Grace McClurg & Katsumi Cook: Determining the Braid Index of Alternating inks

```
In[ ]:= homeScreen[]
```

```
Out[ ]= NotebookObject[ Untitled-1 ]
```

Directions: Determine Orientations throughout Link and Correct Vector

General Cases of M1, M2, and B Links

```
(*Returns the opposite of the input direction,
which is either "L" for left or "R" for right.*)
opposite[direction_] := If[direction == "L", Return["R"], Return["L"]]
```

```
In[ ]:= opposite["R"]
```

```
In[ ]:= opposite["L"]
```

```
(*given the sequence representing the tangle,
and the incoming orientations of the lower left strand and
 the lower right strand in the tangle, outputs the directions list
 of the tangle. twoIncomingOrientations must be input as a list.*)
directionsB[sequence_, twoIncomingOrientations_] := Module[
   {dirList = {}, i = 1},

   dirList = {{opposite[twoIncomingOrientations[[2]]],
      twoIncomingOrientations[[2]], opposite[twoIncomingOrientations[[1]]]}};
   Return[directionsMB[sequence, dirList]];
  ];
```

```
In[ ]:= directionsB[{1, 1, 2}, {"R", "L"}]
```

```
In[ ]:= directionsB[{1, 1, 2}, {"R", "L"}]
```

```
In[ ]:= directionsB[{2, 3, 3, 5, 3}, {"L", "R"}]
```

```
In[ ]:= directionsB[{2, 3, 3, 5, 3}, {"R", "L"}]
```

```
In[ ]:= directionsB[{2, 3, 3, 5, 3}, {"L", "L"}]
```

```
(*Input:
 sequence of numbers which correspond to the continued fraction representing a tangle,
and the class (M1 or M2) of the link which this tangle is a part of

 Output: List representing the directions of each of the three strands
   which precede each crossing set. This method returns the directions
   of the three strands to the left of each crossing set in the tangle,
starting from the rightmost crossing set
 (which appears at the bottom when viewed as a part of a standard M link diagram)
 and proceeding to the left,
when the tangle is drawn in an orientation where the crossing set corresponding
 to the last element in its sequence appears leftmost in the diagram.
*)
directionsM[sequence_, mVariety_] := Module[
  {dirList = {}, i = 1},

  (*At all times, exactly one strand will be
   oriented left to right. This strand is represented by R.*)
  (*We represent the other two strands with Ls.*)
  If [mVariety == "M1",
   (*In the M1 case, all tangles start RRL,
   and in the M2 case, all tangles start LRL.*)
   dirList = {{"R", "R", "L"}},
   dirList = {{"L", "R", "L"}};
  ];
  Return[directionsMB[sequence, dirList]];
 ]
```

In[ ]:= `directionsM[{1, 1, 1}, "M2"]`

In[ ]:= `directionsM[{2, 1, 1, 1, 1}, "M1"]`

```
(*Input: sequence of numbers which correspond to the continued
  fraction representing a tangle, and the starting directions of the tangle

 Output: List representing the directions of each of the three strands
   which precede each crossing set. This method returns the directions
   of the three strands to the left of each crossing set in the tangle,
starting from the rightmost crossing set
 (which appears at the bottom when viewed as a part of a standard M link diagram)
 and proceeding to the left,
when the tangle is drawn in an orientation where the crossing set corresponding
 to the last element in its sequence appears leftmost in the diagram.
*)

directionsMB[sequence_, dirList_] := Module[
  { i = 1, allDirections = dirList},

  While[i ≤ Length[sequence],

   If[EvenQ[sequence[[i]]],
    (*In the case that the next set of crossings has an even number of crossings,
    the orientation of the strands is not affected.*)
    PrependTo[allDirections, allDirections[[1]]];
    ,

    (*In the case where the next set of crossings has an odd number of crossings,
    either the bottom two strands switch orientations,
    or the top two strands switch orientations. *)
    If[OddQ[i],
      (*If on an odd index,
      switch the orientations between the bottom two crossings.*)
      PrependTo[allDirections, {allDirections[[1, 1]],
         allDirections[[1, 3]], allDirections[[1, 2]]}];
      ,
      (*If on an even index,
      switch the orientations between the top two crossings.*)
      PrependTo[allDirections, {allDirections[[1, 2]],
         allDirections[[1, 1]], allDirections[[1, 3]]}];
     ];
   ];
   i++;
  ];
  (*Remove the auxiliary direction set representing the
   directions which come after the first crossing set in the tangle*)
  allDirections = Take[allDirections, Length[allDirections] - 1];
  Return[allDirections];
 ]
```

Specific to Two-Bridge Links

```
(*determine the direction of each set of three strands preceding each crossing set,
starting from the left and going right.*)
```

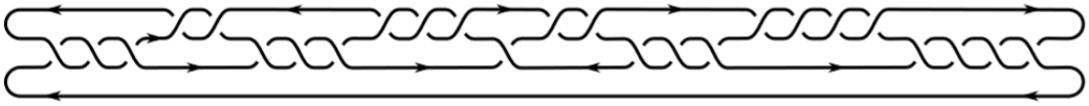
```
directions[sequence_, choiceOfOrientation_] := Module[
  {dirList = {{"L", "R", "R"}}, i = 2, reversedSequence = Reverse[sequence]},
  (*At all times, exactly one strand will be oriented right-
   to-left. This strand is represented by L.*)
  (*We represent the other two strands with Rs.*)
  (*Deciding whether the leftmost orientations
   in the link have the L strand on top or in the middle.*)
  dirList = {leftMostOrientations[reversedSequence]};
  (*If the sequence represents a link,
  user input determines the leftmost starting orientations.*)
  If[EvenQ[Numerator[FromContinuedFraction[sequence]]],

   dirList = {choiceOfOrientation};

  ];

  (*Generate the list of three-letter lists by indexing through the sequence.*)
  While[i ≤ Length[reversedSequence],

   If[EvenQ[reversedSequence[[i - 1]]],
    (*In the case that the previous set of crossings had an even number of crossings,
    the orientation of the strands is not affected.*)
    AppendTo[dirList, dirList[[i - 1]]]

    ,
    (*If the previous set had an odd number of crossings,
    either the bottom two or top two strands will switch orientations. The
     bottom two strands are used for the 1st crossing; after this,
    the switching alternates between the top two and bottom two strands.*)


    If[OddQ[i],
      (*If on an odd index,
      switch the orientations between the bottom two crossings.*)
      AppendTo[dirList, {dirList[[i - 1, 2]], dirList[[i - 1, 1]], dirList[[i - 1, 3]]}]
      ,
      (*If on an even index,
      switch the orientations between the top two crossings.*)
      AppendTo[dirList, {dirList[[i - 1, 1]], dirList[[i - 1, 3]], dirList[[i - 1, 2]]}]

     ];

   ];
   i++;
  ];
  Return[dirList]
 ]
```

Direction labeling

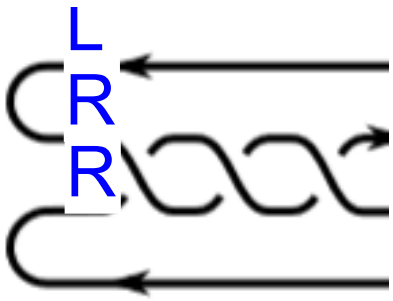
We attach a list of directions to a two-bridge link to determine its signed vector. The directions of the link are represented by a list of 2k+1 lists of three strings each, where 2k+1 is the length of the sequence representing the link. Each string can either be “L,” representing a strand which is oriented right-to-left, or “R,” representing a strand oriented left-to-right.

The three strings in the n^{th} list represent the orientations of the three strands preceding the n^{th} from the left set of crossings in the link, from top to bottom. For example, the list representing the leftmost set of crossings in this two-bridge link would be {“L”, “R”, “R”}.

In[ ]:=

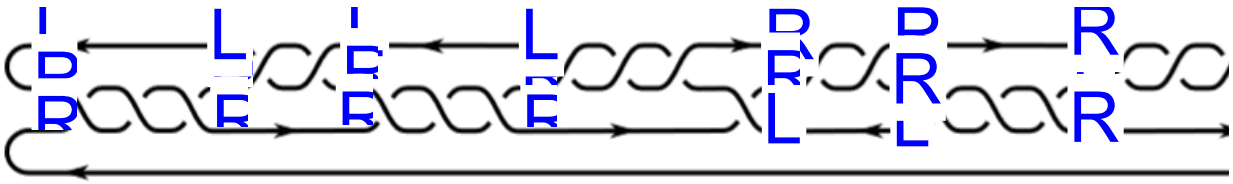


In[ ]:=



The full directions list for this link is {{“L”, “R”, “R”},{“L”, “R”, “R”},{“L”, “R”, “R”},{“L”, “R”, “R”}, {“R”, “R”, “L”}, {“R”, “R”, “L”}, {“R”, “L”, “R”},{“R”, “L”, “R”}}

In[ ]:=



All directions in the link depend on the leftmost set of directions, the parity of the index of the current set of crossings, and the parity of the number of crossings in the current set. Because of the convention that the bottom strand is oriented right-to-left, we can assume that the 3rd element in the first list of the directions of any link is “R”. We can trace the third strand around the knot to determine the orientations of the other two strands at the leftmost position.
(In the case that the sequence represents a link, the other two orientations can be either “L”, “R” or “R”, “L”.)

The leftmostOrientations method determines the directions of the strands preceding the leftmost set of crossings in the link using the movementRightLeft and movement functions to trace the third strand around the knot.

From this information, the directions method determines the directions of the strands preceding all other sets of crossings in the link, in order from left to right. The toSignedVector method uses this information to determine the signs of each set of crossings.

```
(*Traces a strand either to the right or left depending on
 the beginning index and the strand number specified by position*)
movementRightLeft[sequence_, beginning_, position1_] := Module[
  {i = beginning, endpoint, position = position1, sign},
  (*if you are tracing the strand from left to right*)
  If[beginning == 1,
   While[i ≤  Length[sequence],
    position = movement[sequence, position, i];
    i++;
   ];
   (*once you finish tracing the strand from left to right, you have to *)
   Which[position == 1, position = 2,
position == 2, position = 1,
position == 3, position = 1],
   (*if you are tracing the strand from right to left*)
   While[i >  0,
     position = movement[sequence, position, i];
     i--;
    ];
  ];
  Return[position];
 ]
```

In[ ]:= `movementRightLeft[{1, 2, 2}, 1, 3]`

In[ ]:= `movementRightLeft[{1, 2, 2}, 3, 2]`

In[ ]:= `movementRightLeft[{3}, 1, 3]`

```
(*depending on the the index of crossings sets and the number of crossings
 within the set you are currently on, determine position movement*)
movement[sequence_, position1_, index_] := Module[
  {i = index, position = position1},
  (*if there is an odd number of crossings within a section,
  at the end of the section, the position will have changed*)
  If[OddQ[sequence[[i]]],
   (*when the index is odd, the crossing interactions only involve strands 3 and 2,
   meaning the strands will switch to the opposite position to their current position,
   theire only other option being either 2 or 3 depending on their current position*)
        If[OddQ[i],

        Which[position == 3, position = 2, position == 2, position = 3
        ]
   ,
     (*when the index is even, the crossing interactions only involve strands 1 and
      2, meaning the strands will switch to the opposite of their current position,
     their only other option being either 1 or 2 depending on the current position*)
        Which[position == 1, position = 2, position == 2, position = 1]
   ];
  ];
  Return[position];
 ]
```

In[ ]:= `movement[{1, 2, 2}, 2, 2]`

In[ ]:= `movement[{1, 3, 7, 2, 4}, 2, 3]`

In[ ]:= `movement[{1, 3, 5, 9, 7}, 2, 4]`

```
(*Determines whether the orientations of the strands preceding the leftmost set of crossings in th
are RLR or LRR by tracing the third strand around the knot/link*)
leftMostOrientations[sequence_]:= Module[

{i, sequenceOdd = correctSequence[sequence],position,finalPosition,positionForward,positionLeft,p

(*move forward through the diagram, tracing starting from the leftmost set of crossings to determ

positionRight=movementRightLeft[sequenceOdd,1,3];
(*wrap around to the right to see how strand 3 is entering*)
If[Length[sequence]>1,
positionLeft=movementRightLeft[sequenceOdd,Length[sequenceOdd],positionRight],
positionLeft=positionRight;
];
(*if the thrid strand wraps around to the top, then the strand oriented to the left located at the
If[positionLeft == 1, Return[{"L","R","R"}]
];
(*if the third stand wraps around to the middle, then the strand oriented to the left is in the m
If[positionLeft == 2, Return[{"R","L","R"}]
];
(*The third strand never wraps around to the same, third position*)
Return[positionLeft];
]
```

In[•]:= `leftMostOrientations[{3}]`

In[•]:= `leftMostOrientations[{1, 2, 2}]`

In[•]:= `leftMostOrientations[{2, 1, 2}]`

In[•]:= `leftMostOrientations[{3, 2, 1, 5, 1}]`

In[•]:= `leftMostOrientations[{3, 3, 3}]`

In[•]:= `leftMostOrientations[{2, 3, 3}]`

```
(*given an input sequences, determines all possible direction lists. For knots,
two copies of the same direction list are returned*)
bothDirections[sequence_] :=
 Return[{directions[sequence, {"L", "R", "R"}], directions[sequence, {"R", "L", "R"}]}]
```

In[•]:= `bothDirections[{1, 2, 4}]`

In[•]:= `FromContinuedFraction[{1, 2, 4}]`

In[•]:= `bothDirections[{1, 2, 3}]`

In[•]:= `FromContinuedFraction[{1, 2, 3}]`

In[•]:= `bothDirections[{6, 4, 9, 7, 9}]`

In[•]:= `FromContinuedFraction[{6, 4, 9, 7, 9}]`

In[•]:= `directions[{3}, {"R", "L", "R"}]`

```
In[ ]:= directions[{2, 3, 4}, {"R", "L", "R"}]
In[ ]:= directions[{1, 2, 3}, {"R", "L", "R"}]
In[ ]:= directions[{2, 1, 1, 2, 2}, {"R", "L", "R"}]
In[ ]:= directions[{1, 5, 2}, {"R", "L", "R"}]
```

Functions needed for B Links

Classification

This function is also relevant in the classification not only of a tangle, but of the link itself, so is used to evaluate unspecified links.

```
(*Based on the parity of the numerator and
 denominator of the value of a continued fraction sequence,
returns the type of tangle represented by that sequence.*)
classifyTangle[sequence_] := Module[
  {frac = FromContinuedFraction[sequence], num, denom},
  num = Numerator[frac];
  denom = Denominator[frac];

  If[OddQ[num] && OddQ[denom],
   (*If the continued fraction is of the form odd/odd,
   the upper left strand connects to the lower right strand,
   and the upper right strand connects to the lower left strand.*)
   Return["cross"]
  ];
  If[OddQ[num] && EvenQ[denom],
   (*If the continued fraction is of the form odd/even,
   the upper left strand connects to the upper right strand,
   and the lower left strand connects to the lower right strand.*)
   Return["zero"]
  ];
  (*If the continued fraction is of the form even/odd,
  the upper left strand connects to the lower left strand,
  and the upper right strand connects to the lower right strand.*)
  Return["inf"]

 ]
```

```
In[ ]:= classifyTangle[{1, 2, 2}]
In[ ]:= classifyTangle[{2, 3, 3}]
In[ ]:= classifyTangle[{1, 2, 3}]
```

```
(*Returns the Seifert Parity of the input case: Parity 1, 2, or 3*)
seifertParity[case_] := Which[
  case == "i", Return["Parity 1"],

  case == "ii" || case == "iv", Return["Parity 2"],

  case == "v" || case == "vii", Return["Parity 3"]

 ]
```

In[•]:= `seifertParity["i"]`

In[•]:= `seifertParity["v"]`

```
(*Returns a list of strings representing which of the eight categories each tangle decomposes into
representation.*)
eightArcCases[fourDirectionMLink_]:= Module[
{iInstance={"R","R","R","R"},iiInstance={"L","R","L","R"},ivInstance={"R","L","R","L"},vInstance=
instanceList},
instanceList = Table[{""}, Length[fourDirectionMLink]];
While[counter≤ Length[fourDirectionMLink],
(*Based on the incoming/outgoing orientations from the fourDirections list, labels each tangle as
Which[fourDirectionMLink[[counter]]== iInstance,
instanceList[[counter]]="i",
fourDirectionMLink[[counter]]== iiInstance,
instanceList[[counter]]="ii",
fourDirectionMLink[[counter]]== ivInstance,
instanceList[[counter]]="iv",
fourDirectionMLink[[counter]]== vInstance,
instanceList[[counter]]="v",
fourDirectionMLink[[counter]]== viiInstance,
instanceList[[counter]]="vii"
];
counter++;
];
Return[instanceList];
]
```

In[•]:= `eightArcCases[fourDirectionsOfmLink[{{4}, {1, 1, 2}, {3}, 1}]]`

In[•]:=
```
eightArcCases[
 fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}][[1]]]
```

In[•]:=
```
eightArcCases[
 fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}][[2]]]
```

