the top part of the worksheet on handout #8 to pencil in the distances.

Write down the distances between any two neighbor cities (i.e. two cities that can be reached without passing through another city). Don't forget the distance from the start to San Francisco.

We now know the distances between the cities, but what are the routes that Ozobot can take without visiting a city more than once? There are 6 options.

Write down all 6 possible routes. Again, you can do this on your own or use the bottom portion of the worksheet on handout #8. You may want to write this out as a combinational problem with a graph like we did before with the colors of a code. The solution graph can be found on handout #9.

Now we are ready to compute the total distance of each of the options. This will give us the solution, i.e. the shortest route in which Ozobot visits each city only once.

Use the distances between the cities and the 6 different routes to compute how long each route is. You can use the bottom part of the worksheet #8 to pencil in the numbers.

Once all of this is done we can easily figure out that the shortest route is of length 11 and goes as follows:

Start -> San Francisco -> Houston -> Orlando -> Chicago -> Toronto

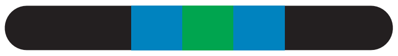
Well done! Now we have to tell Ozobot about this. Use the code slots to fill in directional codes (Go Straight, Go Left, Go Right) to guide Ozobot through the map. If you don't need a code slot, just black it out. Finally, put Ozobot on the map and check if he's going on the correct route.

Where is this used?

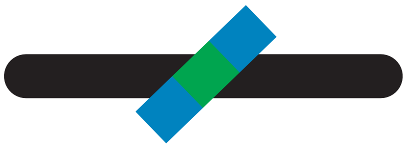
Combinatorics and graph theory is used to solve many different kinds of problems. What does the traveling salesman graph look like? It does look like a map of sorts. Indeed, graph theory can be used in navigation systems to find the shortest way, the fastest way, etc. But there are many other applications: networks of almost any kind (f.e. our electrical grid, traffic system), logistics (how to transport items most efficiently), electronic chip design, chemistry, physics and biology, just to name a few.

HOW TO USE CODES

1. Place static codes on BLACK path. Flash codes work on paths of any color.
2. Align with path



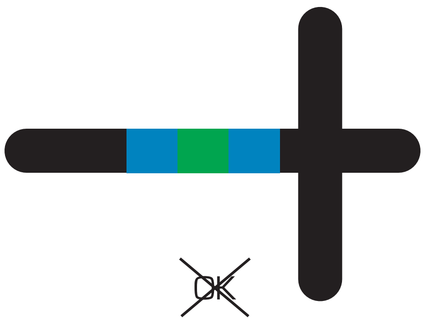
OK



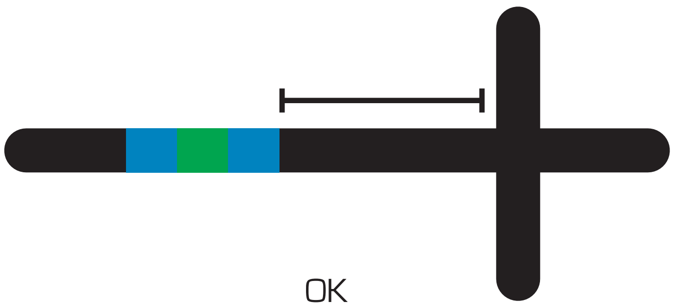
~~OK~~

▸ Rotate until the code is aligned with the path
+ move code by dragging

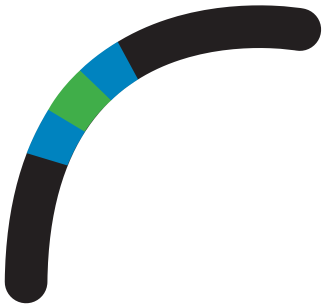
3. Don't place code too close to intersections or curves



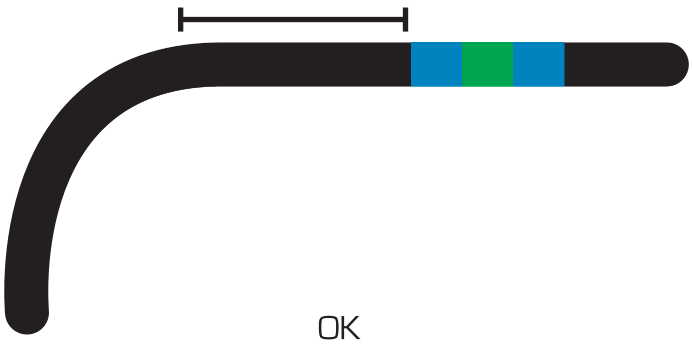
~~OK~~



OK



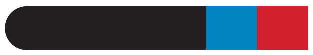
~~OK~~

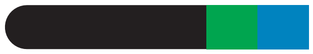


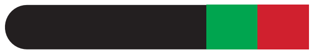
OK

4. Some codes are used on line ends only:

These are the 2-color codes:

U-turn 

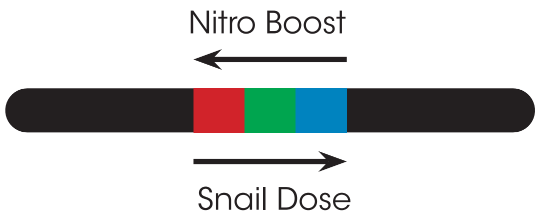
Exit / Win (play again) 

Exit / Win (game over) 

