

Pensieve header: A program to enumerate w-knots.

```

SetDirectory["C:\\drorbn\\AcademicPensieve\\2015-03"]
C:\\drorbn\\AcademicPensieve\\2015-03

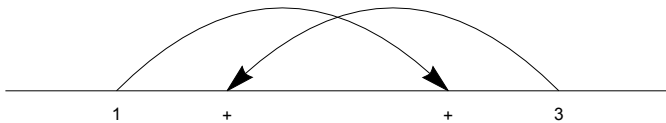
A_List \\ B_List := Complement[A, B];

Draw[w_wLDiag | w_wCDiag] := Module[{n, w1},
  n = Length[w];
  w1 = Abs /@ w;
  Graphics[{
    Line[{{0, 0}, {n + 1, 0}}],
    Table[
      {
        Arrow[BezierCurve[
          {{w1[[j]] - 0.5, 0}, {(w1[[j]] + j - 0.5) / 2, 0.5 Abs[j - w1[[j]] + 0.5]}, {j, 0}}]],
        Text[If[w[[j]] > 0, "+", "-"], {j, -0.1}],
        Text[w1[[j]], {w1[[j]] - 0.5, -0.1}]
      },
      {j, n}
    ]
  ]];

Draw[expr_] := expr /. w_wLDiag | w_wCDiag -> Draw[w]

Draw[wLDiag[3, 1]]

```



```

AllLinearDiagrams[n_] := Flatten@Table[
  wLDiag@@@Tuples[Range[k + 1] U (-Range[k + 1]), k],
  {k, 0, n}
]

```

AllLinearDiagrams[2]

```
{wLdiag[], wLdiag[-2], wLdiag[-1], wLdiag[1], wLdiag[2], wLdiag[-3, -3],
  wLdiag[-3, -2], wLdiag[-3, -1], wLdiag[-3, 1], wLdiag[-3, 2], wLdiag[-3, 3],
  wLdiag[-2, -3], wLdiag[-2, -2], wLdiag[-2, -1], wLdiag[-2, 1], wLdiag[-2, 2],
  wLdiag[-2, 3], wLdiag[-1, -3], wLdiag[-1, -2], wLdiag[-1, -1], wLdiag[-1, 1],
  wLdiag[-1, 2], wLdiag[-1, 3], wLdiag[1, -3], wLdiag[1, -2], wLdiag[1, -1],
  wLdiag[1, 1], wLdiag[1, 2], wLdiag[1, 3], wLdiag[2, -3], wLdiag[2, -2],
  wLdiag[2, -1], wLdiag[2, 1], wLdiag[2, 2], wLdiag[2, 3], wLdiag[3, -3],
  wLdiag[3, -2], wLdiag[3, -1], wLdiag[3, 1], wLdiag[3, 2], wLdiag[3, 3]}
```

```
wCDiag /: RotateLeft[w_wCDiag] := Module[{n},
  n = Length[w];
  wCDiag @@ (RotateLeft[List@@w] /. j_Integer => Which[
    j == 1, n,
    j == -1, -n,
    j > 1, j - 1,
    j < -1, j + 1
  ])
]
```

```
RotateLeft[wCDiag[-3, 1, 3, -2]]
```

```
wCDiag[4, 2, -1, -2]
```

```
RotateToMinimal[w_wCDiag] := RotateToMinimal[w] = Module[
  {bestw = w, rotatedw = RotateLeft[w]},
  While[rotatedw != w,
    bestw = First[Sort[{bestw, rotatedw}]];
    rotatedw = RotateLeft[rotatedw]
  ];
  bestw
];
```

```
wDiag[5, 2, -1, -2] // RotateToMinimal
```

```
wDiag[-2, 5, 2, -1]
```

```
wCDiag[w_wLDiag] := Module[{n},
  n = Length[w];
  RotateToMinimal[wCDiag@@w /. {n+1 -> 1, -n-1 -> -1}]
]
```

```
AllCircularDiagrams[n_] :=
```

```
AllCircularDiagrams[n] = Union[RotateToMinimal /@ Flatten@Table[
  wCDiag@@@Tuples[Range[k] ∪ (-Range[k]), k],
  {k, 0, n}
]]
```

AllCircularDiagrams [2]

```
{wCDiag[], wCDiag[-1], wCDiag[1], wCDiag[-2, -2],
  wCDiag[-2, -1], wCDiag[-2, 1], wCDiag[-2, 2], wCDiag[-1, -2],
  wCDiag[-1, 1], wCDiag[-1, 2], wCDiag[1, 1], wCDiag[1, 2], wCDiag[2, 1]}
```

```
RemoveR1 [w_wLDiag] := RemoveR1 [w] = Module [ {j, k = 0},
  Do [ If [ MemberQ [ {j, j + 1}, Abs [w[[j]]] ], k = j ], {j, Length [w]} ];
  If [ k == 0, w,
    Delete [w, k] /. j_Integer /; Abs [j] > k => Sign [j] (Abs [j] - 1)
  ]
]
```

```
RemoveR1 [wLDiag [-4, 1, 3, -4]]
```

```
wLDiag [-4, 1, 3]
```

RemoveR1 /@AllLinearDiagrams [2]

```
{wLDiag[], wLDiag[], wLDiag[], wLDiag[], wLDiag[], wLDiag[-2], wLDiag[-2],
  wLDiag[-3, -1], wLDiag[-3, 1], wLDiag[-2], wLDiag[-2], wLDiag[-2], wLDiag[-2],
  wLDiag[-1], wLDiag[1], wLDiag[-2], wLDiag[-2], wLDiag[-1], wLDiag[-1], wLDiag[-1],
  wLDiag[1], wLDiag[-1], wLDiag[-1], wLDiag[1], wLDiag[1], wLDiag[-1], wLDiag[1],
  wLDiag[1], wLDiag[1], wLDiag[2], wLDiag[2], wLDiag[-1], wLDiag[1], wLDiag[2],
  wLDiag[2], wLDiag[2], wLDiag[2], wLDiag[3, -1], wLDiag[3, 1], wLDiag[2], wLDiag[2]}
```

```
RemoveR1 [wCDiag []] = wCDiag [];
```

```
RemoveR1 [w_wCDiag] := RemoveR1 [w] = Module [ {n, j, k = 0},
  n = Length [w];
  Do [ If [ MemberQ [ {j, j + 1}, Abs [w[[j]]] ], k = j ], {j, n - 1} ];
  If [ k != 0,
    Delete [w, k] /. j_Integer /; Abs [j] > k => Sign [j] (Abs [j] - 1),
    (*else*) If [ ! MemberQ [ {1, n}, Abs [Last [w]] ], w,
      Drop [w, -1] /. {n -> 1, -n -> -1}
    ]
  ]
]
```