In[•]:=
```
eightArcCases[
 fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}][[3]]]
```

In[•]:=
```
eightArcCases[
 fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}][[4]]]
```

```
(*Returns a list of lists of length 4. The nth of these length-
 4 lists represents the orientations of the strands entering/exiting the nth tangle
  in the link. Positions 1 and 2 represent the bottom left and top left strands,
respectively; positions 3 and 4 represent the bottom right and top right strands,
respectively. *)
fourDirectionsOfmLink[mLink_] := Module[
  {tangleList = Take[mLink, Length[mLink] - 1], tangleTypes = {}, outputList,
   currentPosition = 2, currentTangleIndex = 1, direction = "R", count = 1},
  outputList = Table[{"", "", "", ""}, Length[tangleList]];
  tangleTypes =
   Table[classifyTangle[tangleList[[count]]], {count, Length[tangleList]}];
  outputList[[currentTangleIndex, currentPosition]] = direction;
  While[count ≤  4 * Length[tangleList](*The total number of entering/exiting
    strands to be labeled is 4 times the number of tangles in the link,
   so terminate the labeling process once this many strands have been visited.*),
   (*Travel to the next position in the tangle*)
   currentPosition =
    switchPositionsInTangle[tangleList[[currentTangleIndex]], currentPosition];
   (*Change directions if necessary*)
   direction =
    switchDirections[mLink, {currentTangleIndex, currentPosition}, direction];
   (*Set the new position to the new direction*)
   outputList[[currentTangleIndex, currentPosition]] = direction;
   (*Travel to the appropriate position
    in the adjacent tangle. Direction remains constant.*)
   currentPosition = switchPositionsBetweenTangles[mLink,
      {currentTangleIndex, currentPosition}, direction][[2]];
   currentTangleIndex = switchPositionsBetweenTangles[mLink,
      {currentTangleIndex, currentPosition}, direction][[1]];
   (*Set the new position to the current direction*)
   outputList[[currentTangleIndex, currentPosition]] = direction;
   count++;
  ];
  Return[outputList]

 ]
```

In[•]:= `fourDirectionsOfmLink[{{4}, {1, 1, 2}, {3}, 1}]`

In[•]:= `fourDirectionsOfmLink[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}]`

In[•]:= `fourDirectionsOfmLink[{{1, 1, 1, 2, 2}, {1, 1, 1}, 2}]`

Movement

```
(*Returns the new orientation of the current strand after
 traveling in the specified direction through the specified tangle*)
switchDirections[mLink_, {tangleIndex_, position_}, direction_] := Module[
  {newDirection},
  (*current direction switches if you reach an infinity tangle,
  or you are traveling to the other side of the link*)
  If[classifyTangle[mLink[[tangleIndex]] ] == "inf"
   ,
   Which[
    direction == "L", Return["R"],
    direction == "R", Return["L"];
   ]
  ];

  Return[direction]

 ]
```

In[ ]:= `switchDirections[{{1, 2, 3}, {1, 2, 2}, 2}, {2, 1}, "R"]`

In[ ]:= `switchDirections[{{1, 2, 3}, {1, 2, 2}, 2}, {2, 4}, "R"]`

In[ ]:= `fourDirectionsOfmLink[{{2, 1, 2, 1, 1}, {3}, {2}, 0}]`

```
(*given the tangle written as a sequence and a current position,
returns new position within the tangle
 note: counter clockwise order from top left: 2,1,3,4
*)
switchPositionsInTangle[tangle_, previousPosition_] := Module[
  {tangleType = classifyTangle[tangle], newCrossPositions = {4, 3, 2, 1},
   newInfPositions = {2, 1, 4, 3}, newZeroPositions = {3, 4, 1, 2}},
  Which[tangleType == "cross",
   Return[newCrossPositions[[previousPosition]]],
   tangleType == "inf",
   Return[newInfPositions[[previousPosition]]],
   tangleType ==  "zero",
   Return[newZeroPositions[[previousPosition]]];

  ]
 ]
```

```
(*Returns the index of the tangle reached after
 traveling in the specified directions from the current tangle*)
updatedTangleIndex[currentTangleIndex_, mLink_, direction_] := Module[
  {movement, newTangleIndex},
  If[direction == "L",
   movement = -1,
   movement = 1
  ];
  newTangleIndex = Mod[currentTangleIndex + movement, Length[mLink] - 1];
  If[newTangleIndex == 0,
   newTangleIndex = Length[mLink] - 1;
  ];
  Return[newTangleIndex];
 ]
```

```
(*Returns the new position after switching from the first to last tangle (or vice versa) in the s
switchBetween[position_,direction_,loneCrossingMovement_]:= Module[
{possibleChanges,indexOfNewPosition,newPosition},

If[loneCrossingMovement== True,
possibleChanges={{1,4},{2,3}},

possibleChanges={{2,4},{1,3}}
];
If[direction== "L",
indexOfNewPosition=2,
indexOfNewPosition=1];

newPosition=Flatten[Select[possibleChanges,ContainsAny[#,{position}]&]][[indexOfNewPosition]];
Return[newPosition];
]
```

In[•]:= `switchBetween[3, "R", False]`

In[•]:= `switchBetween[2, "L", False]`

`switchBetween[3, "R", True]`

```
(*currentCoordinates is a list of the form (tangleIndex, position), and direction
 specifies whether you are moving right or left. Returns new coordinates*)
switchPositionsBetweenTangles[mLink_,
  {tangleIndex_, position_}, direction_] := Module[
  {newTangleIndex, newPosition, forwardBackwardPositions = {{1, 3}, {2, 4}},
   loneCrossingPositions = {{1, 4}, {2, 3}}, loneCrossingMovement = False},

  newTangleIndex = updatedTangleIndex[tangleIndex, mLink, direction];

  If[(tangleIndex == 1 && OddQ[Last[mLink]]) && direction == "L" ||
    (tangleIndex == (Length[mLink] - 1) && OddQ[Last[mLink]] && direction == "R"),
   loneCrossingMovement = True;
  ];

  newPosition = switchBetween[position, direction, loneCrossingMovement];
  Return[{newTangleIndex, newPosition}];

 ]
```

```
classifyTangle[{1, 11, 1}]
```

In[ ]:=
```
classifyTangle /@ {{1, 11, 1}, {2, 3}, {2, 3, 4}, 1}
```

In[ ]:=
```
switchPositionsBetweenTangles[{{1, 11, 1}, {2, 3}, {2, 3, 4}, 1}, {3, 3}, "R"]
```

Braid Index for Links with Multiple Components

```
(*Returns a list of the fourDirections of the input mLink if each of its
 components' bottommost strands' orientations are listed in orientationList*)
fourDirectionsOfmLinkAllComponents[mLink_, orientationList_] := Module[
  {fourDirections, count = 1},
  fourDirections = fourDirectionsOfmLink[mLink];

  While[count ≤ Length[orientationList],

   fourDirections = fourDirectionsOfmLinkSecondComponent[
     mLink, fourDirections, orientationList[[count]]];
   count++;
  ];

  Return[fourDirections]
 ]
```

In[ ]:=
```
fourDirectionsOfmLinkAllComponents[
 {{3}, {3}, {3}, {1, 1, 2}, {2, 1, 2}, {2, 2, 2}, {2, 2, 2}, {2, 1, 2}, 0}, {"R", "L", "L"}]
```

In[ ]:=
```
fourDirectionsOfmLinkAllComponents[
 {{3}, {3}, {3}, {1, 1, 2}, {2, 1, 2}, {2, 2, 2}, {2, 2, 2}, {2, 1, 2}, 0}, {"R", "R", "R"}]
```

In[ ]:=
```
fourDirectionsOfmLinkAllComponents[{{3}, {3}, {3}, {1, 1, 2}, 0}, {"R"}]
```

```
(*returns the number of components of an mLink.*)
numComponents[mLink_] := Module[
  {count = 1, fourDirections = fourDirectionsOfmLink[mLink]},

  While[Length[Position[fourDirections, ""]] > 0,

   fourDirections = fourDirectionsOfmLinkSecondComponent[mLink, fourDirections, "R"];

   count++;
  ];
  Return[count]
 ]
```

In[ ]:= `numComponents[{{3}, {3}, {3}, {1, 1, 1}, 0}]`

In[ ]:= `numComponents[{{2, 2, 2}, {2, 1, 2}, {2, 1, 2}, {2, 1, 2}, 0}]`

In[ ]:= `numComponents[{{1, 1, 1, 1, 1}, {3}, {3}, 0}]`

```
(*outputs all possible lists of component
 orientations for the given number of components.*)
orientationListGenerator[numComponents_] := Tuples[{"L", "R"}, numComponents - 1]
```

In[ ]:= `orientationListGenerator[4]`

In[ ]:= `orientationListGenerator[3]`

```
(*returns a list of all possible fourDirections for a given mLink.*)
fourDirectionsOfmLinkAllPossible[mLink_] := Module[
  { listOfLists = {}, numComponents = numComponents[mLink], updatedList = mLink,
   tuples = orientationListGenerator[numComponents[mLink]], count = 1},
  If[numComponents == 1, Return[fourDirectionsOfmLink[mLink]]];

  While[count ≤  Length[tuples],

   AppendTo[listOfLists, fourDirectionsOfmLinkAllComponents[mLink, tuples[[count]]]];

   count++;

  ];
  Return[listOfLists]

 ]
```

In[ ]:= `fourDirectionsOfmLinkAllPossible[{{2, 1, 2}, {2, 2, 2}, 0}]`

In[ ]:= `fourDirectionsOfmLinkAllPossible[{{3}, {3}, {3}, {1, 1, 1}, 0}]`

In[ ]:= `fourDirectionsOfmLinkAllPossible[{{3}, {3}, {3}, {1, 2}, 0}]`

```
(*determines which step of the While loop
 happens first in fourDirectionsOfmLinkSecondComponent*)
orderOfSwitching[firstBlankSpot_, direction_] := Module[
  {position = firstBlankSpot[[2]]},
  If[position == 1 || position == 2,
   (*left side of tangle*)
   If[direction == "L",
     (*switch outside of tangle first.*)
     Return["out"],
     (*switch inside the tangle first.*)
     Return["in"]

    ];
   ,
   If[direction == "L",
     (*switch inside of tangle first.*)
     Return["in"],

     (*switch outside the tangle first.*)
     Return["out"]
    ];
  ]

 ]
```

```
(*direction is L or R.*)
fourDirectionsOfmLinkSecondComponent[mLink_,
  fourDirections_, firstUnorientedComponentDirection_] := Module[
  {tangleList = Take[mLink, Length[mLink] - 1], tangleTypes = {},
   outputList = fourDirections,  currentPosition = 2, currentTangleIndex = 1,
   direction = firstUnorientedComponentDirection, count = 1},

  {currentTangleIndex, currentPosition} = Position[outputList, ""][[1]];
  (*Print[{currentTangleIndex, currentPosition}];*)
  direction = firstUnorientedComponentDirection;

  If[orderOfSwitching[Position[outputList, ""][[1]], direction] == "in",
   outputList = movementWithinTangle[mLink,
     fourDirections, currentTangleIndex, currentPosition, direction]
   ,
   outputList = movementOutsideTangle[mLink,
     fourDirections, currentTangleIndex, currentPosition, direction]
  ];
  Return[outputList];
 ]
```

In[ ]:=
```
fourDirectionsOfmLinkSecondComponent[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 0},
 {{"L", "R", "R", "L"}, {"R", "L", "", ""}, {"", "", "", ""}, {"", "", "L", "R"}}, "R"]
```

In[ ]:=
```
fourDirectionsOfmLink[{{2, 1, 2}, {2, 2, 2}, {3}, {2, 1, 2}, {2, 2, 2}, {3}, 0}]
```

In[•]:=
```
fourDirectionsOfmLinkSecondComponent[
 {{2, 1, 2}, {2, 2, 2}, {3}, {2, 1, 2}, {2, 2, 2}, {3}, 0},
 {{"L", "R", "", ""}, {"", "", "", ""}, {"", "", "", ""},
  {"", "", "", ""}, {"", "", "R", "L"}, {"R", "L", "L", "R"}}, "R"]
```

In[•]:=
```
fourDirectionsOfmLinkSecondComponent[
 {{2, 1, 2}, {2, 2, 2}, {3}, {2, 1, 2}, {2, 2, 2}, {3}, 0},
 {{"L", "R", "R", "L"}, {"R", "L", "", ""}, {"", "", "", ""},
  {"", "", "", ""}, {"", "", "R", "L"}, {"R", "L", "L", "R"}}, "L"]
```

```
(*Updates a fourDirections list after a movement within the current tangle.*)
movementWithinTangle[mLink_, outputList2_,
  currentTangleIndex2_, currentPosition2_, direction2_] := Module[
  {count = 1, tangleList = Take[mLink, Length[mLink] - 1],
   outputList = outputList2, direction = direction2,
   currentPosition = currentPosition2, currentTangleIndex = currentTangleIndex2},
  While[count < 4 * Length[mLink],
   outputList[[currentTangleIndex, currentPosition]] = direction;
   currentPosition =
    switchPositionsInTangle[tangleList[[currentTangleIndex]], currentPosition];
   direction = switchDirections[mLink,
     {currentTangleIndex, currentPosition}, direction];
   outputList[[currentTangleIndex, currentPosition]] = direction;
   currentPosition = switchPositionsBetweenTangles[
      mLink, {currentTangleIndex, currentPosition}, direction][[2]];
   currentTangleIndex = switchPositionsBetweenTangles[mLink,
      {currentTangleIndex, currentPosition}, direction][[1]];
   outputList[[currentTangleIndex, currentPosition]] = direction;
   count++;
  ];
  Return[outputList];
 ]
```

```
(*Updates a fourDirections list after a
 movement from the current tangle to the next tangle.*)
movementOutsideTangle[mLink_, outputList2_, currentTangleIndex2_,
  currentPosition2_, direction2_] := Module[
  {count = 1, tangleList = Take[mLink, Length[mLink] - 1],
   outputList = outputList2, currentTangleIndex = currentTangleIndex2,
   currentPosition = currentPosition2, direction = direction2},

  While[count < 4 * Length[mLink],
   outputList[[currentTangleIndex, currentPosition]] = direction;
   currentPosition = switchPositionsBetweenTangles[
      mLink, {currentTangleIndex, currentPosition}, direction][[2]];
   currentTangleIndex = switchPositionsBetweenTangles[mLink,
      {currentTangleIndex, currentPosition}, direction][[1]];
   outputList[[currentTangleIndex, currentPosition]] = direction;
   currentPosition =
    switchPositionsInTangle[tangleList[[currentTangleIndex]], currentPosition];
   direction = switchDirections[mLink,
     {currentTangleIndex, currentPosition}, direction];
   count++;
  ];
  Return[outputList];
 ]
```

```
(*Returns a list of all possible braid indices for an mLink,
which are different depending on its orientation*)
braidIndexBAllOrientations[mLink_] := Module[
  {output = {}, allFourDirections, count = 1},
  allFourDirections = fourDirectionsOfmLinkAllPossible[mLink];

  If[numComponents[mLink] == 1, Return[braidIndexB[mLink]]];

  While[count ≤ Length[allFourDirections],

   AppendTo[output,
    braidIndexBMultipleComponents[mLink, allFourDirections[[count]]]];

   count++;

  ];
  Return[output]
 ]
```

Corrected Sequence

```
(*Changes a sequence representing a
 continued fraction to have an odd number of elements*)
correctSequence[sequence_] := Module[{sequenceCopy},
  sequenceCopy = sequence;

  If[EvenQ[Length[sequence]],
   (*If the length is even, subtract 1 from the last element*)
   sequenceCopy[[Length[sequence]]] = Last[sequence] - 1;
   (*and append a 1 to the end*)
   sequenceCopy = AppendTo[sequenceCopy, 1];
   Return[sequenceCopy]
   ,
   (*If the sequence length is already odd, make no change to the sequence*)
   Return[sequenceCopy]]

 ]
```

In[•]:= `correctSequence[{3, 2, 1, 6}]`

In[•]:= `correctSequence[{1}]`

In[•]:= `correctSequence[{1, 4}]`

In[•]:= `correctSequence[{3, 5, 4, 6, 2, 8}]`

Delta formulas

$\triangle_1\ (A_j)$

$$\triangle_1(A_j) = \frac{-1 + \text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\ 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\ 0\le m\le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

$\triangle_2\ (A_j)$

$$\triangle_2(A_j) = \frac{1 + \text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\ 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\ 0\le m\le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

$\triangle_3\ (A_j)$

$$\triangle_3(A_j) = \frac{-1 + \text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\, 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\, 0\le m\le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

Delta Code

```
(*Given a tangle A represented by a sequence in two-
 bridge notation and its Seifert parity p, where p = 1 or 2,
returns the value of the appropriate delta formula evaluated at that tangle;
i.e. Δp(A)*)
deltas[signedVector_, seifertParityType_] := Module[
  (*oddSequence is the sequence converted to odd length,
  if the sequence is of even length*)
  {q, res = 0, sign, sumBothParities},


  (*Compute the summations in the delta formula. Note that this
   part of the formula is consistent among each of the parity types.*)
  sumBothParities = sumOfValuesAtIndices[signedVector, EvenQ] +
    sumOfValuesAtIndices[signedVector, OddQ];

  sign = Sign[Last[signedVector]];

  If[seifertParityType == 1 || seifertParityType == 3,

   (*Add the appropriate value the summations
    depending on which of the delta formulas is being evaluated. *)
   res = (-1 + sign)/4 + sumBothParities,
   res = (1 + sign)/4 + sumBothParities
  ];
  Return[res]
 ]
```

In[ ]:= `deltas[{-2, 2, -2}, 2]`

In[ ]:= `deltas[{-2, 1, 2}, 2]`

In[ ]:= `deltas[{1, 1, 1, 2, 2}, 1]`

In[ ]:= `deltas[{1, 1, 1}, 1]`

The above results are correct (Examples from the paper. These values were calculated for the M1 knot example.)

