*

Can a Computer Solve a Word Puzzle?

- or -

Can You Change MAN to APE?

ISC1057

Janet Peterson and John Burkardt

Computational Thinking

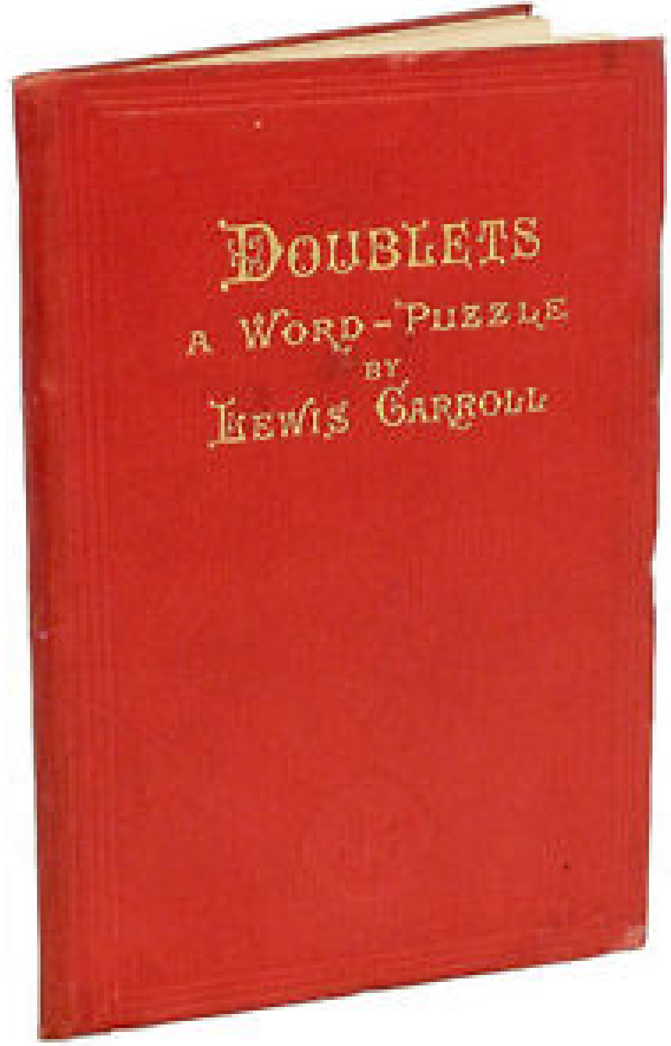
Fall Semester 2016

In this discussion, we will look at a simple word puzzle.

There are several special ways of thinking that we need in order to solve such a puzzle:

- **memory**: we need to know a lot of words;
- **imagination**: we need to imagine possible changes to a word;
- **evaluation**: given several possible changes, we need to choose the one most likely to take us to our goal;
- **backtracking**: when a choice doesn't work out, we need to backtrack and search for an alternate choice;

If we teach a computer to solve these puzzles, then we will have to understand how we do them first, and then try to translate our mental actions into computer actions.


DOUBLETS
A WORD-PUZZLE
BY
LEWIS CARROLL

Lewis Carroll, who wrote the children's book "Alice in Wonderland", was very fond of word games and puzzles.

He asked a riddle that no one has solved: *Why is a raven like a writing desk?*.

He wrote poems like Jabberwocky full of nonsense words, a few of which were absorbed into English: **burbled** and **gallumphing**.

And he invented a word game which he called *"Doublets"*.

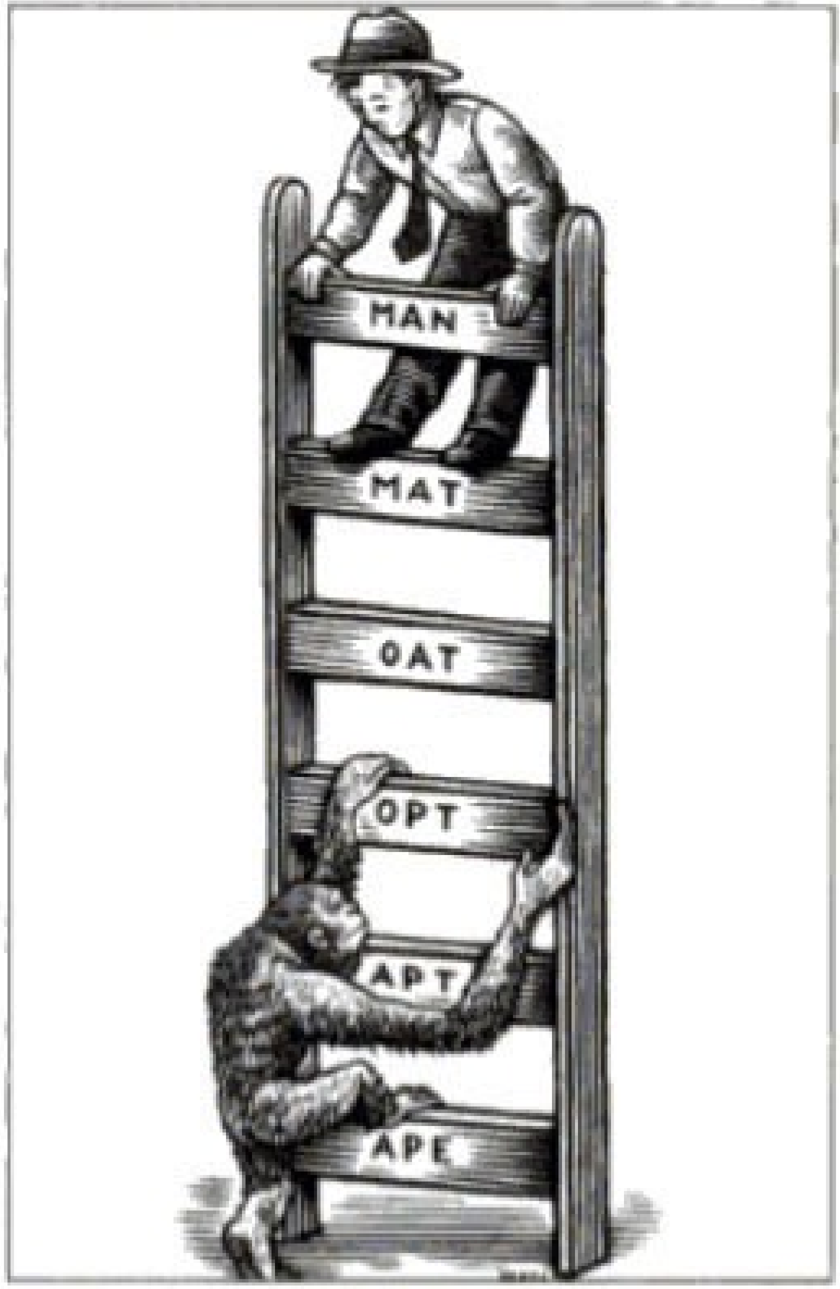


Illustration by Gregory Nemec

Lewis Carroll enjoyed asking friends "Can you turn MAN into APE?"

After getting a puzzled look, he would say: "It's easy!"

```
MAN
MAT (change N to T)
OAT (change M to O)
APT (change A to P)
APE (change T to E)
```

His book called "Doublets" contains more examples of such puzzles.

DOUBLETS ALREADY SET

IN "VANITY FAIR."

March 29.—Drive PIG into STY.
Raise FOUR to FIVE.
Make WHEAT into BREAD.

April 5.—Dip PEN into INK.
Touch CHIN with NOSE.
Change TEARS into SMILE.

April 12.—Change WET to DRY.
Make HARE into SOUP.
PITCH TENTS.

April 19.—Cover EYE with LID.
Prove PITY to be GOOD.
STEAL COINS.

April 26.—Make EEL into PIE.
Turn POOR into RICH.
Prove RAVEN to be MISER.

May 3.—Change OAT to RYE.
Get WOOD from TREE.
Prove GRASS to be GREEN.

May 10.—Evolve MAN from APE.
Change CAIN into ABEL.
Make FLOUR into BREAD.

Digitized by Google

May 17.—Make TEA HOT.
Run COMB into HAIR.
Prove a ROGUE to be a BEAST.

May 24.—Change ELM into OAK.
Combine ARMY and NAVY.
Place BEANS on SHELF.

May 31.—HOOK FISH.
QUELL a BRAVO.
Stow FURIES in BARREL.

June 7.—BUY an ASS.
Get COAL from MINE.
Pay COSTS in PENCE.

June 14.—Raise ONE to TWO.
Change BLUE to PINK.
Change BLACK to WHITE.

June 21.—Change FISH to BIRD.
Sell SHOES for CRUST.
Make KETTLE HOLDER.

N.B. Solutions of these Doublets will be found at *p.* 38.

Digitized by Google

Lewis Carroll came up with a few new puzzles each day and wrote them down.

He didn't explain how he came up with the puzzles, although we can see from the examples that he enjoyed changing a word into its opposite, or using a pair of words that could be used to express a humorous sentence.

Obviously, the pair of words must have the same number of letters, but just because we come up with a pair of words like LOVE and HATE doesn't mean that we can figure out a way to change one into the other, one letter at a time.

MAN	MAN	MAN
MAY	MAR	MAT
PAY	EAR	OAT
PAT	ERR	APT
PIT	ERE	APE
PIE	ARE	
DIE	APE	
DYE		
AYE		
APE		

If we find a way, we don't know if there is a shorter one.

As we can see, in the MAN to APE example, it's easy to drag out the solution, although the best solution is very short.

An obvious strategy is to pick a letter in the goal, and see if you can put it into the current word immediately. If not, sometimes you can see that this is possible in one or two extra steps.

MAN
MAT
PAT
PIT
PIE $\leftarrow$ consonant "T" switches to vowel "E"
DIE
DYE
AYE $\leftarrow$ consonant "D" switches to vowel "A"
APE $\leftarrow$ vowel "Y" switches to consonant "P"

Another thing to notice is whether the vowels and consonants match up. In APE to MAN, every vowel becomes a consonant, and vice versa, and this can make it difficult to do the transformation.

Transform IRON into LEAD.
Move FIRST to THIRD.
From BELOW go ABOVE.

Lewis Carroll worked out a solution to his puzzles before posing them.

But what if we just pull a pair of words out of the air, and try to join them by a chain of transformations?

If we don't find a solution, we really don't know whether we simply didn't try hard enough, or whether there really is no solution.

Don't try these examples! They can't be done!


HALLOWEEN
Word Ladder
MASK
BURN

The game invented by Lewis Carroll is still very popular, and shows up in many magazines and puzzle sites.

When given as a puzzle, sometimes it is helpful to show the number of steps required.

Here, we are asked to turn mask into burn using 3 intermediate words, changing one letter at a time.

One solution is **MASK, BASK, BARK, BARN, BURN**.

Notice that we were able to solve this example simply by changing each of the four letters of **MASK** into a letter of **BURN**, and we just had to figure out the right order in which to do this.

Most word ladder puzzles are harder than this, however!

WORD LADDER: BRITISH PEERAGE RANK

RANDOM JUST FOR FUN OR WORD LADDER QUIZ

Can you name the 4-letter words in this themed word ladder?

by ctom FOLLOW

Updated Oct 27, 2011

60,305 PLAYS 21 2 4.2 99

HOW TO PLAY

Share Tweet Embed

POPULAR TODAY
Green or Red? A Picture Click L... 48,878
Find the US States - No Outlines... 16,075
'I'-Less Asian Countries 11,803
US States (Redux) 9,202
US Cities: West to East 8,620
And more...

PLAY CHALLENGE

SCORE 0/16 TIMER 02:00

Clue	4-Letter Word
Highest British peerage rank	
Sand hill formed by wind	
An inhabitant of Denmark	
It is used to assist someone in walking	
Orange item found around construction sites	
What a scapula is made of	
Last name of a famous secret agent	
To make text darker	
Lower storage portion of a ship	
You might see one in Swiss cheese	
American patriot Nathan	
Racer Earnhardt Jr. or Sr.	
Madonna's 'Truth or ___'	
Mend a hole in a knitted sock	
Make money through work	
British peerage rank between Marquess and Viscount	

To make a long puzzle solvable, sometimes there are clues for the missing words.

The game web site Sporcle at **www.sporcle.com** offers a guided version of Doublets, in which the steps are laid out, with hints. This allows a group of people to cooperatively solve the puzzle, shouting out their guesses, hoping to beat the timer.

Here, we start out with a clue for the word **DUKE** and are clued through a series of steps to the final word **EARL**.

Turn COLD to WARM

COLD

WARM

One doublet puzzle asks us to turn COLD to WARM.

We think of $\mathbf{COLD}$ as the start word, and $\mathbf{WARM}$ as the target word.

It is usually difficult to think of a correct strategy for solving such a puzzle.

This particular puzzle, though, is another illustration of a simple approach that can sometimes work.

The approach, called the greedy method is *Try to replace a letter of the start word with a letter of the target word.*

Thus, starting from $\mathbf{COLD}$, we assume our first step should be to check whether $\mathbf{WOLD}$, $\mathbf{CALD}$, $\mathbf{CORD}$ or $\mathbf{COLM}$ is a word. Thereafter, we keep hoping to take another step by swapping another letter of the starting word for a letter of the target word.

Turn COLD to WARM

COLD
CORD
WORD
WARD
WARM

It is surprising to see that this puzzle can be done simply by swapping one letter at a time.

A greedy person might stumble on this strategy, saying "Let's go for the goal right away! The fastest way is to swap in a target letter on every move!"

The greedy algorithm strives for an immediate obvious payoff. In this example, it reaches the target word by taking a greedy step every time. In other puzzles, it won't always work, but it's always useful to check whether you can take at least one step by swapping in a target letter.

Try the greedy algorithm on these doublets:

```
LEAF    RICH    COME
....    ....    ....
....    ....    ....
....    ....    ....
WORD    DUNE    SALT
```

LEAF	RICH	COME
LEAD	RICE	SOME
LOAD	DICE	SAME
LORD	DINE	SALE
WORD	DUNE	SALT

Here are some harder ones!

```
HEAD    HARD    RISE
....    ....    ....
....    ....    ....
....    ....    ....
....    ....    ....
TAIL    EASY    ....
                FALL
```

HEAD	HARD	RISE
HEAL	HARE	RITE
TEAL	BARE	MITE
TELL	BASE	MILE
TALL	EASE	FILE
TAIL	EASY	FILL
		FALL

In these puzzles, the greedy method doesn't always work. We have to take more steps than usual, and occasionally swap in a letter that later we have to swap back out.

SAGE

FAGE *Not a word!*

SOGE *Not a word!*

SAOE *Not a word!*

SAGL *Not a word!*

FOOL

One of Lewis Carroll's puzzles asks us to turn SAGE into FOOL.

Let's try to think about how we might solve such a puzzle.

Perhaps the first thing to try is simply to hope that we can immediately swap one letter of $\mathbf{SAGE}$ for one of $\mathbf{FOOL}$...after all, we have to do that eventually.

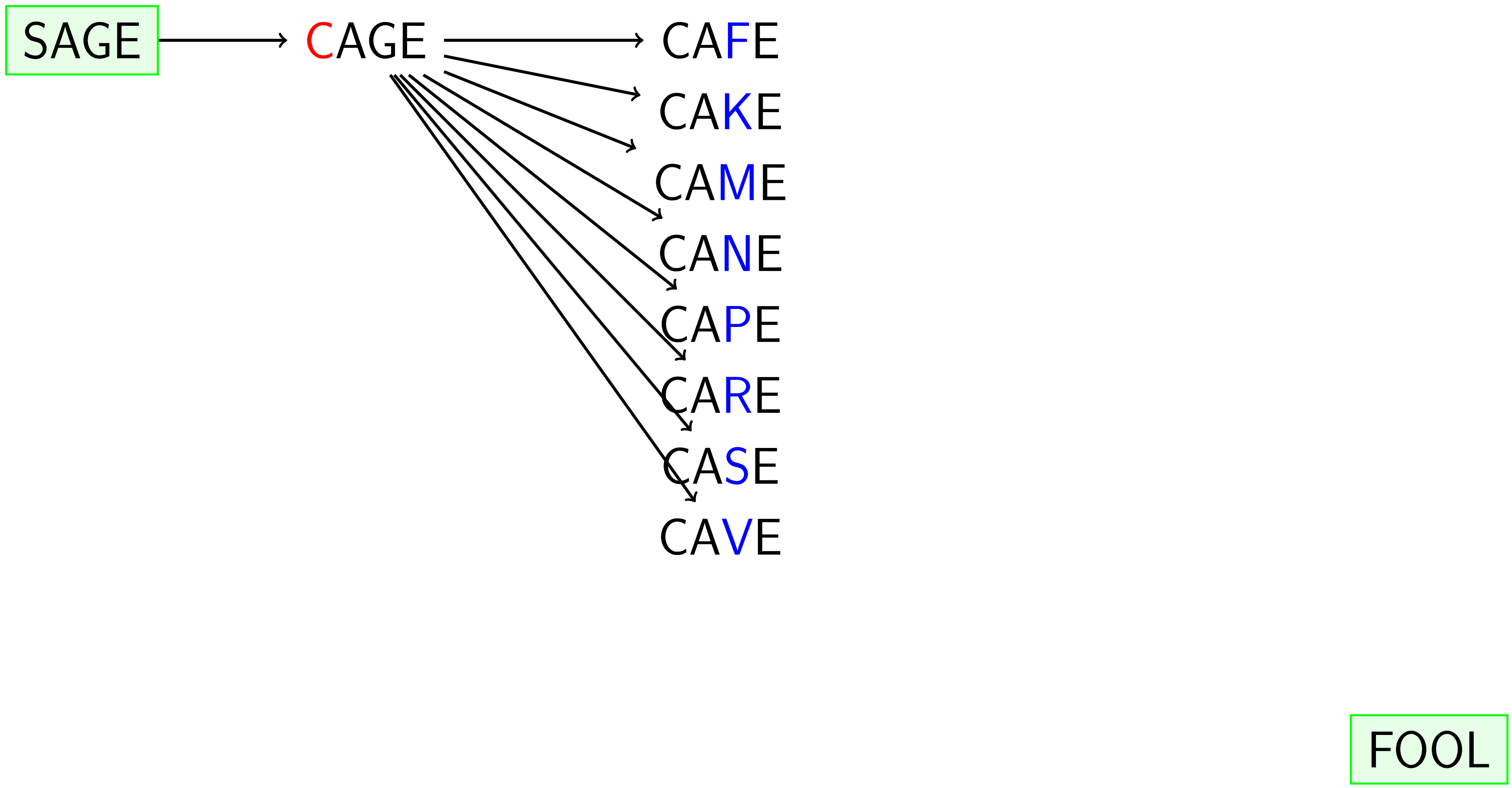
However, we can see that $\mathbf{FAGE}$, $\mathbf{SOGE}$, $\mathbf{SAOO}$ and $\mathbf{SAGL}$ are not words, so we can't make this jump.


SAGE
CAGE
MAGE
PAGE
RAGE
WAGE
SAFE
SALE
SAME
SANE
SAVE
FOOL

So maybe the next thing to consider is ... what words **can** we jump to, and then make a choice of those words.