RemoveR1 /@AllCircularDiagrams [2]

```
{wCDiag[], wCDiag[], wCDiag[], wCDiag[-1], wCDiag[-1], wCDiag[1], wCDiag[1],
  wCDiag[-1], wCDiag[1], wCDiag[1], wCDiag[1], wCDiag[1], wCDiag[1]}
```

```
RemoveR1s [w_wLDiag | w_wCDiag] := RemoveR1s [w] = FixedPoint [RemoveR1, w]
```

RemoveR1s /@AllLinearDiagrams [2] // Union

```
{wLDiag[], wLDiag[-3, -1], wLDiag[-3, 1], wLDiag[3, -1], wLDiag[3, 1]}
```

```
RemoveR1s /@AllCircularDiagrams[4] // Union
```

```
{wCDiag[], wCDiag[-3, -1, -2], wCDiag[-3, -1, 2], wCDiag[-3, 1, -2],
  wCDiag[-3, 1, 2], wCDiag[3, 1, 2], wCDiag[-4, -4, -2, -3], wCDiag[-4, -4, -2, -2],
  wCDiag[-4, -4, -2, 2], wCDiag[-4, -4, -2, 3], wCDiag[-4, -4, -1, -3],
  wCDiag[-4, -4, -1, -2], wCDiag[-4, -4, -1, 2], wCDiag[-4, -4, -1, 3],
  wCDiag[-4, -4, 1, -3], wCDiag[-4, -4, 1, -2], wCDiag[-4, -4, 1, 2],
  wCDiag[-4, -4, 1, 3], wCDiag[-4, -4, 2, -3], wCDiag[-4, -4, 2, -2],
  wCDiag[-4, -4, 2, 2], wCDiag[-4, -4, 2, 3], wCDiag[-4, -1, -2, -3],
  wCDiag[-4, -1, -2, 2], wCDiag[-4, -1, -2, 3], wCDiag[-4, -1, 1, -2],
  wCDiag[-4, -1, 1, 2], wCDiag[-4, -1, 1, 3], wCDiag[-4, -1, 2, -2],
  wCDiag[-4, -1, 2, 2], wCDiag[-4, -1, 2, 3], wCDiag[-4, 1, -2, 2],
  wCDiag[-4, 1, -2, 3], wCDiag[-4, 1, -1, -2], wCDiag[-4, 1, -1, 2],
  wCDiag[-4, 1, -1, 3], wCDiag[-4, 1, 1, -2], wCDiag[-4, 1, 1, 2],
  wCDiag[-4, 1, 1, 3], wCDiag[-4, 1, 2, -2], wCDiag[-4, 1, 2, 2], wCDiag[-4, 1, 2, 3],
  wCDiag[-4, 4, -2, 2], wCDiag[-4, 4, -1, -2], wCDiag[-4, 4, -1, 2],
  wCDiag[-4, 4, -1, 3], wCDiag[-4, 4, 1, -2], wCDiag[-4, 4, 1, 2],
  wCDiag[-4, 4, 1, 3], wCDiag[-4, 4, 2, -2], wCDiag[-4, 4, 2, 2], wCDiag[-4, 4, 2, 3],
  wCDiag[-3, -4, -1, -2], wCDiag[-3, -4, -1, 2], wCDiag[-3, -4, -1, 3],
  wCDiag[-3, -4, 1, 2], wCDiag[-3, -4, 1, 3], wCDiag[-3, -4, 2, 2],
  wCDiag[-3, -4, 2, 3], wCDiag[-3, 1, -1, 2], wCDiag[-3, 1, -1, 3],
  wCDiag[-3, 1, 1, 2], wCDiag[-3, 1, 1, 3], wCDiag[-3, 1, 2, 2], wCDiag[-3, 1, 2, 3],
  wCDiag[-3, 4, -1, 2], wCDiag[-3, 4, 1, 2], wCDiag[-3, 4, 1, 3],
  wCDiag[-3, 4, 2, 2], wCDiag[-3, 4, 2, 3], wCDiag[3, 1, 1, 2], wCDiag[3, 1, 1, 3],
  wCDiag[3, 1, 2, 2], wCDiag[3, 1, 2, 3], wCDiag[3, 4, 1, 2], wCDiag[4, 1, 2, 3]}
```

```
RemoveR2[w_wLDiag] := RemoveR2[w] = Module[{j, k = 0},
  Do[If[w[[j]] + w[[j + 1]] == 0 && !MemberQ[Abs[List@w], j + 1], k = j],
  {j, Length[w] - 1}];
If[k == 0, w,
  Delete[w, {{k}, {k + 1}}] /. j_Integer /; Abs[j] > k => Sign[j] (Abs[j] - 2)
]
```

```
wLDiag[2, -2] // RemoveR2
```

```
wLDiag[2, -2]
```

RemoveR2 /@ **AllLinearDiagrams**[2]

```
{wLdiag[], wLdiag[-2], wLdiag[-1], wLdiag[1], wLdiag[2], wLdiag[-3, -3],
  wLdiag[-3, -2], wLdiag[-3, -1], wLdiag[-3, 1], wLdiag[-3, 2], wLdiag[],
  wLdiag[-2, -3], wLdiag[-2, -2], wLdiag[-2, -1], wLdiag[-2, 1], wLdiag[-2, 2],
  wLdiag[-2, 3], wLdiag[-1, -3], wLdiag[-1, -2], wLdiag[-1, -1], wLdiag[],
  wLdiag[-1, 2], wLdiag[-1, 3], wLdiag[1, -3], wLdiag[1, -2], wLdiag[],
  wLdiag[1, 1], wLdiag[1, 2], wLdiag[1, 3], wLdiag[2, -3], wLdiag[2, -2],
  wLdiag[2, -1], wLdiag[2, 1], wLdiag[2, 2], wLdiag[2, 3], wLdiag[],
  wLdiag[3, -2], wLdiag[3, -1], wLdiag[3, 1], wLdiag[3, 2], wLdiag[3, 3]}
```

AllLinearDiagrams[2]

```
{wLdiag[], wLdiag[-2], wLdiag[-1], wLdiag[1], wLdiag[2], wLdiag[-3, -3],
  wLdiag[-3, -2], wLdiag[-3, -1], wLdiag[-3, 1], wLdiag[-3, 2], wLdiag[-3, 3],
  wLdiag[-2, -3], wLdiag[-2, -2], wLdiag[-2, -1], wLdiag[-2, 1], wLdiag[-2, 2],
  wLdiag[-2, 3], wLdiag[-1, -3], wLdiag[-1, -2], wLdiag[-1, -1], wLdiag[-1, 1],
  wLdiag[-1, 2], wLdiag[-1, 3], wLdiag[1, -3], wLdiag[1, -2], wLdiag[1, -1],
  wLdiag[1, 1], wLdiag[1, 2], wLdiag[1, 3], wLdiag[2, -3], wLdiag[2, -2],
  wLdiag[2, -1], wLdiag[2, 1], wLdiag[2, 2], wLdiag[2, 3], wLdiag[3, -3],
  wLdiag[3, -2], wLdiag[3, -1], wLdiag[3, 1], wLdiag[3, 2], wLdiag[3, 3]}
```