5. All 3- and 4- color codes need a black line before and after

6. Orientation matters for some codes

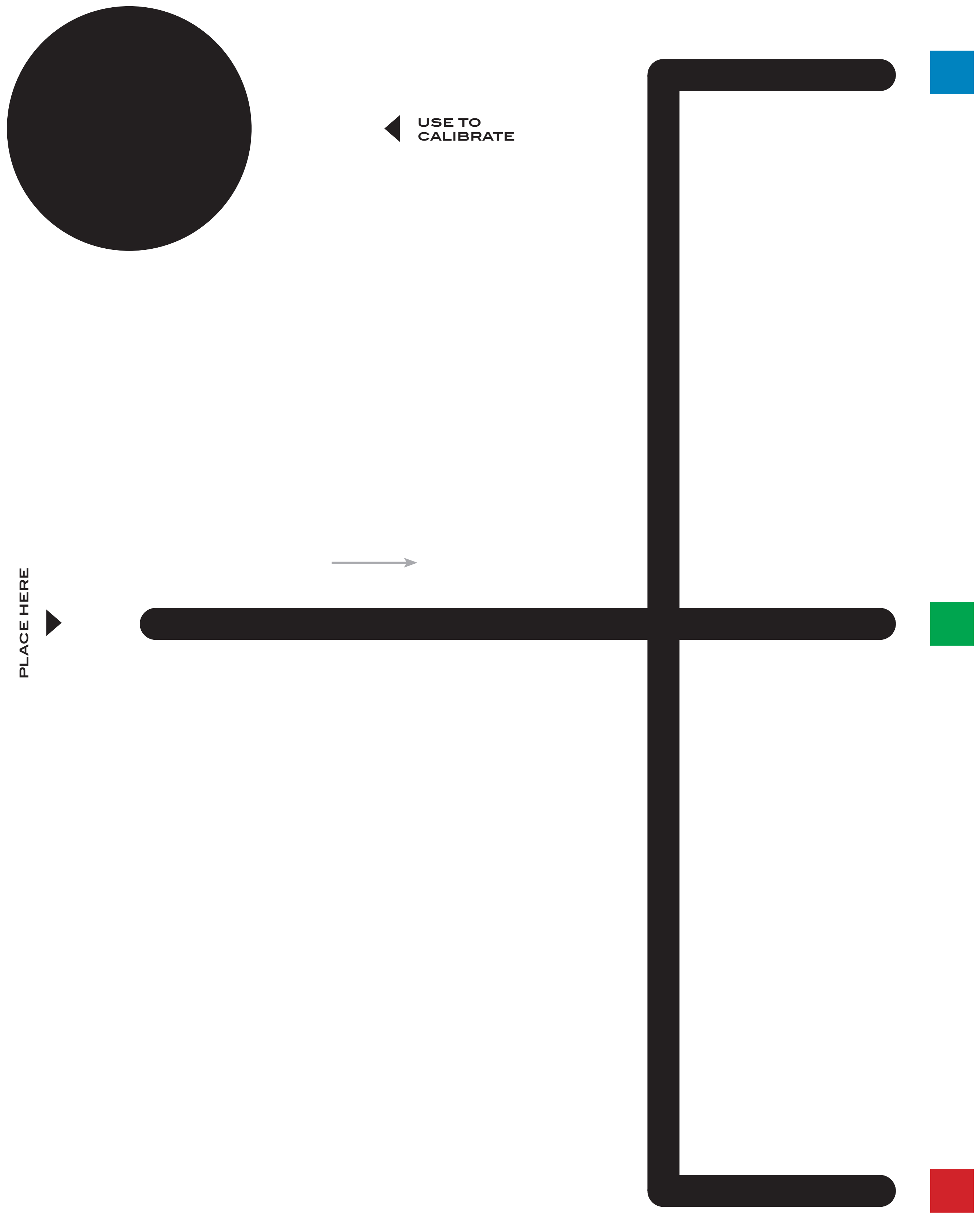
f.e.



7. If you are using the Ozobot app, tap once to toggle between static and flash codes

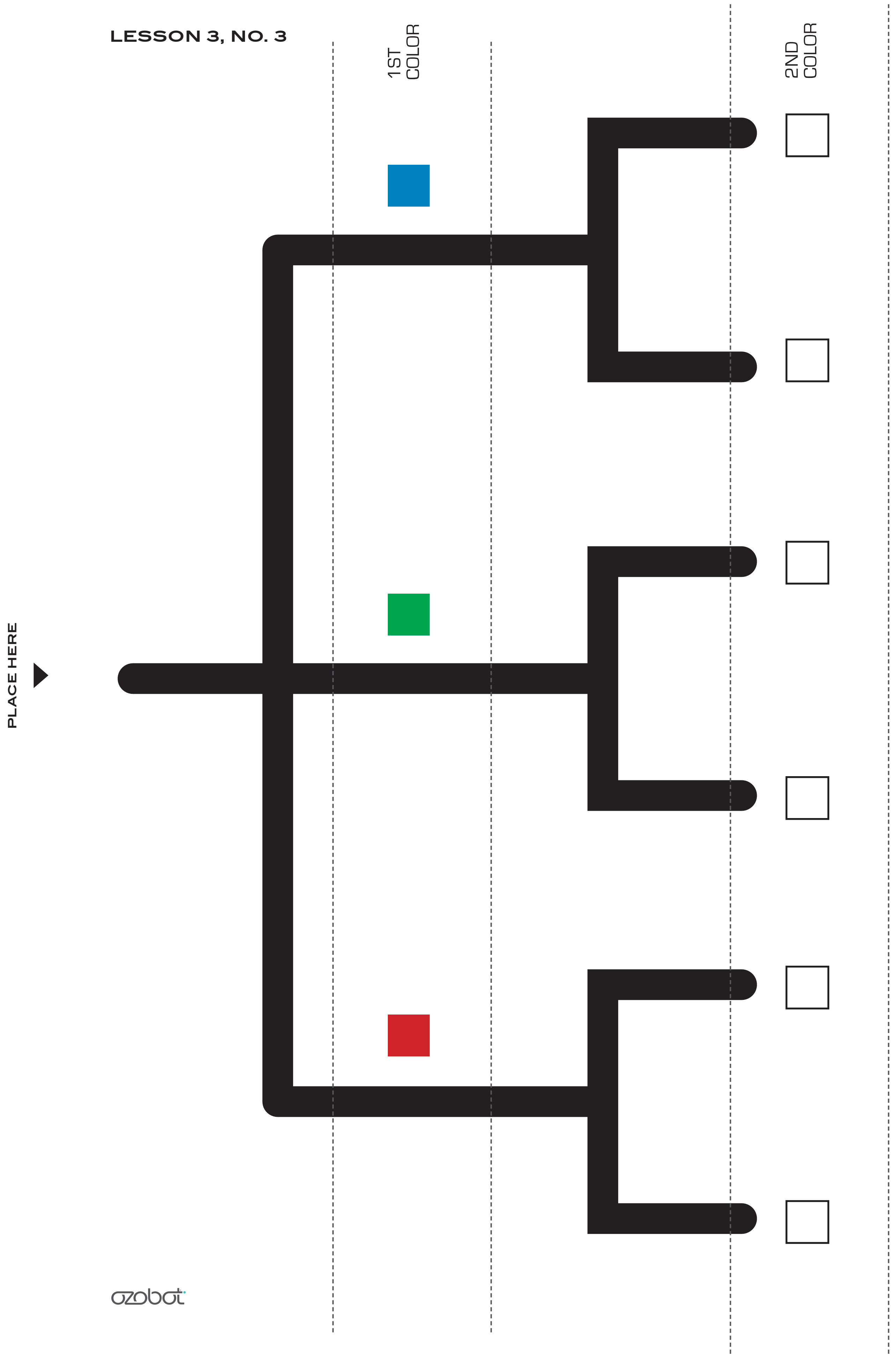
8. If you are using the Ozobot app, drag code off screen to delete.