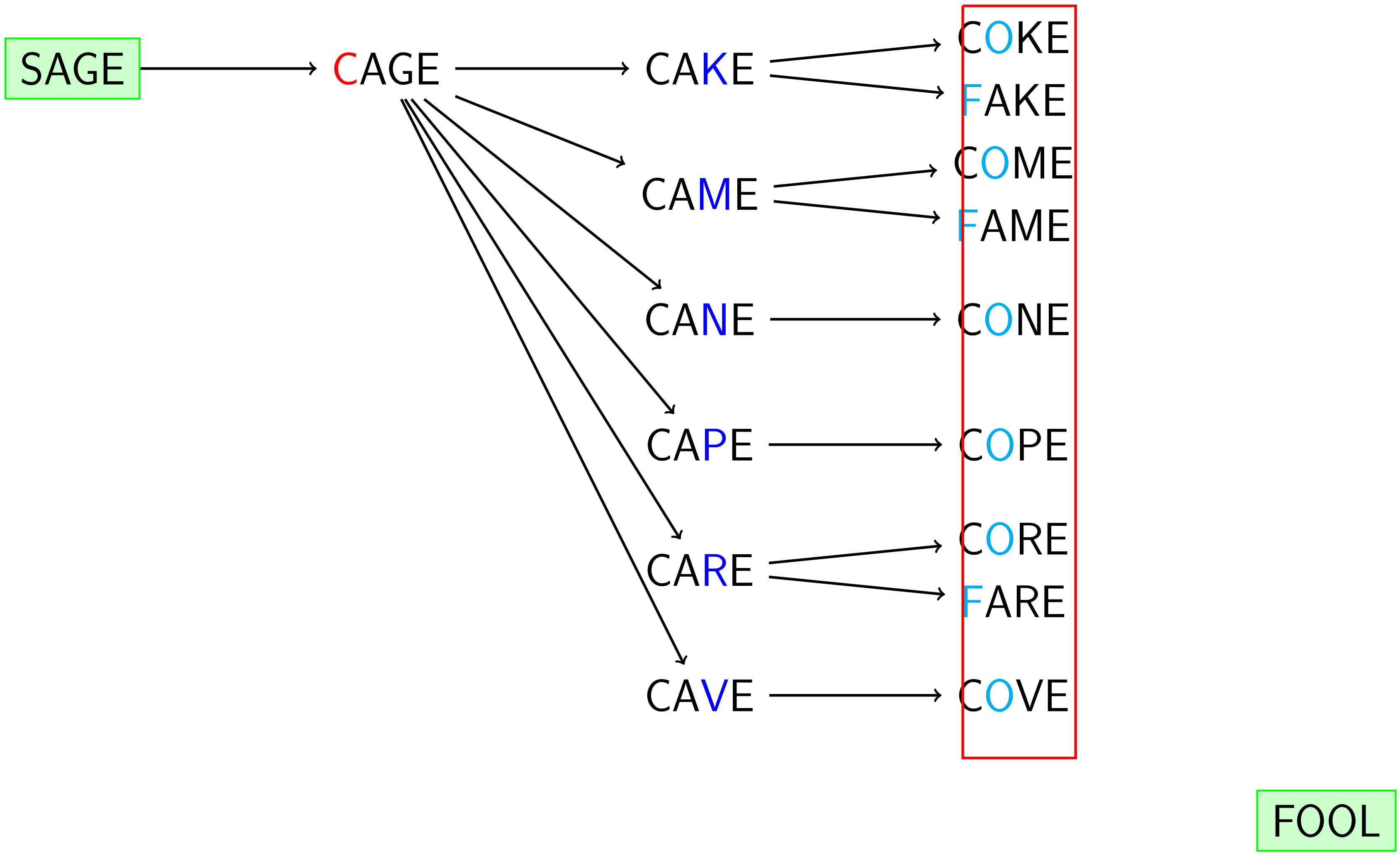
Changing the first letter of **SAGE** gives us CAGE, MAGE, PAGE, RAGE, WAGE.

Changing the third letter of **SAGE** gives us SAFE, SALE, SAME, SANE, SAVE.


SAGE
CAGE
CAFE
CAKE
CAME
CANE
CAPE
CARE
CASE
CAVE
FOOL

Given so many choices, let's focus on the very first one, and then move the others onto the back burner. If our first choice fizzles out, then we can backtrack, that is, come back to these unexplored choices and try them out.

In fact, **CAGE** looks very useful, because there seem to be a lot of words we can get to next: CAFE, CAKE, CAME, CANE, CAPE, CARE, CASE, CAVE.


SAGE
CAGE
CAKE
CAME
CANE
CAPE
CARE
CAVE
COKE
FAKE
COME
FAME
CONE
COPE
CORE
FARE
COVE
FOOL

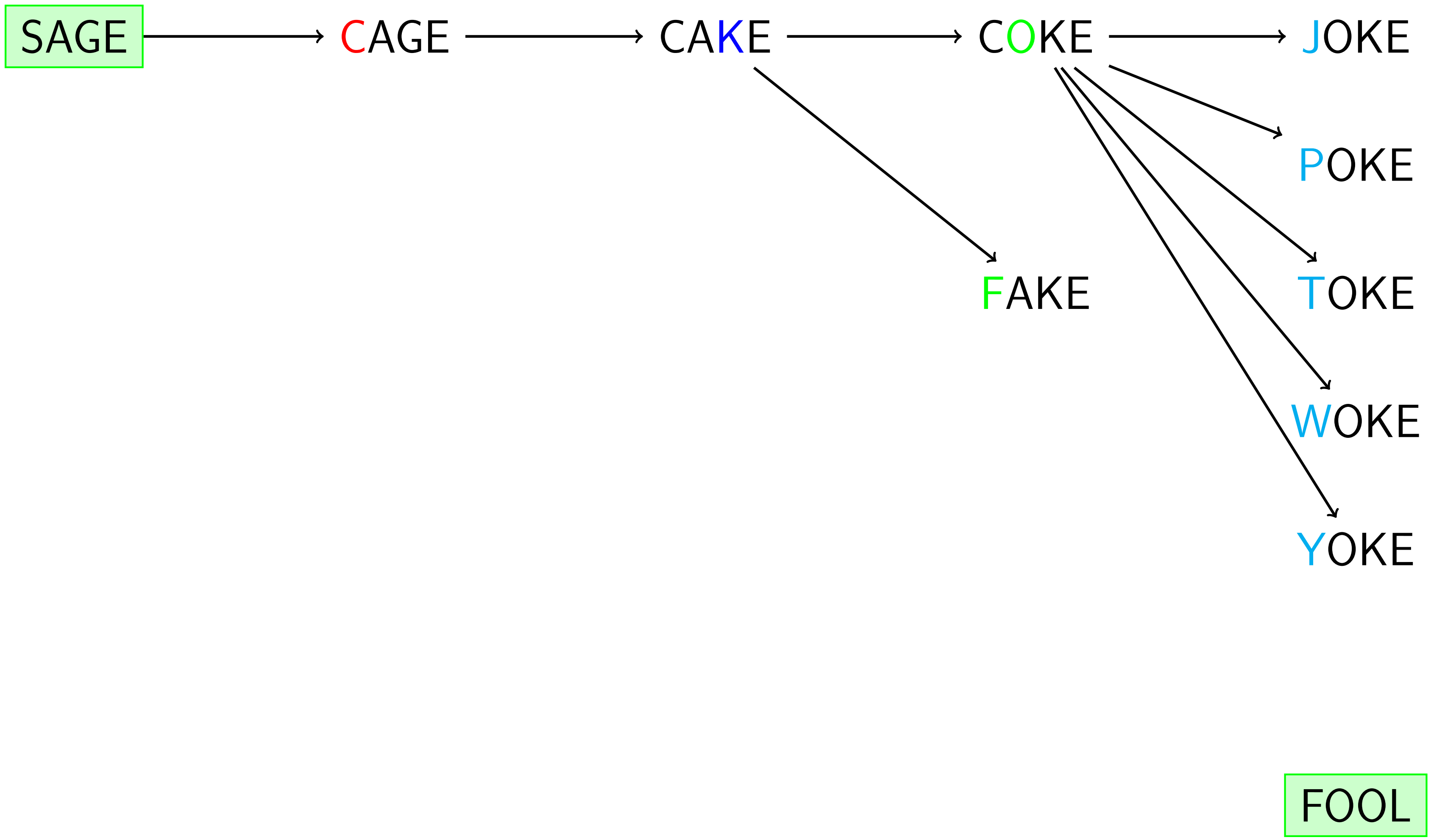
Now we should look at each of these words, and try the greedy approach, that is, whether we can immediately swap in a letter of **FOOL**.

For instance, the word **CAFE** doesn't seem to offer any chance.

But the word **CAKE** can get closer to **FOOL** by transforming into **COKE** or **FAKE**.

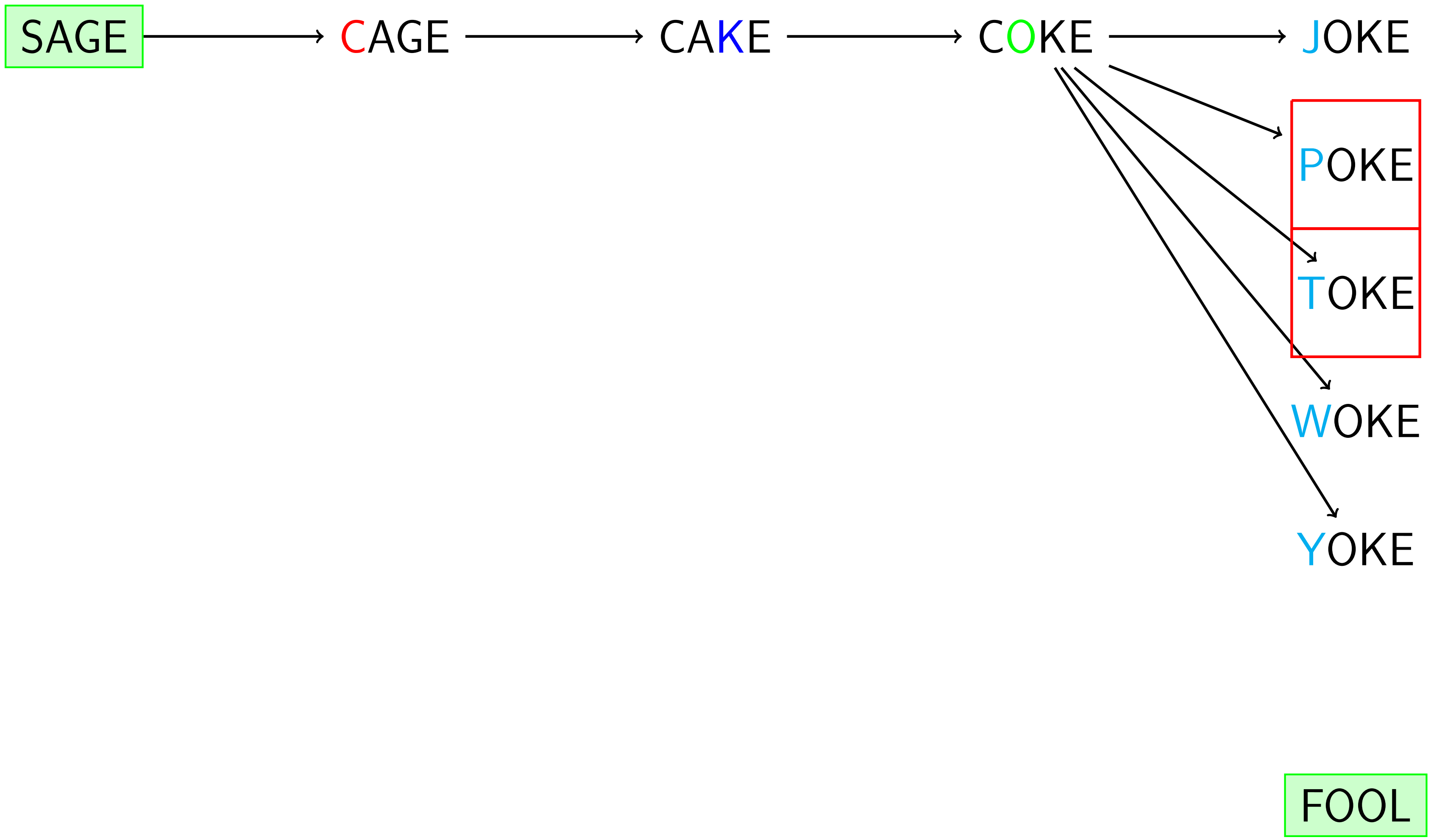
Similarly, **CAME**, **CANE**, **CAPE**, **CARE** and **CAVE** all seem to offer a chance of stepping closer.

So now let's focus on the jump from **CAKE**, and put the other options also on the backburner.


SAGE
CAGE
CAKE
COKE
JOKE
POKE
FAKE
TOKE
WOKE
YOKE
FOOL

Now we have **COKE** and **FAKE** to work with. **FOOL**.

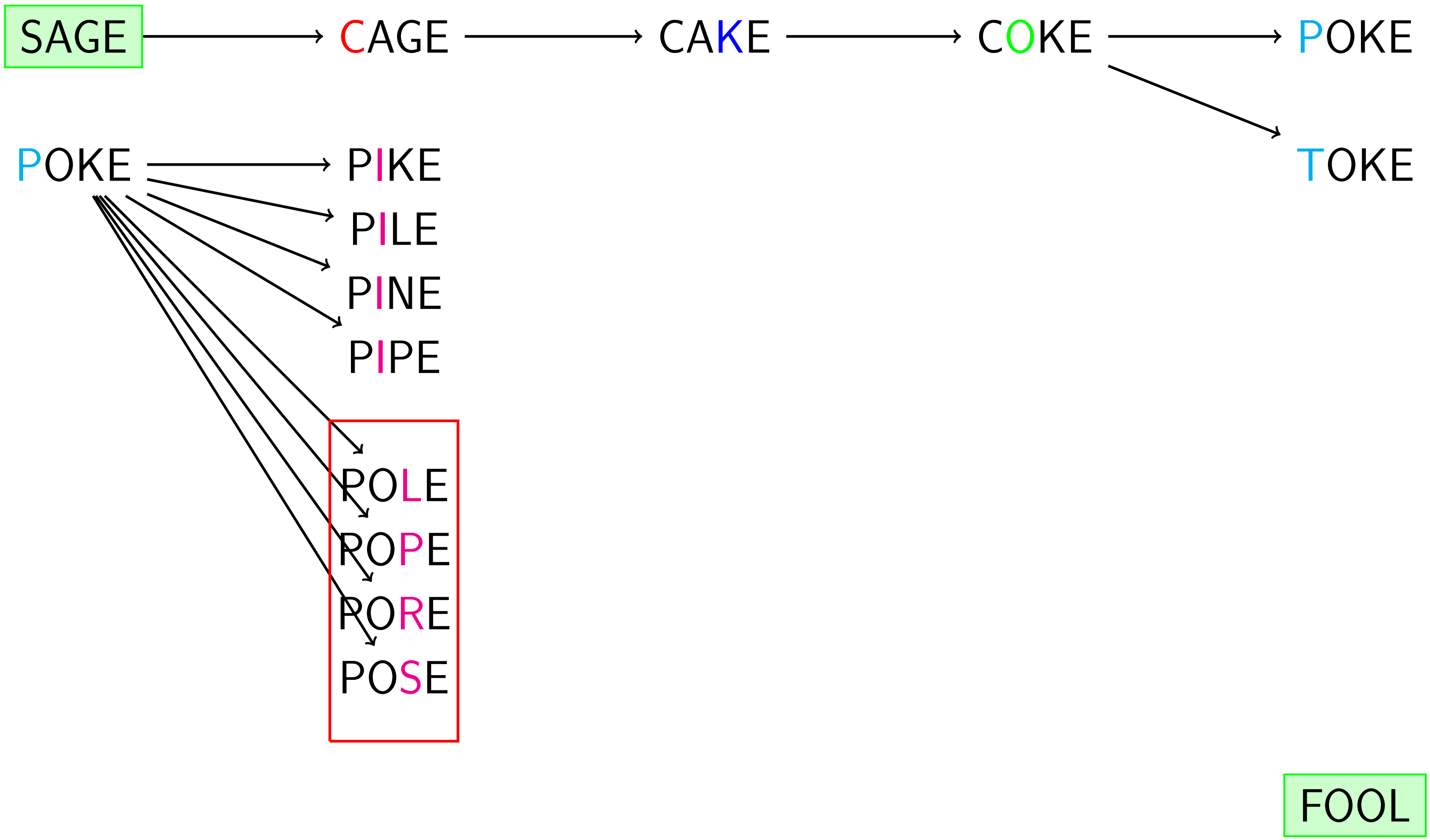
Looking at **COKE**, we can't swap another **FOOL** letter in, so let's just ask what other new words we can get. It seems we have at least JOKE, POKE, TOKE, WOKE, YOKE.


SAGE
CAGE
CAKE
COKE
JOKE
POKE
TOKE
WOKE
YOKE
FOOL

I really don't find **JOKE, WOKE, YOKE** attractive because words with the letters "J", "W" and "Y" don't seem very common. I'd much rather work with **POKE** or **TOKE**.

Let's make POKE our focus, with **TOKE** as our backup, and **JOKE, WOKE, YOKE** as backup backups...

This process of making an on-the-fly evaluation of your opportunities is very important. These rules-of-thumb can help you make a reasonable, if not perfect, choice.

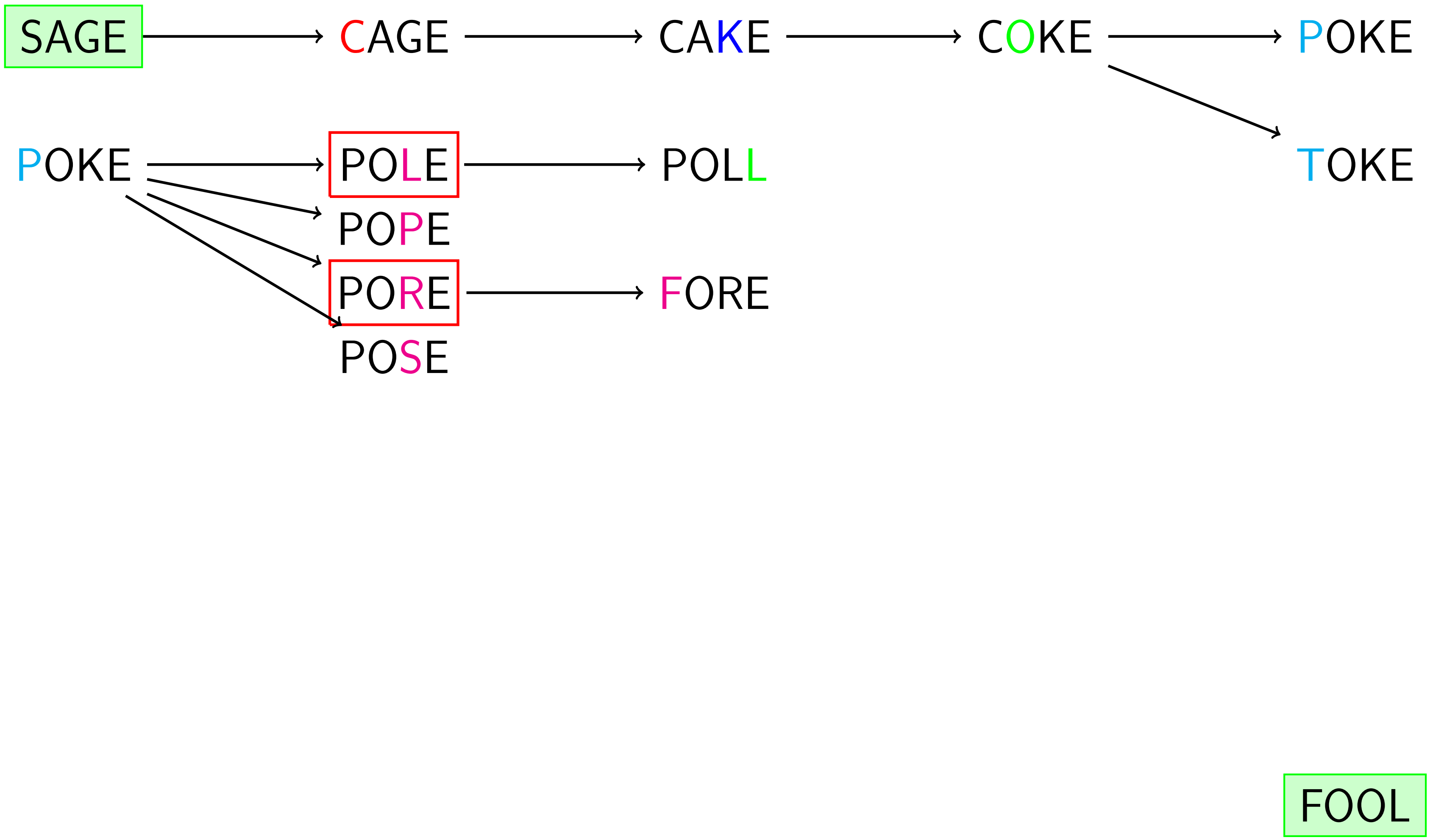


SAGE
CAGE
CAKE
COKE
POKE
TOKE
POKE
PIKE
PILE
PINE
PIPE
POLE
POPE
PORE
POSE
FOOL

What can we do with **POKE**? We got here by changing the first letter.