Select[**AllLinearDiagrams**[2], (# != **RemoveR2**[#]) &]

```
{wLdiag[-3, 3], wLdiag[-1, 1], wLdiag[1, -1], wLdiag[3, -3]}
```

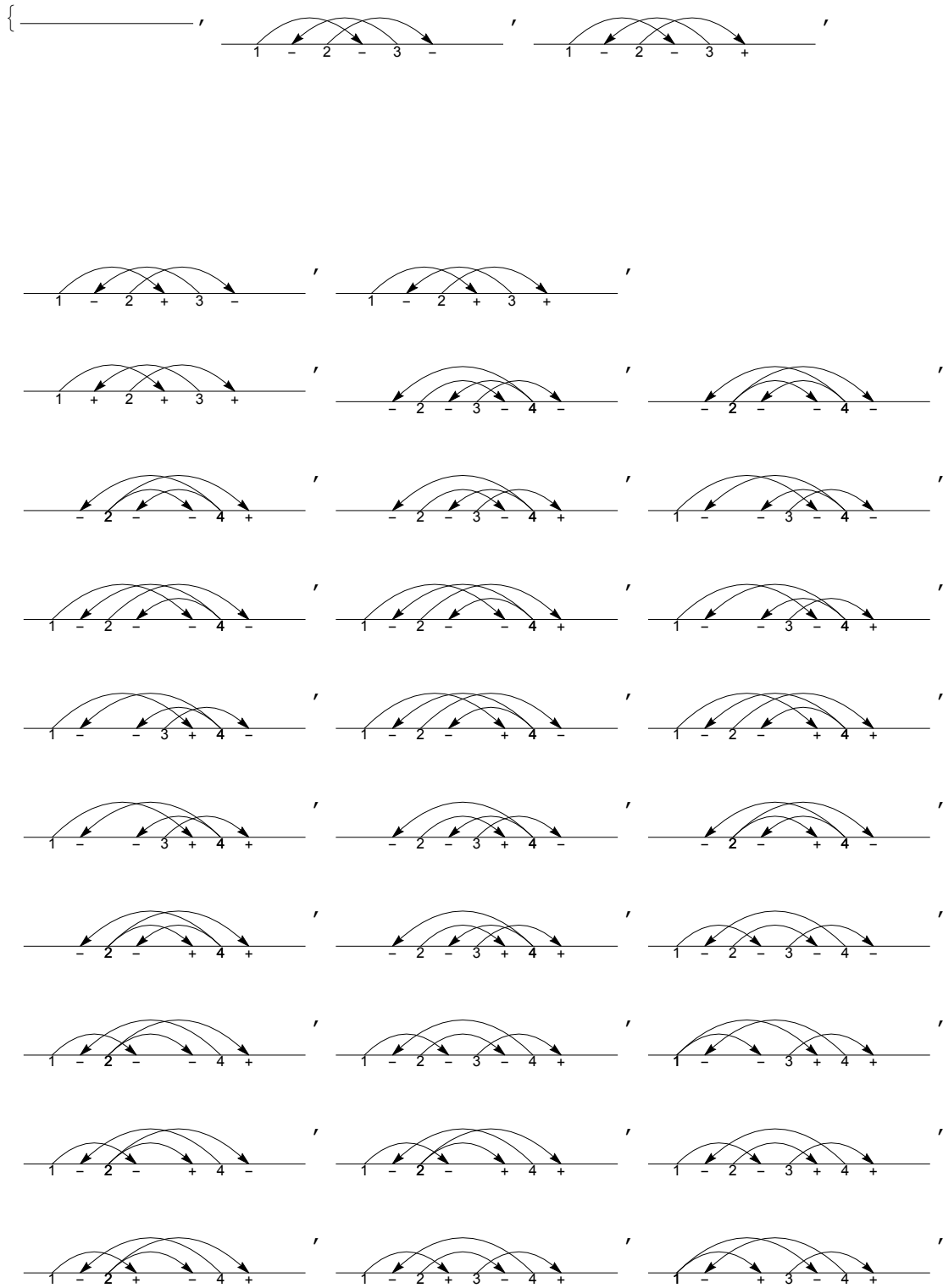
RemoveR2[**w_wCDiag**] /; **Length**[**w**] < 2 := **w**;

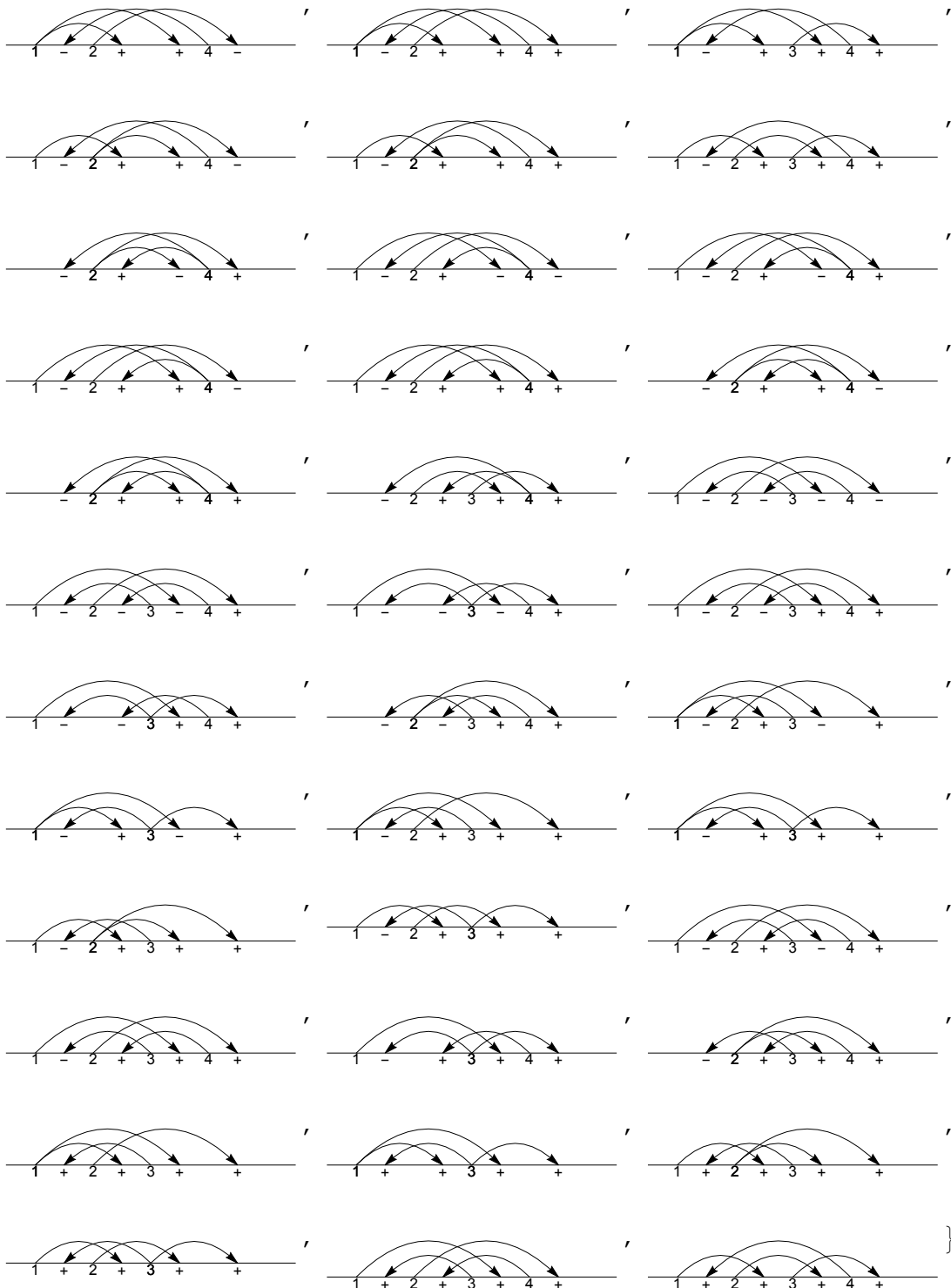
```
RemoveR2[w_wCDiag] := RemoveR2[w] = Module[{n, j, k = 0},
  n = Length[w];
  Do[If[w[[j]] + w[[j + 1]] == 0 && !MemberQ[Abs[List@w], j + 1], k = j], {j, n - 1}];
  If[k ≠ 0,
    Delete[w, {{k}, {k + 1}}] /.
      j_Integer /; Abs[j] > k => Sign[j] (Abs[j] - 2) /. {n - 1 → 1, 1 - n → -1},
    If[w[[1]] + w[[n]] == 0 && !MemberQ[Abs[List@w], 1],
      w[[2 ;; n - 1]] /. j_Integer => Sign[j] (Abs[j] - 1) /. {n - 1 → 1, 1 - n → -1},
      (*else*) w
    ]
  ]
]
```