LESSON 3, NO. 2



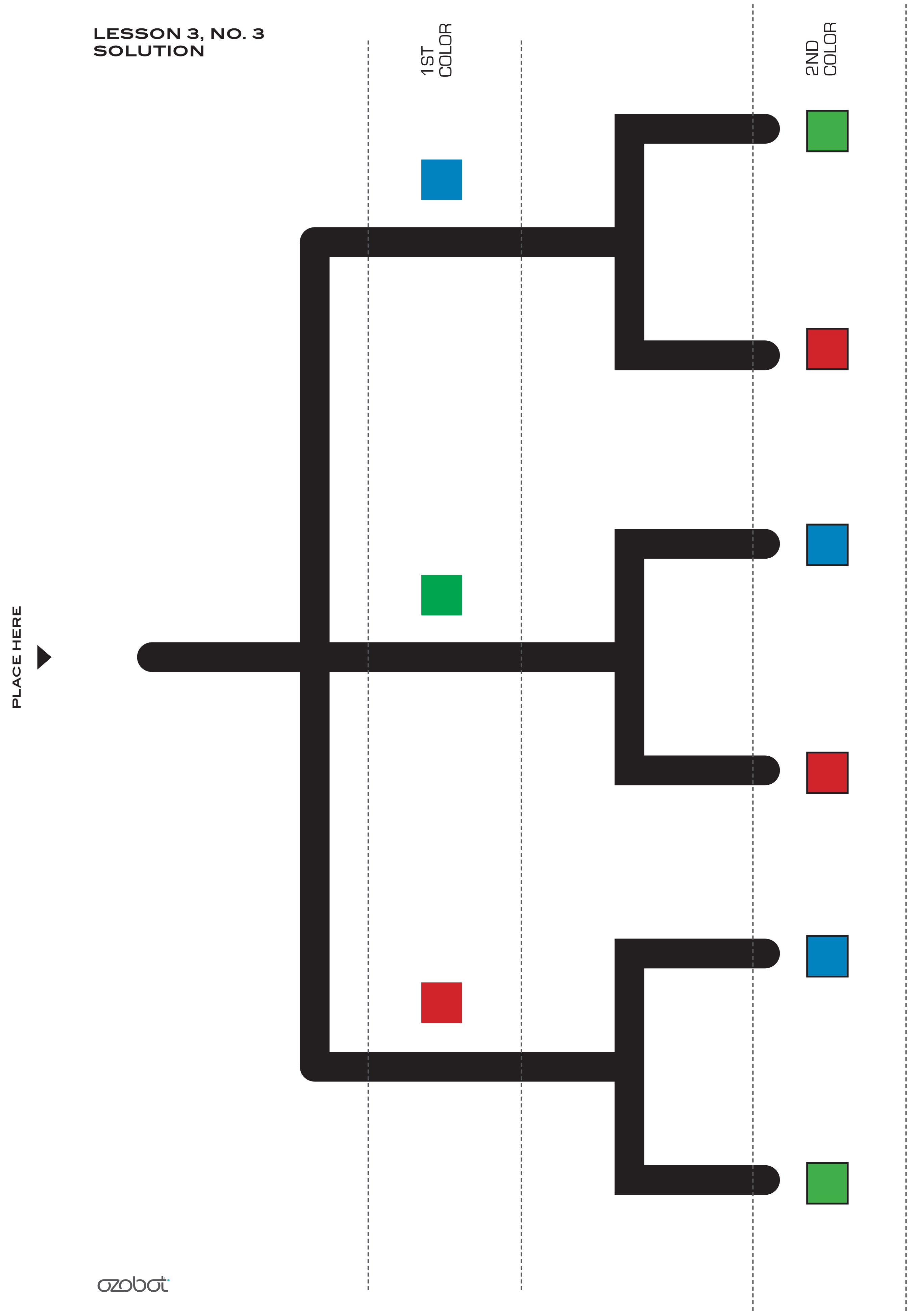
ozobot

LESSON 3, NO. 3



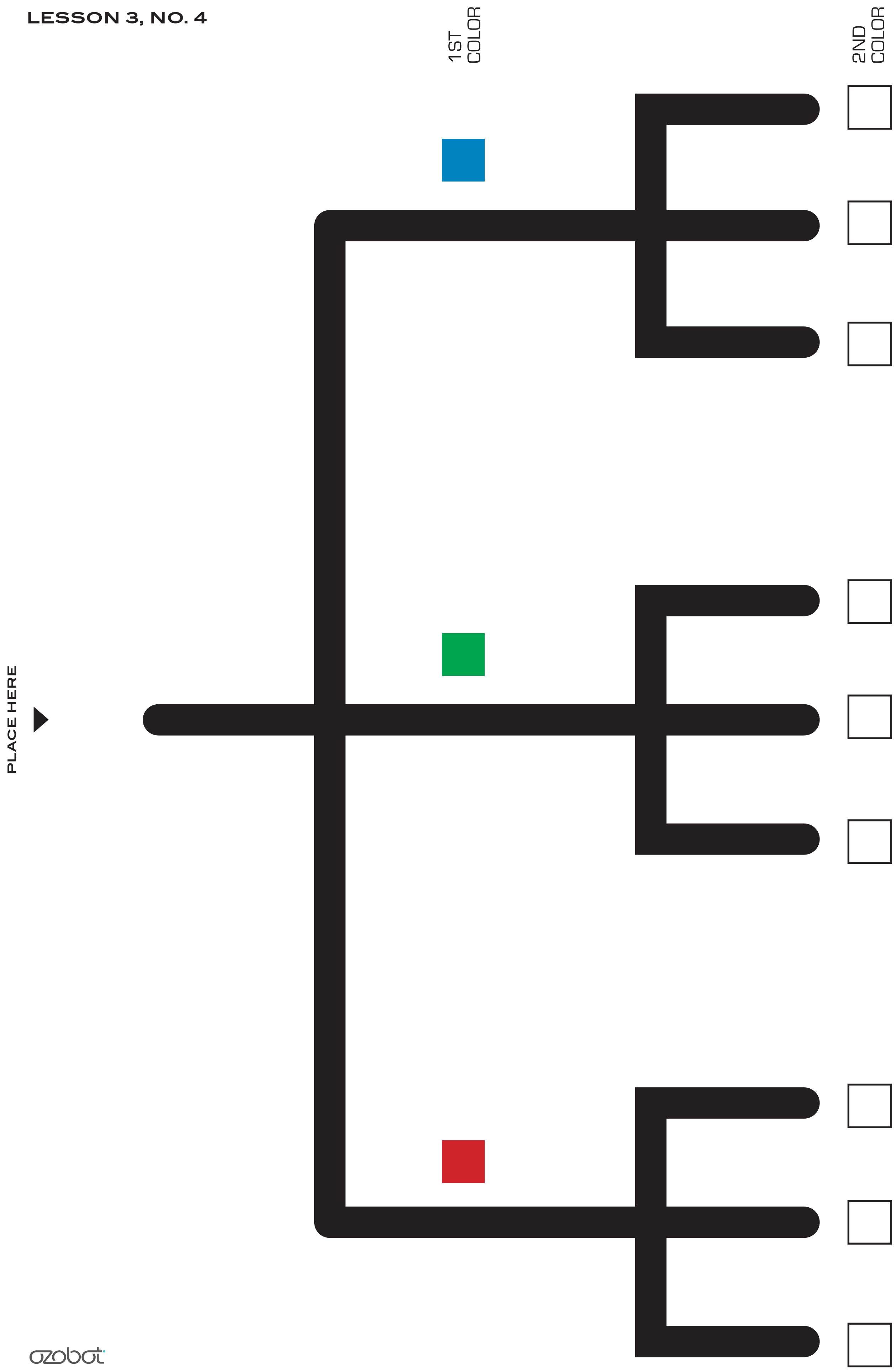
ozobot

LESSON 3, NO. 3