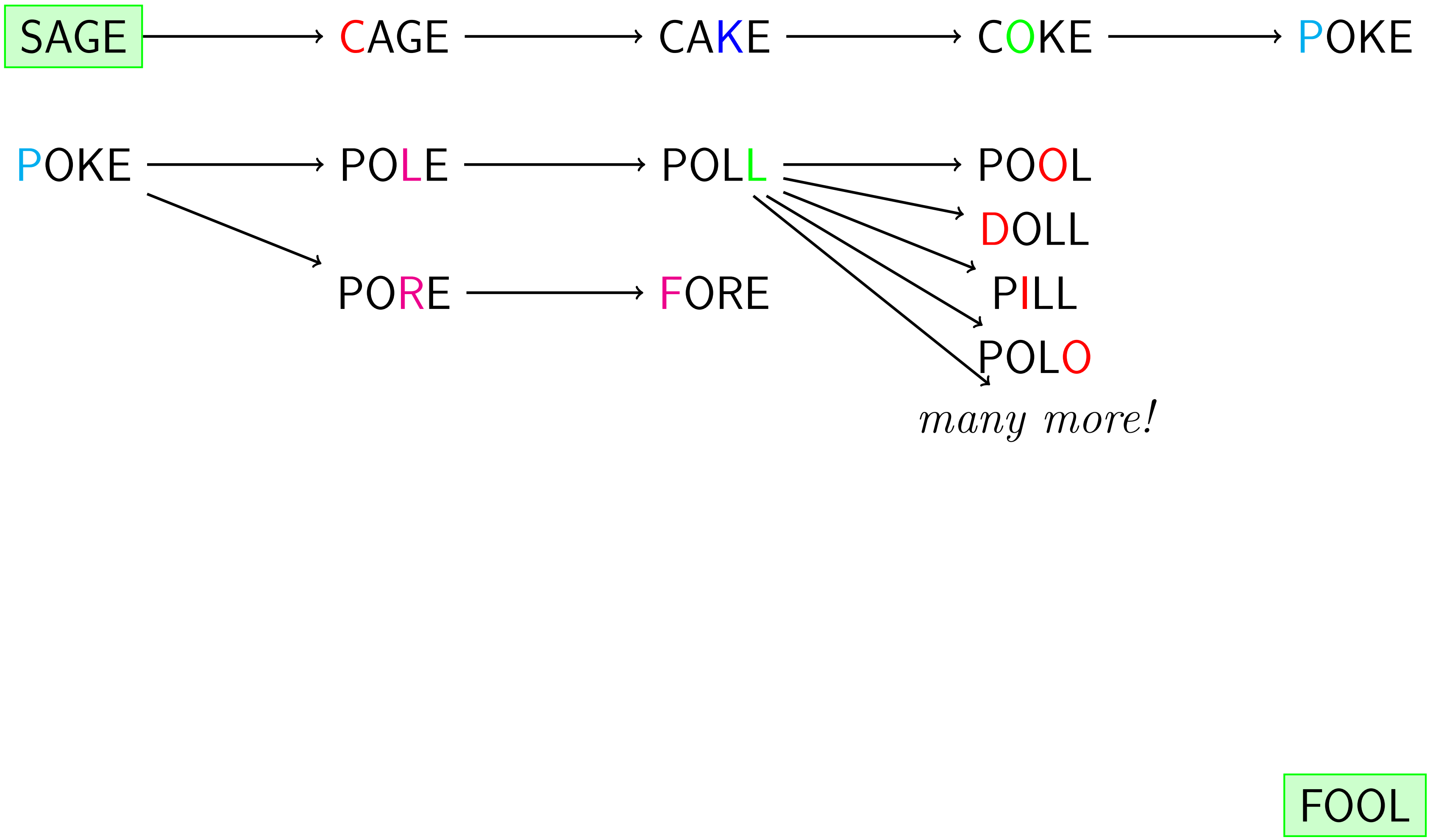
If we change the second letter, we can get PIKE, PILE, PINE, PIPE.

But we'd like to keep the second letter, since that matches **FOOL**. Changing the third letter can get us POLE, POPE, PORE, POSE

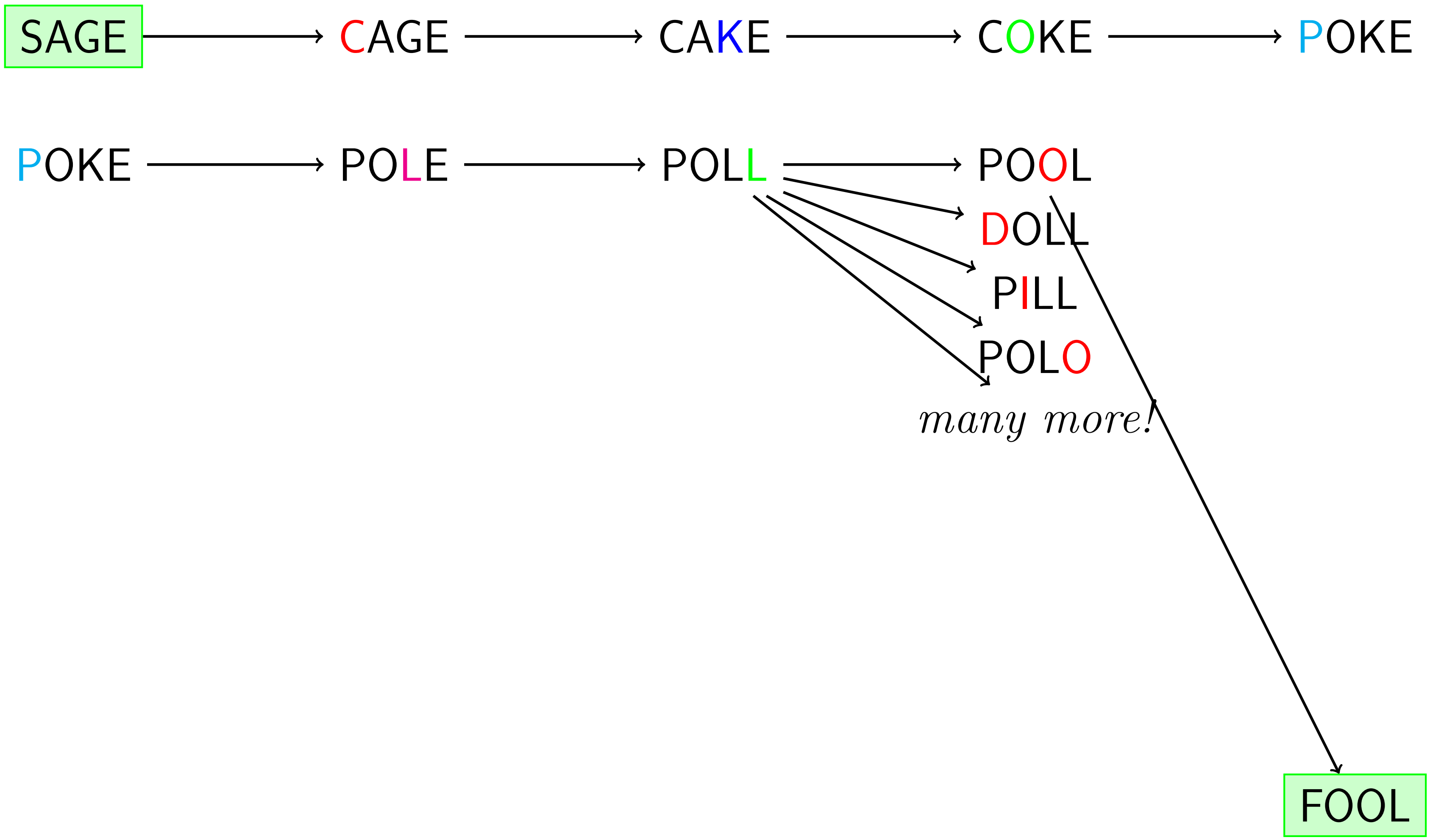

SAGE
CAGE
CAKE
COKE
POKE
TOKE
POKE
POLE
POLL
POPE
PORE
FORE
POSE
FOOL

And now things start to get exciting, because I can see that the greedy choice can work for $\mathbf{POLE}$, giving us POLL, or for $\mathbf{PORE}$, giving us FORE.

Suddenly, we seem to be moving close to our solution!


SAGE
CAGE
CAKE
COKE
POKE
POKE
POLE
POLL
POOL
DOLL
PILL
POLO
PORE
FORE
many more!
FOOL

I am really interested in $\mathbf{POLL}$ because it means that we have swapped out a vowel for a consonant in the fourth position, which is a difficult jump.



SAGE
CAGE
CAKE
COKE
POKE
POKE
POLE
POLL
POOL
DOLL
PILL
POLO
many more!
FOOL

And once we have $\mathbf{POLL}$, we can see two greedy jumps that take us the solution: POOL and then $\mathbf{FOOL}$.

It's strange but true that once we get close to the solution, the last few steps are often easy.

SAGE → CAGE → CAKE → COKE → POKE → POLE → POLL → POOL → FOOL

Now we can display our solution, hiding all the work we did, and all the partial results we kept in backup in case our first guesses didn't work.

$\mathbf{SAGE}$ turned into $\mathbf{FOOL}$ using 7 intermediate words.

SAGE : 4

CAGE : 4

CAKE : 4

COKE : 3

POKE : 3

POLE : 3

POLL : 2

POOL : 1

FOOL : 0

We can measure how close we are getting to the solution simply by counting the number of incorrect letters.

SAGE starts out with all four letters incorrect, and our next two moves don't actually add a correct letter, they are just searching around for a good jump.

When we go from **CAKE** to **COKE**, though, our distance does drop to 3, since "O" is the right letter in the right place.

It takes us some more time wandering around at a distance of 3.

At the end, our distance decreases step by step.

For this example, the distance always went down. We can imagine there are puzzles for which the distance might go up, where we have to temporarily lose a correct letter in order to reach a useful steppingstone word.

POLE $\longrightarrow$ POLL

POLL $\longrightarrow$ POOL

When close, you may "see" the solution.

Otherwise:

1. Always try greedy step;
2. Step to words with many neighbors;
3. Try to avoid words with unusual letters;
4. Watch for chances to correct consonant/vowel mismatch;

Some conclusions we can make from this puzzle solving experience:

Once you get close, the puzzle gets much easier.

When you have several choices, you record the ones you didn't explore, so if the current one fails, you can come back and try others.

One useful way to evaluate choices is the "greedy" check. Can we use this word to swap in another letter of the target word?

Another way to evaluate a choice is whether it helps us match the vowel and consonant pattern of the target word.

We also should prefer choices that don't have unusual letters like J, K, Q, W, X, Y or Z.

Another way to evaluate a choice is whether it has many "neighbors", that is, it that word can be transformed into many new words.

In doublets, we are trying to "travel" from one word to another, but we don't have a map. A map would help us plan the route, and even to take the shortest one.

Road maps may include in small print the distance between two cities that are directly connected by a stretch of road; but we will want to know the distance of the total journey.

So we are looking for a map that answers our question:

What is the shortest path to our goal?

The shortest path problem is a famous case in computing. Versions of this problem arise during many kinds of computation, and our doublets problem is one of them.

So before returning to the doublets question, let's take a moment to consider the shortest path problem for two simple cases, involving a city-to-city driving map, and a maze of connected rooms.

For the driving map problem, suppose we have cities A through F, with a network of roads of varying lengths, and that we wish to start at city B and determine the shortest distance to all the other cities on the map.

We know that the shortest distance from B to itself is 0 miles, so we can fill that in before we start, and we set all the other distances to ∞.

Now we look at all the cities that are immediately connected to B, and pick the closest one, say city A and add it to the $\mathbf{sure}$ set. We are sure that there is no way to shorten the distance from B to A by going through another town, say C, because just getting to C takes longer than getting to A directly.

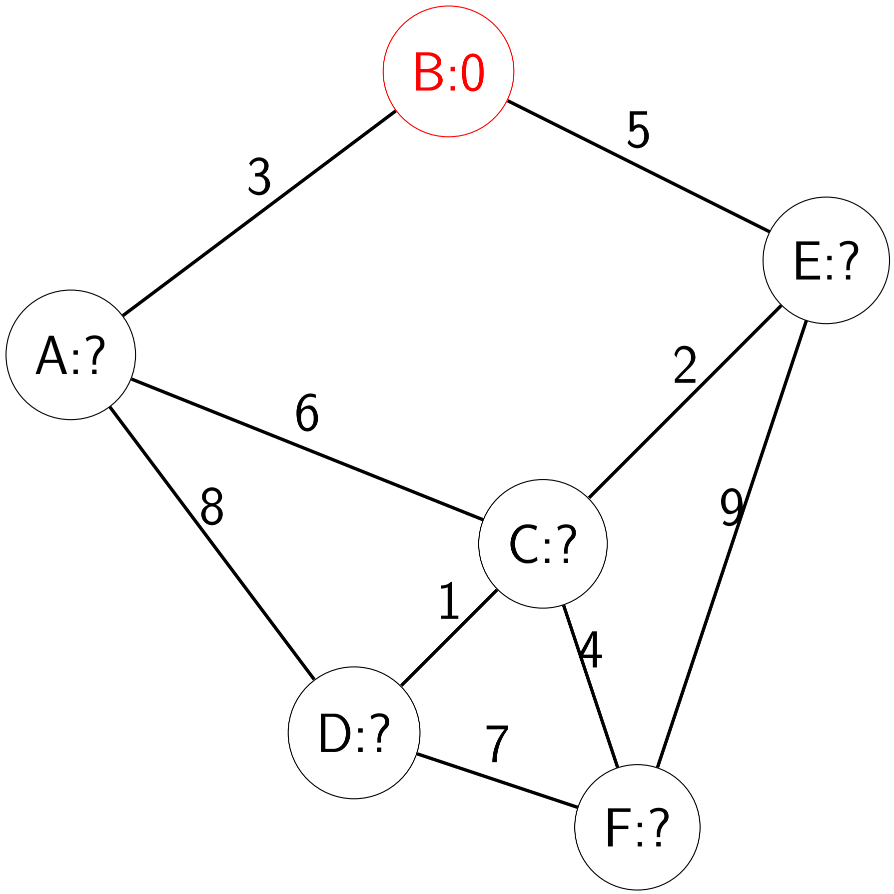
For the next step, we check the distances of trips that start at B, pass through A, and then land at any immediate neighbor of A. Any time such a trip is shorter than what we've already recorded, we put down the new shorter estimate.

After this check, we look at the table for the city, say "F", with the lowest distance in the unsure set, and move it to the sure set.

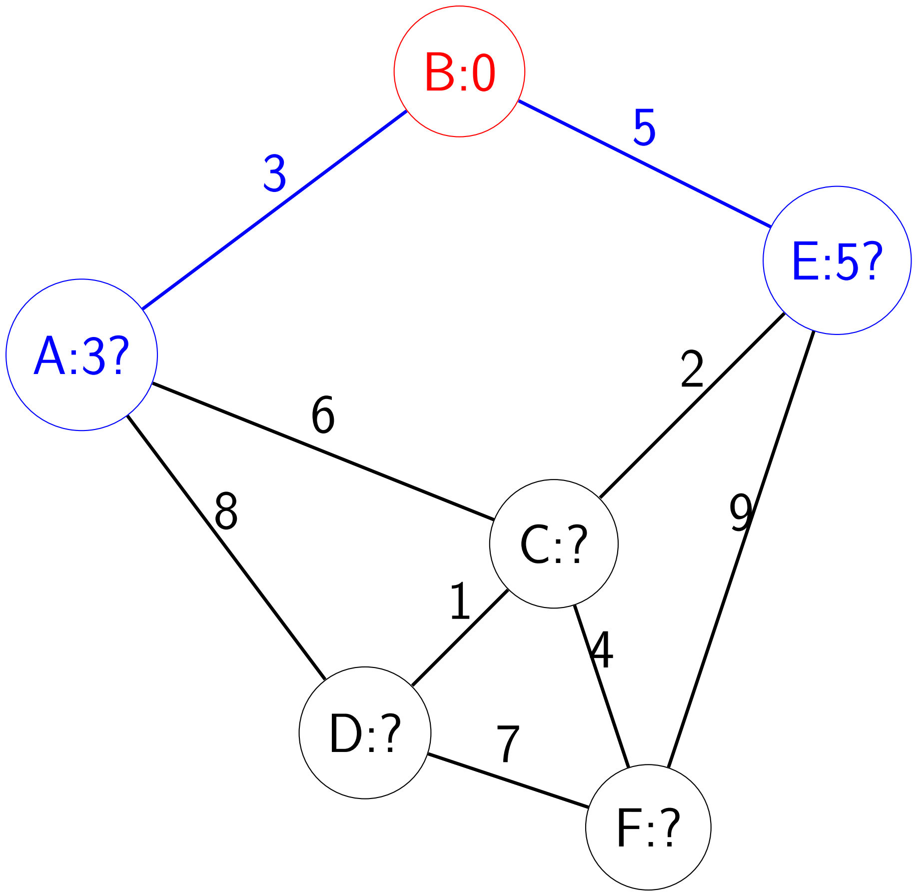
Then we consider the distance of trips that start at B, pass through F, and continue to any one of F's immediate neighbors.

Eventually, we will complete the table and have the shortest distance for trips from B to any other city.

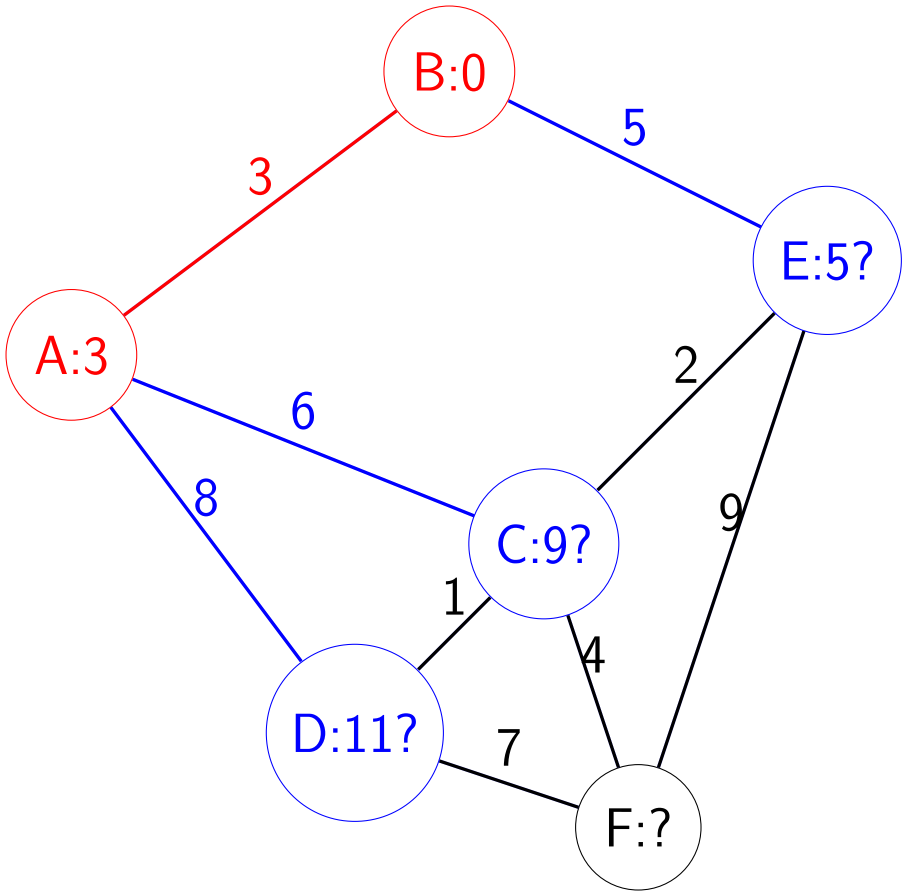
To do a complete table of shortest distances from any city to any city, we have to repeat the whole procedure, picking a new starting city each time.