In[ ]:=
```
deltas[toSignedVectorM[correctSequence[{1, 1, 1, 2, 2}],
  directionsM[correctSequence[{1, 1, 1, 2, 2}], "M2"], "M2"], 2]
```

```
In[ ]:= deltas[toSignedVectorM[correctSequence[{1, 1, 1}],
          directionsM[correctSequence[{1, 1, 1}], "M2"], "M2"], 2]

In[ ]:= deltas[toSignedVectorM[correctSequence[{2}],
          directionsM[correctSequence[{2}], "M2"], "M2"], 2]
```

The above results are correct. These values were calculated in the paper for the M2 knot.

delta0

```
(*Returns Δ0(L) given the Montesinos link L's list of Seifert parity 3 tangles,
and the end lone crossings. *)
delta0[seifert3tangles_, loneCrossings_] := Module[
  {tangleList = {}, result = 0, eta},

  eta = Length[seifert3tangles];

  result = eta + loneCrossings - Min[(eta + loneCrossings)/2 - 1, loneCrossings];
  Return[result]

 ]
```

```
In[ ]:= delta0[{{2}, {2, 1, 2}}, 2]

In[ ]:= delta0[{{4}, {6}, {3, 2, 1}}, 5]
```

Signed Vectors

Signed Vector for M Links

Signed Vector for B Links

```
(*Returns a list of each of the signed vectors in the mLink given its fourDirections list*)
toSignedVectorWholeBSequence[mLink_, fourDirections_]:=Module [
{twoIncomingOrientations,incomingOrienatations,tangleList,signedVector},
tangleList = correctSequence/@Take[mLink, Length[mLink]-1];
(*fourDirections*)
twoIncomingOrientations = Table[{fourDirections[[i,1]],fourDirections[[i,3]]},
{i, Length[fourDirections]}];
(*list of incoing orientations of each tangle this is within the indicated seifert class*)
(*incomingOrienatations = Table[Flatten[twoIncomingOrientations[[i]]], {i,tangleList}];*)
signedVector=Table[toSignedVectorB[tangleList[[i]],directionsB[tangleList[[i]],
twoIncomingOrientations[[i]]]],

{i, Length[tangleList]}];
Return[signedVector];

]
```

In[ ]:=
```
toSignedVectorWholeBSequence[{{3, 2, 1}, {2}, {2, 2}, 1},
 fourDirectionsOfmLinkAllPossible[{{3, 2, 1}, {2}, {2, 2}, 1}][[1]]]
```

In[ ]:=
```
toSignedVectorWholeBSequence[{{4, 3}, {2, 1, 1}, {6}, {2, 7, 2, 6, 1}, 2},
 fourDirectionsOfmLinkAllPossible[{{4, 3}, {2, 1, 1}, {6}, {2, 7, 2, 6, 1}, 2}]]
```

Example 5.8. Let $K = 12a_{304} = M(7/19, 1/3, 1/2, 0)$ in Figure 24. This is an alternating M link of Class B with $e = 0$ and $\eta = 2$. $7/19 = (2, 1, 2, 1, 1)$ has a signed vector of $(2, 1, -2,$ is of Seifert Parity 3, $1/3 = (3)$ has a signed vector of (3) so it is of Seifert Parity 3, $1/2$ a signed vector of (-2) so it is of Seifert Parity 2. We obtain $\Delta_3(7/19) = 1 + 1 = 2$, $\Delta_3($ $\Delta_2(1/2) = 1$ (here we count $\text{sign}(b_1^j) = \text{sign}(b_{2q_j+1}^j) = -1$) and $\Delta_0(L) = 2 - 0 = 2$. By (5.5) $\mathbf{b}(12a_{304}) = 2 + 1 + 2 = 5$.

In[ ]:=
```
toSignedVectorWholeBSequence[{{2, 1, 2, 1, 1}, {3}, {2}, 0},
 fourDirectionsOfmLink[{{2, 1, 2, 1, 1}, {3}, {2}, 0}]]
```

```
toSignedVectorB[sequence_, directions_] := Module[
  {startDir, parallelOrient = {}},
  startDir = directions[[1]];
  If[Count[startDir, "L"] == 2,
   parallelOrient = {"L", "L"},
   parallelOrient = {"R", "R"}
  ];
  Return[toSignedVector[sequence, directions, parallelOrient]];
 ]
```

In[ ]:=
```
toSignedVectorB[{2, 3, 3, 5, 3},
 {{"L", "L", "R"}, {"L", "R", "L"}, {"R", "L", "L"}, {"R", "L", "L"}, {"L", "R", "L"}}]
```

In[ ]:=
```
toSignedVectorB[{2, 3, 3, 5, 3},
 {{"R", "L", "R"}, {"R", "R", "L"}, {"R", "R", "L"}, {"R", "L", "R"}, {"L", "R", "R"}}]
```

```
In[•]:= toSignedVectorB[{4, 6, 2, 2, 1},
 {{"L", "R", "R"}, {"L", "R", "R"}, {"L", "R", "R"}, {"L", "R", "R"}, {"L", "R", "R"}}]
```

Signed Vector for Two Bridge Links

```
(*Given a two-bridge sequence and its set of directions lists,
returns the signed vector*)
toSignedVectorTwoBridge[sequence_, directionsChoice_] := Module[
  {directionsChoice2 = directionsChoice},
  If[OddQ[Numerator[FromContinuedFraction[sequence]]],
   directionsChoice2 = directions[sequence, {"R", "L", "R"}];
  ];
  Return[toSignedVector[sequence, directionsChoice2, {"R", "R"}]]
 ]
```

```
In[•]:= toSignedVectorTwoBridge[{2, 3, 4}, directions[{2, 3, 4}, {"L", "R", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{1, 2, 4}, directions[{1, 2, 4}, {"R", "L", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{1, 2, 4}, directions[{1, 2, 4}, {"L", "R", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{1, 2, 3}, directions[{1, 2, 3}, {"L", "R", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{1, 2, 3}, directions[{1, 2, 3}, {"R", "L", "R"}]]
```

The sequence {1,2,4} represents a knot, so it only has one possible signed vector. The sequence {1,2,3} represents a link, so it has two possible signed vectors.

```
(*given an input sequences, determines all possible signed
 vectors. For knots two copies of the same signed vector are returned*)
bothSignedVectors[sequence_] :=
 Return[{toSignedVectorTwoBridge[sequence, bothDirections[sequence][[1]]],
   toSignedVectorTwoBridge[sequence, bothDirections[sequence][[2]] ]}]
```

```
In[•]:= bothSignedVectors[{2, 10, 8, 2, 6}]
```

```
In[•]:= bothSignedVectors[{1, 2, 3}]
```

```
In[•]:= bothSignedVectors[{1, 2, 4}]
```

Additional toSignedVectorTwoBridge testing

```
In[•]:= toSignedVectorTwoBridge[{3, 3, 2}, directions[{3, 3, 2} , {"L", "R", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{2, 2, 2}, directions[{2, 2, 2}, {"R", "L", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{3, 5, 6}, directions[{3, 5, 6}, {"R", "L", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{2, 2, 2, 2, 2}, directions[{2, 2, 2, 2, 2}, {"R", "L", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{2, 3, 4}, directions[{2, 3, 4}, {"R", "L", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{2, 3, 4}, directions[{2, 3, 4}, {"L", "R", "R"}]]
```

```
In[•]:= toSignedVectorTwoBridge[{3, 5, 6}, directions[{3, 5, 6}, {"L", "R", "R"}]]
```

```
In[ ]:= toSignedVectorTwoBridge[{2, 3, 2, 1, 4}, directions[{2, 3, 2, 1, 4}, {"L", "R", "R"}]]
```

```
In[ ]:= toSignedVectorTwoBridge[{2, 2, 2, 2, 2}, directions[{2, 2, 2, 2, 2}, {"L", "R", "R"}]]
```

```
In[ ]:= toSignedVectorTwoBridge[{2, 3, 4}, directions[{2, 3, 4}, {"L", "R", "R"}]]
```

To Signed Vector for All Cases

```
(*Returns the signed vector of sequence given its directions list.*)
toSignedVector[sequence_, directions_, parallelOrient_] := Module[
  {i = Length[sequence], signList = {},
   signedVector = {}, directionsChoice2 = directions},

  (*Go through the list of orientations in right to left order,
  since the beginning crossing set is drawn rightmost in this view.*)
  While[i > 0,
   If[OddQ[i],
    (*at odd indicies, interactions occur exclusively between strands 2 and 3*)
    If[{directionsChoice2 [[i, 2]], directionsChoice2 [[i, 3]]} == parallelOrient,

     (*if the strands have opposite orientations, the crossing is negative. If
      the strands have parallel orientations, the crossings is positive.*)
     AppendTo[signList, 1]
     ,
     AppendTo[signList, -1]
    ]
    ,

    (*at even indicies, crossings occur exclusively between strands 1 and 2*)
    If[{directionsChoice2[[i, 1]], directionsChoice2 [[i, 2]]} == parallelOrient,

      (*At even indices, if the strands have opposite orientations,
      the crossing is positive. If the strands have parallel orientations,
      the crossings is negative.*)
      AppendTo[signList, -1]
      ,
      AppendTo[signList, 1];
     ];
   ];
   i--;
  ];

  (*Assign the corresponding sign to each value in the sequence*)
  signedVector = Table[sequence[[x]] * signList[[x]], {x, Length[sequence]}];
  Return[signedVector];
 ]
```

```
In[ ]:= toSignedVector[{2, 1, 1, 1, 1}, {{"L", "R", "R"}, {"L", "R", "R"},
    {"R", "L", "R"}, {"R", "R", "L"}, {"R", "R", "L"}}, {"R", "R"}]
```

```
In[ ]:= toSignedVector[{1, 1, 2}, {{"L", "R", "L"}, {"L", "R", "L"}, {"R", "L", "L"}}, {"L", "L"}]
```

```
In[ ]:= toSignedVector[{2, 1, 1, 1, 1}, {{"L", "R", "R"}, {"L", "R", "R"},
          {"R", "L", "R"}, {"R", "R", "L"}, {"R", "R", "L"}}, {"R", "R"}]
```

Braid Index Specified: M1, M2, B, and Two-Bridge.

Notation for All Types: Sum of Values at Specified Indices & Formatting for Conway

```
(*Converts a link written in Conway notation to standard two-
 bridge notation of a Montesinos link*)
formatAsInput[sequence_] := Module[
  {sequenceEdit, lastTangle, loneCrossing},
  (*get rid of unneccessary formatting*)
  sequenceEdit = StringDelete[sequence, " "];
  sequenceEdit = StringDelete[sequenceEdit, "]"];
  sequenceEdit = StringDelete[sequenceEdit, "["];
  sequenceEdit = StringSplit[sequenceEdit, ";"];

  (*individual sequences are grouped as one element,
  so split each into seperate lists*)
  sequenceEdit = Split[sequenceEdit, 1];

  (*if there is a lone crossing*)
  If[StringPosition[Last[sequenceEdit], "+"] ≠ {{}},
   lastTangle = Split[Flatten[StringSplit[Last[sequenceEdit], "+"]], ","][[1]];
   loneCrossing = Flatten[ToExpression[
       Split[Flatten[StringSplit[Last[sequenceEdit], "+"]], ","]]][[2]];
   sequenceEdit = ReplacePart[sequenceEdit, Length[sequenceEdit] → lastTangle];
   sequenceEdit = Table[{ToExpression[StringReverse[sequenceEdit[[i]][[1]]]]},
     {i, Length[sequenceEdit]}];

   sequenceEdit = Table[IntegerDigits[i[[1]]], {i, sequenceEdit}];

   AppendTo[sequenceEdit, loneCrossing]

   ,

   sequenceEdit = Table[{ToExpression[StringReverse[sequenceEdit[[i]][[1]]]]},
     {i, Length[sequenceEdit]}];
   sequenceEdit = Table[IntegerDigits[i[[1]]], {i, sequenceEdit}];
   AppendTo[sequenceEdit, 0]
  ];
  Return[sequenceEdit];
 ]
```

```
In[ ]:= formatAsInput["[2 3 1; 2 1 1;2]"]
```

```
In[ ]:= formatAsInput["[2 8; 1 4 5; 3 + 2]"]
```

```
(*Given a signed vector and a parity (EvenQ or OddQ), returns the value of the summation in the b
elements (EvenQ) or for odd elements (OddQ).*)
sumOfValuesAtIndices[sequence_,parity_]:= Module[
{listOfIndices,vectorValues,sumOfIndices,validValues},
Which[parity== EvenQ && Length[sequence] == 1,
Return[0]
];

(*gives a list of either all odd or even numbers up to and including the length*)
listOfIndices=Select[Table[i,{i,1,Length[sequence]}],parity];
(*list of the vector values at either even or odd indices*)
vectorValues=Table[sequence[[i]],{i,listOfIndices}];

If[listOfIndices[[1]]== 1,
(*Return the value of either the even or odd summation. If the first element of the list of indic
otherwise, return the even summation.*)

(*only indclude values at odd indices that are negative*)
sumOfIndices= Total[Table[Abs[i]/2,{i,Select[vectorValues,#<0&]}]],
(*only indclude values at even indices that are positive*)
sumOfIndices= Total[Table[i/2,{i,Select[vectorValues,#>0&]}]];
];
Return[sumOfIndices]
]
```

In[ ]:= `sumOfValuesAtIndices[{3}, OddQ]`

In[ ]:= `sumOfValuesAtIndices[{-3}, OddQ]`

In[ ]:= `sumOfValuesAtIndices[{-2, 1, 5}, EvenQ]`

Braid Index Two-Bridge

Theorem 5.1. *Let $K = b(\alpha, \beta)$ be a two bridge link diagram with signed vector $(b_1, b_2, ..., b_{2k+1})$ in the normal form, then the braid index is given by*

Out[ ]=

$$\boldsymbol{b}(K) = 1 + \frac{2 + \text{sign}(b_1) + \text{sign}(b_{2k+1})}{4} + \sum_{b_{2j}>0, 1\le j\le k} \frac{b_{2j}}{2} + \sum_{b_{2j+1}<0, 0\le j\le k} \frac{|b_{2j+1}|}{2}. \tag{5.1}$$

```
(*Returns all braid indices determined by the possible signed vectors of a two-
 bridge link. Is given a list of positive integers
  representing the continued fraction for the link*)
braidIndexTwoBridge[sequence_] := Module[
  (*oddSequence is the sequence converted to odd length,
  if the sequence is of even length*)
  {oddSequence = correctSequence[sequence], signedVector1 ,
   signedVector2, braidIndex = 0, sign1, sign2, sumBothParities},
  (*using the sequence that has been updated to odd length,
  update vector to have signs reflecting the signs of each section of crossings*)
  signedVector1 = bothSignedVectors[oddSequence][[1]];
  signedVector2 = bothSignedVectors[oddSequence][[2]];
  If[EvenQ[Numerator[FromContinuedFraction[sequence]]],
   Return[{braidIndexNumber[signedVector1], braidIndexNumber[signedVector2]}],
   Return[braidIndexNumber[signedVector1]]

  ]
 ]
```

In[ ]:= `braidIndexTwoBridge[{1, 2, 3}]`

In[ ]:= `braidIndexTwoBridge[{1, 2, 2}]`

In[ ]:= `toSignedVectorTwoBridge[{1, 2, 2}]`

Below, two braid indices are expected because the continued fraction does not represent a knot. We know this because the sequence's continued fraction has a even numerator.