SOLUTION



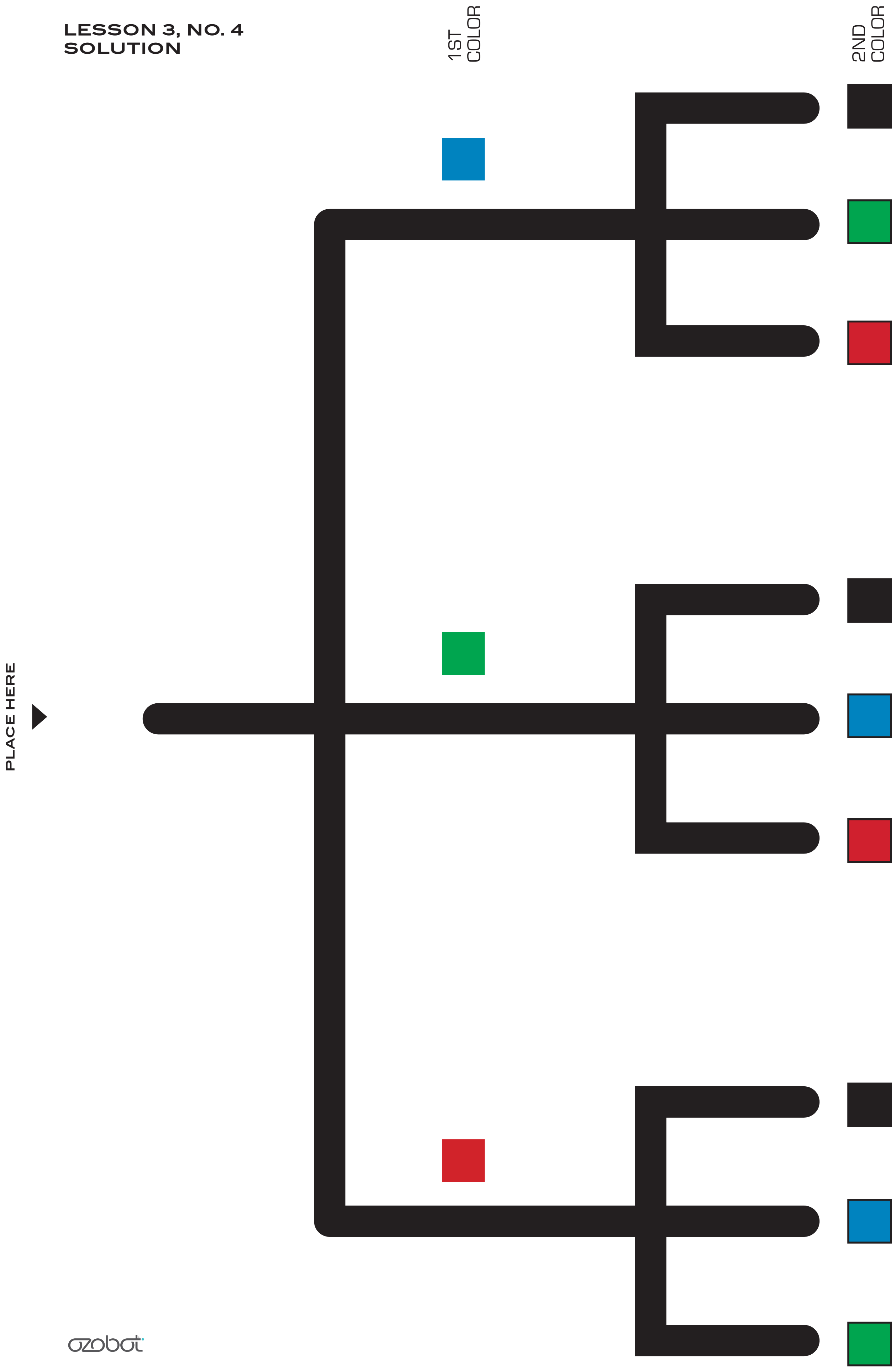
ozobot

LESSON 3, NO. 4



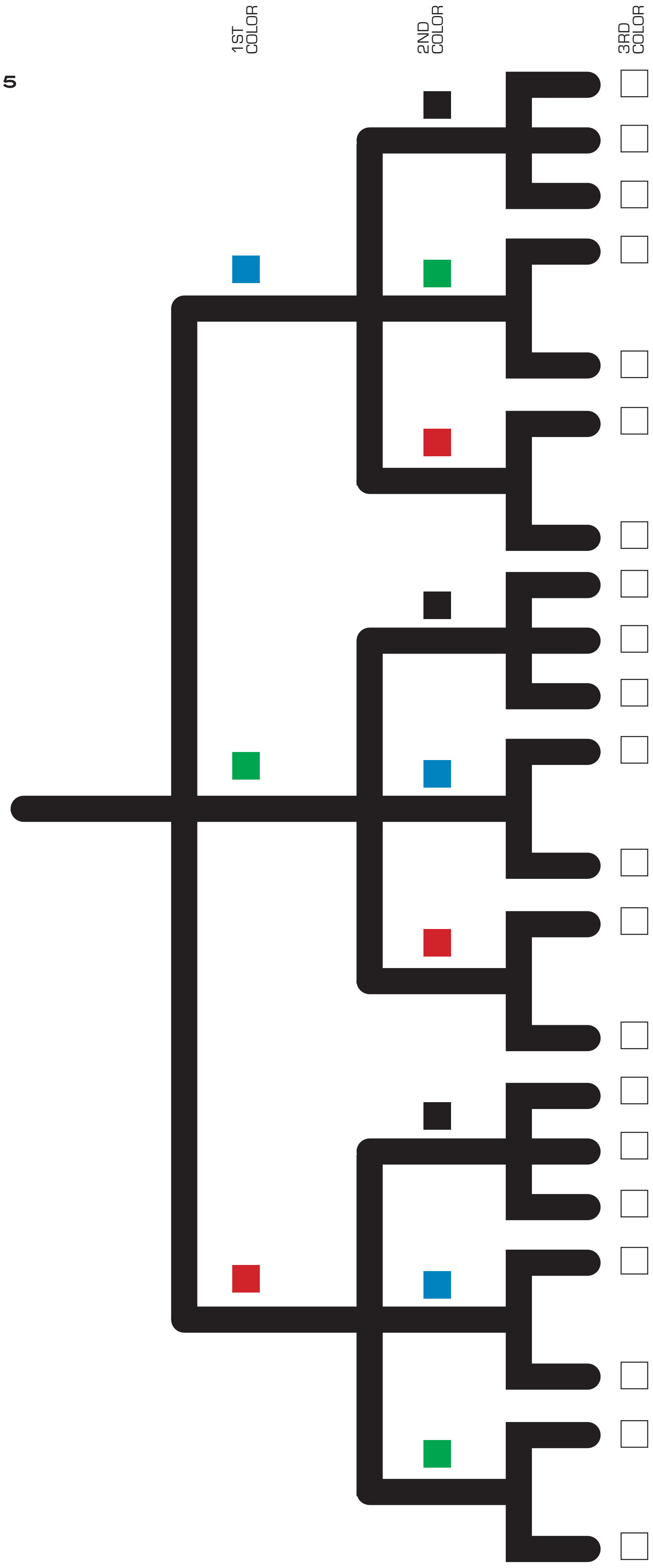
ozobot

LESSON 3, NO. 4
SOLUTION