City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞

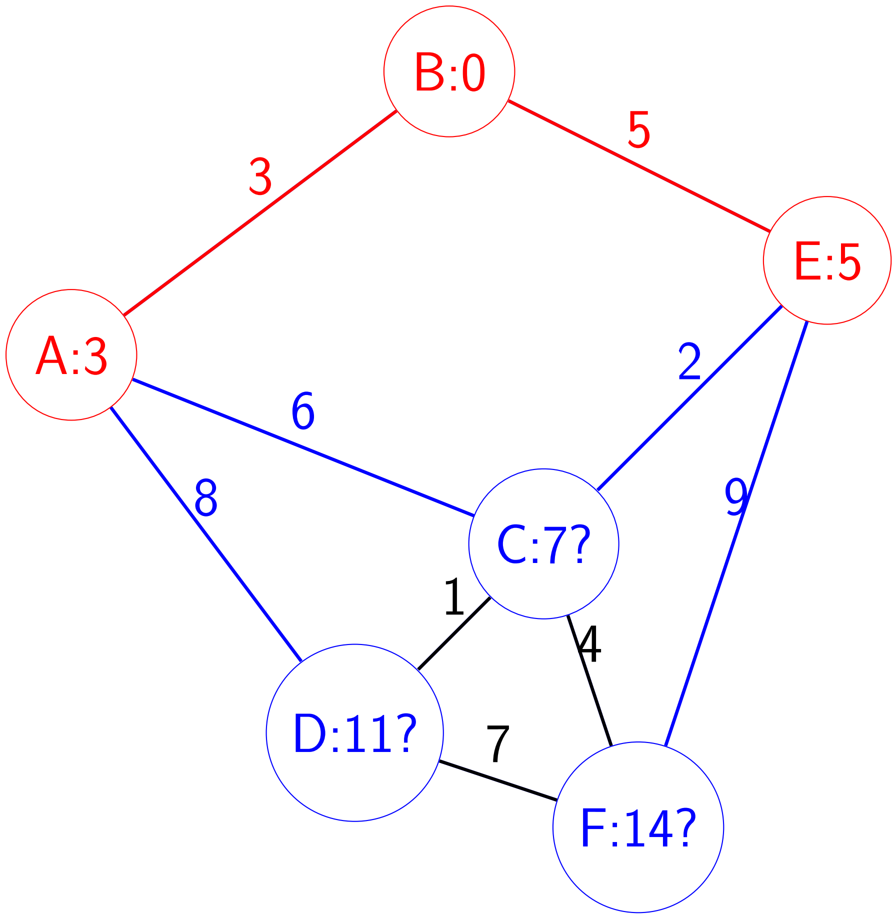


City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞

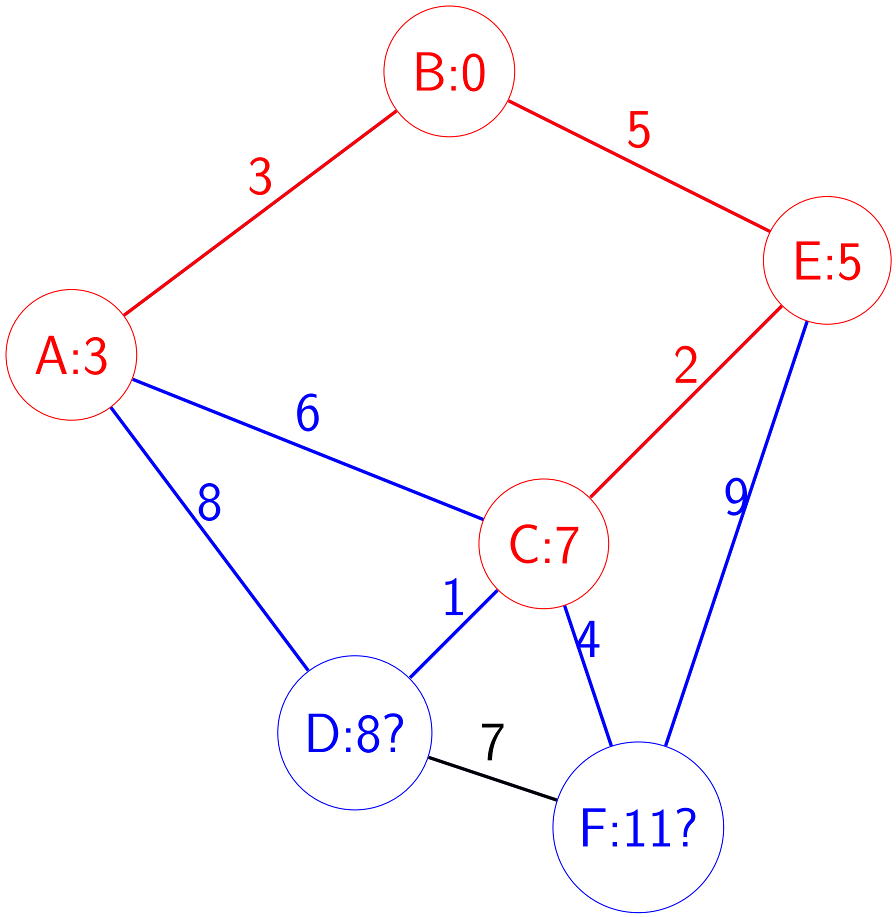


B:0
3
5
E:5?
A:3
2
6
8
C:9?
9
1
4
D:11?
7
F:?

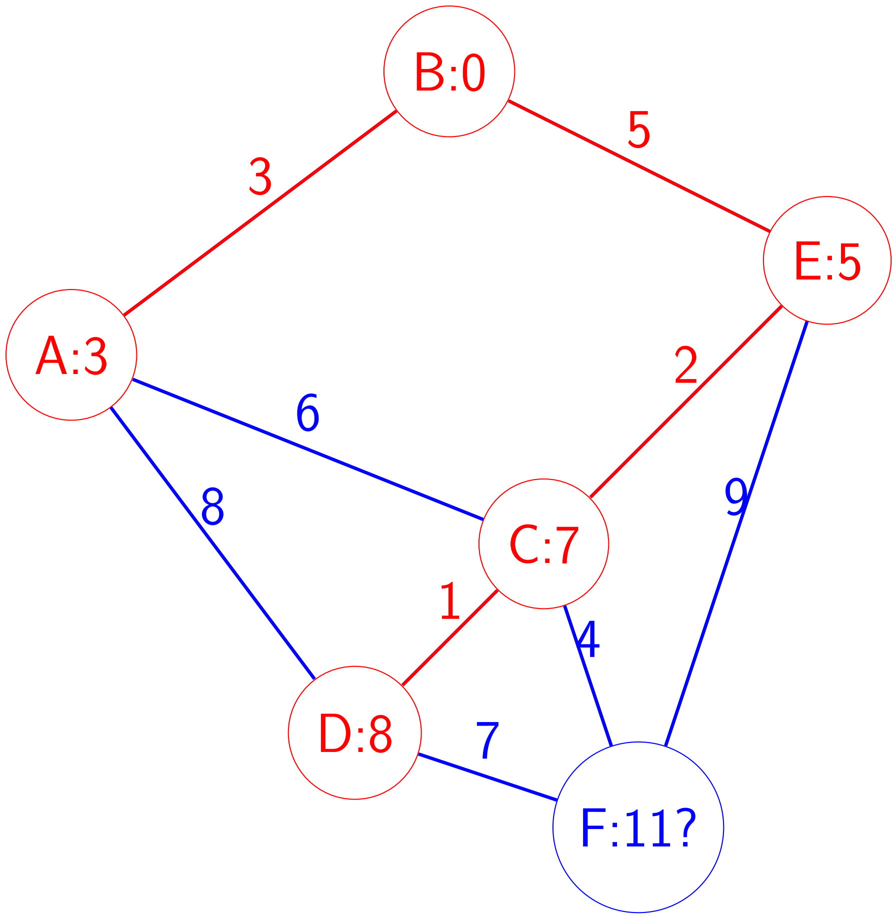
City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞
From A	3	0	9	11	5	∞


B:0
3
5
A:3
E:5
6
2
8
9
C:7?
1
4
D:11?
7
F:14?

City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞
From A	3	0	9	11	5	∞
From E	3	0	7	11	5	14

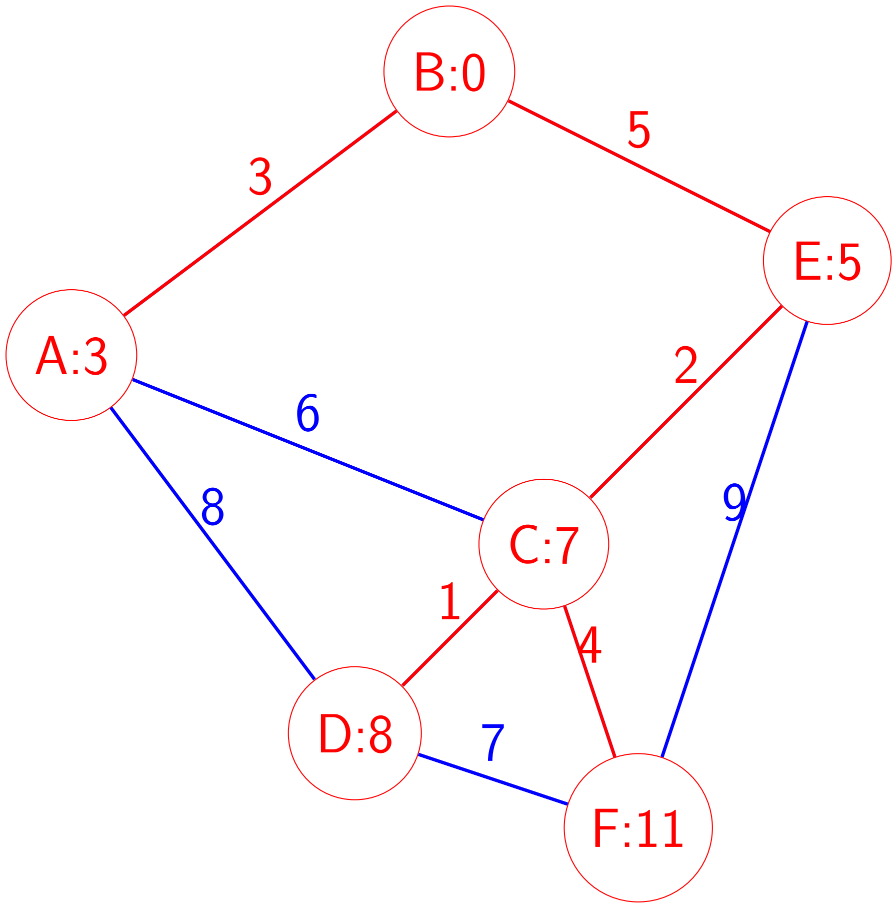

B:0
A:3
E:5
C:7
D:8?
F:11?
3
5
6
2
8
9
1
4
7

City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞
From A	3	0	9	11	5	∞
From E	3	0	7	11	5	14
From C	3	0	7	8	5	11



B:0
A:3
E:5
C:7
D:8
F:11?
3
5
6
2
8
9
1
4
7

City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞
From A	3	0	9	11	5	∞
From E	3	0	7	11	5	14
From C	3	0	7	8	5	14
From D	3	0	7	8	5	11


B:0
3
5
A:3
E:5
2
6
9
8
C:7
1
4
D:8
7
F:11

City	A	B	C	D	E	F
B	∞	0	∞	∞	∞	∞
From B	3	0	∞	∞	5	∞
From A	3	0	9	11	5	∞
From E	3	0	7	11	5	14
From C	3	0	7	8	5	14
From D	3	0	7	8	5	11
Done!	3	0	7	8	5	11

After all that work, we only know the shortest distances for trips that start at city B. To make a complete driving distance table, we need to repeat this process for each possible starting city.

Here's the result for our sample map, with our previous city B results highlighted in red:

	To A	To B	To C	To D	To E	To F
From A	0	3	6	7	8	10
From B	3	0	7	8	5	11
From C	6	7	0	1	2	4
From D	7	8	1	0	3	5
From E	8	5	2	3	0	6
From F	10	11	4	5	6	0

This distance table has some properties that correspond to our ideas of distance in the real world:

- The distance is never negative;
- The distance from a city to itself is always 0;
- The distance from A to B is the same as from B to A;
- The distance from A to B plus the distance from B to C can never be less than the distance from A to C.

Doublets is somewhat like our city distance problem, because we do have a beginning word, an end word that we are trying to reach, and connections from one word to another that we could also think of as roads.

Having a map, and knowing the shortest distance between any pair of words, would be very helpful.

The Doublets game is simpler than the city distance problem, however, because the roads we use don't have different lengths. We count the steps we take in transforming words, so each word we "visit" involves a trip of 1 unit in length.

So for the Doublets problem, determining the shortest distance information can be done in a simpler way.

Suppose we are in a maze of connected rooms, and told to start in one specific room, and to find another “goal” room.

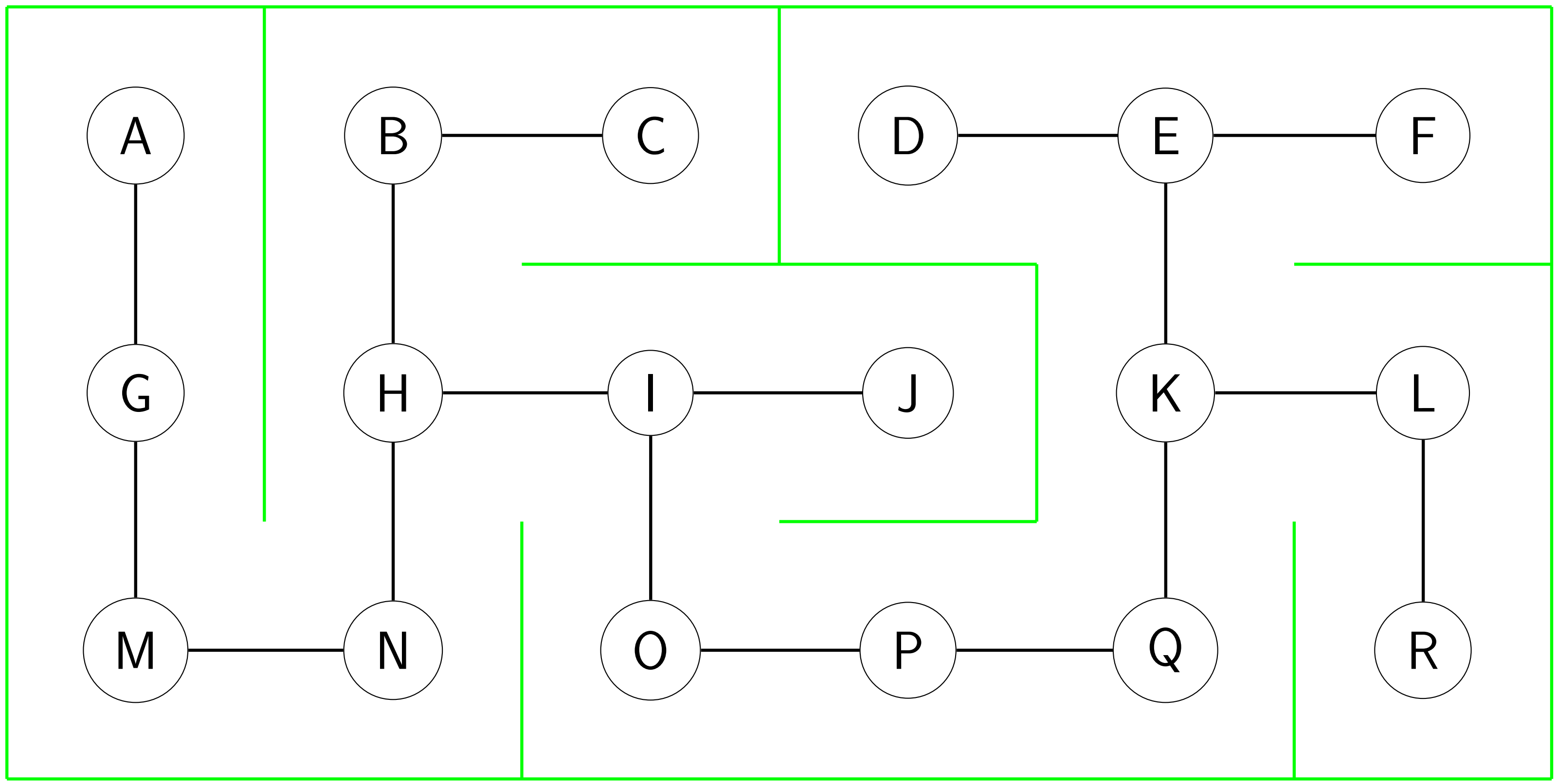
We could seek our goal by aimless wandering, of course.

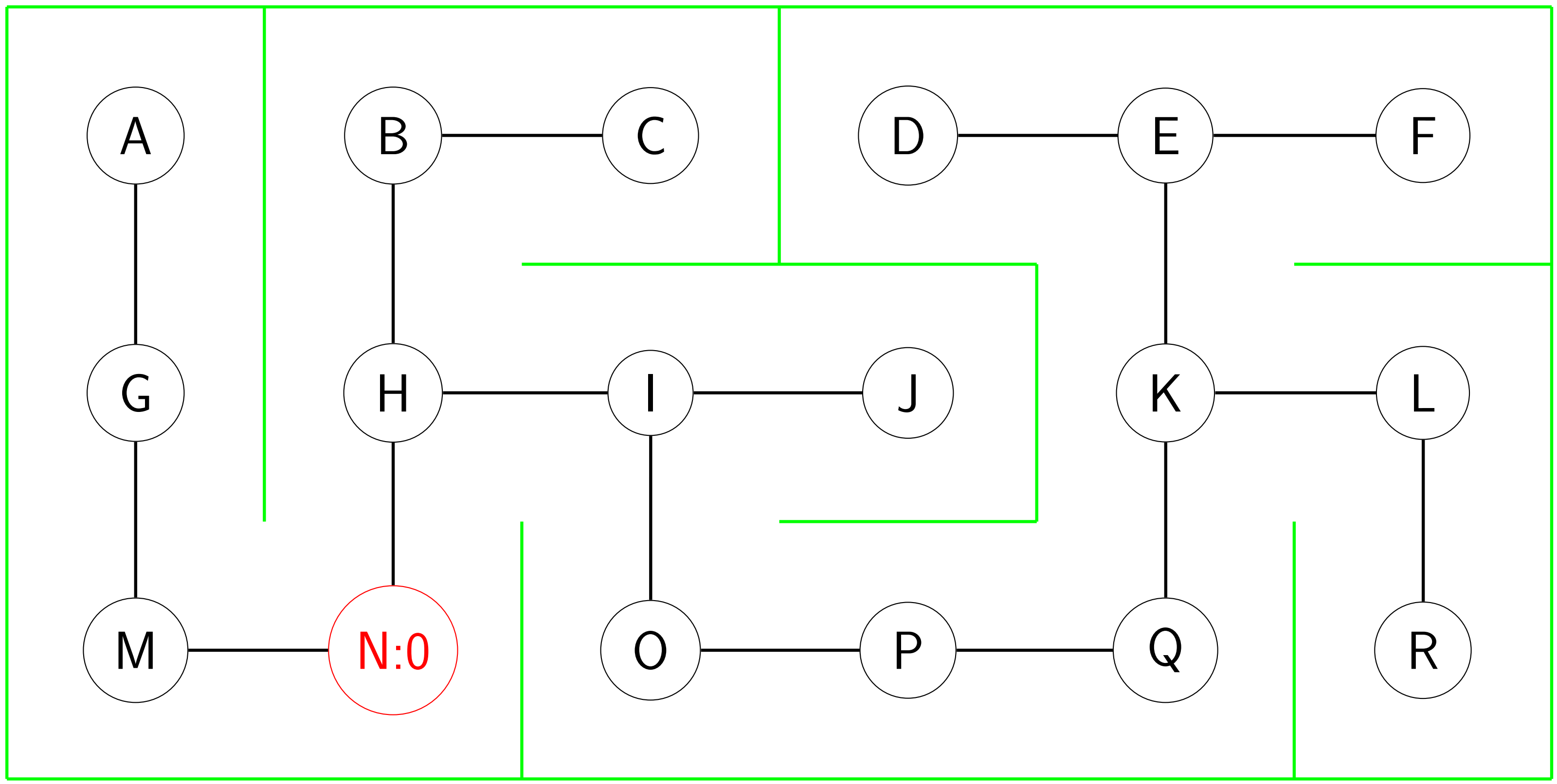
But we can also try a systematic approach, which involves measuring the distance from our starting room to every other room.