In[ ]:= `braidIndexTwoBridge[{2, 5, 7, 2, 1}]`

In[ ]:= `FromContinuedFraction[{2, 5, 7, 2, 1}]`

```
(*determines the braid index for the given signed vector*)
braidIndexNumber[signedVector_] := Module[
  {braidIndex = 0, sign1, sign2, sumBothParities},
  (*using the sequence that has been updated to odd length,
  update vector to have signs reflecting the signs of each section of crossings*)

  sumBothParities = sumOfValuesAtIndices[signedVector, EvenQ] +
    sumOfValuesAtIndices[signedVector, OddQ];

  sign1 = Sign[signedVector[[1]]];
  sign2 = Sign[Last[signedVector]];
  braidIndex = 1 + (2 + sign1 + sign2)/4 + sumBothParities;
  Return[braidIndex]

 ]
```

In[ ]:= `braidIndexNumber[{-2, 1, 5}]`

In[ ]:= `braidIndexNumber[{-3}]`

```
In[ ]:= braidIndexNumber[{3}]
```

```
(*determine the braidIndex for a two-
 bridge link given the continued fraction in numerator/denominator form*)
braidIndexForFraction[frac_] := braidIndexTwoBridge[ContinuedFraction[frac]]
```

```
In[ ]:= braidIndexForFraction[17 / 6]
In[ ]:= braidIndexForFraction[6 / 5]
```

Testing:

```
In[ ]:= ContinuedFraction[17 / 6]
In[ ]:= braidIndexForFraction[17 / 6]
In[ ]:= braidIndexForFraction[3]
In[ ]:= braidIndexForFraction[17 / 6]
In[ ]:= braidIndexForFraction[131 / 30]
In[ ]:= braidIndexForFraction[125 / 33]
In[ ]:= braidIndexForFraction[183 / 67]
In[ ]:= first18Knots = {3, 5/2, 5, 7/3, 9/7, 11/4, 13/5, 7, 11/5, 13/9, 15/11, 17/7, 19/7, 21/8, 13/11, 17/6, 17/4, 19/14};
In[ ]:= braidIndexForfirst18Knots = Table[braidIndexForFraction[first18Knots[[i]]], {i, 18}]
In[ ]:=
In[ ]:= actualBraidIndexForFirst18Knots = {2, 3, 2, 3, 4, 3, 3, 2, 4, 3, 4, 3, 4, 4, 5, 3, 5, 4};
In[ ]:= braidIndexForfirst18Knots == actualBraidIndexForFirst18Knots
In[ ]:= braidIndexForFraction[97 / 21]
In[ ]:= braidIndexForFraction[123 / 89]
In[ ]:= braidIndexForFraction[109 / 30]
In[ ]:= braidIndexForFraction[53 / 8]
In[ ]:= braidIndexForFraction[79 / 61]
In[ ]:= braidIndexForFraction[11]
In[ ]:= braidIndexForFraction[95 / 43]
In[ ]:= braidIndexForFraction[179 / 74]
```

Braid Index M Links

Braid Index B Links

Braid Index B Links with Multiple Components

```
(*given the orientation of all incoming links*)
braidIndexBMultipleComponents[mLink_, fourDirections_] := Module[
  {twoIncomingOrientations, braidIndex},
  twoIncomingOrientations = Table[
    {fourDirections[[i, 1]], fourDirections[[i, 3]]}, {i, Length[fourDirections]}];

  braidIndex = braidIndexBEquation[mLink, twoIncomingOrientations, fourDirections];
  Return[braidIndex];
 ]
```

In[ ]:=
```
seifertClass[{{3, 2, 3}, {2, 2, 2}, {4, 3, 2}, {2}, 1},
 fourDirectionsOfmLinkAllPossible[{{3, 2, 3}, {2, 2, 2}, {4, 3, 2}, {2}, 1}][[1]]]
```

In[ ]:=
```
braidIndexBMultipleComponents[{{3, 2, 3}, {2, 2, 2}, {4, 3, 2}, {2}, 1},
 fourDirectionsOfmLinkAllPossible[{{3, 2, 3}, {2, 2, 2}, {4, 3, 2}, {2}, 1}][[1]]]
```

braidIndexUnspecified

BraidIndex Unspecified for Different Inputs

```
(*Returns the braid index of a Montesinos link given its fourDirections list.*)
braidIndexUnspecified[mLink_, fourDirections_] :=
 Which[
  seifertClass[mLink, fourDirections] == "M1",
  Return[braidIndexM[mLink, "M1"]],
  seifertClass[mLink, fourDirections] == "M2",
  Return[braidIndexM[mLink, "M2"]],
  seifertClass[mLink, fourDirections] == "B",
  Return[braidIndexBMultipleComponents[mLink, fourDirections]]
 ]
```

```
(*Returns the list of signed vectors of the input mLink given its fourDirections list*)
unspecifiedSignedVector[mLink_,fourDirections_]:=
Which[
seifertClass[mLink,fourDirections]== "M1",
Return[toSignedVectorWholeMSequence[mLink,"M1"]],
seifertClass[mLink,fourDirections]== "M2",
Return[toSignedVectorWholeMSequence[mLink,"M2"]],
seifertClass[mLink,fourDirections] == "B",
Return[toSignedVectorWholeBSequence[mLink, fourDirections]];
]
```

In[ ]:=
```
unspecifiedSignedVector[{{1, 3, 2}, {1, 1, 2}, {2}, 0},
 fourDirectionsOfmLink[{{1, 3, 2}, {1, 1, 2}, {2}, 0}]]
```

```
(*Computes the braid index of an input mLink represented
 as a list of fractions given its fourDirections list*)
braidIndexUnspecifiedForFrac[mLink_, fourDirections_] := Module[
  {continuedFractionForm, tangleList},
  tangleList = Table[ContinuedFraction[1/mLink[[i]]], {i, Length[mLink] - 1}];
  continuedFractionForm = AppendTo[tangleList, Last[mLink]];
  Return[braidIndexUnspecified[continuedFractionForm, fourDirections]];
 ]
```

In[•]:= `fourDirectionsOfmLinkAllPossible[{{2, 1, 2, 2}, {3}, {2}, 0}]`

In[•]:= `ContinuedFraction /@ {7/19, 1/3, 1/2, 0}`

In[•]:= `braidIndexUnspecifiedForFrac[{7/19, 1/3, 1/2, 0}, fourDirectionsOfmLinkAllPossible[{{2, 1, 2, 2}, {3}, {2}, 0}]]`

```
(*Returns the Seifert Class (M1, M2, or B) of an mLink,
given its fourDirections list *)
seifertClass[mLink_, fourDirections_] := Module[
  {parityList = {}},

  parityList = seifertParity /@ eightArcCases[fourDirections];

  If[ContainsAny[parityList, {"Parity 3"}], Return["B"]];
  If[ContainsOnly[parityList, {"Parity 1"}], Return["M1"]];
  If[ContainsOnly[parityList, {"Parity 2"}], Return["M2"]];

 ]
```

In[•]:= `seifertClass[{{3}, {3}, {3}, {1, 2}, 0}]`

In[•]:= `seifertClass[{{3}, {3}, {3}, {1, 2}, 0}]`

In[•]:= `seifertClass[{{2, 1, 2, 2}, {3}, {2}, 0}, fourDirectionsOfmLink[{{2, 1, 2, 2}, {3}, {2}, 0}]]`

In[•]:= `seifertClass[{{4}, {1, 1, 2}, {3}, 1}]`

In[•]:= `seifertClass[{{1, 1, 1, 2, 2}, {1, 1, 1}, {2}, 0}]`

```
(*Computes the braid index of an input mLink
 represented in Conway notation given its fourDirections list*)
braidIndexUnspecifiedForConway[conway_, fourDirections_] := Module[
  {mLink = formatAsInput[conway]},
  Which[
    seifertClass[mLink, fourDirections] == "M1",
    Return[braidIndexM[mLink, "M1"]],
    seifertClass[mLink, fourDirections] == "M2",
    Return[braidIndexM[mLink, "M2"]],
    seifertClass[mLink, fourDirections] == "B",
    Return[braidIndexBMultipleComponents[mLink, fourDirections]]
   ];
 ]
```

In[ ]:=
```
braidIndexUnspecifiedForConway["[2 2 2; 2 1; 4 +1]",
 fourDirectionsOfmLinkAllPossible[formatAsInput["[2 2 2; 2 1; 4 +1]"]][[1]]]
```

Testing for M1 links

The data for tests 2-13 comes from KnotInfo.

https://www.indiana.edu/~knotinfo/results.php?searchmode=%3C%3Fphp+%24show_mobile%3F+1+%3A+0%3F%3E&searchmode=0&category%5B%5D=%2B0%3C%3D6&category%5B%5D=%3D%2711a%27&category%5B%5D=%3D%2712a1%27&category%5B%5D=%3D%2712a2%27&category%5B%5D=%3D%2712a3%27&category%5B%5D=%3D%2712a4%27&category%5B%5D=%3D%2712a5%27&category%5B%5D=%3D%2712a6%27&category%5B%5D=%3D%2712a7%27&name=%3D1&conway_notation=%3D1&braid_index=%3D1&startrow=0&rows=1662

Test 1: M(12/19,2/3,2). (From paper.) Braid Index: 5.

Test 2: 12a_0750. Conway: [3;3;3;2 1]. Braid Index: 6.

In[ ]:=
```
braidIndexMForConway["[3;3;3;2 1]", "M1"]
```

In[ ]:=
```
braidIndexM[{{3}, {3}, {3}, {1, 2}, 0}, "M1"]
```

Test 3: 12a_0208. Conway: [2 3;2 1;2 1+ 1]. Braid Index: 6.

In[ ]:=
```
braidIndexMForConway["[2 3;2 1;2 1+ 1]", "M1"]
```

Test 4: 12a_0504. Conway: [2 2;3;2 1+ 2]. Braid Index: 5

In[ ]:=
```
braidIndexMForConway["[2 2;3;2 1+ 2]", "M1"]
```

Test 5: 12a_0505. Conway: [3 1 1;2 1;2 1+ 1]. Braid index: 5

```
In[•]:= braidIndexMForConway["[3 1 1;2 1;2 1+ 1]", "M1"]
In[•]:= braidIndexM[{{1, 1, 3}, {1, 2}, {1, 2}, 1}, "M1"]
```

Test 6: 12_a_0513. Conway: [2 2;2 1 1;3+ 1]. Braid index: 5

```
In[•]:= braidIndexMForConway["[2 2;2 1 1;3+ 1]", "M1"]
In[•]:= braidIndexM[{{2, 1, 1}, {1, 1, 2}, {3}, 1}, "M1"]
```

Test 7: 12a_0515. Conway: [2 1 1;2 1;2 1+ 2]. Braid index: 5

```
In[•]:= braidIndexMForConway["[2 1 1;2 1;2 1+ 2]", "M1"]
In[•]:= braidIndexM[{{1, 1, 2}, {1, 2}, {1, 2}, 2}, "M1"]
```

Test 8: 12a_0516. Conway:[2 1 1;2 1 1;2 1+ 1]. Braid index: 5

```
In[•]:= braidIndexMForConway["[2 1 1;2 1 1;2 1+ 1]", "M1"]
In[•]:= braidIndexM[{{1, 1, 2}, {1, 1, 2}, {1, 1, 1}, 1}, "M1"]
```

Test 9: 12a_0524. Conway: [2 2 1;3;2 1+ 1]. Braid index: 6

```
In[•]:= braidIndexMForConway["[2 2 1;3;2 1+ 1]", "M1"]
In[•]:= braidIndexM[{{1, 2, 2}, {3}, {1, 2}, 1}, "M1"]
```

Test 10: 12a_0525. Conway: [4 1;2 1;2 1+ 1]. Braid index: 6

```
In[•]:= braidIndexMForConway["[4 1;2 1;2 1+ 1]", "M1"]
In[•]:= braidIndexM[{{1, 4}, {1, 2}, {1, 2}, 1}, "M1"]
```

Test 11: 12a_0542. Conway: [2 1 1 2;3;3]. Braid index: 6

```
In[•]:= braidIndexMForConway["[2 1 1 2;3;3]", "M1"]
In[•]:= braidIndexM[{{2, 1, 1, 2}, {3}, {3}, 0}, "M1"]
```

Test 12: 12a_0229. Conway: [2 1 1 2;2 1;2 1]. Braid index: 6

```
In[•]:= braidIndexMForConway["[2 1 1 2;2 1;2 1]", "M1"]
In[•]:= braidIndexM[{{2, 1, 1, 2}, {1, 2}, {1, 2}, 0}, "M1"]
```

Test 13: 12_a_332. Conway: [2 3 1;2 1;2 1]. Braid index: 6

```
In[•]:= braidIndexMForConway["[2 3 1;2 1;2 1]", "M1"]
```

```
In[ ]:= braidIndexM[{{3}, {3}, {3}, {1, 2}, 0}, "M1"]
```

Testing for M2 links

The data for tests 1-12 comes from KnotInfo.

https://www.indiana.edu/~knotinfo/results.php?searchmode=%3C%3Fphp+%24show_mobile%3F+1+%3A+0%3F%3E&searchmode=0&category%5B%5D=%2B0%3C%3D6&category%5B%5D=%3D%2711a%27&category%5B%5D=%3D%2712a1%27&category%5B%5D=%3D%2712a2%27&category%5B%5D=%3D%2712a3%27&category%5B%5D=%3D%2712a4%27&category%5B%5D=%3D%2712a5%27&category%5B%5D=%3D%2712a6%27&category%5B%5D=%3D%2712a7%27&name=%3D1&conway_notation=%3D1&braid_index=%3D1&startrow=0&rows=1662

Test 1. 12a_0648. Conway:[3 3;2 1;2 1]. Braid Index: 5

```
In[ ]:= braidIndexM[formatAsInput["[3 3;2 1;2 1]"], "M2"]
Out[ ]= 5
```

```
In[ ]:= formatAsInput["[3 3;2 1;2 1]"]
Out[ ]= {{3, 3}, {1, 2}, {1, 2}, 0}
```

Test 2. 12a_0679. Conway: [6;2 1;2 1]. Braid Index: 6

```
In[ ]:= braidIndexM[formatAsInput["[6;2 1;2 1]"], "M2"]
Out[ ]= 6
```

12a_0877. Conway: [4 1;3 1 1;2]. Braid Index: 5

```
In[ ]:= braidIndexM[formatAsInput["[4 1;3 1 1;2]"], "M2"]
Out[ ]= 5
```

Test 3. 12_a_0880. Conway: [3 1 1;3 1 1;2]. Braid Index: 6

```
In[ ]:= braidIndexM[formatAsInput["[3 1 1;3 1 1;2]"], "M2"]
Out[ ]= 6
```

Test 4. 12a_0082. Conway:[2 2 2 1;2 1;2]. Braid Index: 5

```
In[ ]:= braidIndexMForConway["[2 2 2 1;2 1;2]", "M2"]
Out[ ]= 5
```

Test 5. 12a_0096. Conway: [4 3;2 1;2]. Braid Index: 5

```
In[ ]:= braidIndexMForConway["[4 3;2 1;2]", "M2"]
Out[ ]= 5
```

Test 6. 12a_0097. Conway: [3 1 3;2 1;2]. Braid Index: 6

```
In[ ]:= braidIndexMForConway["[3 1 3;2 1;2]", "M2"]
Out[ ]= 6
```

Test 7. 11a_43. Conway: [21;21;21;2]. Braid Index: 5

```
In[ ]:= braidIndexMForConway["[21;21;21;2]", "M2"]
Out[ ]= 5
```

Test 8. 12a_0117. Conway: [2 2;2 1;2 1;2]. Braid Index: 6

```
In[ ]:= braidIndexMForConway["[2 2;2 1;2 1;2]", "M2"]
Out[ ]= 6

In[ ]:= formatAsInput["[2 2;2 1;2 1;2]"]
Out[ ]= {{2, 2}, {1, 2}, {1, 2}, {2}, 0}
```

Test 9. 12a_0880. Conway: [3 1 1;3 1 1;2]. Braid Index: 6

```
In[ ]:= braidIndexMForConway["[3 1 1;3 1 1;2]", "M2"]
Out[ ]= 6
```

Test 10. 12a_0877. Conway: [4 1;3 1 1;2]. Braid Index: 5

```
In[ ]:= braidIndexMForConway["[4 1;3 1 1;2]", "M2"]
Out[ ]= 5
```

Test 11. 12a_0679. Conway: [6;2 1;2 1]. Braid Index: 6

```
In[ ]:= braidIndexMForConway["[6;2 1;2 1]", "M2"]
Out[ ]= 6
```

Test 12. 12a_0648. Conway: [3 3;2 1;2 1]. Braid Index: 5.

```
In[ ]:= braidIndexMForConway["[3 3;2 1;2 1]", "M2"]
Out[ ]= 5
```

Test 13. M(12/19,2/3,1/2). (From paper.) Braid Index: 6

```
In[ ]:= braidIndexMForFraction[{12/19, 2/3, 1/2, 0}, "M2"]
Out[ ]= 6
```

Testing for type B knots

Test1. 12a_0017. Braid Index: 5.

```
In[ ]:= seifertClass[formatAsInput["[2 1 1 1 1;2 1 1;2]"],
         fourDirectionsOfmLink[formatAsInput["[2 1 1 1 1;2 1 1;2]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[2 1 1 1 1;2 1 1;2]"]]
Out[ ]= 5
```

Test 2. 12a_0021. Braid Index: 5.

```
In[ ]:= seifertClass[formatAsInput["[2 2 1;2 1 1;2+ 1]"],
         fourDirectionsOfmLink[formatAsInput["[2 2 1;2 1 1;2+ 1]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[2 2 1;2 1 1;2+ 1]"]]
Out[ ]= 5
```