RemoveR12s[**w_wLDiag** | **w_wCDiag**] :=

```
RemoveR12s[w] = FixedPoint[RemoveR2[RemoveR1[#]] &, w]
```

Union[**RemoveR12s** /@ **AllCircularDiagrams**[4]] // **Draw**

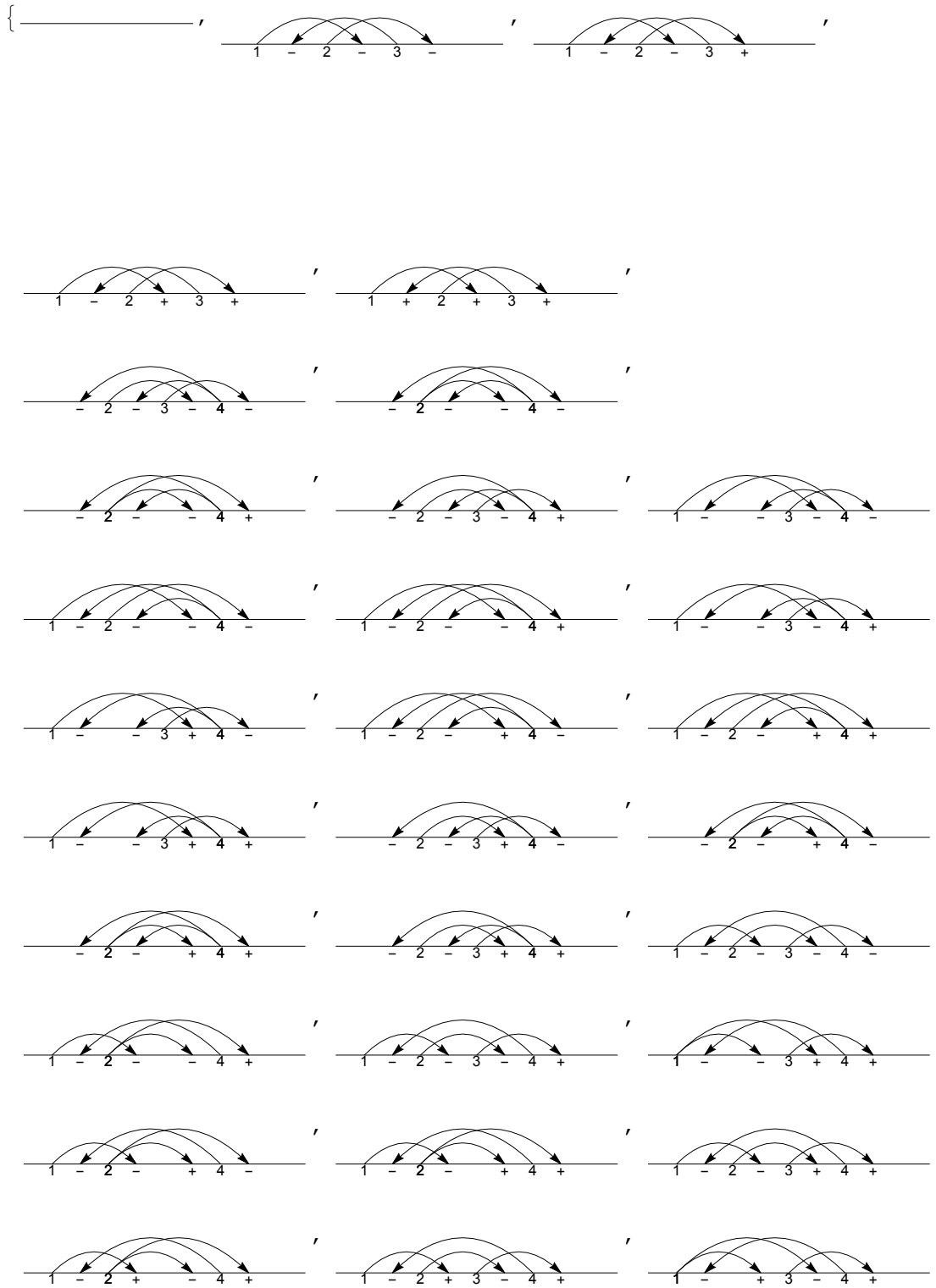


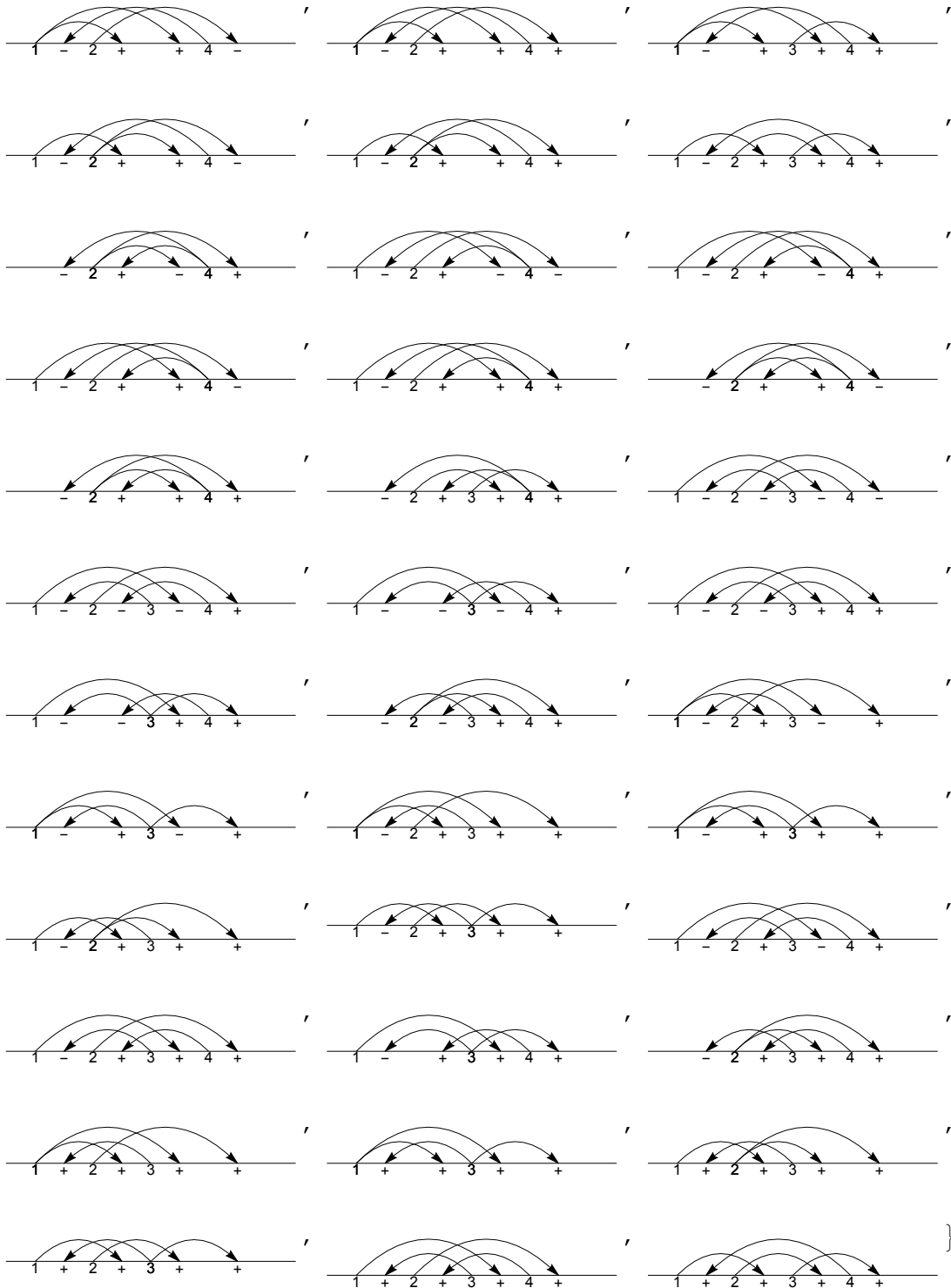


```

RF[w_wCDiag] := RF[w] = RotateToMinimal[RemoveR12s[w]];
RF[w_wLDiag] := RemoveR12s[w];

Union[RF /@ AllCircularDiagrams[4]] // Draw
    
```

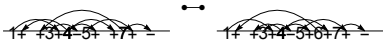




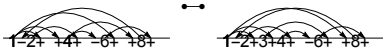
```
wLDiag /:
  Resolve[wLDiag[R3[top_, mid_, bot_, s1_, s2_, s3_], ts_]] := UndirectedEdge[
    RF@ReplacePart[wLDiag@ts, {bot + (1 - s3) / 2 → s2 s3 top,
      bot + (1 + s3) / 2 → s1 s3 (mid + 1), mid → s2 top}],
    RF@ReplacePart[wLDiag@ts, {bot + (1 - s3) / 2 → s1 s3 mid,
      bot + (1 + s3) / 2 → s2 s3 top, mid → s2 top}]
  ];
```

```
wCDiag /: Resolve[wCDiag[R3[top_, mid_, bot_, s1_, s2_, s3_], ts_]] :=
  (RF[wCDiag[#]]) & /@ Resolve[wLDiag[R3[top, mid, bot, s1, s2, s3], ts]]
```

```