We know the starting room has distance 0, of course. Now step into each room immediately connected to the starting room and paint a “1” on the floor.

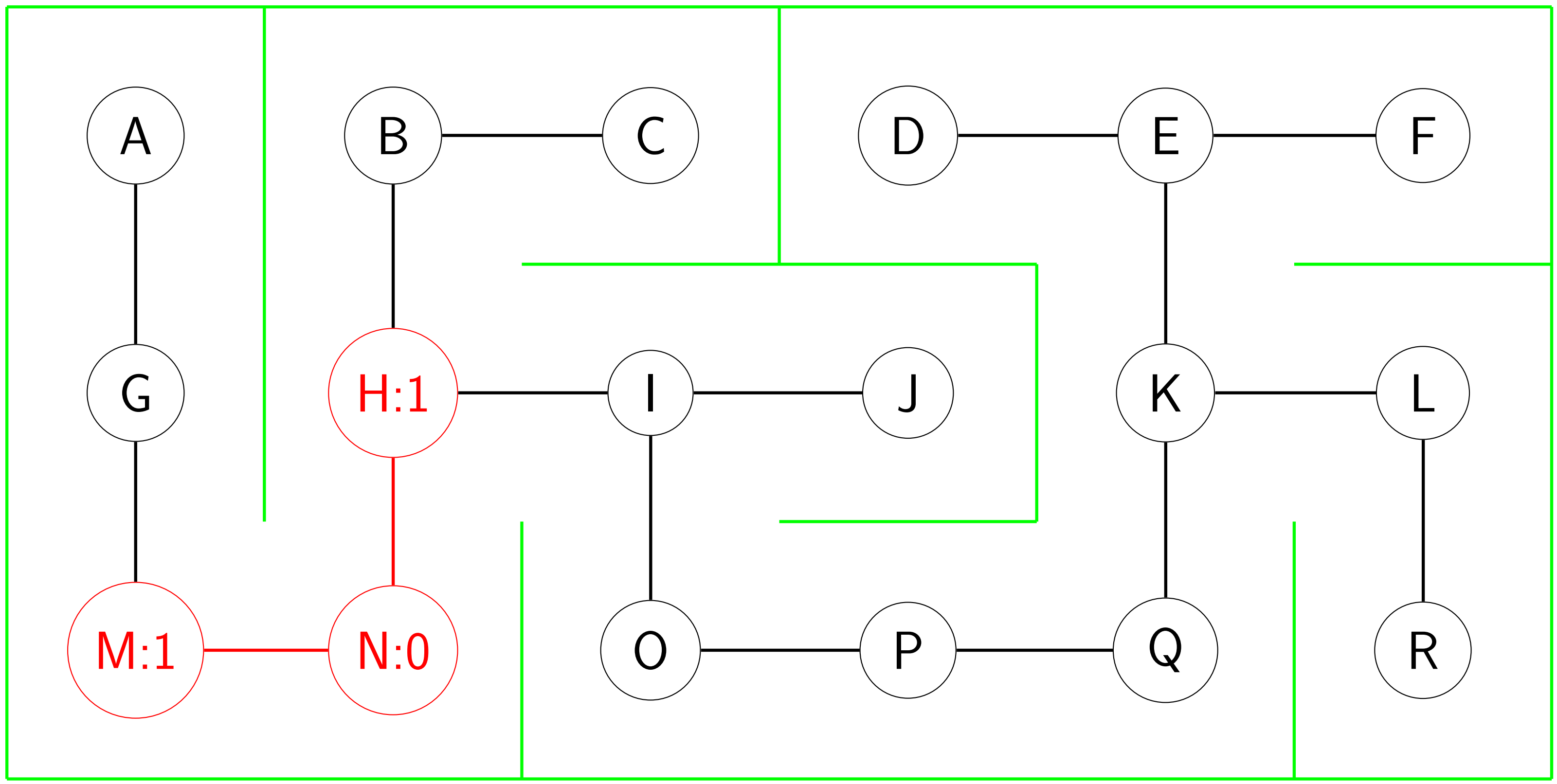
Then, from every “1” room, step into unpainted neighboring rooms and mark them “2”.

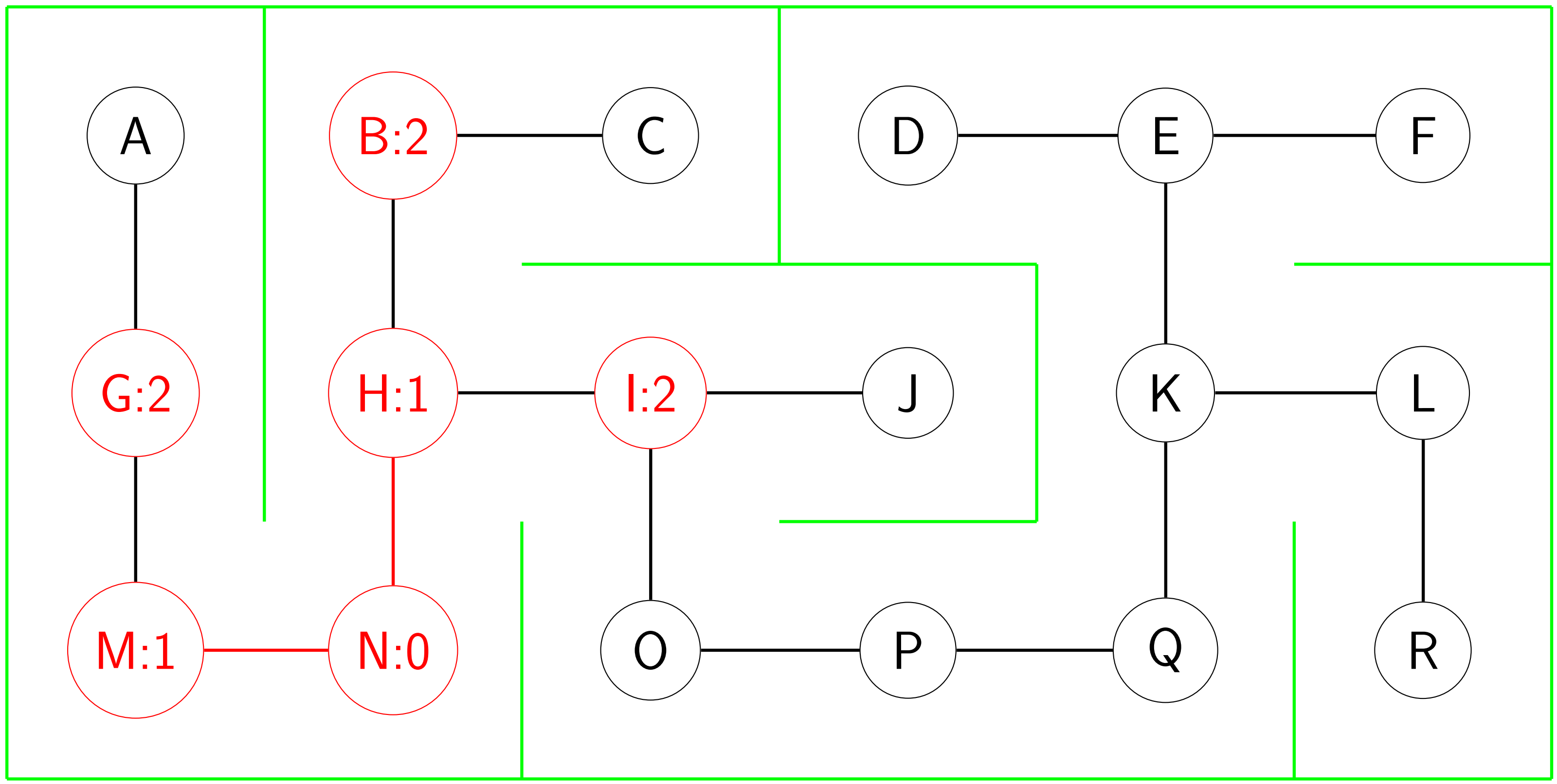
Repeating this process gets you to the goal room, tells you how far the goal room is from the start, and even gives you a trail to follow back to the starting room.