Test 3. 12a_0022. Braid Index: 6.

```
In[ ]:= seifertClass[formatAsInput["[4 1 1;2 1 1;2]"],
         fourDirectionsOfmLink[formatAsInput["[4 1 1;2 1 1;2]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[4 1 1;2 1 1;2]"]]
Out[ ]= 6
```

Test 4. 12a_0027. Braid Index: 6.

```
In[ ]:= seifertClass[formatAsInput["[3 1 1;2 1 1;2+ 1]"],
         fourDirectionsOfmLink[formatAsInput["[3 1 1;2 1 1;2+ 1]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[3 1 1;2 1 1;2+ 1]"]]
Out[ ]= 6
```

Test 5. 12a_0034. Braid index: 4.

```
In[ ]:= seifertClass[formatAsInput["[5;2 1 1;2+ 1]"],
         fourDirectionsOfmLink[formatAsInput["[5;2 1 1;2+ 1]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[5;2 1 1;2+ 1]"]]
Out[ ]= 4
```

Test 6. 12a_0003. Braid Index: 6.

```
In[ ]:= seifertClass[formatAsInput["[2 3 1;2 1 1;2]"],
         fourDirectionsOfmLink[formatAsInput["[2 3 1;2 1 1;2]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput["[2 3 1;2 1 1;2]"]]
Out[ ]= 6
```

Test 7. 12a_0019. Braid Index: 5.

```
In[ ]:= seifertClass[formatAsInput["[3 1 1 1;2 1 1;2]"],
         fourDirectionsOfmLink[formatAsInput["[3 1 1 1;2 1 1;2]"]]]
Out[ ]= B

In[ ]:= braidIndexB[formatAsInput@"[3 1 1 1;2 1 1;2]"]
Out[ ]= 5
```

Test 8. 12a_0020. Braid Index: 5.

```
In[ ]:= seifertClass[formatAsInput["[2 1 1 2;2 1 1;2]"],
         fourDirectionsOfmLink[formatAsInput["[2 1 1 2;2 1 1;2]"]]]
Out[ ]= B

In[ ]:= braidIndexBForConway["[2 1 1 2;2 1 1;2]"]
Out[ ]= 5
```

Test 9. 12a_0026. Braid index: 5.

```
In[ ]:= seifertClass[formatAsInput@"[4 1;2 1 1;2+ 1]",
         fourDirectionsOfmLink[formatAsInput["    [4 1;2 1 1;2+ 1]"]]]
Out[ ]= B

In[ ]:= braidIndexBForConway["    [4 1;2 1 1;2+ 1]"]
Out[ ]= 5
```

Test 10. 12a_0027. Braid Index: 6.

```
In[•]:= seifertClass[formatAsInput@"[2 2;2 1 1;2+ 2]",
         fourDirectionsOfmLink[formatAsInput["[2 2;2 1 1;2+ 2]"]]]
Out[•]= B

In[•]:= braidIndexBForConway["[2 2;2 1 1;2+ 2]"]
Out[•]= 6
```

Test 11. 12a_0029. Braid Index: 5

```
In[•]:= seifertClass[formatAsInput@"[2 1 1;2 1;2 1;2]",
         fourDirectionsOfmLink[formatAsInput["[2 1 1;2 1;2 1;2]"]]]
Out[•]= B

In[•]:= braidIndexBForConway["[2 1 1;2 1;2 1;2]"]
Out[•]= 5
```

Test 12. (Example 5.8 from paper.) Braid Index: 5.

```
In[•]:= braidIndexBForFraction[{7/19, 1/3, 1/2, 0}]
Out[•]= 5
```

Test 13. (Example 5.9 from paper.) Braid Index: 4.

```
In[•]:= braidIndexBForFraction[{1/4, 3/5, 1/3, 1}]
Out[•]= 4
```

Testing for braidIndexUnspecified

Test 1: M(12/19,2/3,2). (From paper.) Braid Index: 5. Type M1.

```
In[•]:= ContinuedFraction /@ {12/19, 2/3, 2}
Out[•]= {{0, 1, 1, 1, 2, 2}, {0, 1, 2}, {2}}

In[•]:= fourDirectionsOfmLinkAllPossible[{{1, 1, 1, 2, 2}, {1, 2}, 2}]
Out[•]= {{{L, R, L, R}, {L, R, L, R}}, {{R, R, R, R}, {R, R, R, R}}}
```

Choose the orientation which produces an M1 link:

```
In[•]:= braidIndexUnspecified[{{1, 1, 1, 2, 2}, {1, 2}, 2},
         fourDirectionsOfmLinkAllPossible[{{1, 1, 1, 2, 2}, {1, 2}, 2}][[2]]]
Out[•]= 5
```

Test 2: 12a_0750. Conway: [3;3;3;2 1]. Braid Index: 6. Type M1.

```
In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[3;3;3;2 1]"]]
Out[•]= {{R, R, R, R}, {R, R, R, R}, {R, R, R, R}, {R, R, R, R}}

In[•]:= braidIndexUnspecified[formatAsInput["[3;3;3;2 1]"],
         {{"R", "R", "R", "R"}, {"R", "R", "R", "R"}, {"R", "R", "R", "R"}, {"R", "R", "R", "R"}}]
Out[•]= 6
```

Test 3: 12a_0208. Conway: [2 3;2 1;2 1+ 1]. Braid Index: 6. Type M1.

```
In[•]:= formatAsInput["[2 3;2 1;2 1+ 1]"]
Out[•]= {{3, 2}, {1, 2}, {1, 2}, 1}

In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[2 3;2 1;2 1+ 1]"]]
Out[•]= {{R, R, R, R}, {R, R, R, R}, {R, R, R, R}}

In[•]:= braidIndexUnspecified[formatAsInput["[2 3;2 1;2 1+ 1]"],
         {{"R", "R", "R", "R"}, {"R", "R", "R", "R"}, {"R", "R", "R", "R"}}]
Out[•]= 6
```

Test 4: 12a_0504. Conway: [2 2;3;2 1+ 2]. Braid Index: 5. Type M1.

```
In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[2 2;3;2 1+ 2]"]]
Out[•]= {{R, R, R, R}, {R, R, R, R}, {R, R, R, R}}

In[•]:= braidIndexUnspecified[formatAsInput["[2 2;3;2 1+ 2]"],
         {{"R", "R", "R", "R"}, {"R", "R", "R", "R"}, {"R", "R", "R", "R"}} ]
Out[•]= 5
```

Test 5. 12a_0877. Conway: [4 1;3 1 1;2]. Braid Index: 5. Type M2.

```
In[•]:= formatAsInput[" [4 1;3 1 1;2]"]
Out[•]= {{1, 4}, {1, 1, 3}, {2}, 0}

In[•]:= fourDirectionsOfmLinkAllPossible[{{1, 4}, {1, 1, 3}, {2}, 0}]
Out[•]= {{L, R, L, R}, {L, R, L, R}, {L, R, L, R}}

In[•]:= braidIndexUnspecified[{{1, 4}, {1, 1, 3}, {2}, 0},
         {{"L", "R", "L", "R"}, {"L", "R", "L", "R"}, {"L", "R", "L", "R"}}]
Out[•]= 5
```

Test 6. 12a_0679. Conway: [6;2 1;2 1]. Braid Index: 6. Type M2.

```
In[•]:= formatAsInput[" [6;2 1;2 1]"]
Out[•]= {{6}, {1, 2}, {1, 2}, 0}

In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput[" [6;2 1;2 1]"]]
Out[•]= {{L, R, L, R}, {L, R, L, R}, {L, R, L, R}}

In[•]:= braidIndexUnspecified[formatAsInput[" [6;2 1;2 1]"],
         fourDirectionsOfmLink[formatAsInput[" [6;2 1;2 1]"]]]
Out[•]= 6
```

Test 7. 12a_0648. Conway: [3 3;2 1;2 1]. Braid Index: 5. Type M2.

```
In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[3 3;2 1;2 1]"]]
Out[•]= {{L, R, L, R}, {L, R, L, R}, {L, R, L, R}}

In[•]:= braidIndexUnspecified[formatAsInput["[3 3;2 1;2 1]"],
         fourDirectionsOfmLink[formatAsInput["[3 3;2 1;2 1]"]]]
Out[•]= 5
```

Test 8. M(12/19,2/3,1/2). (From paper.) Braid Index: 6. Type M2.

```
In[•]:= ContinuedFraction /@ {12 / 19, 2 / 3, 1 / 2, 0}
Out[•]= {{0, 1, 1, 1, 2, 2}, {0, 1, 2}, {0, 2}, {0}}

In[•]:= fourDirectionsOfmLinkAllPossible[{{1, 1, 1, 2, 2}, {1, 2}, {2}, 0}]
Out[•]= {{L, R, L, R}, {L, R, L, R}, {L, R, L, R}}

In[•]:= braidIndexUnspecified[{{1, 1, 1, 2, 2}, {1, 2}, {2}, 0},
         {{"L", "R", "L", "R"}, {"L", "R", "L", "R"}, {"L", "R", "L", "R"}} ]
Out[•]= 6
```

Test 9. 12a_0026. Braid index: 5. Knot of type B.

```
In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[4 1;2 1 1;2+ 1]"]]
Out[•]= {{L, R, L, R}, {L, R, R, L}, {R, L, R, L}}

In[•]:= braidIndexUnspecified[formatAsInput["[4 1;2 1 1;2+ 1]"],
         fourDirectionsOfmLink[formatAsInput["[4 1;2 1 1;2+ 1]"]]]
Out[•]= 5
```

Test 10. 12a_0027. Braid Index: 6. Knot of type B.

```
In[•]:= fourDirectionsOfmLinkAllPossible[formatAsInput["[2 2;2 1 1;2+ 2]"]]
Out[•]= {{L, R, L, R}, {L, R, R, L}, {R, L, L, R}}

In[•]:= braidIndexUnspecified[formatAsInput["[2 2;2 1 1;2+ 2]"],
         fourDirectionsOfmLink[formatAsInput["[2 2;2 1 1;2+ 2]"]]]
Out[•]= 6
```

Test 11. (Example 5.12 from paper.) Type B. Braid Index: 9.

```
In[•]:= ContinuedFraction /@ {17 / 44, 7 / 10, 19 / 26, 2}
Out[•]= {{0, 2, 1, 1, 2, 3}, {0, 1, 2, 3}, {0, 1, 2, 1, 2, 2}, {2}}

In[•]:= numComponents[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}]
Out[•]= 3

In[•]:= fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}]
Out[•]= {{{L, R, L, R}, {L, R, L, R}, {L, R, L, R}}, {{L, R, L, R}, {L, R, R, L}, {R, L, L, R}},
         {{L, R, R, L}, {R, L, L, R}, {L, R, L, R}}, {{L, R, R, L}, {R, L, R, L}, {R, L, L, R}}}
```

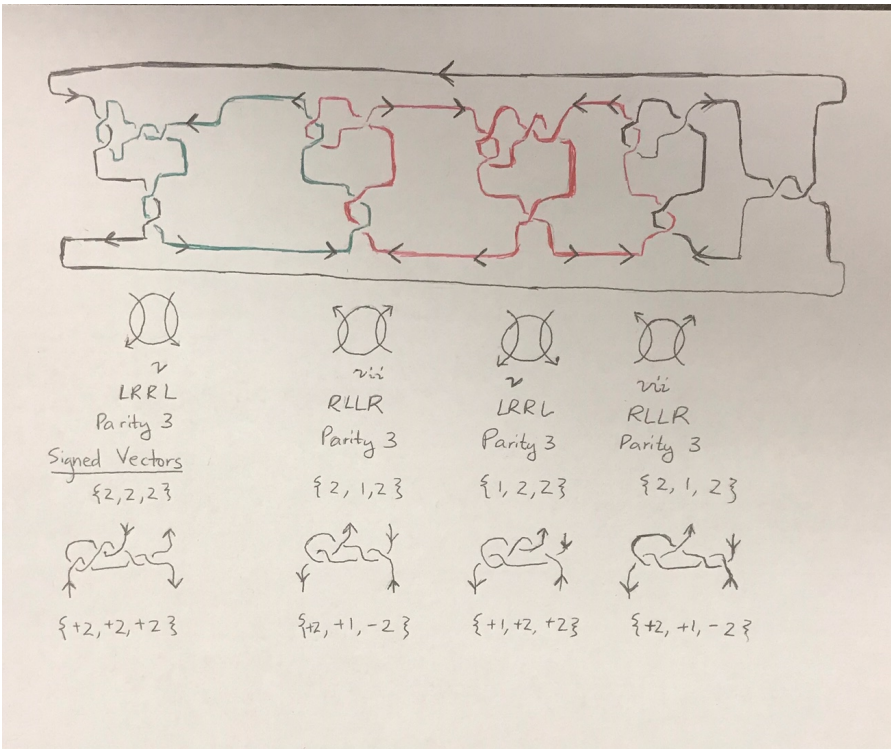
All possible braid indices of this link, up to orientation:

```
In[•]:= Table[braidIndexUnspecified[
          {{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}, fourDirectionsOfmLinkAllPossible[
            {{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}][[count]]], {count,
          Length@fourDirectionsOfmLinkAllPossible[{{2, 1, 1, 2, 3}, {1, 2, 3}, {1, 2, 1, 2, 2}, 2}]}]
Out[•]= {9, 9, 9, 9}
```

Test 12. (Verified by hand.) Type B link with multiple components. Braid Index: 8.

```
In[•]:= fourDirectionsOfmLinkAllPossible[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 2}]
Out[•]= {{{L, R, L, R}, {L, R, L, R}, {L, R, R, L}, {R, L, L, R}},
         {{L, R, L, R}, {L, R, R, L}, {R, L, L, R}, {L, R, L, R}},
         {{L, R, R, L}, {R, L, L, R}, {L, R, R, L}, {R, L, L, R}},
         {{L, R, R, L}, {R, L, R, L}, {R, L, L, R}, {L, R, L, R}}}

In[•]:= seifertClass[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 2},
         fourDirectionsOfmLinkAllPossible[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 2}][[3]]]
Out[•]= B

In[•]:= braidIndexUnspecified[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 2},
         fourDirectionsOfmLinkAllPossible[{{2, 2, 2}, {2, 1, 2}, {1, 2, 2}, {2, 1, 2}, 2}][[3]]]
Out[•]= 8
```

Diagram of Montesinos link



Calculation of braid index:

Formula for braid index of a type B link:

$$b(L) = \triangle_0(L) + \sum_{A_j \in \Omega_2} \triangle_2(A_j) + \sum_{A_j \in \Omega_3} \triangle_3(A_j)$$

$\triangle_0$ formula:

$$\triangle_0(\mathrm{L}) = \eta + e - \min\{(\eta + e)/2 - 1,\ e\}$$

The number of Parity 3 tangles, η, is 4, and the number of lone crossings, e, is 2.

$$\triangle_0(\mathrm{L}) = 4 + 2 - \min\left\{\frac{4+2}{2} - 1, 2\right\} = 4$$

Formula for Parity 2 tangles (there are no Parity 2 tangles in this link).

$$\sum_{A_j \in \Omega_2} \triangle_2(A_j) = 0$$

Formula for Parity 3 tangles.

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = \triangle_3(A_1) + \triangle_3(A_2) + \triangle_3(A_3)$$

$$\triangle_3(A_j) = \frac{-1 + \mathrm{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\, 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\, 0\le m\le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = \left[\left(\frac{-1+1}{4}\right) + \frac{2}{2} + \frac{|\,0\,|}{2}\right] + \left[\right.$$

$$\left(\frac{-1-1}{4}\right) + \frac{1}{2} + \frac{|\,-2\,|}{2}\right] + \left[\left(\frac{-1+1}{4}\right) + \frac{2}{2} + \frac{|\,0\,|}{2}\right] + \left[\left(\frac{-1-1}{4}\right) + \frac{1}{2} + \frac{|\,-2\,|}{2}\right]$$

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = [1] + [1] + [1] + [1] = 4$$

$$b(L) = \triangle_0(L) + \sum_{A_j \in \Omega_3} \triangle_3(A_j) = 4 + 4 = 8$$

The braid index is 8.

