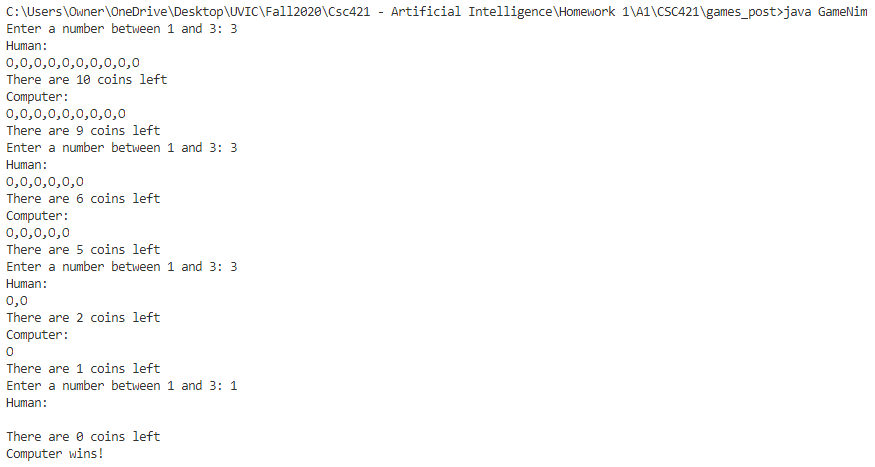
For question 1, we used the same template as TicTacToe, but instead modified to the Nim game, here is the output of the code. (We couldn’t find a way to beat the computer by the way!)



Enter a number between 1 and 3: 3

Human:

O,O,O,O,O,O,O,O,O,O

There are 10 coins left

Computer:

O,O,O,O,O,O,O,O,O

There are 9 coins left

Enter a number between 1 and 3: 3

Human:

O,O,O,O,O,O

There are 6 coins left

Computer:

O,O,O,O,O

There are 5 coins left

Enter a number between 1 and 3: 3

Human:

O,O

There are 2 coins left

Computer:

O

There are 1 coins left

Enter a number between 1 and 3: 1

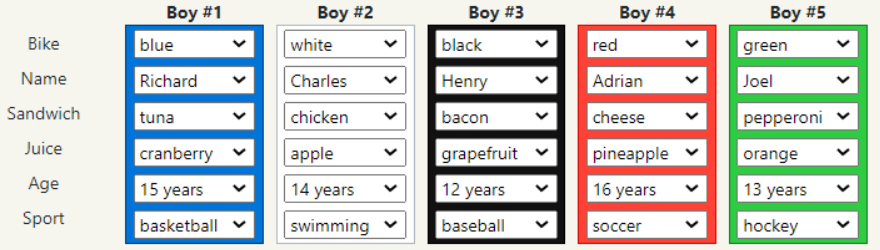
Human:

There are 0 coins left

Computer wins!

Formulation of zebra problem (Bike Riding variation)

I solved the problem in my own time and found the solution.



It is beneficial to get the unary constraints out the way by leveraging the indices of different bike riders.

My strategy here will be to implement A) Unary Constraints B) Binary Constraints

1. Unary Constraints (Based on some definitive index)
2. **In the middle is the boy that likes baseball**

        for(int i=1; i<=5; i++)

            if(i != 3)

                csp.D.get("baseball").remove(i);

1. **In the fifth position is the 13 year old boy**

        for(int i=1; i<=5; i++)

            if(i != 5)

                csp.D.get("13 years").remove(i);

1. **The Boy who likes Hockey is in the fifth position**

        for(int i=1; i<=5; i++)

            if(i != 5)

                csp.D.get("hockey").remove(i);

1. **The Boy who rides a black bicycle is at the third position**

        for(int i=1; i<=5; i++)

            if(i != 3)

                csp.D.get("black").remove(i);

1. **The boy who is going to drink pineapple juice is at the fourth position**

for(int i=1; i<=5; i++)

            if(i != 4)

                csp.D.get("pineapple").remove(i);

1. **Tuna is at eaten on one of the ends**

   for(int i=1; i<=5; i++){

            if((i != 1)){

                if (i!= 5){

                    csp.D.get("tuna").remove(i);

                }

            }

        }

1. **Green bike is rode on one of the ends**

        for(int i=1; i<=5; i++){

            if((i != 1)){

                if (i!= 5){

                    csp.D.get("green").remove(i);