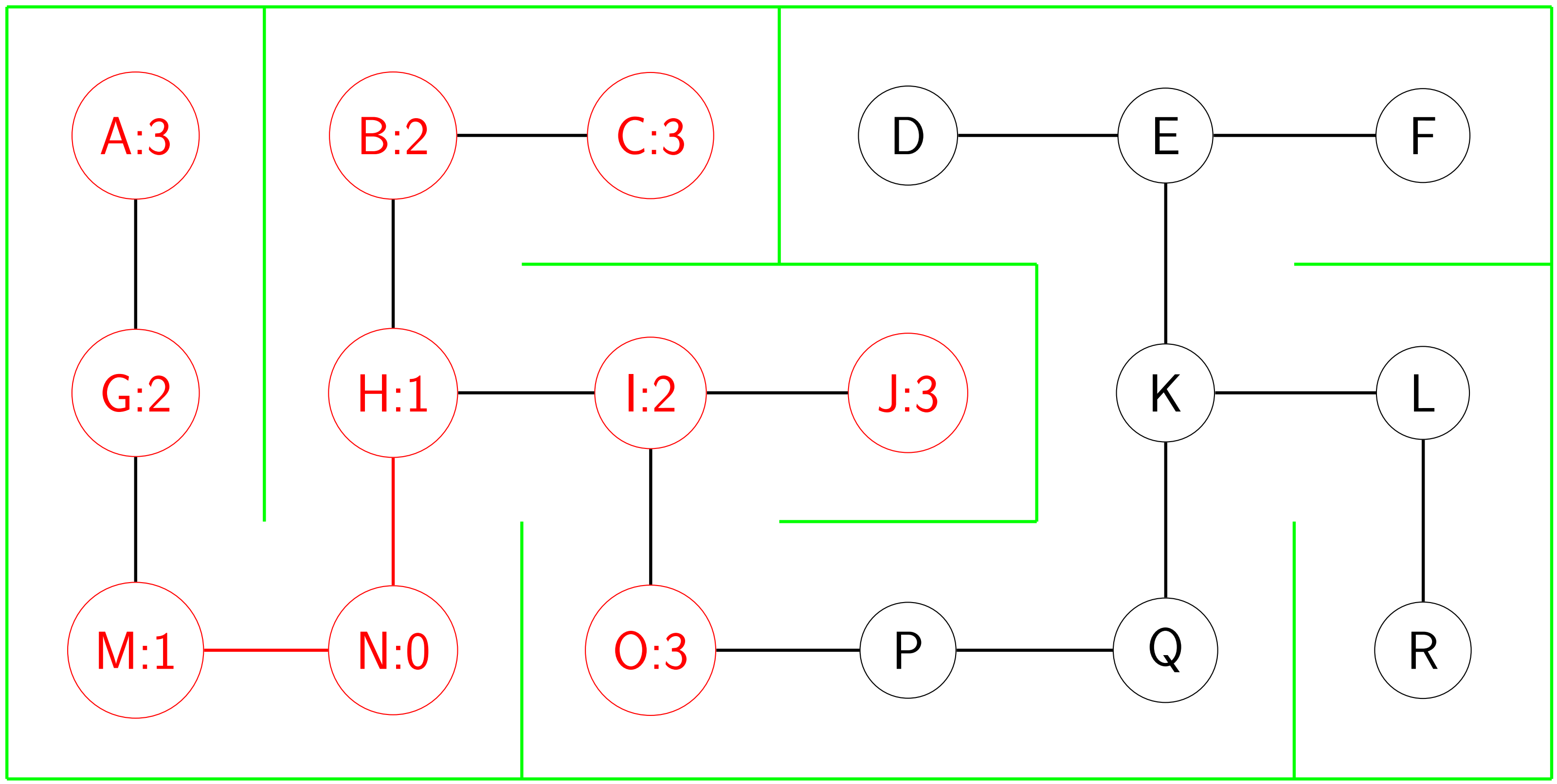

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R

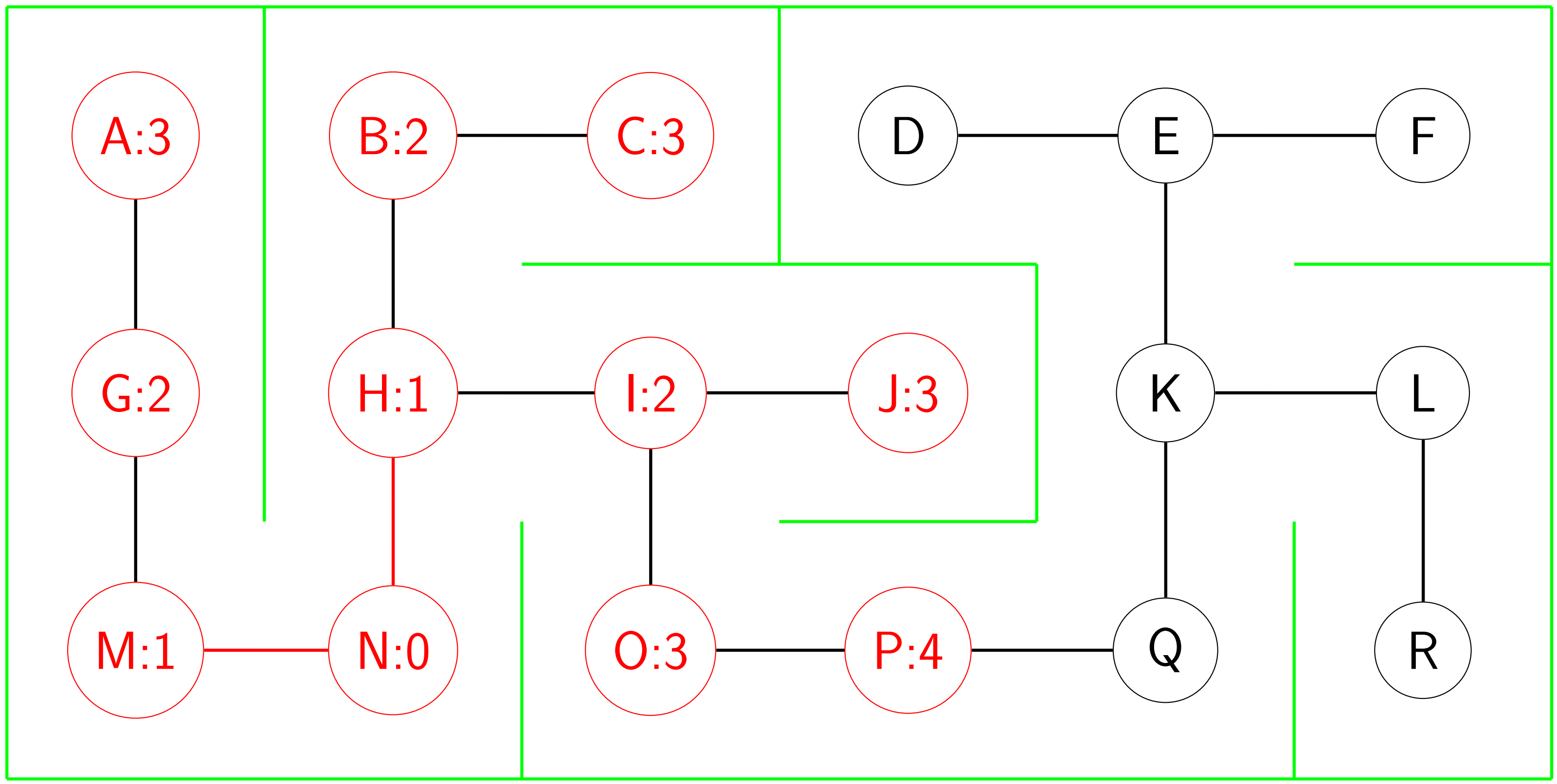


A
B
C
D
E
F
G
H
I
J
K
L
M
N:0
O
P
Q
R

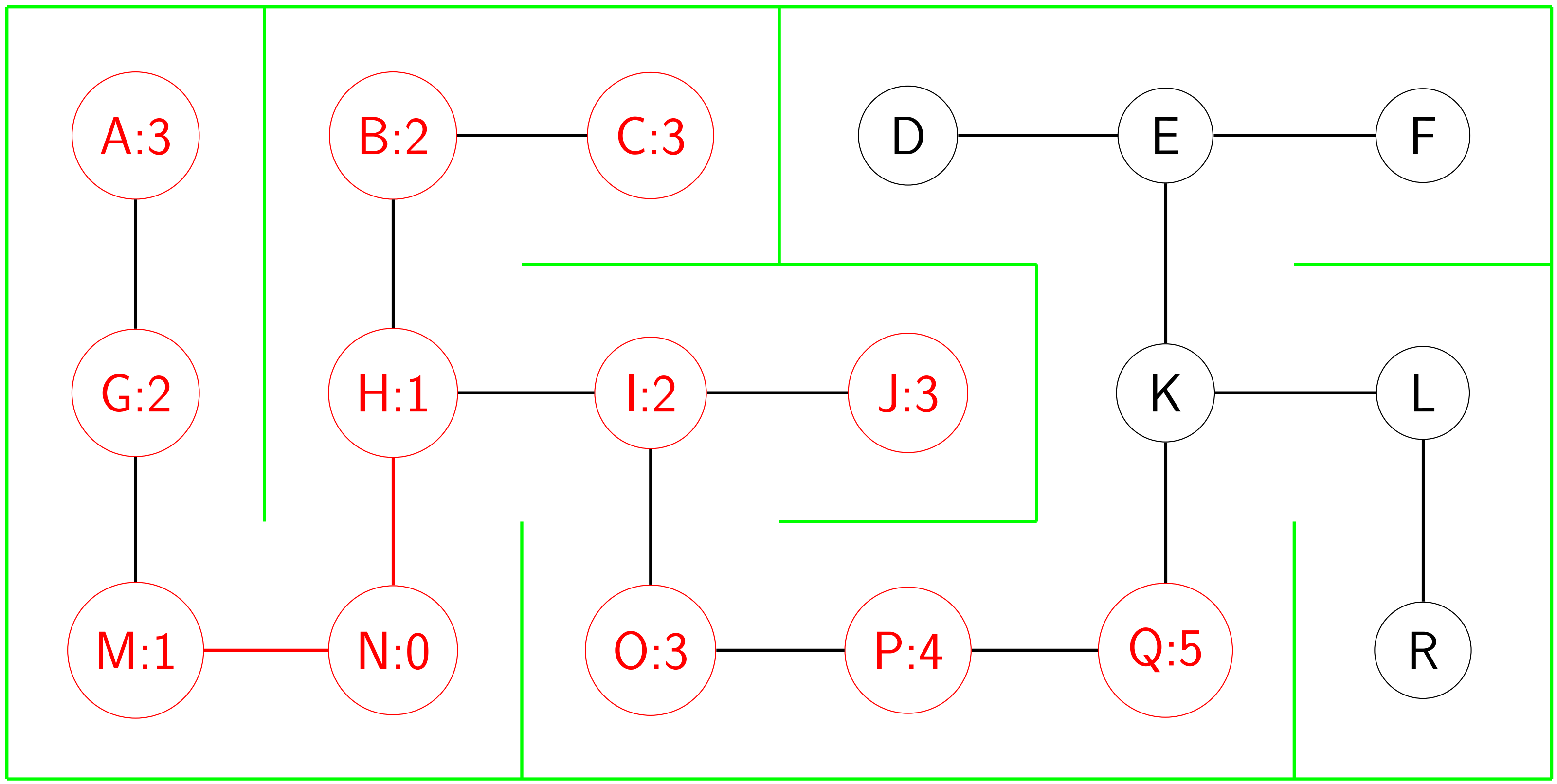

A
B
C
D
E
F
G
H:1
I
J
K
L
M:1
N:0
O
P
Q
R


A
B:2
C
D
E
F
G:2
H:1
I:2
J
K
L
M:1
N:0
O
P
Q
R

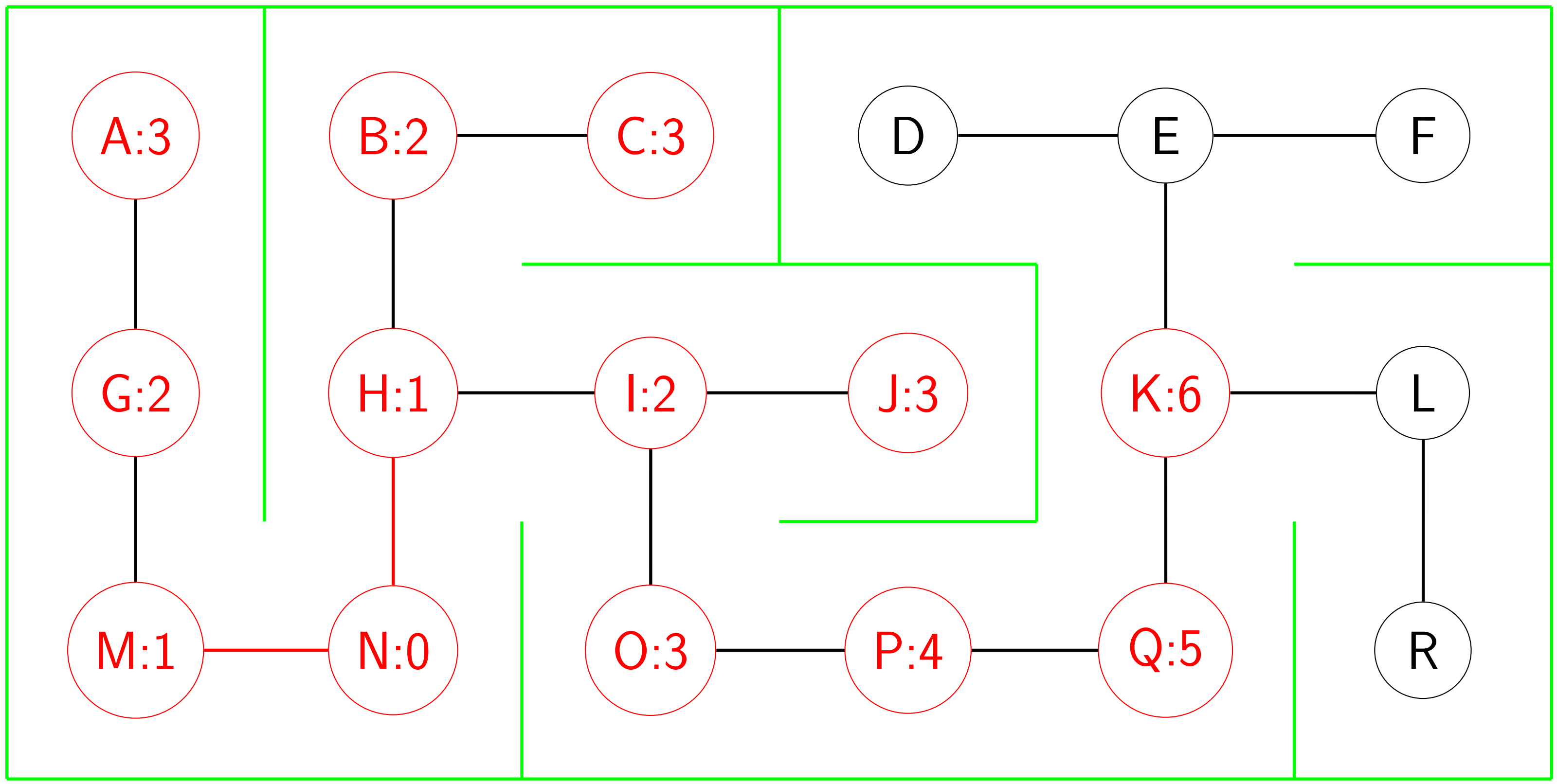

A:3
B:2
C:3
D
E
F
G:2
H:1
I:2
J:3
K
L
M:1
N:0
O:3
P
Q
R



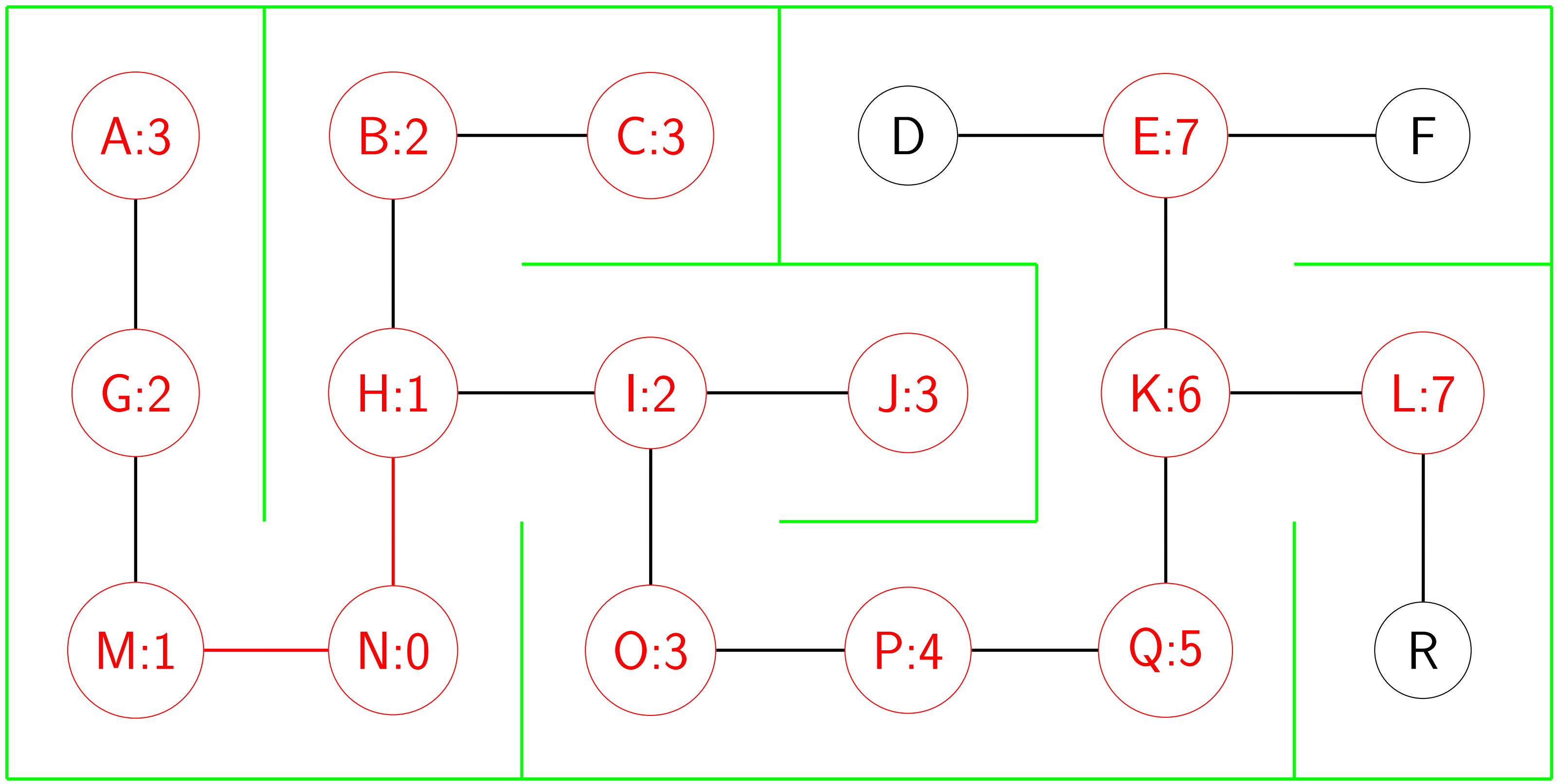
A:3
B:2
C:3
D
E
F
G:2
H:1
I:2
J:3
K
L
M:1
N:0
O:3
P:4
Q
R

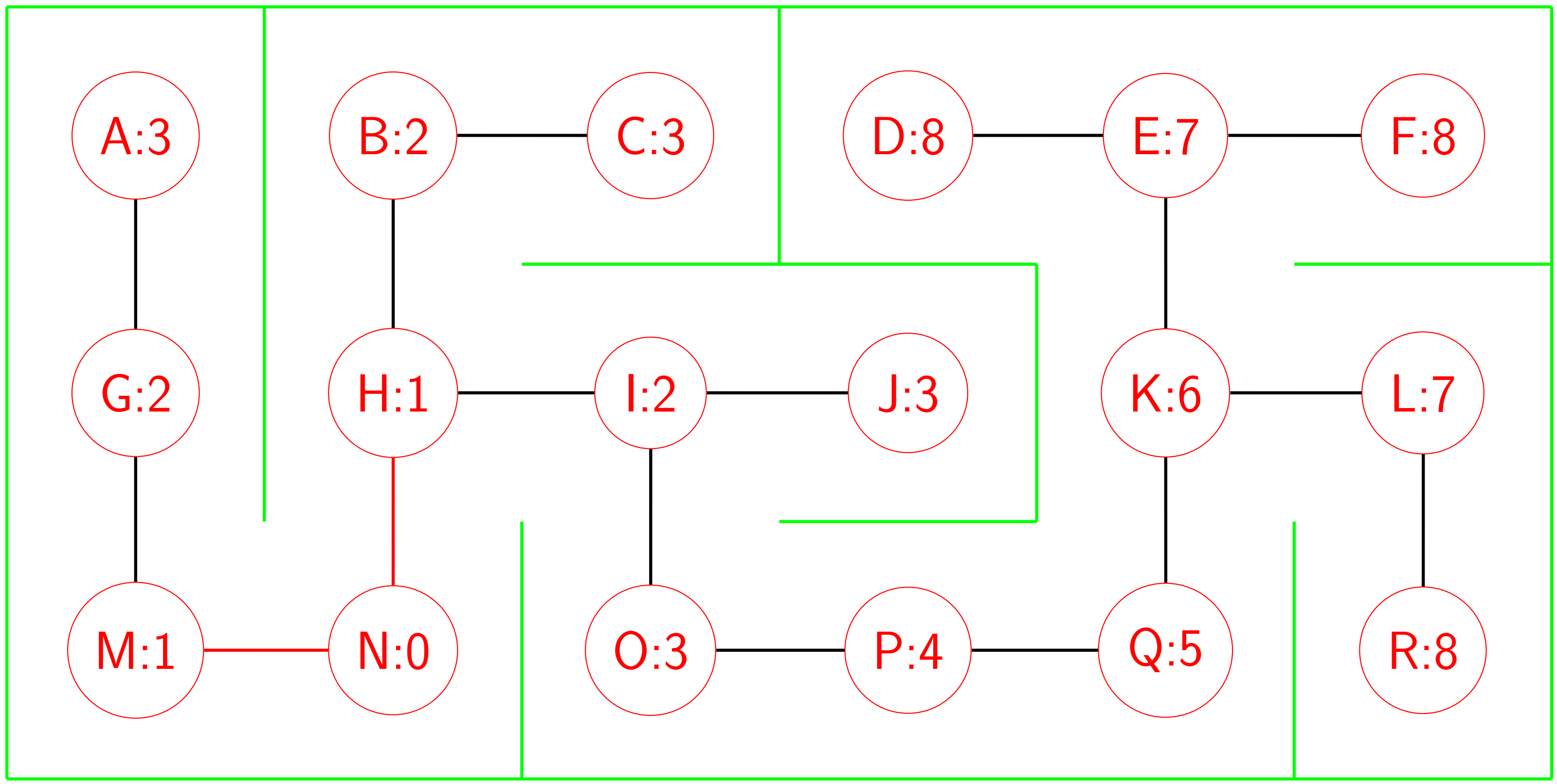


A:3
B:2
C:3
D
E
F
G:2
H:1
I:2
J:3
K
L
M:1
N:0
O:3
P:4
Q:5
R



A:3
B:2
C:3
D
E
F
G:2
H:1
I:2
J:3
K:6
L
M:1
N:0
O:3
P:4
Q:5
R


A:3
B:2
C:3
D
E:7
F
G:2
H:1
I:2
J:3
K:6
L:7
M:1
N:0
O:3
P:4
Q:5
R



Now our diagram of the maze has turned into a shortest distance table for trips that start at position $\mathbf{N}$. This means, for example, that the shortest distance from $\mathbf{N}$ to $\mathbf{K}$ requires 6 steps.

So the shortest distance problem is simpler to work on when the connections or road all have length 1.

We simply pick our starting point, and then all the immediate neighbors are guaranteed to be one unit away.

All their neighbors (if we haven't already seen them) are 2 units away, and so on.

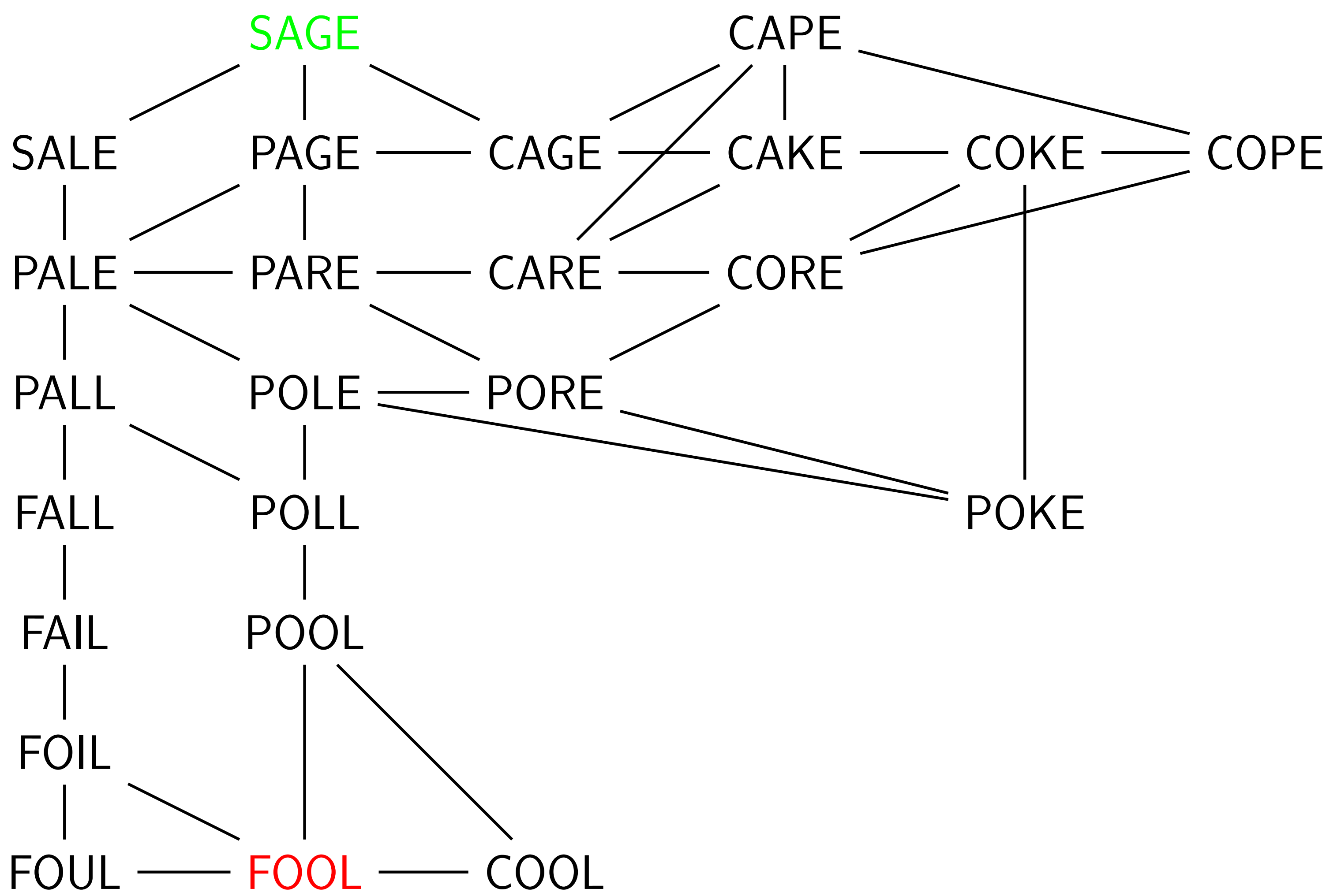
By marking each spot with its distance, we get a table of distances, and we can even work out the path back to our starting point.

Now that we've thought about maps and shortest distances, let's return to our doublets problem and use these ideas.

We can't afford to draw a map of all possible four-letter words, so let's draw a reduced map with a limited vocabulary.

Two words are connected if they differ by a single letter.

We plan to start at one word (SAGE) and try to reach another word (FOOL) and we want to do this in the shortest possible number of steps.

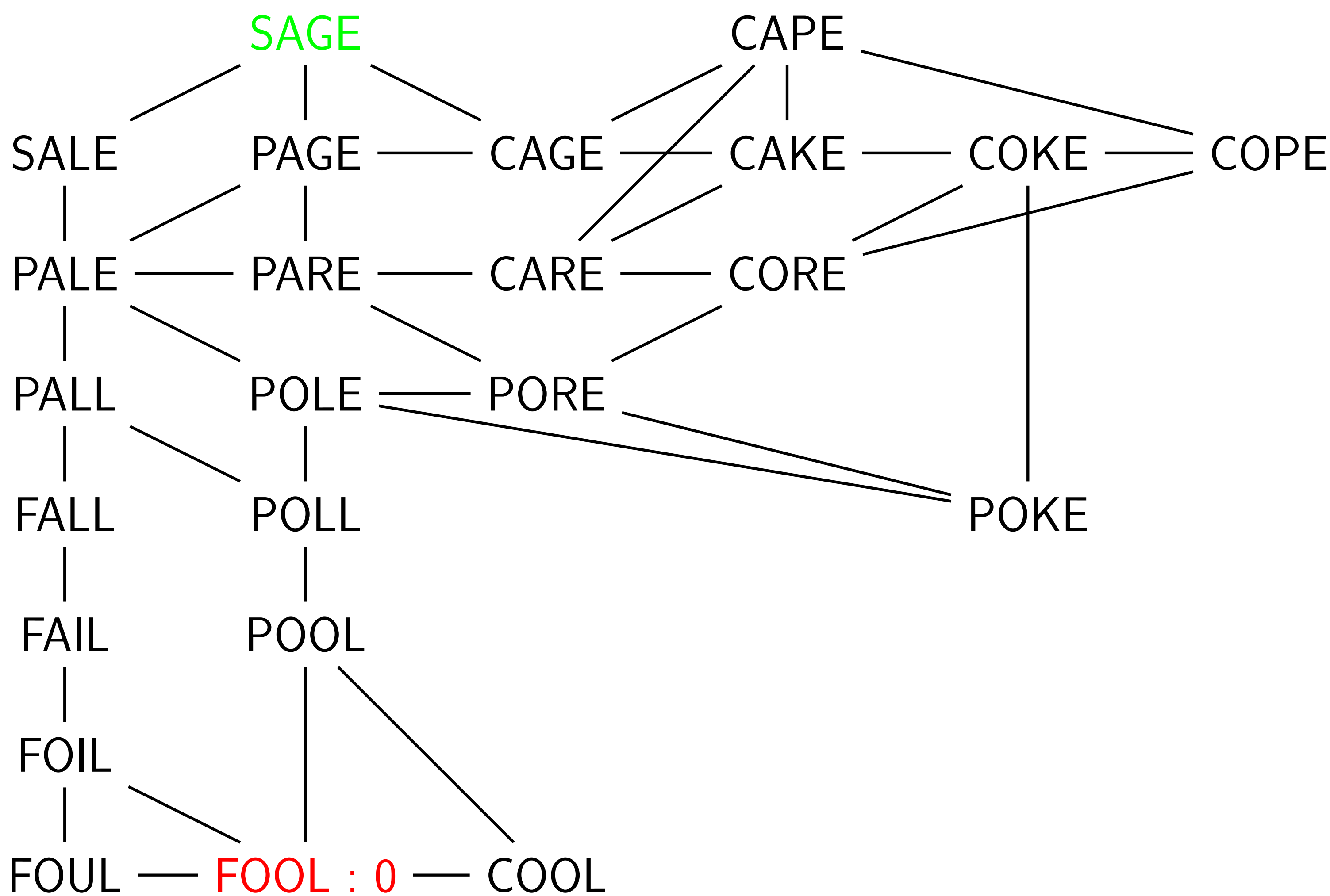

SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE
PARE
CARE
CORE
PALL
POLE
PORE
FALL
POLL
POKE
FAIL
POOL
FOIL
FOUL
FOOL
COOL

Here is a sort of map of our word problem for transforming SAGE to FOOL.