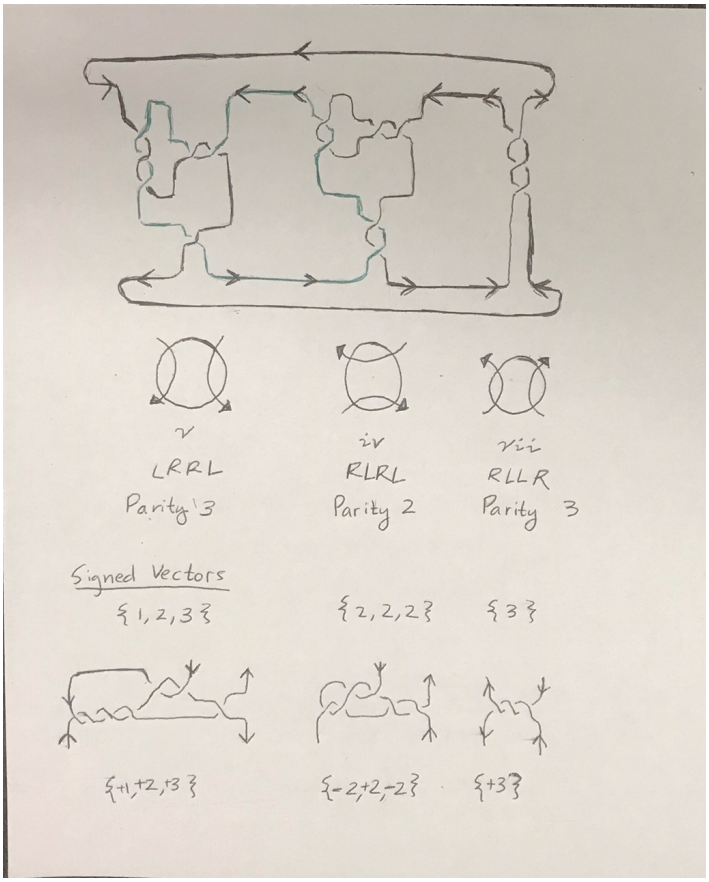
Test 13. (Verified by hand.) Type B link with multiple components. Braid Index:

6. (finish calculating)

```
In[•]:= fourDirectionsOfmLinkAllPossible[{{1, 2, 3}, {2, 2, 2}, {3}, 0}]
Out[•]= {{{L, R, L, R}, {L, R, R, L}, {R, L, L, R}}, {{L, R, R, L}, {R, L, R, L}, {R, L, L, R}}}

In[•]:= seifertClass[{{1, 2, 3}, {2, 2, 2}, {3}, 0},
         fourDirectionsOfmLinkAllPossible[{{1, 2, 3}, {2, 2, 2}, {3}, 0}][[2]]]
Out[•]= B

In[•]:= braidIndexUnspecified[{{1, 2, 3}, {2, 2, 2}, {3}, 0},
         fourDirectionsOfmLinkAllPossible[{{1, 2, 3}, {2, 2, 2}, {3}, 0}][[2]]]
Out[•]= 6
```

```
{{1, 2, 3}, {2, 2, 2}, {3}, 0},
{{"L", "R", "R", "L"}, {"R", "L", "R", "L"}, {"R", "L", "L", "R"}}
```

Diagram of Montesinos link



Calculation of braid index:

Formula for braid index of a type B link:

$$b(L) = \triangle_0(L) + \sum_{A_j \in \Omega_2} \triangle_2(A_j) + \sum_{A_j \in \Omega_3} \triangle_3(A_j)$$

$\triangle_0$ formula:

$$\triangle_0(\mathrm{L}) = \eta + e - \min\{(\eta + e)/2 - 1,\ e\}$$

The number of Parity 3 tangles, η, is 2, and the number of lone crossings, e, is 0.

$$\triangle_0(\mathrm{L}) = 2 + 0 - \min\left\{\frac{2+0}{2} - 1,\ 0\right\} = 2$$

Formula for Parity 2 tangles.

$$\sum_{A_j \in \Omega_2} \triangle_2(A_j) = \triangle_2(A_2)$$

$$\triangle_2(A_j) = \frac{1 + \mathrm{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m} > 0,\ 1 \le m \le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1} < 0,\ 0 \le m \le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

$$\triangle_2(A_2) = \frac{1 + (-1)}{4} + \left(\frac{2}{2}\right) + \frac{|\,-2 - 2\,|}{2} = 3$$

Formula for Parity 3 tangles.

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = \triangle_3(A_1) + \triangle_3(A_3)$$

$$\triangle_3(A_j) = \frac{-1+\text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\,1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\,0\le m\le q_j} \frac{|\,b^j_{2\,m+1}\,|}{2}$$

$$\sum_{A_j\in\Omega_3} \triangle_3(A_j) = \left[\left(\frac{-1+1}{4}\right)+\frac{2}{2}+\frac{|\,0\,|}{2}\right] + \left[\left(\frac{-1+1}{4}\right)+\frac{0}{2}+\frac{|\,0\,|}{2}\right]$$

$$\sum_{A_j\in\Omega_3} \triangle_3(A_j) = [1] + [0] = 1$$

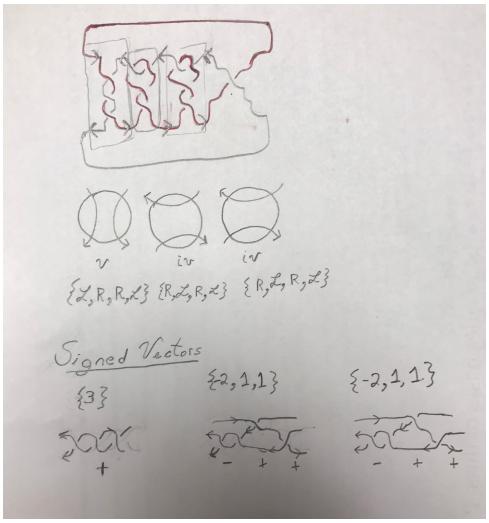
$$b(L) = \triangle_0(L) + \sum_{A_j\in\Omega_3} \triangle_3(A_j) = 6$$

The braid index is 8.

Test 14. (to be verified by hand.) Type B link with multiple components. Braid Index: 6. calculation, image not here yet

```
In[•]:= fourDirectionsOfmLinkAllPossible[{{3}, {2, 1, 1}, {2, 2}, 1}]
Out[•]= {{{L, R, R, L}, {R, L, R, L}, {R, L, R, L}}, {{R, R, R, R}, {R, R, R, R}, {R, R, R, R}}}

In[•]:= seifertClass[{{3}, {2, 1, 1}, {2, 2}, 1},
         fourDirectionsOfmLinkAllPossible[{{3}, {2, 1, 1}, {2, 2}, 1}][[1]]]
Out[•]= B

In[•]:= braidIndexUnspecified[{{3}, {2, 1, 1}, {2, 2}, 1},
         fourDirectionsOfmLinkAllPossible[{{3}, {2, 1, 1}, {2, 2}, 1}][[1]]]
Out[•]= 6
```

Drawing of Link



Calculation of braid index:

Formula for braid index of a type B link:

$$b(L) = \triangle_0(L) + \sum_{A_j \in \Omega_2} \triangle_2(A_j) + \sum_{A_j \in \Omega_3} \triangle_3(A_j)$$

$\triangle_0$ formula:

$$\triangle_0(\mathrm{L}) = \eta + e - \min\{(\eta + e)/2 - 1,\ e\}$$

The number of Parity 3 tangles, η, is 1, and the number of lone crossings, e, is 1.

$$\triangle_0(\mathrm{L}) = 1 + 1 - \min\left\{\frac{1+1}{2} - 1,\ 1\right\} = 2$$

Formula for Parity 2 tangles.

$$\sum_{A_j \in \Omega_2} \triangle_2(A_j) = \triangle_2(A_2) + \triangle_2(A_3)$$

$$\triangle_2(A_j) = \frac{1 + \text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\, 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\, 0\le m\le q_j} \frac{|\, b^j_{2\,m+1}\,|}{2}$$

$$\sum_{A_j \in \Omega_2} \triangle_2(A_j) = \left[\frac{1+(1)}{4} + \left(\frac{1}{2}\right) + \frac{|-2|}{2}\right] + \left[\frac{1+(1)}{4} + \left(\frac{1}{2}\right) + \frac{|-2|}{2}\right] = 4$$

Formula for Parity 3 tangles.

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = \triangle_3(A_1)$$

$$\triangle_3(A_j) = \frac{-1 + \text{sign}\left(b^j_{2\,q_j+1}\right)}{4} + \sum_{b^j_{2\,m}>0,\, 1\le m\le q_j} \frac{b^j_{2\,m}}{2} + \sum_{b^j_{2\,m+1}<0,\, 0\le m\le q_j} \frac{|\, b^j_{2\,m+1}\,|}{2}$$

$$\sum_{A_j \in \Omega_3} \triangle_3(A_j) = \left[\left(\frac{-1+1}{4}\right) + \frac{0}{2} + \frac{|\,0\,|}{2}\right] = 0$$

$$b(L) = \triangle_0(L) + \sum_{A_j \in \Omega_2} \triangle_2(A_j) + \sum_{A_j \in \Omega_3} \triangle_3(A_j) = 2 + 4 = 6$$

The braid index is 6.

Complete Two Bridge Testing (362 cases from KnotInfo)

In[•]:=
```
data = Import[
    "https://www.indiana.edu/~knotinfo/results.php?searchmode=%3C%3Fphp+%24show_mobile%3
      F+1+%3A+0%3F%3E&searchmode=0&category%5B%5D=%2B0%3C13&two_bridge_notation=%3
      D1&braid_index=%3D1&startrow=0&rows=2977", "Data"][[1]];
```

... Part: Part 1 of {} does not exist.

In[ ]:=

In[ ]:=
```
fractionsBraidIndices = Select[data, Length[#] > 1 && IntegerQ[#[[2]]] &];
```

... Part: Part called with 0 arguments; 1 or more arguments are expected.

In[ ]:=
```
braidIndicesFromKnotInfo1 = Table[ToExpression[i[[2]]], {i, fractionsBraidIndices}]
```

Out[ ]=
```
{2, 3, 2, 3, 4, 3, 3, 2, 4, 3, 4, 3, 4, 4, 5, 3, 5, 4, 4, 3, 4, 3, 4, 5, 4, 4, 2, 5, 3, 4, 5, 3,
 4, 5, 3, 4, 4, 5, 4, 5, 5, 4, 4, 5, 4, 5, 4, 4, 4, 4, 6, 3, 6, 5, 3, 4, 5, 4, 3, 5, 5, 4,
 6, 4, 4, 5, 3, 5, 4, 5, 4, 4, 4, 5, 4, 4, 4, 5, 5, 5, 5, 4, 5, 5, 6, 5, 5, 5, 4, 4, 5, 5,
 5, 5, 5, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 5, 5, 5, 6, 5, 6, 5, 5, 6, 6, 6, 6, 4,
 4, 4, 4, 5, 4, 4, 4, 5, 4, 5, 4, 5, 5, 4, 5, 5, 6, 4, 5, 5, 4, 5, 5, 6, 6, 5, 5, 5, 6, 6,
 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 4, 5, 5, 6, 6, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 4, 5, 5,
 6, 3, 4, 2, 6, 6, 7, 6, 5, 6, 5, 6, 5, 6, 6, 6, 5, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6, 5, 6, 6,
 6, 6, 6, 6, 6, 6, 7, 7, 7, 5, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 5, 6, 4, 5, 5, 5, 5, 5, 4,
 5, 5, 5, 6, 4, 5, 5, 5, 5, 6, 6, 6, 6, 5, 4, 5, 5, 5, 5, 6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 5,
 6, 6, 6, 7, 7, 4, 4, 5, 3, 4, 4, 4, 5, 3, 4, 4, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 5,
 4, 5, 5, 4, 5, 5, 6, 6, 6, 5, 6, 5, 6, 6, 7, 4, 5, 4, 5, 5, 6, 5, 6, 5, 6, 7, 3, 4, 4, 4,
 5, 5, 3, 4, 4, 5, 6, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 3, 4, 5, 4, 5, 5, 6, 5, 6, 7}
```

In[•]:= **twoBridgeFractionNotation1 = Table[ToExpression[i[[1]]], {i, fractionsBraidIndices}]**

Out[•]= $\left\{3, \frac{5}{2}, 5, \frac{7}{3}, \frac{9}{7}, \frac{11}{4}, \frac{13}{5}, 7, \frac{11}{5}, \frac{13}{9}, \frac{15}{11}, \frac{17}{7}, \frac{19}{7}, \frac{21}{8}, \frac{13}{11}, \frac{17}{6}, \frac{17}{4}, \frac{19}{14}, \frac{23}{10}, \frac{23}{9}, \frac{25}{9}, \frac{25}{7}, \frac{27}{10},\right.$
$\frac{29}{12}, \frac{29}{11}, \frac{31}{12}, 9, \frac{15}{7}, \frac{19}{13}, \frac{21}{5}, \frac{23}{17}, \frac{27}{5}, \frac{29}{13}, \frac{31}{11}, \frac{31}{9}, \frac{33}{23}, \frac{33}{14}, \frac{35}{13}, \frac{37}{27}, \frac{37}{14}, \frac{39}{16}, \frac{39}{14}, \frac{41}{17},$
$\frac{41}{16}, \frac{41}{15}, \frac{43}{18}, \frac{45}{19}, \frac{47}{18}, \frac{49}{19}, \frac{55}{21}, \frac{17}{15}, \frac{23}{8}, \frac{25}{6}, \frac{27}{20}, \frac{33}{13}, \frac{37}{16}, \frac{43}{16}, \frac{29}{6}, \frac{39}{28}, \frac{45}{17}, \frac{43}{13}, \frac{47}{17}, \frac{53}{22},$
$\frac{57}{22}, \frac{43}{19}, \frac{47}{33}, \frac{41}{9}, \frac{55}{23}, \frac{51}{14}, \frac{35}{16}, \frac{45}{16}, \frac{49}{36}, \frac{59}{23}, \frac{55}{24}, \frac{65}{24}, \frac{61}{44}, \frac{71}{27}, \frac{53}{19}, \frac{63}{26}, \frac{67}{26}, \frac{57}{25}, \frac{69}{29},$
$\frac{65}{18}, \frac{37}{13}, \frac{49}{20}, \frac{51}{20}, \frac{53}{23}, \frac{59}{25}, \frac{61}{22}, \frac{75}{29}, \frac{71}{26}, \frac{81}{31}, \frac{73}{27}, \frac{79}{30}, \frac{89}{34}, \frac{61}{28}, \frac{43}{23}, \frac{59}{32}, \frac{83}{36}, \frac{131}{76}, \frac{101}{57},$
$\frac{107}{47}, \frac{119}{44}, \frac{87}{64}, \frac{129}{50}, \frac{93}{52}, \frac{73}{33}, \frac{121}{50}, \frac{77}{18}, \frac{97}{35}, \frac{103}{37}, \frac{117}{49}, \frac{77}{34}, \frac{109}{64}, \frac{119}{50}, \frac{65}{17}, \frac{73}{56}, \frac{83}{22}, \frac{67}{37},$
$\frac{111}{65}, \frac{59}{45}, \frac{79}{51}, \frac{105}{41}, \frac{111}{31}, \frac{97}{21}, \frac{123}{89}, \frac{57}{20}, \frac{89}{64}, \frac{73}{60}, \frac{115}{34}, \frac{87}{68}, \frac{109}{30}, \frac{95}{39}, \frac{67}{14}, \frac{85}{67}, \frac{83}{19}, \frac{97}{71},$
$\frac{95}{66}, \frac{53}{8}, \frac{63}{11}, \frac{101}{71}, \frac{91}{66}, \frac{47}{40}, \frac{85}{26}, \frac{105}{74}, \frac{73}{16}, \frac{67}{12}, \frac{85}{23}, \frac{89}{27}, \frac{53}{42}, \frac{71}{20}, \frac{71}{55}, \frac{51}{43}, \frac{37}{5}, \frac{71}{49}, \frac{99}{29},$
$\frac{65}{53}, \frac{47}{9}, \frac{69}{49}, \frac{41}{13}, \frac{19}{17}, \frac{105}{76}, \frac{83}{18}, \frac{71}{15}, \frac{93}{25}, \frac{61}{47}, \frac{79}{61}, \frac{65}{14}, \frac{49}{9}, \frac{75}{17}, \frac{59}{11}, \frac{89}{63}, \frac{55}{13}, \frac{61}{19}, \frac{29}{25},$
$\frac{31}{27}, \frac{45}{7}, \frac{79}{55}, \frac{91}{27}, \frac{31}{5}, \frac{53}{43}, \frac{57}{47}, \frac{35}{29}, \frac{25}{3}, \frac{51}{35}, 11, \frac{71}{33}, \frac{49}{23}, \frac{69}{32}, \frac{173}{76}, \frac{105}{47}, \frac{169}{66}, \frac{181}{75}, \frac{87}{40},$
$\frac{127}{57}, \frac{133}{60}, \frac{163}{71}, \frac{159}{59}, \frac{97}{23}, \frac{107}{28}, \frac{191}{80}, \frac{115}{52}, \frac{155}{68}, \frac{147}{61}, \frac{153}{64}, \frac{147}{64}, \frac{157}{69}, \frac{95}{43}, \frac{127}{45}, \frac{71}{17},$
$\frac{77}{20}, \frac{151}{62}, \frac{161}{66}, \frac{179}{74}, \frac{81}{37}, \frac{149}{65}, \frac{121}{43}, \frac{103}{27}, \frac{85}{38}, \frac{169}{70}, \frac{93}{22}, \frac{209}{81}, \frac{207}{76}, \frac{233}{89}, \frac{167}{60}, \frac{199}{55},$
$\frac{91}{37}, \frac{185}{68}, \frac{129}{56}, \frac{193}{81}, \frac{125}{51}, \frac{151}{64}, \frac{187}{79}, \frac{145}{52}, \frac{157}{34}, \frac{111}{25}, \frac{133}{36}, \frac{113}{48}, \frac{173}{73}, \frac{183}{67}, \frac{125}{33}, \frac{137}{31},$
$\frac{163}{44}, \frac{175}{47}, \frac{137}{29}, \frac{179}{50}, \frac{83}{13}, \frac{145}{44}, \frac{165}{49}, \frac{153}{41}, \frac{143}{63}, \frac{111}{26}, \frac{149}{34}, \frac{103}{18}, \frac{131}{30}, \frac{177}{49}, \frac{69}{11}, \frac{119}{36},$
$\frac{131}{39}, \frac{161}{45}, \frac{143}{31}, \frac{181}{50}, \frac{139}{30}, \frac{81}{14}, \frac{123}{26}, \frac{109}{25}, \frac{127}{34}, \frac{99}{23}, \frac{113}{30}, \frac{127}{27}, \frac{165}{46}, \frac{97}{17}, \frac{155}{46}, \frac{107}{29},$
$\frac{135}{41}, \frac{89}{20}, \frac{77}{12}, \frac{139}{39}, \frac{107}{19}, \frac{169}{50}, \frac{43}{5}, \frac{89}{28}, \frac{141}{41}, \frac{113}{21}, \frac{171}{50}, \frac{29}{3}, \frac{63}{20}, \frac{107}{31}, \frac{103}{19}, \frac{157}{46}, \frac{133}{29},$
$\frac{167}{46}, \frac{105}{22}, \frac{95}{18}, \frac{73}{14}, \frac{141}{43}, \frac{119}{37}, \frac{113}{35}, \frac{79}{12}, \frac{61}{8}, \frac{59}{8}, \frac{113}{31}, \frac{61}{9}, \frac{111}{34}, \frac{139}{41}, \frac{51}{7}, \frac{97}{30}, \frac{133}{39},$
$\frac{91}{20}, \frac{89}{16}, \frac{87}{16}, \frac{63}{13}, \frac{85}{24}, \frac{57}{11}, \frac{83}{24}, \frac{47}{15}, \frac{21}{2}, \frac{127}{29}, \frac{149}{40}, \frac{81}{19}, \frac{91}{19}, \frac{107}{25}, \frac{121}{32}, \frac{137}{37}, \frac{115}{26}, \frac{101}{23},$
$\frac{119}{26}, \frac{97}{22}, \frac{59}{9}, \frac{105}{23}, \frac{125}{27}, \frac{73}{11}, \frac{131}{40}, \frac{159}{47}, \frac{53}{7}, \frac{103}{32}, \frac{147}{43}, \frac{79}{14}, \frac{101}{18}, \frac{97}{18}, \frac{79}{15}, \frac{117}{34}, \frac{73}{23}, \frac{35}{4},$
$\left.\frac{39}{5}, \frac{77}{16}, \frac{113}{24}, \frac{75}{14}, \frac{69}{13}, \frac{103}{24}, \frac{67}{16}, \frac{33}{4}, \frac{61}{11}, \frac{95}{17}, \frac{149}{44}, \frac{75}{13}, \frac{121}{37}, \frac{41}{6}, \frac{67}{10}, \frac{109}{33}, \frac{63}{10}, \frac{37}{6}\right\}$