Resolve@wLDiag[R3[4, 6, 1, 1, 1, 1], 0, 0, +1, -3, +4, 0, +5, -7] // Draw
```



```
Resolve@wCDiag[R3[4, 6, 1, 1, 1, 1], 0, 0, +1, -3, +4, 0, +5, -7] // Draw
```



```
AllLinearR3s[n_] /; n < 3 := {};
```

```
AllLinearR3s[n_] := Flatten@Table[
  Prepend[
    ReplacePart[wLDiag@@Table[0, {n}],
      Thread[Range[n] \ {bot, bot + 1, mid} → #]],
    R3[top, mid, bot, s1, s2, s3]
  ] & /@ Tuples[Range[-n - 1, n + 1] \ {-bot - 1, 0, bot + 1}, n - 3],
  {bot, Range[n - 1]},
  {mid, Range[n] \ {bot, bot + 1}}, {top, Range[n + 1] \ {bot + 1}},
  {s1, {-1, 1}}, {s2, {-1, 1}}, {s3, {-1, 1}}
];
```

```
AllCircularR3s[n_] /; n < 3 := {};
```

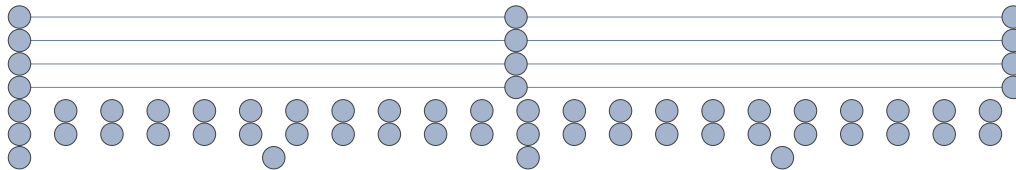
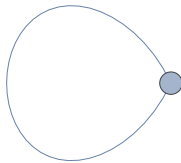
```
AllCircularR3s[n_] := Flatten@Table[
  Prepend[
    ReplacePart[wCDiag@@Table[0, {n}], Thread[Range[n] \ {1, 2, mid} → #]],
    R3[top, mid, 1, s1, s2, s3]
  ] & /@ Tuples[Range[-n, n] \ {-2, 0, 2}, n - 3],
  {mid, Range[n] \ {1, 2}}, {top, Range[n] \ {2}},
  {s1, {-1, 1}}, {s2, {-1, 1}}, {s3, {-1, 1}}
];
```