Of course, we have left out many many possible words, but this map gives us some very interesting information.

It shows us that there are many solutions to the problem.

It shows us that there are dead ends, and worthless steps that just lengthen our journey.


SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE
PARE
CARE
CORE
PALL
POLE
PORE
FALL
POLL
POKE
FAIL
POOL
FOIL
FOUL
FOOL : 0
COOL

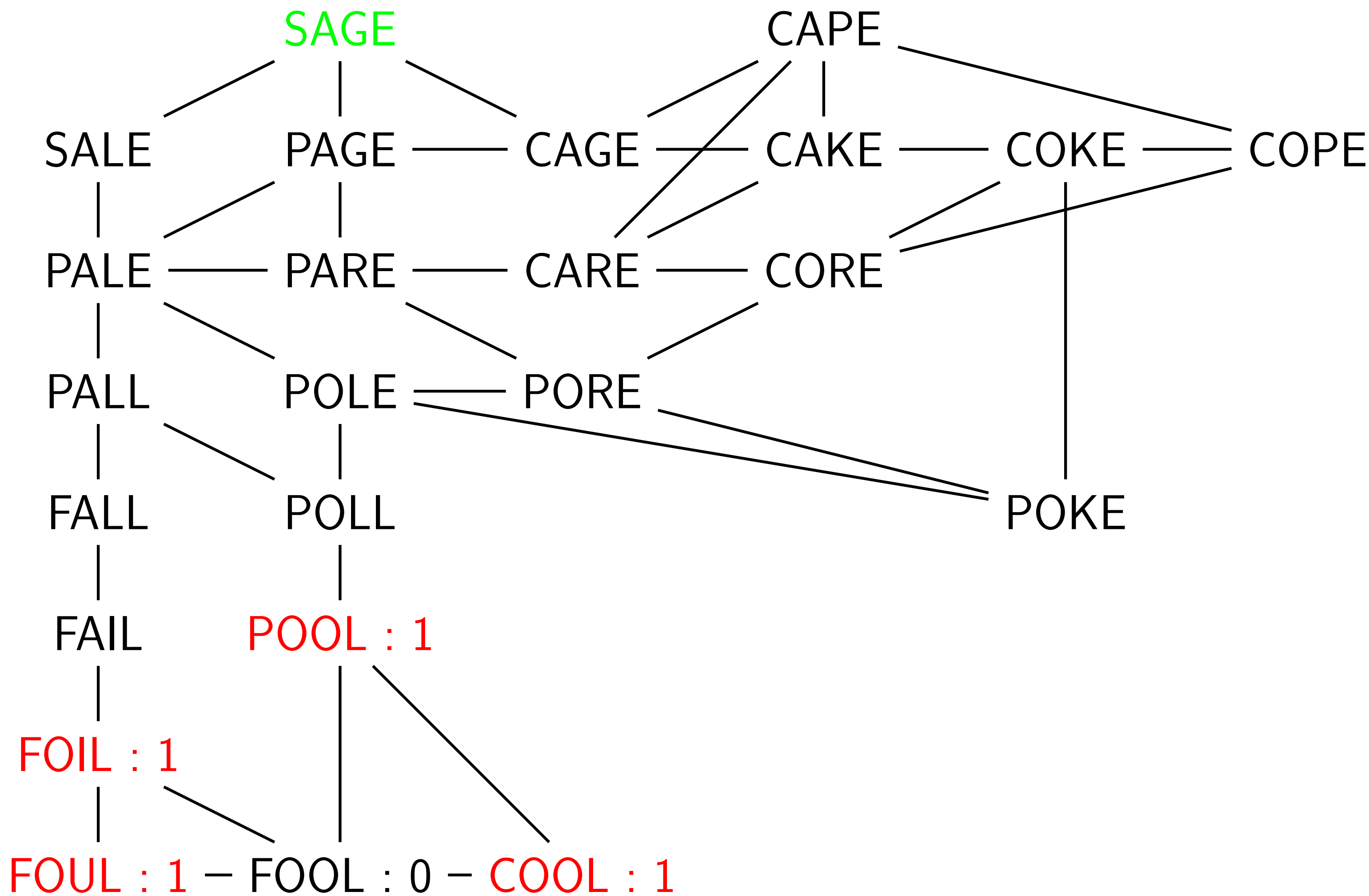
We can even determine the number of steps necessary to transform ANY word into FOOL.

Mark FOOL's distance as "0".

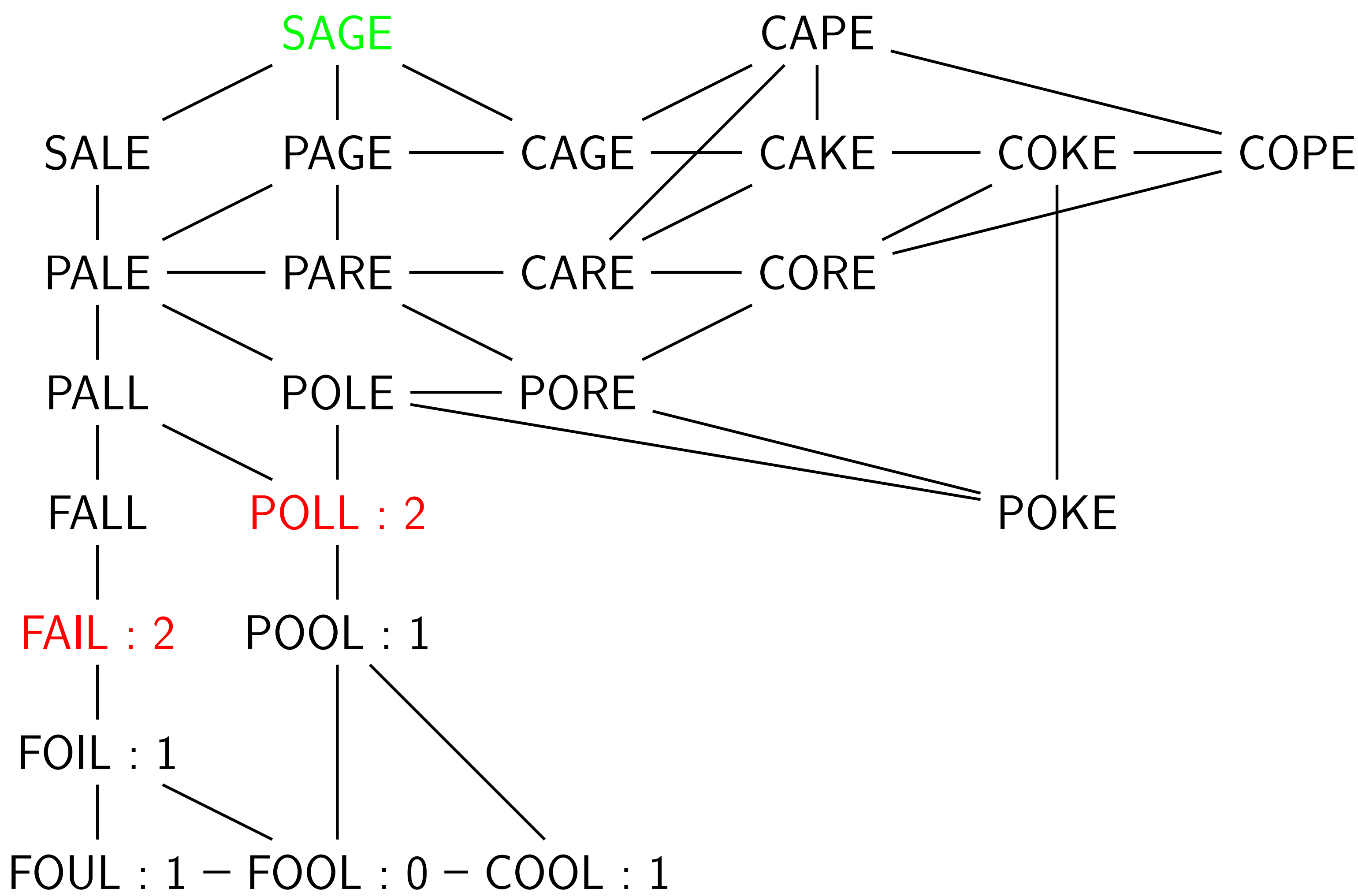
Every word in the map that touches FOOL now has distance 1.

Any unmarked word that touches a word of distance 1 now has distance 2.

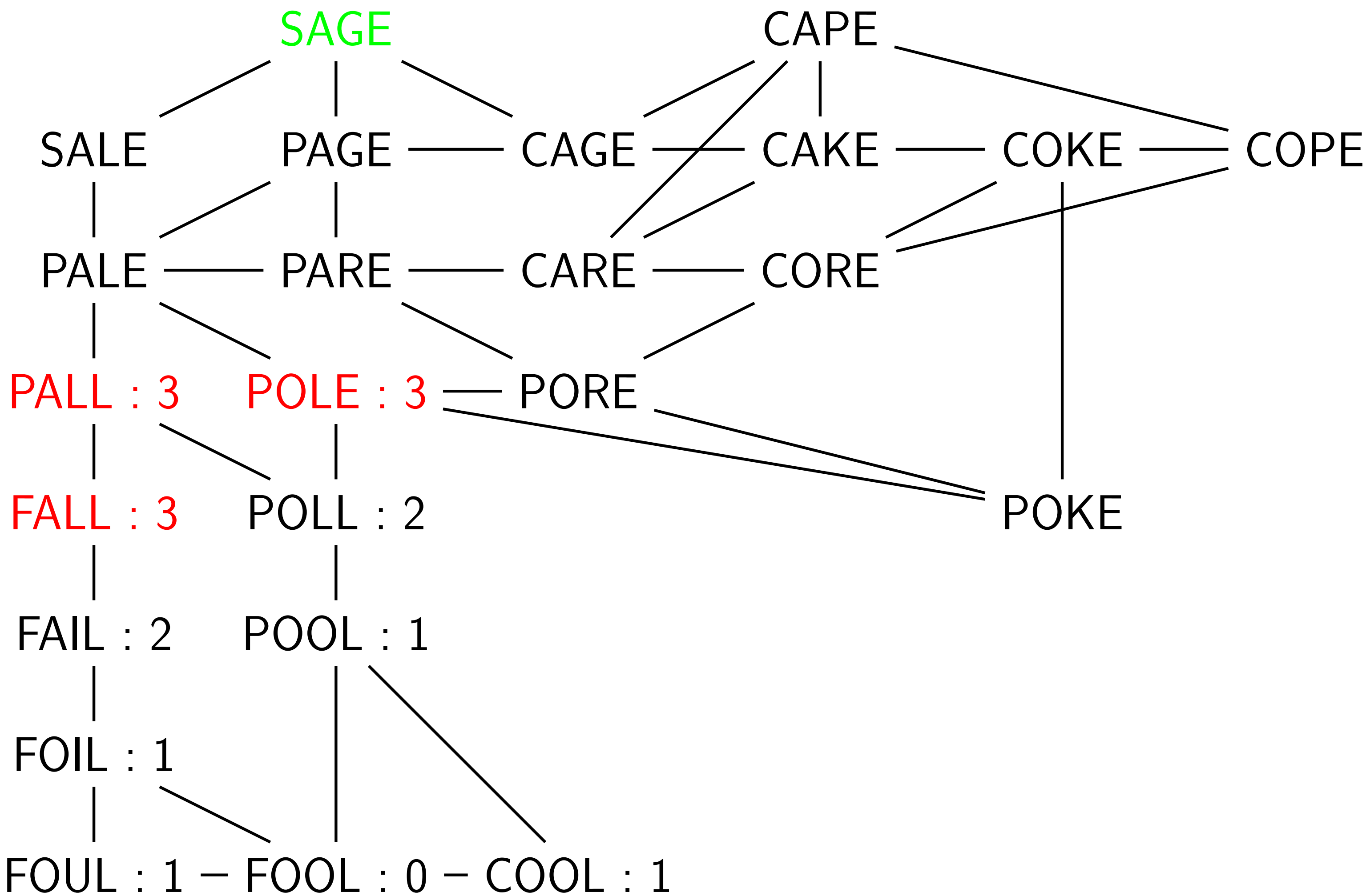
Keep going until you can reach no more words. If any words remain unmarked, you can't transform them to FOOL!


SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE
PARE
CARE
CORE
PALL
POLE
PORE
FALL
POLL
POKE
FAIL
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

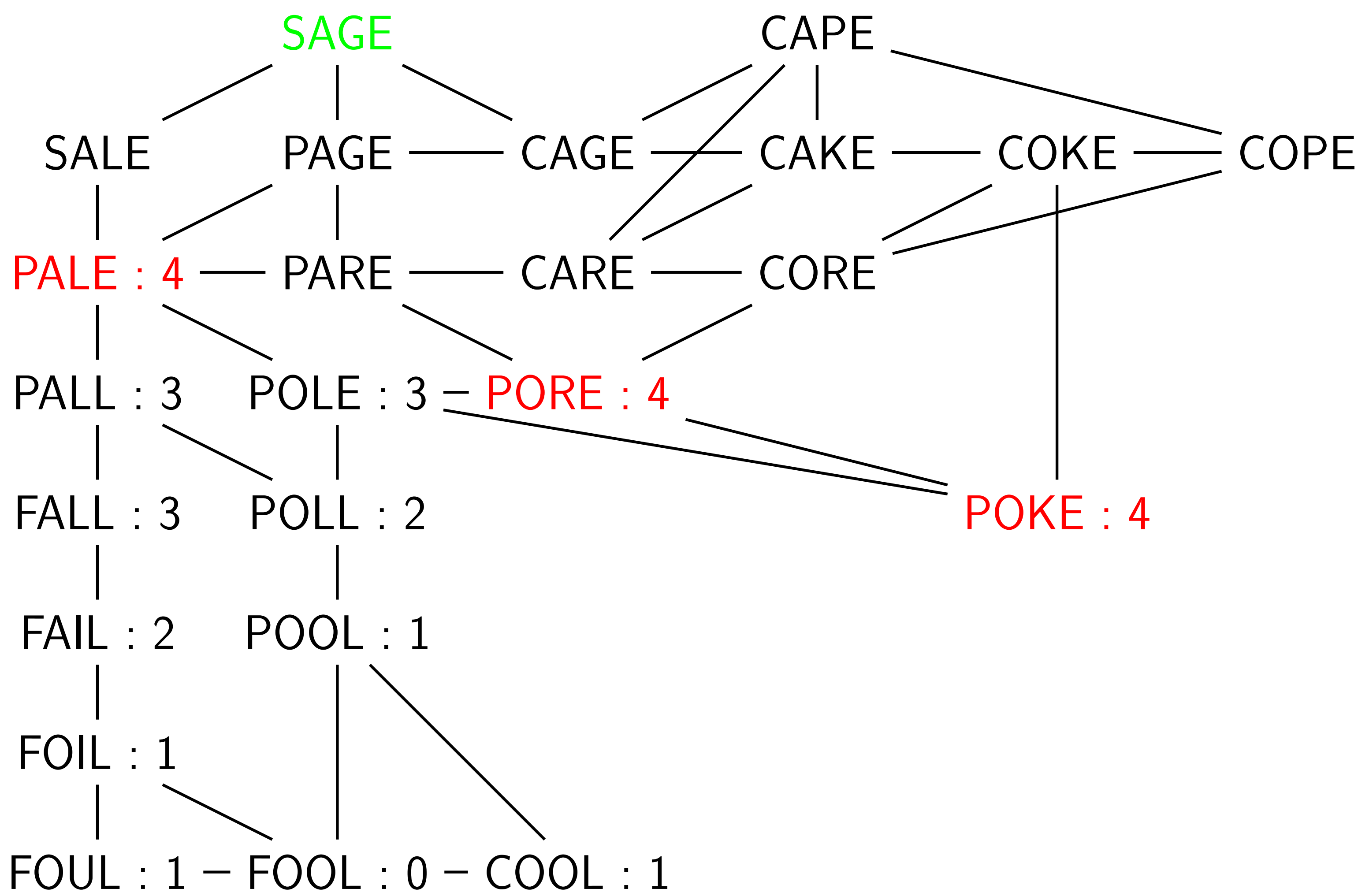
On step 1, we can add FOUL, FOIL, POOL and COOL.


SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE
PARE
CARE
CORE
PALL
POLE
PORE
FALL
POLL : 2
POKE
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

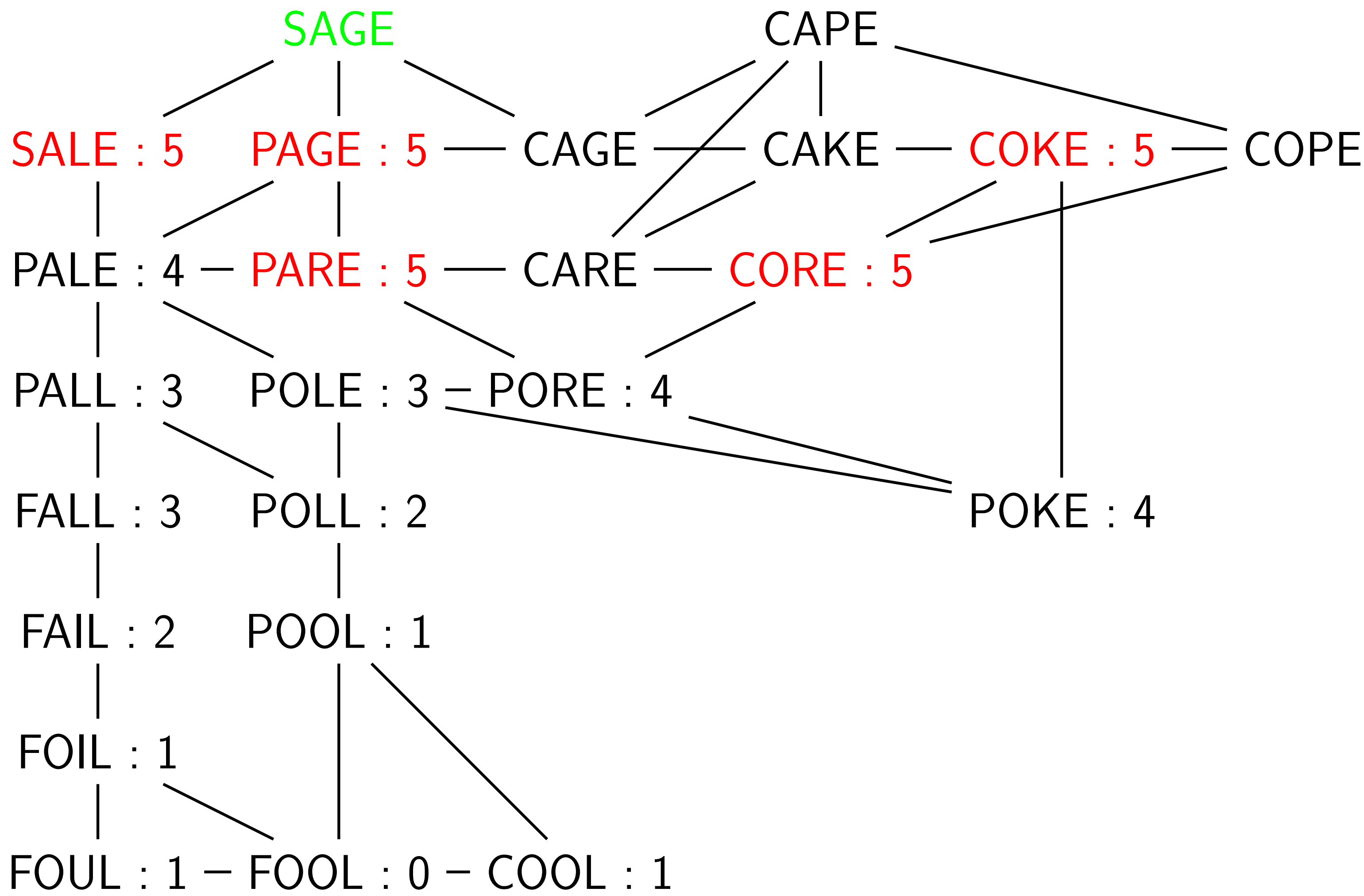
On step 2, we add POLL and FAIL.


SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE
PARE
CARE
CORE
PALL : 3
POLE : 3
PORE
FALL : 3
POLL : 2
POKE
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

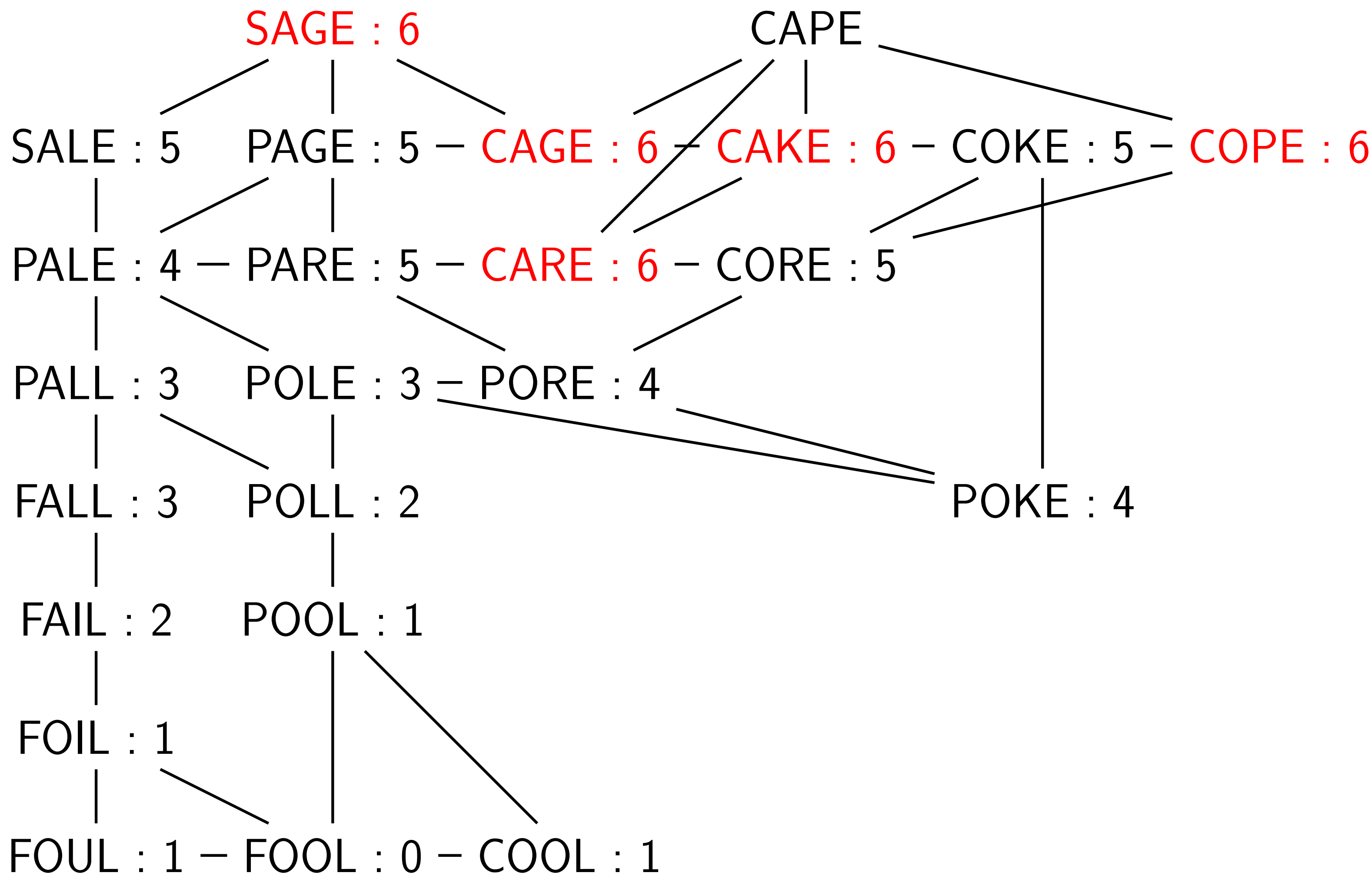
On step 3, we can add PALL, POLL and FALL.


SAGE
CAPE
SALE
PAGE
CAGE
CAKE
COKE
COPE
PALE : 4
PARE
CARE
CORE
PALL : 3
POLE : 3
PORE : 4
FALL : 3
POLL : 2
POKE : 4
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

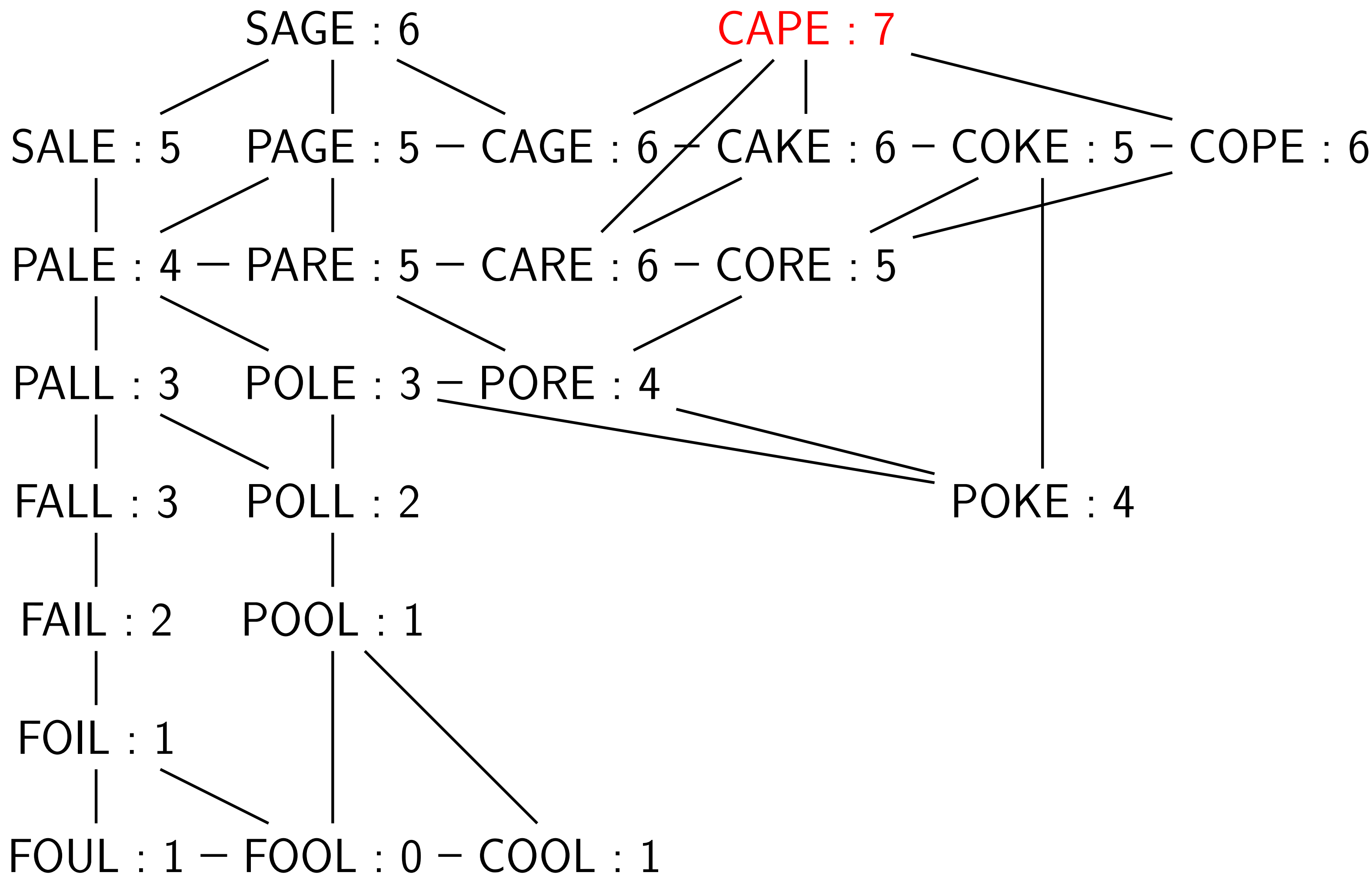
On step 4, we can add PALE, PORE, POKE.


SAGE
CAPE
SALE : 5
PAGE : 5
CAGE
CAKE
COKE : 5
COPE
PALE : 4
PARE : 5
CARE
CORE : 5
PALL : 3
POLE : 3
PORE : 4
FALL : 3
POLL : 2
POKE : 4
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

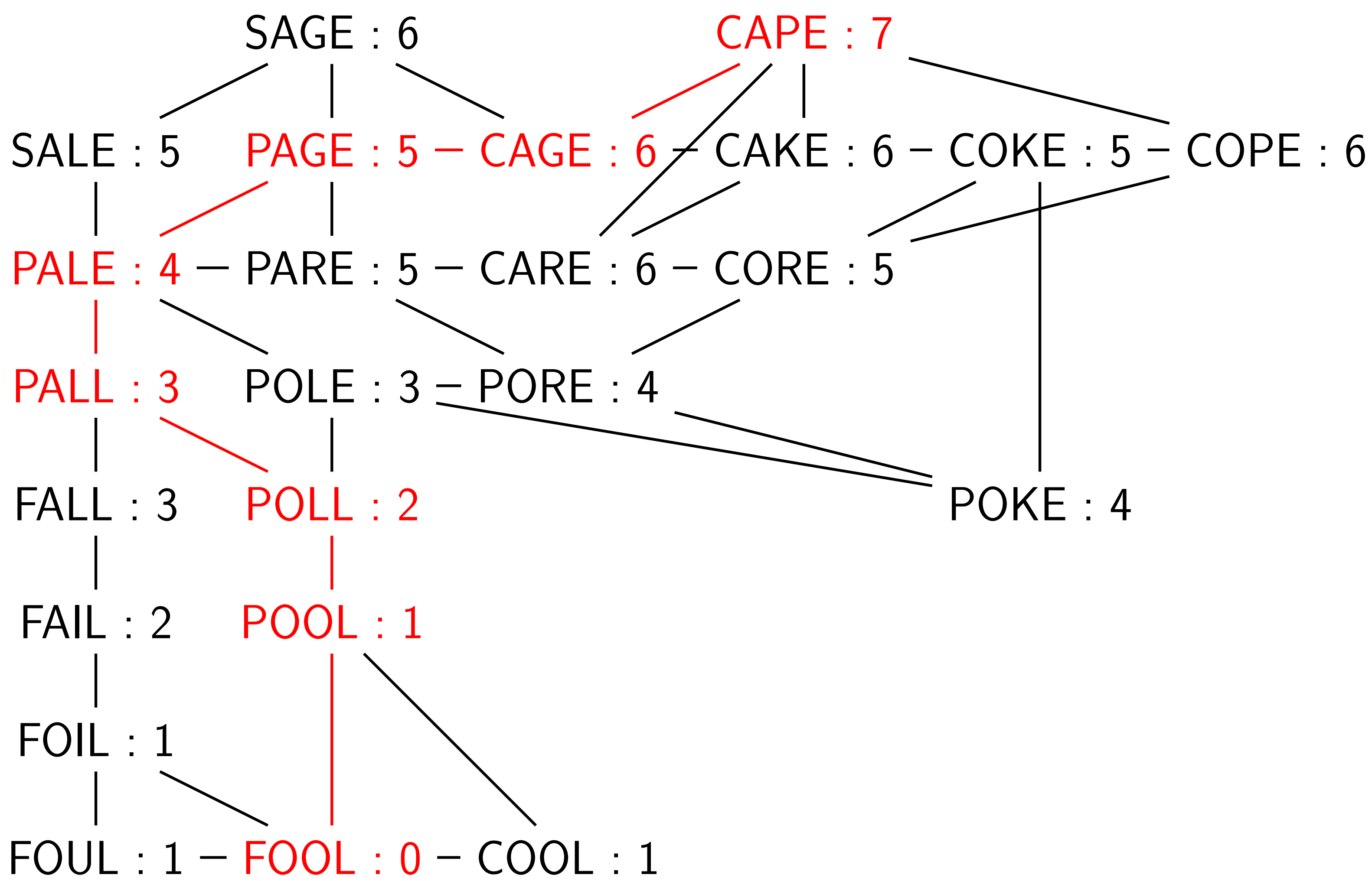
On step 5, we can add SALE, PAGE, COKE, PARE, CORE.


SAGE : 6
CAPE
SALE : 5
PAGE : 5
CAGE : 6
CAKE : 6
COKE : 5
COPE : 6
PALE : 4
PARE : 5
CARE : 6
CORE : 5
PALL : 3
POLE : 3
PORE : 4
FALL : 3
POLL : 2
POKE : 4
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

On step 6, we can add SAGE, CAGE, CAKE, COPE, CARE.


SAGE : 6
CAPE : 7
SALE : 5
PAGE : 5
CAGE : 6
CAKE : 6
COKE : 5
COPE : 6
PALE : 4
PARE : 5
CARE : 6
CORE : 5
PALL : 3
POLE : 3
PORE : 4
FALL : 3
POLL : 2
POKE : 4
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

On step 7, we add CAPE.



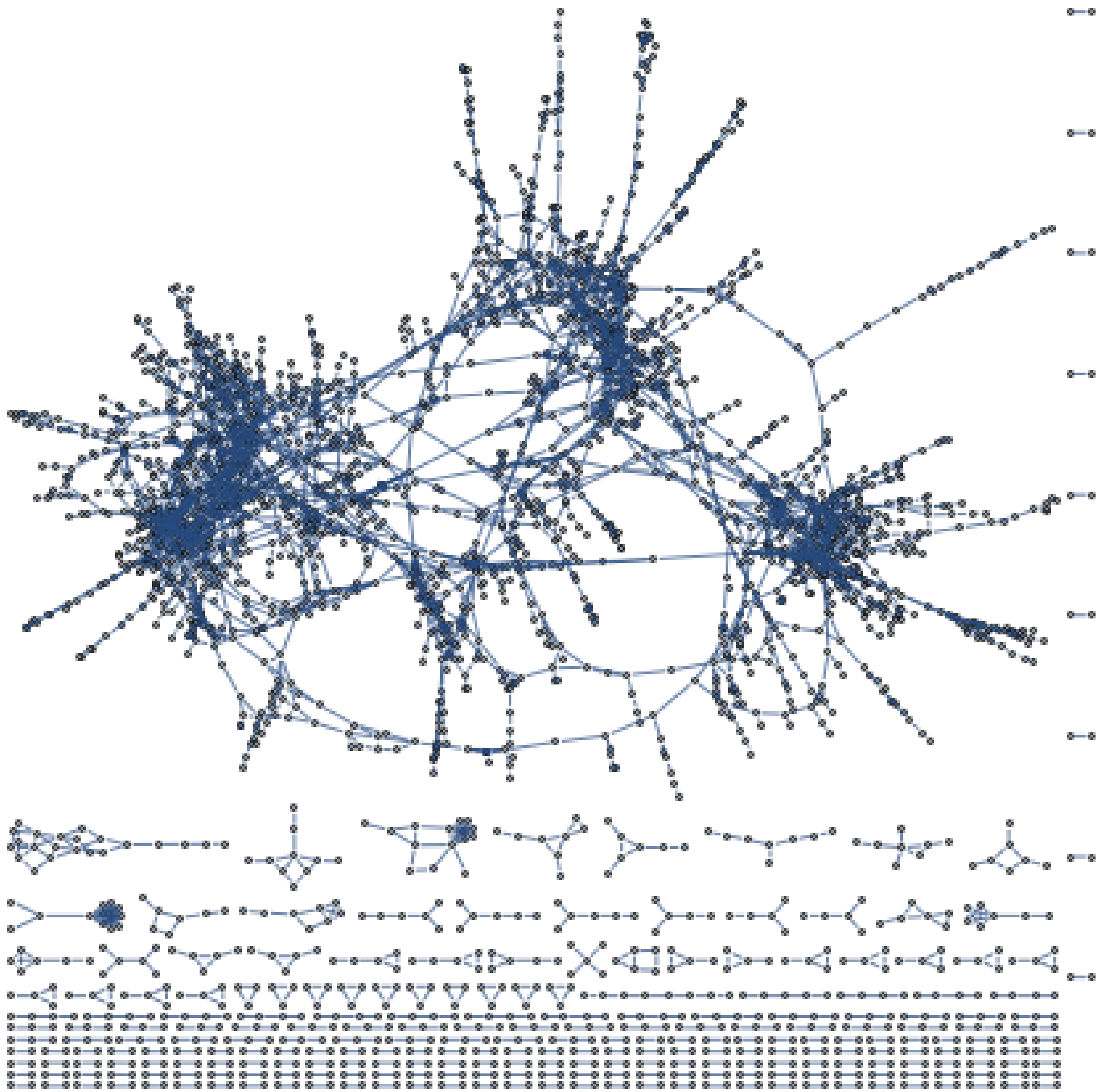
SAGE : 6
CAPE : 7
SALE : 5
PAGE : 5
CAGE : 6
CAKE : 6
COKE : 5
COPE : 6
PALE : 4
PARE : 5
CARE : 6
CORE : 5
PALL : 3
POLE : 3
PORE : 4
FALL : 3
POLL : 2
POKE : 4
FAIL : 2
POOL : 1
FOIL : 1
FOUL : 1
FOOL : 0
COOL : 1

We can use our marked map to determine the transformation of any word into FOOL.

Pick a starting word, such as “CAPE”. It has a distance 7. To find the solution, move to any neighboring word that is one unit closer, and keep doing it til you reach FOOL.

One such path is CAPE, CAGE, PAGE, PALE, PALL, POLL, POOL, FOOL.

Out[4]=



If we were playing Doublets using 5 letter words, and we had a computer, we could make a map of all the connections.

Here is such a map, using more than 5,000 five letter words. In this map, each word appears only as a dot, so we are just seeing the abstract connection pattern.

Most words are connected, although there are some disconnected sets, and even solitary, unconnected words. One of them is **ALOOF**.

You can see a few cases where words are connected but very far apart. One such pair is COMEDY and CHARGE which can be connected using a sequence of 48 words, some of them uncommon.

The fact that we can make such a map means that this is actually a fairly simple problem...for a computer.

Is an algorithm more like Captain Kirk or like Mr Spock?

We have seen two ways to turn SAGE into FOOL.

One way is haphazard - we check to see if we can make a greedy move, otherwise we look at our choices and evaluate them, taking the best and saving the rest.

The other method spends a great deal of time preparing a map, and then calmly says "Go here, then here, then here, and that's the fastest way."

The mathematical, organized method is nice if you can discover it, and have the time to set it up. The one-step at a time, rule-of-thumb approach may not always work, may not be the fastest, but it may be better at handling problems where the data changes, or it's really hard to see the big picture.