In[•]:= **Length[twoBridgeFractionNotation1]**

Out[•]= 362

In[∘]:= **Length[braidIndicesFromKnotInfo1]**

Out[∘]= 362

In[∘]:=
```
outputOfBraidIndexCalc =
 Table[braidIndexForFraction[twoBridgeFractionNotation1[[i]]], {i, 362}]
```

Out[∘]=
```
{2, 3, 2, 3, 4, 3, 3, 2, 4, 3, 4, 3, 4, 4, 5, 3, 5, 4, 4, 3, 4, 3, 4, 5, 4, 4, 2, 5, 3, 4, 5, 3,
 4, 5, 3, 4, 4, 5, 4, 5, 5, 4, 4, 5, 4, 5, 4, 4, 4, 4, 6, 3, 6, 5, 3, 4, 5, 4, 3, 5, 5, 4,
 6, 4, 4, 5, 3, 5, 4, 5, 4, 4, 4, 5, 4, 4, 4, 5, 5, 5, 5, 4, 5, 5, 6, 5, 5, 5, 4, 4, 5, 5,
 5, 5, 5, 6, 6, 6, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 5, 5, 5, 6, 5, 6, 5, 5, 6, 6, 6, 6, 4,
 4, 4, 4, 5, 4, 4, 4, 5, 4, 5, 4, 5, 5, 4, 5, 5, 6, 4, 5, 5, 4, 5, 5, 6, 6, 5, 5, 5, 6, 6,
 6, 3, 4, 4, 5, 4, 5, 5, 6, 4, 5, 4, 5, 5, 6, 6, 3, 4, 4, 5, 4, 5, 5, 6, 3, 4, 4, 4, 5, 5,
 6, 3, 4, 2, 6, 6, 7, 6, 5, 6, 5, 6, 5, 6, 6, 6, 5, 6, 6, 6, 6, 5, 6, 6, 6, 6, 6, 5, 6, 6,
 6, 6, 6, 6, 6, 6, 7, 7, 7, 5, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 5, 6, 4, 5, 5, 5, 5, 5, 4,
 5, 5, 5, 6, 4, 5, 5, 5, 5, 6, 6, 6, 6, 5, 4, 5, 5, 5, 5, 6, 6, 6, 6, 5, 6, 5, 6, 5, 6, 5,
 6, 6, 6, 7, 7, 4, 4, 5, 3, 4, 4, 4, 5, 3, 4, 4, 4, 5, 4, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 5,
 4, 5, 5, 4, 5, 5, 6, 6, 6, 5, 6, 5, 6, 6, 7, 4, 5, 4, 5, 5, 6, 5, 6, 5, 6, 7, 3, 4, 4, 4,
 5, 5, 3, 4, 4, 5, 6, 5, 4, 5, 5, 6, 4, 5, 5, 6, 5, 6, 6, 7, 3, 4, 5, 4, 5, 5, 6, 5, 6, 7}
```

In[∘]:= **error = outputOfBraidIndexCalc − braidIndicesFromKnotInfo1**

Out[∘]=
```
{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

In[∘]:= **outputOfBraidIndexCalc == braidIndicesFromKnotInfo1**

Out[∘]= True

In[∘]:=

In[∘]:= **braidIndexForFraction[10 / 3]**

Out[∘]= {3, 3}

In[∘]:= braidIndexForFraction[2000 / 333]

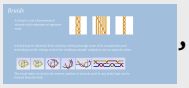
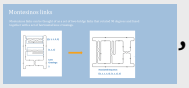
Out[∘]= {85, 88}

In[∘]:= **braidIndexForFraction[201 / 35]**

Out[∘]= 4

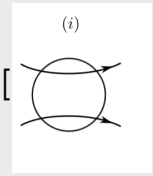
Interface

Instructions

```
instructions[] := CreateWindow[DocumentNotebook[{Column[
      {  ,  ,  ,  ,  ,
       Button[Back to main menu, NotebookClose[], ImageSize → Small]
      }]}, {Magnification → 4.5}
  ],
Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
```

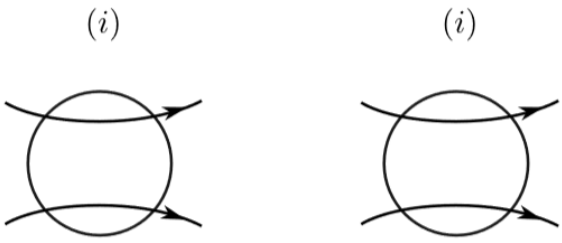
General

```
calculatorPageConway[braidIndexFunction_, signedVectorFunction_] :=
 CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator: Conway", White, Italic, 30]],
      Row[{Text[Style["Insert either a sequence of
             fractions or a sequence of continued fractions.",
          White, Italic, 18]], Button["Help", helpContinuedFractions[]]}],
      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "", braidIndex = "Answer Will Appear Here"},
        Column[{InputField[Dynamic[braidIndexUnspecifiedSequence]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}]*)
          Button["Submit", Dynamic[braidIndex = formatAsInputInterface[
              braidIndexUnspecifiedSequence, "braidIndex",
              braidIndexFunction]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           formatAsInputInterface[braidIndexUnspecifiedSequence, "additionalInfo",
            braidIndexFunction, signedVectorFunction], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

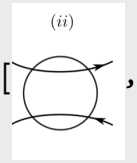
```
arcCaseM1[sequence_] :=

 GraphicsRow[Table[ 

, {i, Length[sequence] - 1}], ImageSize → Medium]
```

In[ ]:= **arcCaseM1[{{1}, {2, 3}, 0}]**

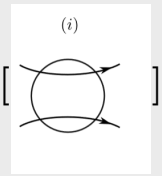
Out[ ]=



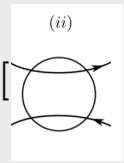

```
arcCaseM2[sequence_] :=
 GraphicsRow[Table[ , {i, Length[sequence] - 1}], ImageSize → Medium]
```



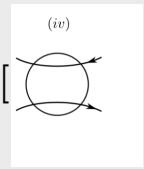
```
eightArcCaseToImage[arcCase_]:= Which[

arcCase == "i", Return[
```



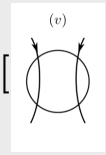

```
],

arcCase == "ii", Return[
```



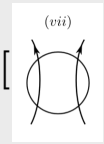

```
],

arcCase == "iv", Return[
```




```
],

arcCase == "v", Return[
```




```
],

arcCase == "vii", Return[
```




```
]

]
```

```
makeGraphicOfSeifertDecomp[fourDirections_, sequence_, mType_] := Module[
  {},
  Which[mType == "M1",
   Return[arcCaseM1[sequence]],
   mType == "M2",
   Return[arcCaseM2[sequence]],
   mType == "Unspecified",
   Return[GraphicsRow[
     eightArcCaseToImage /@ eightArcCases[fourDirections], ImageSize → Medium]]
  ]
 ]
```

```
helpContinuedFractions[] :=
MessageDialog["Input a list of continued fractions representing a Montesinos link."];
```

```
helpConway[] :=
MessageDialog[
   "Input Conway notation as a string. Each tangle sequence is separated by a
     semicolon, and elements in a sequence are separated by spaces.
     The string is enclosed in brackets. The number of lone crossings
     appears after a \" + \" appended to the last element of the last
     sequence.(Example of Conway notation: \"[2 1 2;4; 1 2 + 1]\")"];
```

In[ ]:= `homeScreen[]`

Out[ ]= `NotebookObject[ Untitled-7 ]`

```
homeScreen[]:=

CreateWindow[DocumentNotebook[{Style[ 

],Row[{Button[Text["Instructions"],instruction

Button["Calculator",NotebookClose[];chooseType[],ImageSize→200,Background→LightBlue]}]}],

Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
```

```
calculatorPageM[braidIndexFunction_, mType_] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator M", White, Italic, 30]],
      Row[{Text[Style["Insert either a sequence of
             fractions or a sequence of continued fractions.",
          White, Italic, 18]], Button["Help", helpContinuedFractions[]]}],

      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "", braidIndex = "Answer Will Appear Here"},
        Column[{
          InputField[Dynamic[braidIndexUnspecifiedSequence]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}]*)
          Button["Submit", Dynamic[braidIndex = braidIndexFunction[
              braidIndexUnspecifiedSequence]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           detailsM[braidIndexUnspecifiedSequence, mType], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

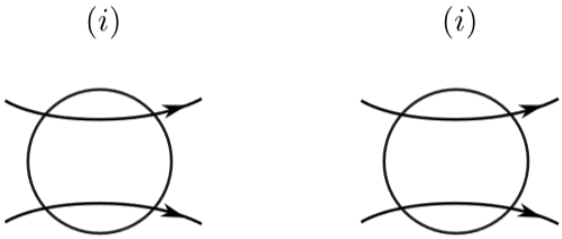
```
calculatorPageMConway[braidIndexFunction_, mType_] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator: Conway", White, Italic, 30]],
      Row[{Text[
         Style["Insert a string representing a Montesinos link in Conway notation.",
          White, Italic, 18]], Button["Help", helpConway[]]}],
      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "", braidIndex = "Answer Will Appear Here"},
        Column[{InputField[Dynamic[braidIndexUnspecifiedSequence]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}],*)
          Button["Submit", Dynamic[braidIndex = formatAsInputInterfaceM[
              braidIndexUnspecifiedSequence, "braidIndex",
              braidIndexFunction, mType]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           formatAsInputInterfaceM[braidIndexUnspecifiedSequence,
            "additionalInfo", braidIndexFunction, mType], Background → LightBlue]
         }]
       ]
      ], Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
(*determines braidIndex but also checks if the input is invalid*)
formatAsInputInterfaceM[sequence1_,
  specifiedPage_, braidIndexFunction_, mType_] := Module[
  {sequence = sequence1},
  (*If[(Head[sequence]===List),
    Return[
     MessageDialog["Not a valid input. If in continued fraction form, please use the
        corresponding calculator page."]]
   ];*)
  If[specifiedPage == "braidIndex",
   Return[braidIndexFunction[formatAsInput[sequence]]],
   Return[
    detailsM[formatAsInput[sequence1], mType]]
  ]
 ]
```

```
toSignedVectorWholeMSequence[mLink_,mVariety_]:=Module[
{twoIncomingOrientations,incomingOrienatations,tangleList,signedVector},
tangleList = correctSequence/@Take[mLink, Length[mLink]-1];
(*fourDirections*)
(*list of incoing orientations of each tangle this is within the indicated seifert class*)
(*incomingOrienatations = Table[Flatten[twoIncomingOrientations[[i]]], {i,tangleList}];*)
signedVector=Table[toSignedVectorM[tangleList[[i]],directionsM[tangleList[[i]],
mVariety],mVariety],

{i, Length[tangleList]}];
signedVector=AppendTo[signedVector,Last[mLink]];
Return[signedVector];

]
```

In[ ]:=
```
makeGraphicOfSeifertDecomp[
 fourDirectionsOfmLink[{{1, 1, 1, 2, 2}, {1, 2}, 2}], {{1, 1, 1, 2, 2}, {1, 2}, 2}]
```

Out[ ]=



```
detailsM[sequence1_,mType_]:=Module[
{link=sequence1,tangleList},
If[NumberQ[link[[1]]]== True && IntegerQ[link[[1]]]== False,
tangleList = Table[ContinuedFraction[1/link[[i]]],{i, Length[link]-1}];
 link = AppendTo[tangleList, Last[link]];
];
If[!(Head[link]===List),
CreateWindow[DocumentNotebook[{
Column[{Text[Style["Invalid Input!",White,24]],
Button["Back to Previous Screen",NotebookClose[]]},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
,

CreateWindow[DocumentNotebook[{
Column[{Text[Style["Seifert Decomposition:",White,Italic,24]],
makeGraphicOfSeifertDecomp[fourDirectionsOfmLink[link],link,mType],
Text[Style["Signed Vector:",White,Italic,24]],
Text[Style[toSignedVectorWholeMSequence[link,mType],20]],
Text[Style["Class:",White,Italic,24]],
Text[Style[mType,20]],
Text[Style["Number of Components:",White,Italic,24]],
Text[Style[numComponents[link],20]],
Button["Back to Previous Screen",NotebookClose[]]
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
];
]
```

M1, M2, Unspecified Interface

```
linkMultipleChoices[] := CreateWindow[DocumentNotebook[
{Column[{
Row[{Style[Text["Choose whether to specify all incoming orientations throughout
             the link or recieve all possible Braid Indices"],
          Italic, White, 24, TextAlignment → Center]}],
Button["Specify Orientations", NotebookClose[];
        linkConwayFracMulti[], ImageSize → 300, Background → LightBlue],
Button["View All Possible Orientations", NotebookClose[];
        directionChoicesFormat[], ImageSize → 300, Background → LightBlue]
}, Alignment → Center]
}, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
]
```