Union[RF /@AllLinearDiagrams[4]]

```
{wLDiag[], wLDiag[-3, -1], wLDiag[-3, 1], wLDiag[3, -1], wLDiag[3, 1],
wLDiag[-4, -4, -2], wLDiag[-4, -4, -1], ... 1163 ..., wLDiag[5, 5, 2, 3],
wLDiag[5, 5, 5, -3], wLDiag[5, 5, 5, -2], wLDiag[5, 5, 5, -1],
wLDiag[5, 5, 5, 1], wLDiag[5, 5, 5, 2], wLDiag[5, 5, 5, 3]}
```

large output | show less | show more | show all | set size limit...

```
n = 3;
vs = Union[RF /@AllLinearDiagrams[n]];
es = Union[Resolve /@AllLinearR3s[n]] /. Thread[vs -> Range[Length@vs]];
g = Graph[Range[Length@vs], es]
```



ConnectedComponents[g]

```
{{58, 4, 39}, {3, 28, 9}, {61, 5, 44}, {23, 2, 6}, {15}, {16},
{33}, {25}, {7}, {34}, {59}, {29}, {40}, {48}, {20}, {37}, {50}, {43},
{22}, {46}, {32}, {41}, {17}, {53}, {35}, {60}, {51}, {13}, {19},
{27}, {30}, {42}, {45}, {8}, {11}, {14}, {56}, {26}, {47}, {57}, {10},
{12}, {49}, {18}, {24}, {36}, {52}, {54}, {21}, {31}, {1}, {55}, {38}}
```

vs[[Flatten@ConnectedComponents[g]]]

```
{wLdiag[4, 4, -2], wLdiag[3, -1], wLdiag[3, -1, -1], wLdiag[-3, 1], wLdiag[-3, 1, 1],
wLdiag[-4, -4, 2], wLdiag[4, 4, 2], wLdiag[3, 1], wLdiag[3, 1, 1],
wLdiag[-3, -1, -1], wLdiag[-3, -1], wLdiag[-4, -4, -2], wLdiag[-4, 1, 2],
wLdiag[-4, 4, -2], wLdiag[-3, 4, 2], wLdiag[-3, -1, 2], wLdiag[-4, -4, -1],
wLdiag[3, -4, -2], wLdiag[4, 4, -1], wLdiag[-3, 1, 2], wLdiag[3, -1, 1],
wLdiag[3, 4, 1], wLdiag[-3, -4, 1], wLdiag[3, -4, 2], wLdiag[4, -4, -2],
wLdiag[3, 1, -1], wLdiag[-3, -1, -2], wLdiag[3, 4, -2], wLdiag[-3, 4, 1],
wLdiag[3, -1, 2], wLdiag[-4, 4, 2], wLdiag[4, -1, -1], wLdiag[3, -4, -1],
wLdiag[4, 4, 1], wLdiag[4, -4, 2], wLdiag[-4, 1, -2], wLdiag[-3, -4, -1],
wLdiag[-3, 1, -1], wLdiag[-3, 4, -2], wLdiag[3, 1, -2], wLdiag[3, 1, 2],
wLdiag[-4, -4, 1], wLdiag[-4, -1, -1], wLdiag[-4, 1, 1], wLdiag[4, 1, 1],
wLdiag[-3, 1, -2], wLdiag[3, 4, -1], wLdiag[4, 1, 2], wLdiag[-4, -1, -2],
wLdiag[-4, -1, 2], wLdiag[3, 4, 2], wLdiag[-3, -4, -2], wLdiag[-3, -1, 1],
wLdiag[3, -4, 1], wLdiag[4, -1, -2], wLdiag[4, -1, 2], wLdiag[-3, -4, 2],
wLdiag[-3, 4, -1], wLdiag[], wLdiag[4, 1, -2], wLdiag[3, -1, -2]}
```

wCDiag /@ vs[[Flatten@ConnectedComponents[g]]] // Union

```
{wCDiag[], wCDiag[-2, -2], wCDiag[-2, 2], wCDiag[-1, 1],
wCDiag[1, 1], wCDiag[-3, -3, -3], wCDiag[-3, -3, -2], wCDiag[-3, -3, 2],
wCDiag[-3, -3, 3], wCDiag[-3, -2, -2], wCDiag[-3, -2, 2],
wCDiag[-3, -1, -2], wCDiag[-3, -1, 1], wCDiag[-3, -1, 2], wCDiag[-3, -1, 3],
wCDiag[-3, 1, -3], wCDiag[-3, 1, -1], wCDiag[-3, 1, 1], wCDiag[-3, 1, 2],
wCDiag[-3, 1, 3], wCDiag[-3, 2, 2], wCDiag[-3, 3, -3], wCDiag[-3, 3, 2],
wCDiag[-2, 1, 2], wCDiag[-2, 2, 2], wCDiag[-1, 1, 1], wCDiag[-1, 1, 2],
wCDiag[1, 1, 1], wCDiag[1, 1, 2], wCDiag[2, 1, 2], wCDiag[3, 1, 2]}
```

Timing[n = 5;

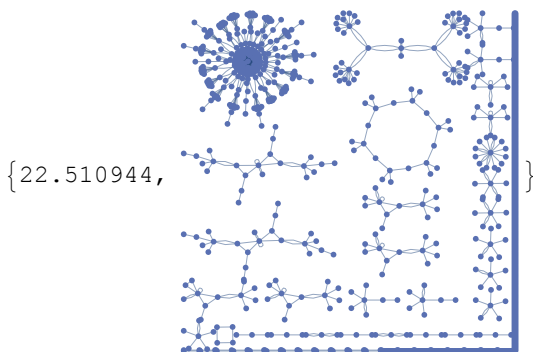
vs = Union[RF /@ AllCircularDiagrams[n]];

es = Union[Resolve /@ AllCircularR3s[n] /. w_wLdiag => RF[wCDiag[w]]] /.

Thread[vs -> Range[Length@vs]];

g = Graph[Range[Length@vs], es]

]



```
vs[[Flatten@ConnectedComponents[g]]] // Length
```

1387

```
Select[vs[[Flatten@ConnectedComponents[g]]], Length[#] == 3 &] // Length
```

4

```
Timing[n = 6;
```

```
vs = Union[RF /@ AllCircularDiagrams[n]];

```

```
es = Union[Resolve /@ AllCircularR3s[n]] /. Thread[vs -> Range[Length@vs]];

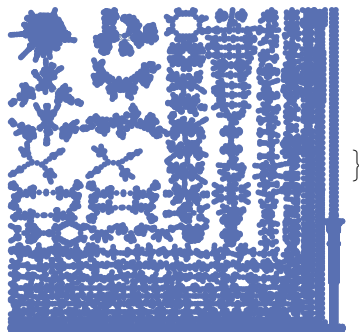
```

```
g = Graph[Range[Length@vs], es]

```

]

{1106.936296,



}

```
Select[vs[[Flatten@ConnectedComponents[g]]], Length[#] == 4 &] // Length
```