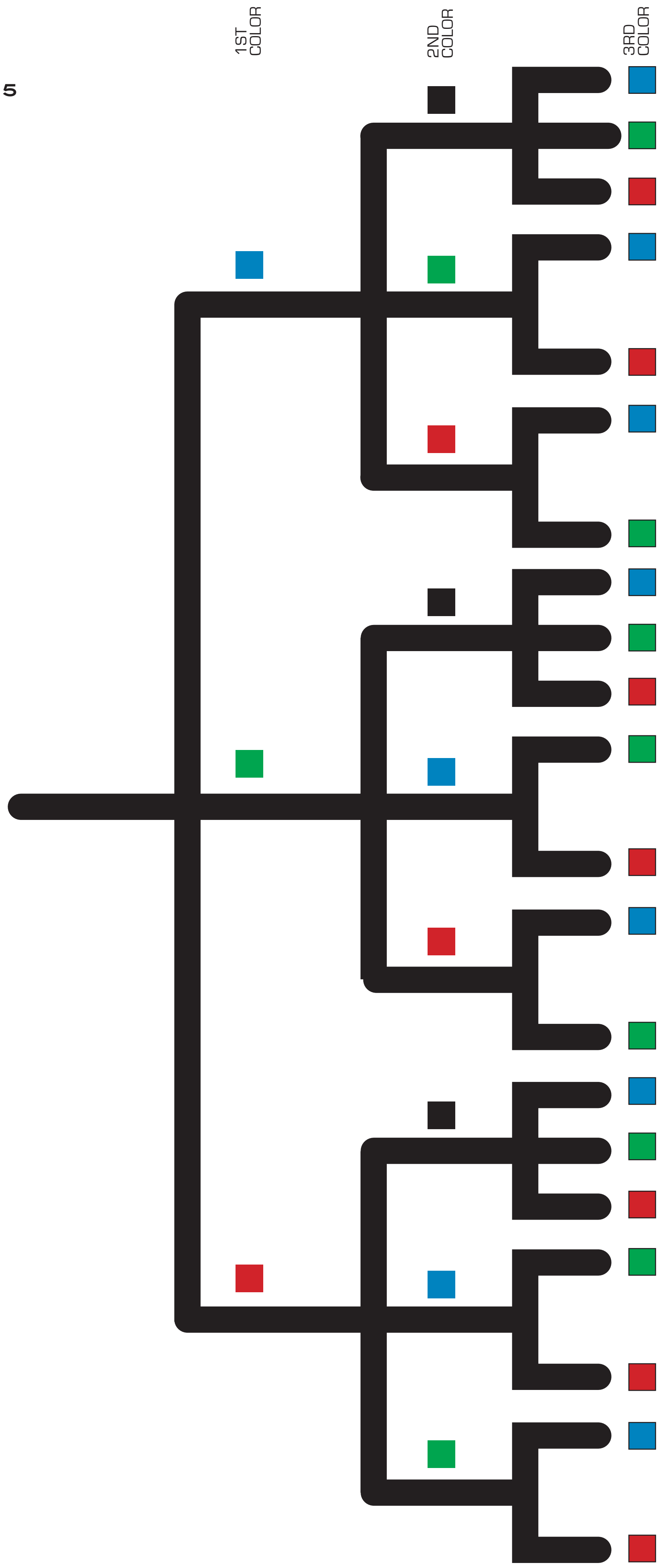
ozobot

LESSON 3, NO. 5



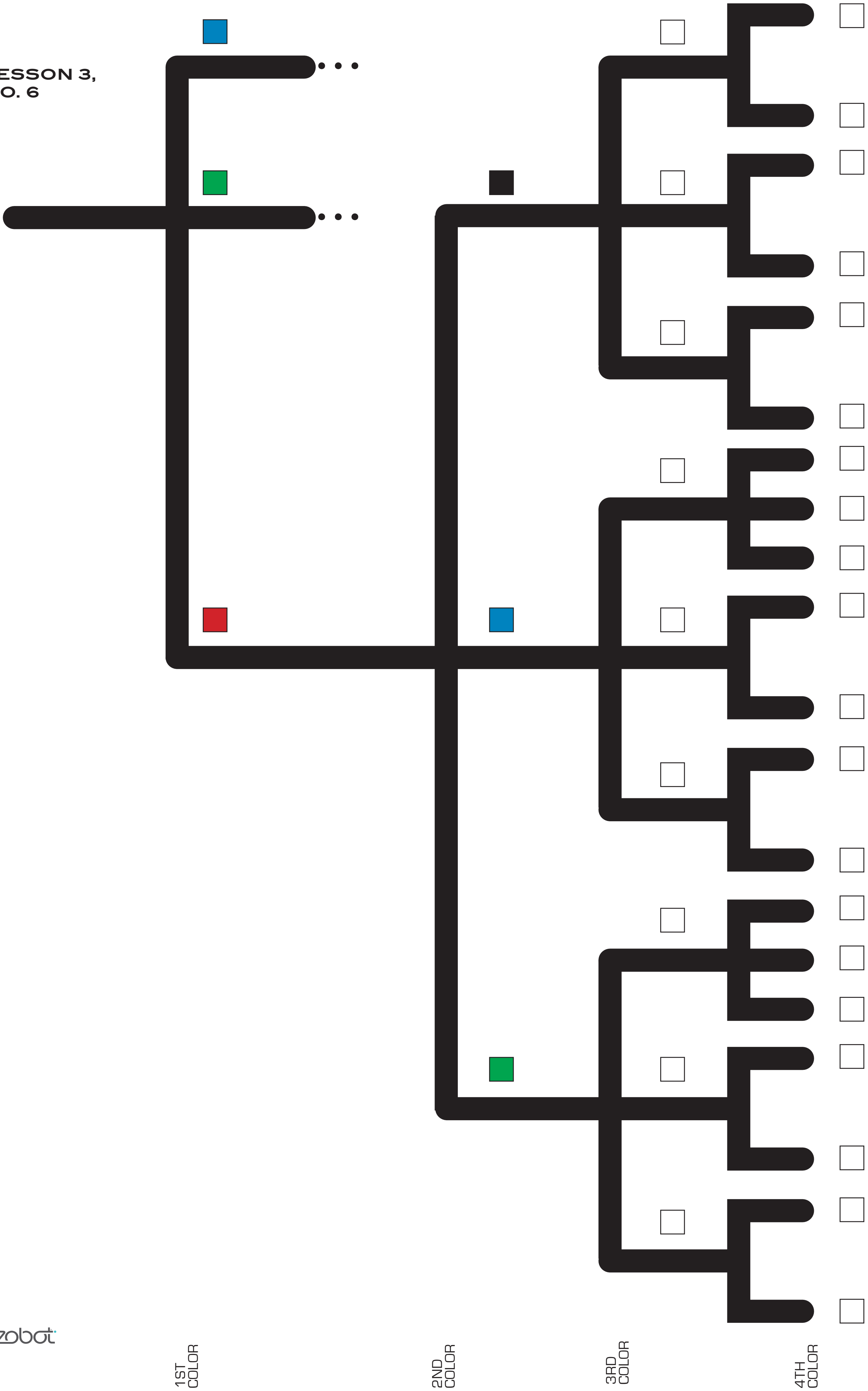
ozobot

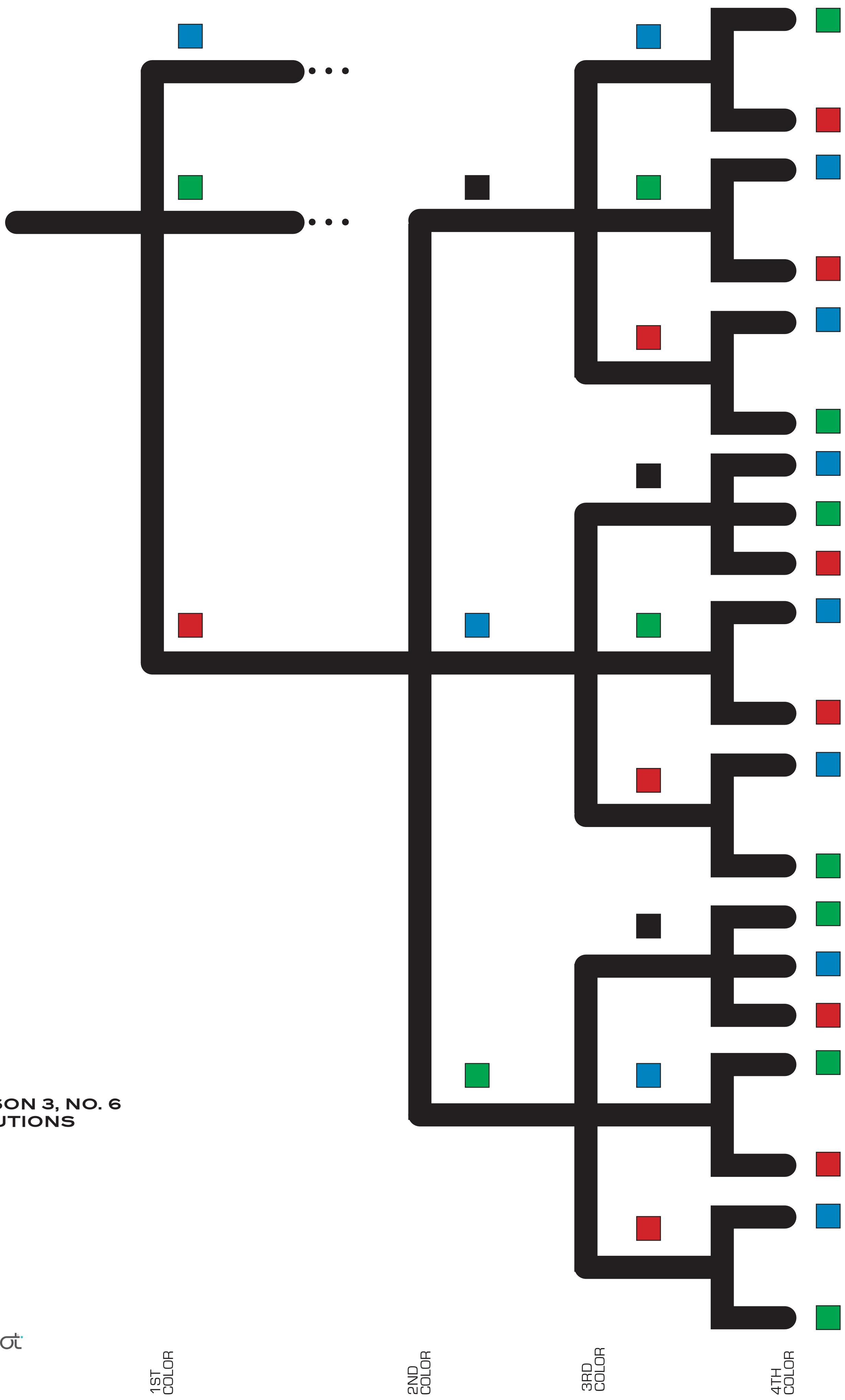