```
linkConwayFracMulti[]:= CreateWindow[DocumentNotebook[
{Column[{
Row[{Style[Text["Choose whether to specify all incoming orientations throughout the link or recie
Button["Fractional or Continued Fraction",NotebookClose[];calculatorPageMulti[],ImageSize→300,Bac
Button["Conway Notation",NotebookClose[];calculatorPageConwayMulti[],ImageSize→300,Background→Lig
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
directionChoicesFormat[]:= CreateWindow[DocumentNotebook[
{Column[{
Row[{Style[Text["Choose whether the link's notation is in conway or fractional form"],Italic,Whit
Button["Fractional or Continued Fraction",NotebookClose[];directionsPossibilities[],ImageSize→300
Button["Conway Notation",NotebookClose[];directionsPossibilitiesConway[],ImageSize→300,Background
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
directionsPossibilitiesConway[] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["All Possible Directions", White, Italic, 30]],
      Row[{Text[Style["Insert either a sequence of
            fractions or a sequence of continued fractions.",
          White, Italic, 18]], Button["Help", helpConway[]]}],
      Panel[DynamicModule[
        {sequence = "", listOfDirections = ""},
        Column[{InputField[Dynamic[sequence]],
          Button["Submit", Dynamic[listOfDirections = allDirectionsInterface[
              formatAsInput[sequence]]], Background → LightBlue],
          Style[Text[Dynamic[listOfDirections]]],
          Button["Evaluate the link at one of these directions",
           calculatorPageConwayMulti[], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
directionsPossibilities[] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["All Possible Directions", White, Italic, 30]],
      Row[{Text[Style["Insert either a sequence of
            fractions or a sequence of continued fractions.",
          White, Italic, 18]], Button["Help", helpContinuedFractions[]]}],
      Panel[DynamicModule[
        {sequence = "", listOfDirections = ""},
        Column[{InputField[Dynamic[sequence]],
          Button["Submit", Dynamic[listOfDirections = allDirectionsInterface[sequence]],
           Background → LightBlue],
          Style[Text[Dynamic[listOfDirections]]],
          Button["Evaluate the link at one of these directions",
           calculatorPageMulti[], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
allDirectionsInterface[sequence_]:= Module[
{tangleList,link=sequence},
If[NumberQ[sequence[[1]]]== True && IntegerQ[sequence[[1]]]== False,
tangleList = Table[ContinuedFraction[1/sequence[[i]]],{i, Length[sequence]-1}];
link = AppendTo[tangleList, Last[sequence]];
];
Return[fourDirectionsOfmLinkAllPossible[link]];

]
```

```
(*determines braidIndex but also checks if the input is invalid*)
formatAsInputInterfaceMulti[sequence1_, specifiedPage_, fourDirections_] := Module[
  {sequence = sequence1},
  If[(Head[sequence] === List),
   Return[
    MessageDialog["Not a valid input. If in continued fraction form, please use the
       corresponding calculator page."]]
  ];
  If[specifiedPage == "braidIndex",
   Return[braidIndexUnspecified[formatAsInput[sequence], fourDirections]],
   Return[detailsMulti[formatAsInput[sequence1], fourDirections]]]
 ]
```

```
calculatorPageConwayMulti[] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator: Conway", White, Italic, 30]],
      Row[{Text[
         Style["Insert a string representing a Montesinos link in Conway notation.",
          White, Italic, 18]], Button["Help", helpConway[]]}],
      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "",
         braidIndex = "Answer Will Appear Here", fourDirections = ""},
        Column[{InputField[Dynamic[braidIndexUnspecifiedSequence]],
          InputField[Dynamic[fourDirections]],
          Button["Submit", Dynamic[
            braidIndex = formatAsInputInterfaceMulti[braidIndexUnspecifiedSequence,
              "braidIndex", fourDirections]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           formatAsInputInterfaceMulti[braidIndexUnspecifiedSequence,
            "additionalInfo", fourDirections], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
calculatorPageMulti[] := CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator", White, Italic, 30]],
      Row[{Text[
         Style["Insert in the first input field a sequence of fractions or a sequence
            of continued fractions. In the second, insert a list of
            the four orientations entering at each tangle.",
          White, Italic, 18]], Button["Help", helpContinuedFractions[]]}],

      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "",
         fourDirections = "", braidIndex = "Answer Will Appear Here"},
        Column[{
          InputField[Dynamic[braidIndexUnspecifiedSequence]],
          InputField[Dynamic[fourDirections]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}]*)
          Button["Submit", Dynamic[braidIndex = braidIndexUnspecified[
              braidIndexUnspecifiedSequence,
              fourDirections]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           detailsMulti[braidIndexUnspecifiedSequence,
            fourDirections], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
detailsMulti[sequence1_,orientations_]:=Module[
{link=sequence1,tangleList},
If[NumberQ[link[[1]]]== True && IntegerQ[link[[1]]]== False,
tangleList = Table[ContinuedFraction[1/link[[i]]],{i, Length[link]-1}];
 link = AppendTo[tangleList, Last[link]];
];

If[!(Head[link]===List),
CreateWindow[DocumentNotebook[{
Column[{Text[Style["Invalid Input!",White,24]],
Button["Back to Previous Screen",NotebookClose[]]},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
,

CreateWindow[DocumentNotebook[{
Column[{Text[Style["Seifert Decomposition:",White,Italic,24]],
makeGraphicOfSeifertDecomp[orientations,link,"Unspecified"],
Text[Style["Signed Vector:",White,Italic,24]],
Text[Style[unspecifiedSignedVector[link,orientations],20]],
Text[Style["Class:",White,Italic,24]],
Text[Style[seifertClass[link,orientations],20]],
Text[Style["Number of Components:",White,Italic,24]],
Text[Style[numComponents[link],20]],
Button["Back to Previous Screen",NotebookClose[]]
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
];
]
```

```
chooseType[]:=
CreateWindow[DocumentNotebook[
{Column[{
Style[Text["Which type of link are you evaluating?"],Italic,White, 24,TextAlignment→Center],
Button["Unspecified Link",NotebookClose[];linkMultipleChoices[],ImageSize→300,Background→LightBlu
Button["Two-Bridge",NotebookClose[];conwayOrFractionsTwoBridge[],ImageSize→300,Background→LightB
Button["M1 Link",NotebookClose[];conwayOrFractionsM1[],ImageSize→300,Background→LightBlue],
Button["M2 Link",NotebookClose[];conwayOrFractionsM2[],ImageSize→300,Background→LightBlue],
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
(*determines braidIndex but also checks if the input is invalid*)
braidBInterface[sequence1_] := Module[
  {sequence = sequence1},
  If[! (Head[sequence] === List),
   Return["Not a valid input"]
  ];
  If[NumberQ[sequence[[1]]] == True && IntegerQ[sequence[[1]]] == False,
   Return[braidIndexBForFraction[sequence]]
  ];
  Return[braidIndexB[sequence]]
 ]
```

```
(*determines braidIndex but also checks if the input is invalid*)
braidM1Interface[sequence1_] := Module[
  {sequence = sequence1},
  If[! (Head[sequence] === List),
   Return["Not a valid input"]
  ];
  If[NumberQ[sequence[[1]]] == True && IntegerQ[sequence[[1]]] == False,
   Return[braidIndexMForFraction[sequence, "M1"]]
  ];
  Return[braidIndexM[sequence, "M1"]]
 ]
```

```
(*determines braidIndex but also checks if the input is invalid*)
braidM2Interface[sequence1_] := Module[
  {sequence = sequence1},
  If[! (Head[sequence] === List),
   Return["Not a valid input"]
  ];
  If[NumberQ[sequence[[1]]] == True && IntegerQ[sequence[[1]]] == False,
   Return[braidIndexMForFraction[sequence, "M2"]]
  ];
  Return[braidIndexM[sequence, "M2"]]
 ]
```

```
(*determines braidIndex but also checks if the input is invalid*)
braidIndexUnspecifiedInterface[sequence1_] := Module[
  {sequence = sequence1},
  If[! (Head[sequence] === List),
   Return["Not a valid input"]
  ];
  If[IntegerQ[sequence[[1]]] == False,
   Return[braidIndexUnspecified[sequence]]
  ];
  Return[braidIndexUnspecifiedForFrac[sequence]]
 ]
```

```
conwayOrFractionsUnspecified[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPage[braidIndexUnspecifi
Row[{Button["Conway Notation",NotebookClose[];calculatorPageConway[braidIndexUnspecifiedInterface
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
conwayOrFractionsM1[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPageM[braidM1Interface,"
Row[{Button["Conway Notation",NotebookClose[];calculatorPageMConway[braidM1Interface,"M1"],ImageS
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
conwayOrFractionsM2[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPageM[braidM2Interface,"
Row[{Button["Conway Notation",NotebookClose[];calculatorPageMConway[braidM2Interface,"M2"],ImageS
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
conwayOrFractionsUnspecified[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPage[braidIndexUnspecifi
Row[{Button["Conway Notation",NotebookClose[];calculatorPageConway[braidIndexUnspecifiedInterface
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
conwayOrFractionsB[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPage[braidM1Interface,to
Row[{Button["Conway Notation",NotebookClose[];calculatorPageConway[braidBInterface,toSignedVector
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

Two-bridge Interface

```
details2Bridge[sequence1_,braidIndex_]:=Module[
{link=sequence1,tangleList,signedVector},
If[NumberQ[link[[1]]]== True && IntegerQ[link[[1]]]== False,
link = Table[ContinuedFraction[1/link[[i]]],{i, Length[link]}];

];

If[!(Head[link]===List),
CreateWindow[DocumentNotebook[{
Column[{Text[Style["Invalid Input!",White,24]],
Button["Back to Previous Screen",NotebookClose[]]},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
,

CreateWindow[DocumentNotebook[{
Column[{
Text[Style["Signed Vector(s):",White,Italic,24]],
Text[Style[bothSignedVectors[link],20]],
Text[Style["Number of Components:",White,Italic,24]],
Text[Style[numComponents[{link,0}],20]],
Button["Back to Previous Screen",NotebookClose[]]
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]]
];
]
```

```
conwayOrFractionsTwoBridge[]:=
CreateWindow[DocumentNotebook[
{Column[{
Text[Style["Which format is your input in?",Italic,White, 24,TextAlignment→Center]],
Row[{Button["Fractional or Continued Fraction",NotebookClose[];calculatorPage2Bridge[braidTwoBrid
Row[{Button["Conway Notation",NotebookClose[];calculatorPageConway2Bridge[braidTwoBridgeInterface
},Alignment→Center]
},Background→RGBColor[0.63,0.75,0.86],TextAlignment→Center]
]
```

```
(*determines braidIndex but also checks if the input is invalid*)
braidTwoBridgeInterface[sequence1_] := Module[
  {sequence = sequence1},
  If[! (Head[sequence] === List),
   Return["Not a valid input"]
  ];
  If[NumberQ[sequence[[1]]] == True && IntegerQ[sequence[[1]]] == False,
   Return[braidIndexForFraction[sequence]]
  ];
  Return[braidIndexTwoBridge[sequence]]
 ]
```

```
calculatorPageConway2Bridge[braidIndexFunction_, signedVectorFunction_] :=
 CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator: Conway", White, Italic, 30]],
      Row[{Text[
         Style["Insert a string representing a two-bridge link in Conway notation.",
          White, Italic, 18]], Button["Help", helpConway[]]}],
      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "", braidIndex = "Answer Will Appear Here"},
        Column[{InputField[Dynamic[braidIndexUnspecifiedSequence]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}]*)
          Button["Submit", Dynamic[braidIndex = formatAsInputInterface2Bridge[
              braidIndexUnspecifiedSequence, "braidIndex", braidIndexFunction,
              signedVectorFunction]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           formatAsInputInterface2Bridge[
            braidIndexUnspecifiedSequence, "additionalInfo",
            braidIndexFunction, signedVectorFunction], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
calculatorPage2Bridge[braidIndexFunction_, signedVectorFunction_] :=
 CreateWindow[DocumentNotebook[{
    Column[{
      Text[Style["Braid Index Calculator", White, Italic, 30]],
      Row[{Text[Style["Insert a sequence of integers.", White, Italic, 18]],
        Button["Help", helpContinuedFractions[]]}],

      Panel[DynamicModule[
        {braidIndexUnspecifiedSequence = "", braidIndex = "Answer Will Appear Here"},
        Column[{
          InputField[Dynamic[braidIndexUnspecifiedSequence]],
          (*Graphics[{{GrayLevel[0.5],Rectangle[{0,0},{1,1}]},Text[Dynamic[
              braidIndexUnspecified[braidIndexUnspecifiedSequence]],{0.5,0.5}]}]*)
          Button["Submit", Dynamic[braidIndex = braidIndexFunction[
              braidIndexUnspecifiedSequence]], Background → LightBlue],
          Text["Braid Index: "],
          Graphics[{{GrayLevel[0.9], Rectangle[{0, 0}, {12, 4}]},
            Text[Dynamic[braidIndex], {6, 2}]}],
          Button["Additional Information",
           details2Bridge[
            braidIndexUnspecifiedSequence, braidIndex], Background → LightBlue]
         }]
       ]
      ]
     }, Alignment → Center]
   }, Background → RGBColor[0.63, 0.75, 0.86], TextAlignment → Center]
 ]
```

```
(*determines braidIndex but also checks if the input is invalid*)
formatAsInputInterface2Bridge[sequence1_, specifiedPage_,
  braidIndexFunction_, signedVectorFunction_] := Module[
  {sequence = sequence1},
  If[(Head[sequence] === List),
   Return[
    MessageDialog["Not a valid input. If in continued fraction form, please use the
       corresponding calculator page."]]
  ];
  If[specifiedPage == "braidIndex",
   Return[braidIndexFunction[formatAsInput[sequence][[1]]]],
   Return[details2Bridge[formatAsInput[sequence1][[1]],
     braidIndexFunction[formatAsInput[sequence][[1]]]]]
 ]
```

Tests for efficiency

```
In[•]:= braidIndexUnspecified[{{2, 2, 2, 2, 2}, {1, 2, 4, 5, 3},
          {2, 4, 2}, {3, 4}, {3}, {1, 2}, {4}, {10}, {12, 3, 4, 5, 2}, 1},
         fourDirectionsOfmLinkAllPossible[{{2, 2, 2, 2, 2}, {1, 2, 4, 5, 3},
            {2, 4, 2}, {3, 4}, {3}, {1, 2}, {4}, {10}, {12, 3, 4, 5, 2}, 1}][[1]]]
Out[•]= 30

In[•]:= braidIndexUnspecified[
         {{2, 2, 2, 2, 2}, {1, 2, 4, 5, 3}, {2, 4, 2}, {3, 4}, {3}, {1, 2}, {4}, {10}, {12, 3, 4, 5, 2},
          {12, 3, 4, 5, 2}, {12, 3, 4, 5, 2}, {12, 3, 4, 5, 2}, 1}, fourDirectionsOfmLinkAllPossible[
          {{2, 2, 2, 2, 2}, {1, 2, 4, 5, 3}, {2, 4, 2}, {3, 4}, {3}, {1, 2}, {4}, {10},
           {12, 3, 4, 5, 2}, {12, 3, 4, 5, 2}, {12, 3, 4, 5, 2}, {12, 3, 4, 5, 2}, 1}][[1]]]
Out[•]= 63

In[•]:= test1 = {{2, 3, 4}, {3, 4, 2}, {1, 3, 6}, {7}, {2, 3, 500 000}, {3, 3, 3}, {2, 2, 4, 5}, 0}
Out[•]= {{2, 3, 4}, {3, 4, 2}, {1, 3, 6}, {7}, {2, 3, 500 000}, {3, 3, 3}, {2, 2, 4, 5}, 0}

In[•]:= test2 = test1 * 1 000 000
Out[•]= {{2 000 000, 3 000 000, 4 000 000}, {3 000 000, 4 000 000, 2 000 000},
         {1 000 000, 3 000 000, 6 000 000}, {7 000 000}, {2 000 000, 3 000 000, 500 000 000 000},
         {3 000 000, 3 000 000, 3 000 000}, {2 000 000, 2 000 000, 4 000 000, 5 000 000}, 0}

In[•]:= fourDirectionsOfmLinkAllPossible[test1]
Out[•]= {{{L, R, L, R}, {L, R, R, L}, {R, L, L, R}, {L, R, R, L},
          {R, L, R, L}, {R, L, R, L}, {R, L, L, R}}, {{L, R, R, L}, {R, L, L, R},
          {L, R, R, L}, {R, L, L, R}, {L, R, R, L}, {R, L, R, L}, {R, L, L, R}}}

In[•]:= braidIndexUnspecified[test1, fourDirectionsOfmLinkAllPossible[test1][[1]]]
Out[•]= 22
```

This test case took less than 2 seconds to evaluate.

```
In[•]:= braidIndexUnspecified[test2, fourDirectionsOfmLinkAllPossible[test2][[1]]]
Out[•]= 250 031 000 001
```

This case failed to evaluate after 15 minutes of running.

```
In[•]:= Length[test3]
Out[•]= 29
```

```
In[•]:= test3 = {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
          {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
          {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
          {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 0}
Out[•]= {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
         {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
         {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
         {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 0}

In[•]:= braidIndexUnspecified[test3, fourDirectionsOfmLinkAllPossible[test3][[1]]]
Out[•]= $Aborted

In[•]:= homeScreen[]
Out[•]= NotebookObject[ Untitled-143 ]

In[•]:= test4 = {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
          {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 1}
Out[•]= {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
         {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 1}

In[•]:= braidIndexUnspecified[test4, fourDirectionsOfmLinkAllPossible[test4][[1]]]
Out[•]= 30

In[•]:= test5 = {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
          {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
          {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 1}
Out[•]= {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
         {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
         {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, 1}

In[•]:= braidIndexUnspecified[test5, fourDirectionsOfmLinkAllPossible[test5][[1]]]
Out[•]= $Aborted

In[•]:= test6 = {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2},
          {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 3, 5}, 1}
Out[•]= {{2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5},
         {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 2, 2}, {2, 3, 5}, {2, 3, 5}, 1}

In[•]:= Length[test6]
Out[•]= 16

In[•]:= braidIndexUnspecified[test6, fourDirectionsOfmLinkAllPossible[test6][[1]]]
Out[•]= 45
```