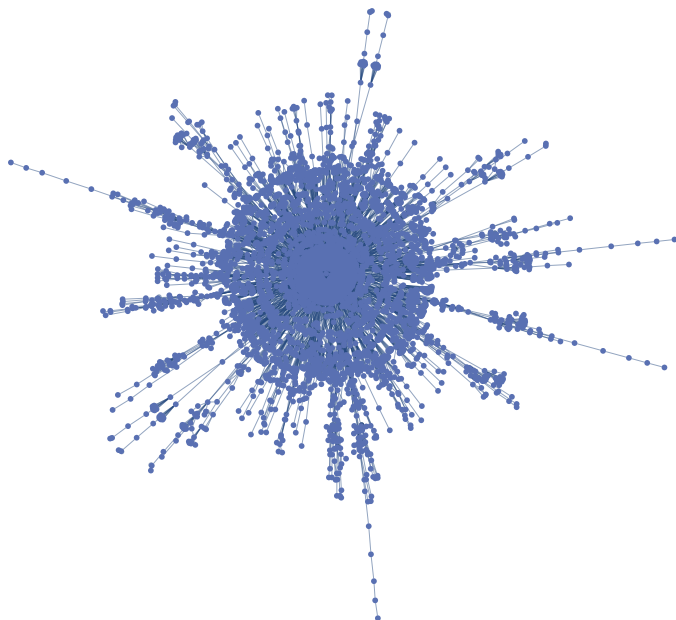
62

```
cc = ConnectedComponents[g];

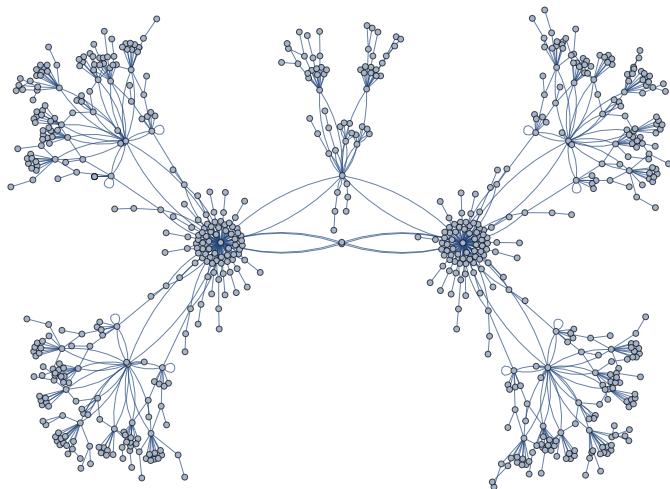
```

```
Subgraph[g, cc[[1]]]

```



Subgraph[g, cc[[2]]]



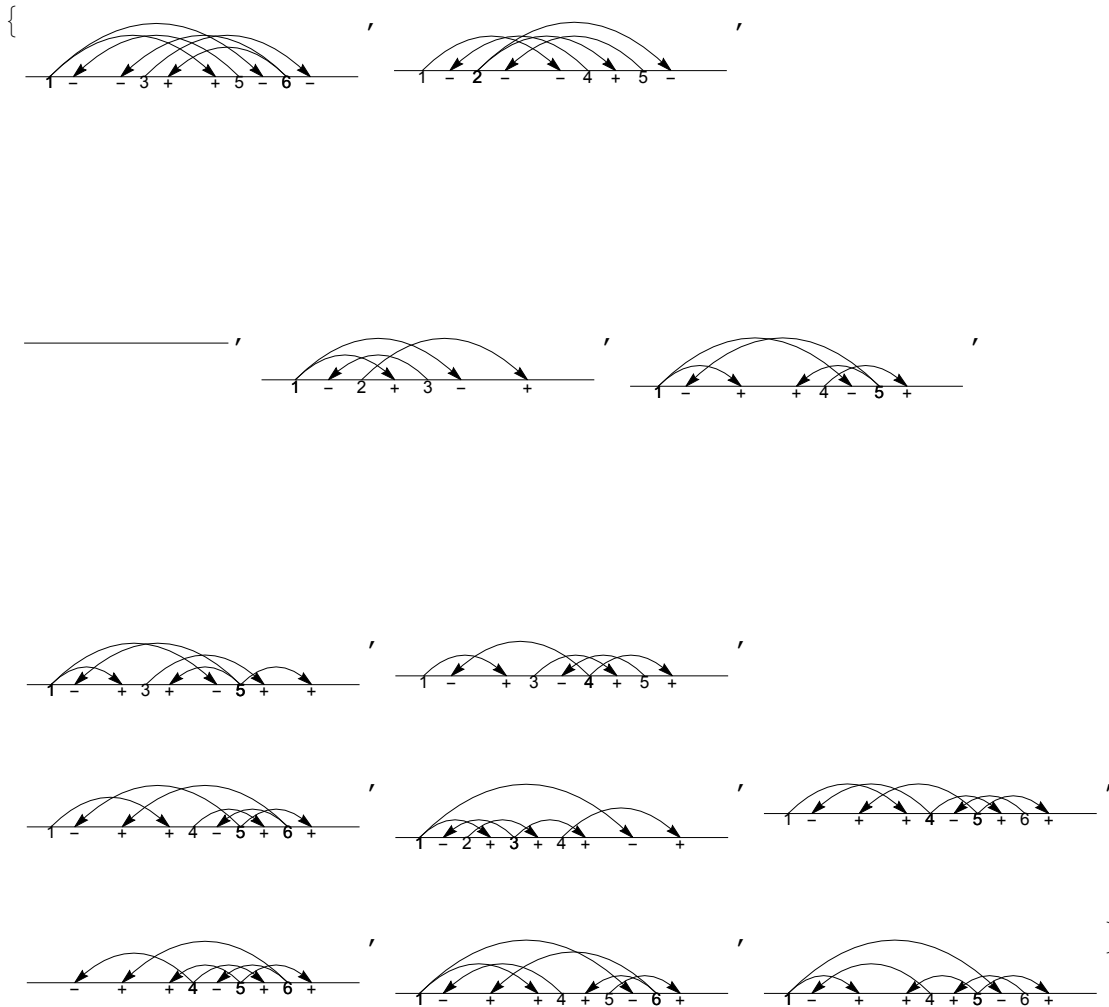
Subgraph[g, cc[[1000]]]



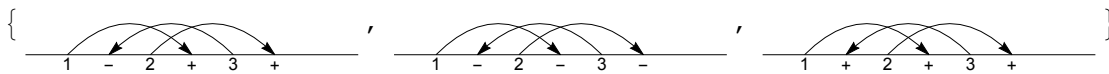
Length@cc[[1]]