                }

            }

        }

This concludes the unary constraints, we now may formulate what else is needed via the binary constraints which are bit more complex and tricky, however some are still quite simple

1. Binary constraints
2. **The 16-year-old brought Cheese sandwich.**

        if(X.equals("16 years") && Y.equals("cheese") && !x.equals(y)){

            return false;

        }

1. **Henry is exactly to the left of the soccer fan**

        if(X.equals("Henry") && Y.equals("soccer") && (Integer)x-(Integer)y!=-1){

            return false;

        }

1. **Joel is next to the 16 year old cyclist**

if(X.equals("Joel") && Y.equals("16 years") && Math.abs((Integer)x-(Integer)y)!=1){

            return false;

        }

1. **The one who likes Swimming is next to the friend who likes Baseball**

if(X.equals("swimming") && Y.equals("baseball") && Math.abs((Integer)x-(Integer)y)!=1){

            return false;

        }

1. **The baseball fan is next to the one who drinks apple juice**

if(X.equals("baseball") && Y.equals("apple") && Math.abs((Integer)x-(Integer)y)!=1){

            return false;

        }

1. **The boy who likes the sport on ice (hockey) will eat pepperoni**

        if(X.equals("hockey") && Y.equals("pepperoni") && !x.equals(y)){

            return false;

        }

1. **The boy who is going to eat Bacon sandwich is somewhere to the right of the owner of the White bicycle.**

        if(X.equals("white") && Y.equals("bacon") && ((Integer)x-(Integer)y) <= 0){

            return false;

        }

“In between cases’: From here on out, there is data given to us saying that a certain variable value is between two domain values, so we split the binary constraints intuitively to bound the restriction.

For example

* The owner of the White bike is somewhere between the 15-year-old boy and the youngest boy, in that order.

This implies two things:

* the White bike belongs to a bike rider strictly to the right of the bike rider who is 15 (lower)
* the White bike belongs to a bike rider strictly to the left of the bike rider who is 12 (upper)

So, we created two binary constraint in these cases, representing two bidirectional arcs for each one of these “in between” cases. We can do this lower or higher bound strategies for all the info we are given with “in between” cases.

There is also a third case implied here

* The blue 15 year old rider comes strictly before the 12 year old rider

So for each “in between” statement, we have three cases to consider

        //The owner of the white bike is in between the 15 year old boy and 12 year old boy

        //A: Lower Bound isGood

        if(X.equals("15 years") && Y.equals("white") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("12 years") && Y.equals("white") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: 15 before 12

        if(X.equals("15 years") && Y.equals("12 years") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //The boy who is going to drink Grapefruit juice

        //is somewhere between who brought Tuna sandwich and who brought Pineapple juice, in that order.

        //A: Lower Bound isGood

        if(X.equals("tuna") && Y.equals("grapefruit") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("pineapple") && Y.equals("grapefruit") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: tuna before pineapple juice

        if(X.equals("tuna") && Y.equals("pineapple") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //The boy riding the White bike

        //is somewhere between the boys riding the blue and the black bicycles, in that order

        //A: Lower Bound isGood

        if(X.equals("blue") && Y.equals("white") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("black") && Y.equals("white") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: Blue before

black

        if(X.equals("blue") && Y.equals("black") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //The cyclist that brought Pineapple juice

        //is somewhere between the 14-year-old and the boy that brought Orange juice, in that order

        //A: Lower Bound isGood

        if(X.equals("14 years") && Y.equals("pineapple") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("orange") && Y.equals("pineapple") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: 14 year old

before orange juice

        if(X.equals("14 years") && Y.equals("orange") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //The 12-year-old is somewhere between the 14-year-old and the oldest boy, in that order.

        //A: Lower Bound isGood

        if(X.equals("14 years") && Y.equals("12 years") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("16 years") && Y.equals("12 years") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: 14 years before 16 years

        if(X.equals("14 years") && Y.equals("16 years") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //The cyclist riding the White bike

        //is somewhere between Richard and the boy riding the Red bike, in that order

        //A: Lower Bound isGood

        if(X.equals("Richard") && Y.equals("white") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("red") && Y.equals("white") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: Richard before red bike

        if(X.equals("Richard") && Y.equals("red") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //Charles is somewhere between Richard and Adrian, in that order.