LESSON 3, NO. 5
SOLUTIONS



ozobot

LESSON 3, NO. 6

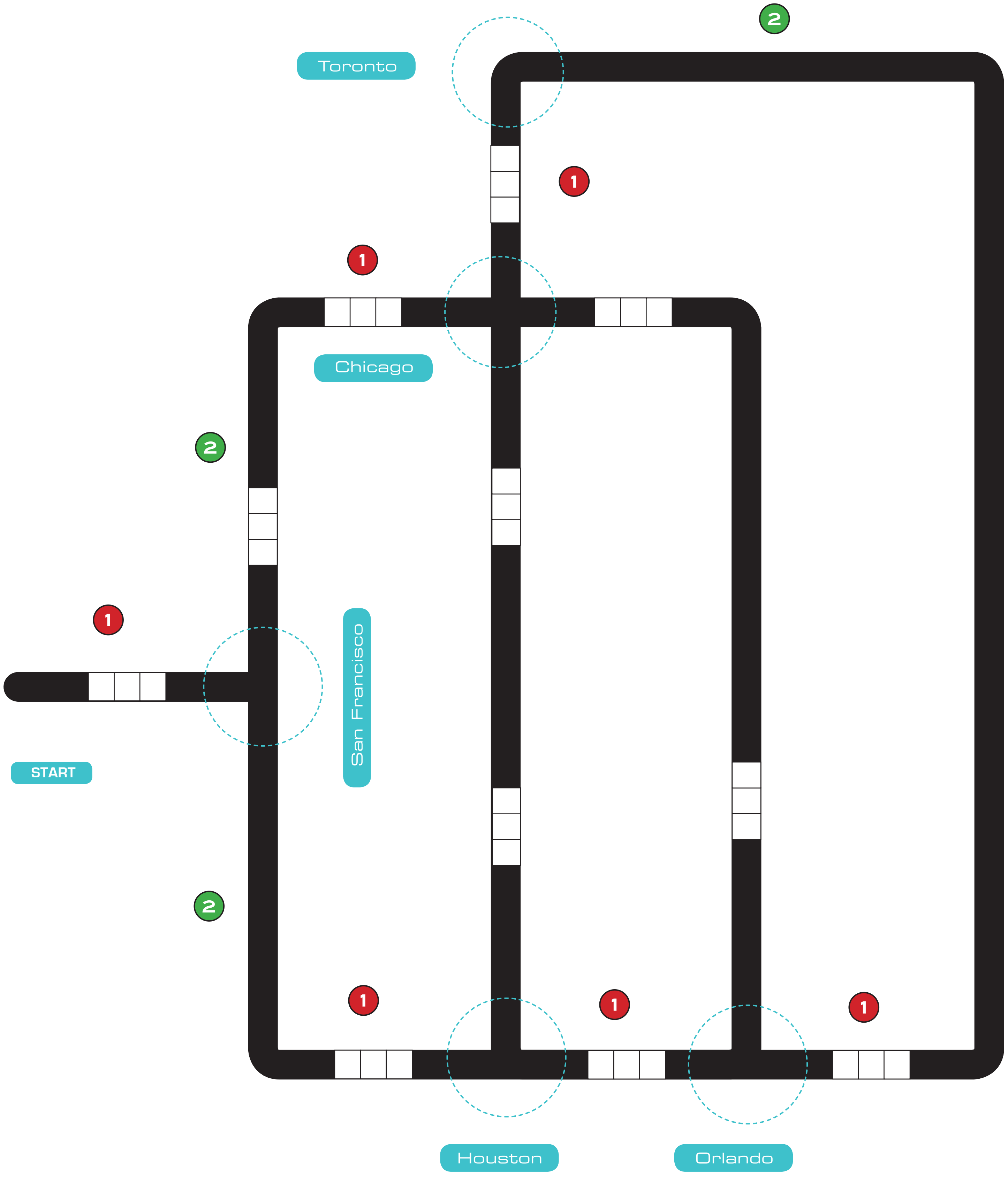




LESSON 3, NO. 6
SOLUTIONS

ozobot

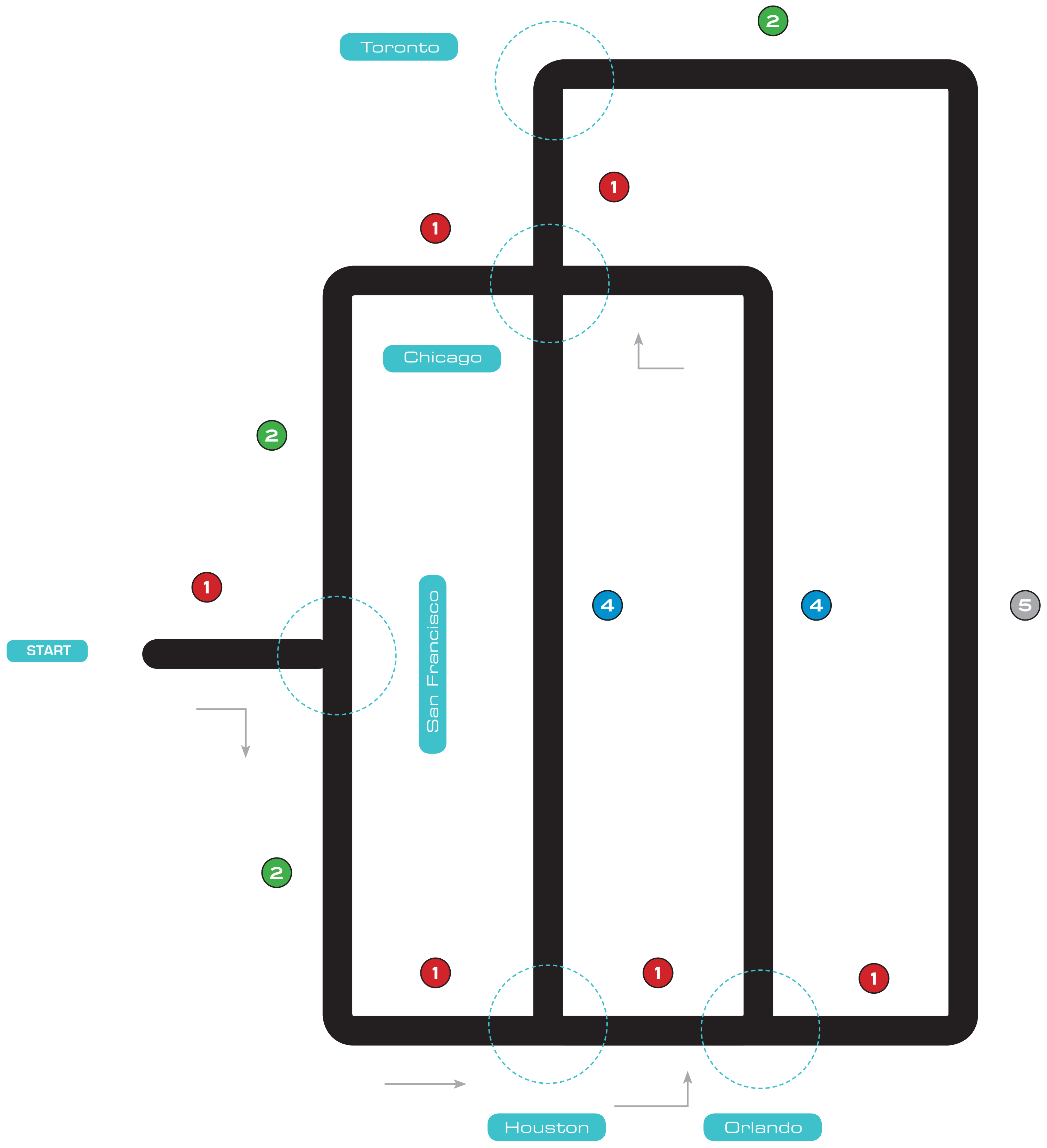
TRAVELLING SALESMAN MAP



ozobot

LESSON 3, NO. 7
SOLUTIONS

TRAVELLING SALESMAN MAP



Compute the distance between each of these cities:

San Francisco	$\longleftrightarrow$	Chicago	=
San Francisco	$\longleftrightarrow$	Houston	=
Houston	$\longleftrightarrow$	Orlando	=
Houston	$\longleftrightarrow$	Chicago	=
Chicago	$\longleftrightarrow$	Toronto	=
Chicago	$\longleftrightarrow$	Orlando	=
Orlando	$\longleftrightarrow$	Toronto	=
Start	$\longleftrightarrow$	San Francisco	=

Options for routes. Fill in this cities and compute the total distance:

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

Start ⟶ San Francisco ⟶ .... ⟶ .... ⟶ .... ⟶ ....
= 1 + ... + ... + ... + ... = $\boxed{}$

LESSON 3, NO. 8 SOLUTIONS
TRAVELLING SALESMAN WORKSHEET

Compute the distance between each of these cities:

San Francisco	$\longleftrightarrow$	Chicago	=	3
San Francisco	$\longleftrightarrow$	Houston	=	3
Houston	$\longleftrightarrow$	Orlando	=	1
Houston	$\longleftrightarrow$	Chicago	=	4
Chicago	$\longleftrightarrow$	Toronto	=	1
Chicago	$\longleftrightarrow$	Orlando	=	5
Orlando	$\longleftrightarrow$	Toronto	=	8
Start	$\longleftrightarrow$	San Francisco	=	1

Options for paths: what is the total distance?

Start $\longrightarrow$ San Francisco $\longrightarrow$ Chicago $\longrightarrow$ Houston $\longrightarrow$ Orlando $\longrightarrow$ Toronto
= 1+ 3 + 4 + 1 + 8 = 17

Start $\longrightarrow$ San Francisco $\longrightarrow$ Chicago $\longrightarrow$ Toronto $\longrightarrow$ Orlando $\longrightarrow$ Houston
= 1 + 3 + 1 + 8 + 1 = 14

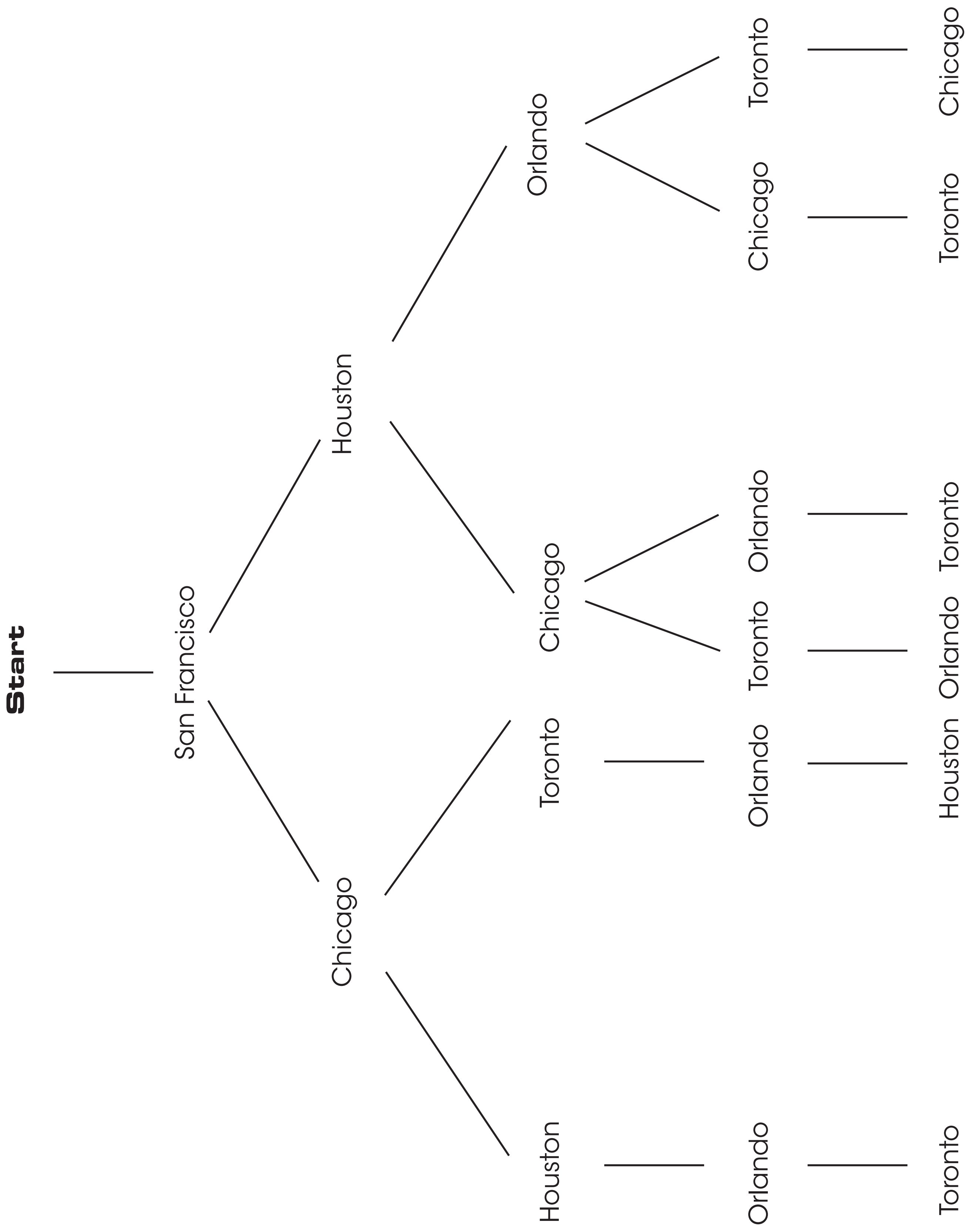
Start $\longrightarrow$ San Francisco $\longrightarrow$ Houston $\longrightarrow$ Chicago $\longrightarrow$ Toronto $\longrightarrow$ Orlando
= 1 + 3 + 4 + 1 + 8 = 17

Start $\longrightarrow$ San Francisco $\longrightarrow$ Houston $\longrightarrow$ Chicago $\longrightarrow$ Orlando $\longrightarrow$ Toronto
= 1 + 3 + 4 + 5 + 8 = 21

Start $\longrightarrow$ San Francisco $\longrightarrow$ Houston $\longrightarrow$ Orlando $\longrightarrow$ Chicago $\longrightarrow$ Toronto
= 1 + 3 + 1 + 5 + 1 = 11 $\longleftarrow$

Start $\longrightarrow$ San Francisco $\longrightarrow$ Houston $\longrightarrow$ Orlando $\longrightarrow$ Toronto $\longrightarrow$ Chicago
= 1 + 3 + 1 + 8 + 1 = 14

TRAVELLING SALESMAN GRAPH


Start
San Francisco
Houston
Chicago
Orlando
Chicago
Toronto
Houston
Toronto
Chicago
Orlando
Toronto
Orlando
Orlando
Chicago
Toronto
Toronto
Orlando
Houston
Toronto