4649

```
vs[[#]] & /@ FindShortestPath[g, cc[[1, 1]], cc[[1, 4649]] // Draw
```



```
Select[Table[
  First@MinimalBy[vs[[#]] & /@ c, Length],
  {c, cc}
], Length[#] == 3 &] // Draw
```



```
Select[Table[
  First@MinimalBy[vs[[#]] & /@ c, Length],
  {c, cc}
], Length[#] == 3 &]
{wCDiag[-3, 1, 2], wCDiag[-3, -1, -2], wCDiag[3, 1, 2]}
```

```

Position[vs, wCDiag[3, -1, -2] // RF]
{{3}}

Position[cc, 3]
{{2, 74}}

Position[vs[[#]] & /@ cc[[2]], wCDiag[-3, 1, 2]]
{{2}}

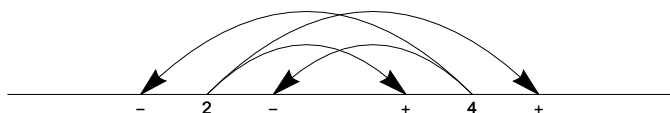
cc[[2, 2]]
4

vs[[4]]
wCDiag[-3, 1, 2]

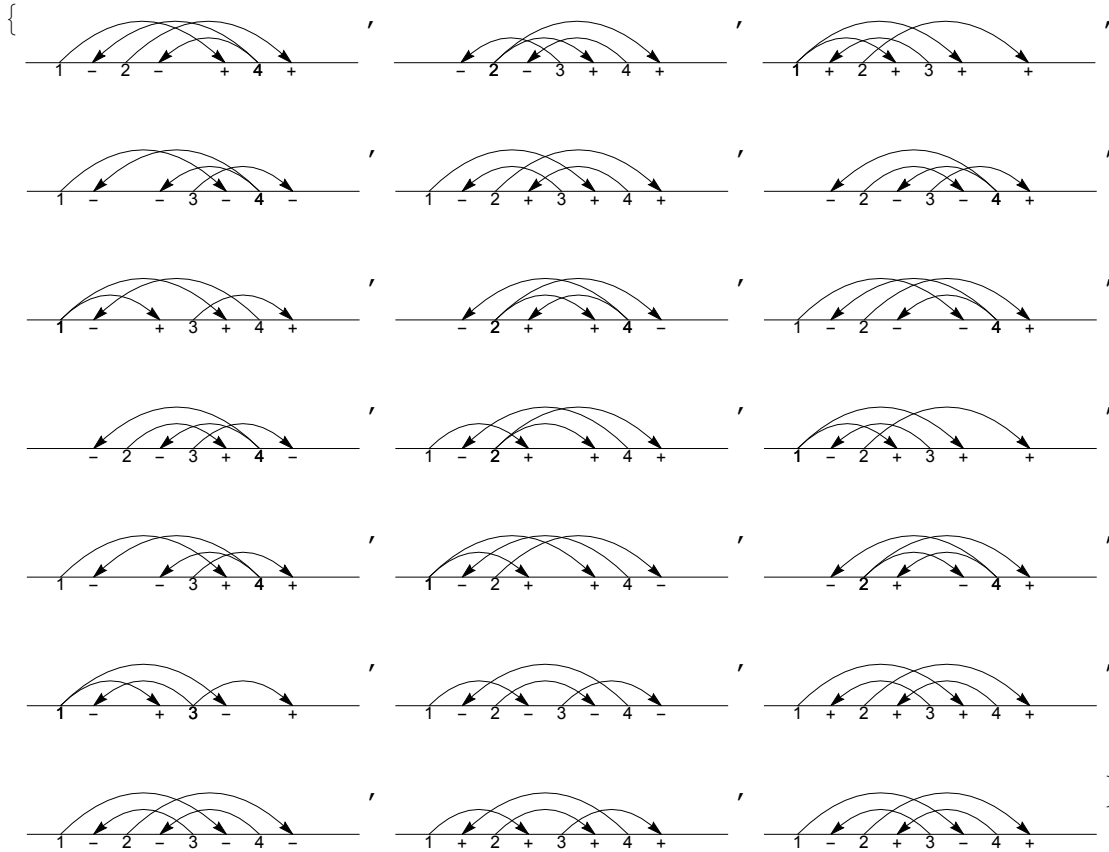
FindShortestPath[g, 3, 4]
{3, 20, 4}

vs[[20]] // Draw

```




```
Select[Table[
  First@MinimalBy[vs[##] & /@ c, Length],
  {c, cc}
], Length[##] == 4 &] // Draw
```



```
wK6[4] = Select[Table[
  First@MinimalBy[vs[##] & /@ c, Length],
  {c, cc}
], Length[##] == 4 &]
```

- {wCDiag[-4, -4, 1, 2], wCDiag[-3, -4, 2, 2], wCDiag[3, 1, 1, 2],
- wCDiag[-4, -4, -1, -3], wCDiag[-3, 4, 1, 2], wCDiag[-4, -4, -2, 3],
- wCDiag[-4, 1, 1, 3], wCDiag[-4, 4, 2, -2], wCDiag[-4, -4, -1, 2],
- wCDiag[-4, -4, 2, -3], wCDiag[-4, 1, 2, 2], wCDiag[-3, 1, 1, 2],
- wCDiag[-4, -4, 1, 3], wCDiag[-4, 1, 1, -2], wCDiag[-4, 4, -2, 2],
- wCDiag[-3, 1, -1, 3], wCDiag[-4, -1, -2, -3], wCDiag[3, 4, 1, 2],
- wCDiag[-3, -4, -1, -2], wCDiag[4, 1, 2, 3], wCDiag[-3, 4, -1, 2]}

```

wK7[4] = {wCDiag[-4, -4, 1, 2], wCDiag[-4, -4, -1, 3], wCDiag[3, 1, 1, 2],
  wCDiag[-4, -4, -2, -3], wCDiag[-4, 1, 2, 2], wCDiag[-4, -4, 2, -3],
  wCDiag[-3, 4, 1, 2], wCDiag[-4, 1, 1, -2], wCDiag[-4, -4, 1, 3],
  wCDiag[-4, -4, 1, -2], wCDiag[-4, 1, 1, 3], wCDiag[-4, 4, 2, -2],
  wCDiag[-3, 1, -1, 3], wCDiag[-4, 4, -2, 2], wCDiag[-4, 1, -2, 3], wCDiag[4, 1, 2, 3],
  wCDiag[-4, -1, -2, -3], wCDiag[-3, -4, -1, -2], wCDiag[3, 4, 1, 2]}
{wCDiag[-4, -4, 1, 2], wCDiag[-4, -4, -1, 3], wCDiag[3, 1, 1, 2],
wCDiag[-4, -4, -2, -3], wCDiag[-4, 1, 2, 2], wCDiag[-4, -4, 2, -3],
wCDiag[-3, 4, 1, 2], wCDiag[-4, 1, 1, -2], wCDiag[-4, -4, 1, 3],
wCDiag[-4, -4, 1, -2], wCDiag[-4, 1, 1, 3], wCDiag[-4, 4, 2, -2],
wCDiag[-3, 1, -1, 3], wCDiag[-4, 4, -2, 2], wCDiag[-4, 1, -2, 3], wCDiag[4, 1, 2, 3],
wCDiag[-4, -1, -2, -3], wCDiag[-3, -4, -1, -2], wCDiag[3, 4, 1, 2]}

```