        //A: Lower Bound isGood

        if(X.equals("Richard") && Y.equals("Charles") && (Integer)x-(Integer)y >= 0){

            return false;

        }

        //B: Higher Bound isGood

        if(X.equals("Adrian") && Y.equals("Charles") && (Integer)x-(Integer)y <= 0){

            return false;

        }

        //C: Implied relationship between lower and upper bound: Richard comes before Adrian

        if(X.equals("Richard") && Y.equals("Adrian") && (Integer)x-(Integer)y >= 0){

            return false;

        }

**Bidirectional Arcs: Binary Constraints**

   //binary constraints: add constraint arcs

        //The owner of the White bike is somewhere between the 15-year-old boy and the youngest boy, in that order.

        csp.addBidirectionalArc("15 years", "white");

        csp.addBidirectionalArc("white", "12 years");

        csp.addBidirectionalArc("15 years", "12 years");

        //Henry is exactly to the left of the Soccer fan.

        csp.addBidirectionalArc("Henry", "soccer");

        //The boy who is going to drink Grapefruit juice is

        //somewhere between who brought Tuna sandwich and who brought Pineapple juice, in that order.

        csp.addBidirectionalArc("tuna", "grapefruit");

        csp.addBidirectionalArc("grapefruit", "pineapple");

        csp.addBidirectionalArc("tuna", "pineapple");

        //The one who likes Swimming is next to the friend who likes Baseball.

        csp.addBidirectionalArc("swimming", "baseball");

        //The cyclist that brought Pineapple juice

        //is somewhere between the 14-year-old and the boy that brought Orange juice, in that order.

        csp.addBidirectionalArc("14 years", "pineapple");

        csp.addBidirectionalArc("pineapple", "orange");

        csp.addBidirectionalArc("14 years", "orange");

        //The boy who likes the sport played on ice is going to eat Pepperoni sandwich.

        csp.addBidirectionalArc("hockey", "pepperoni");

        //The boy riding the White bike

        //is somewhere between the boys riding the blue and the black bicycles, in that order.

        csp.addBidirectionalArc("blue", "white");

        csp.addBidirectionalArc("white", "black");

        csp.addBidirectionalArc("blue", "black");

        //Joel is next to the 16-year-old cyclist.

        csp.addBidirectionalArc("Joel", "16 years");

        //Adrian is exactly to the left of the boy who is going to eat Pepperoni sandwich.

        csp.addBidirectionalArc("Adrian", "pepperoni");

        //The 12-year-old is somewhere between the 14-year-old and the oldest boy, in that order

        csp.addBidirectionalArc("14 years", "12 years");

        csp.addBidirectionalArc("12 years", "16 years");

        csp.addBidirectionalArc("14 years", "16 years");

        //The boy who is going to eat Bacon sandwich

        //is somewhere to the right of the owner of the White bicycle.

        csp.addBidirectionalArc("bacon", "white");

        //The 16-year-old brought Cheese sandwich.

        csp.addBidirectionalArc("cheese", "16 years");

        //The cyclist riding the White bike is somewhere

        //between Richard and the boy riding the Red bike, in that order.

        csp.addBidirectionalArc("Richard", "white");

        csp.addBidirectionalArc("white", "red");

        csp.addBidirectionalArc("Richard", "red");

        //The Baseball fan is next to the boy who is going to drink Apple juice.

        csp.addBidirectionalArc("baseball", "apple");

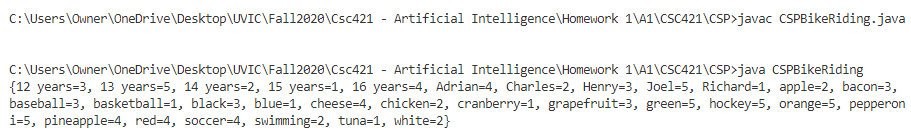
        //Charles is somewhere between Richard and Adrian, in that order

        csp.addBidirectionalArc("Richard", "Charles");

        csp.addBidirectionalArc("Charles", "Adrian");

        csp.addBidirectionalArc("Richard", "Adrian");

**Finally,** after running we get the output



{12 years=3, 13 years=5, 14 years=2, 15 years=1, 16 years=4, Adrian=4, Charles=2, Henry=3, Joel=5, Richard=1, apple=2, bacon=3, baseball=3, basketball=1, black=3, blue=1, cheese=4, chicken=2, cranberry=1, grapefruit=3, green=5, hockey=5, orange=5, pepperoni=5, pineapple=4, red=4, soccer=4, swimming=2, tuna=1, white=2}

