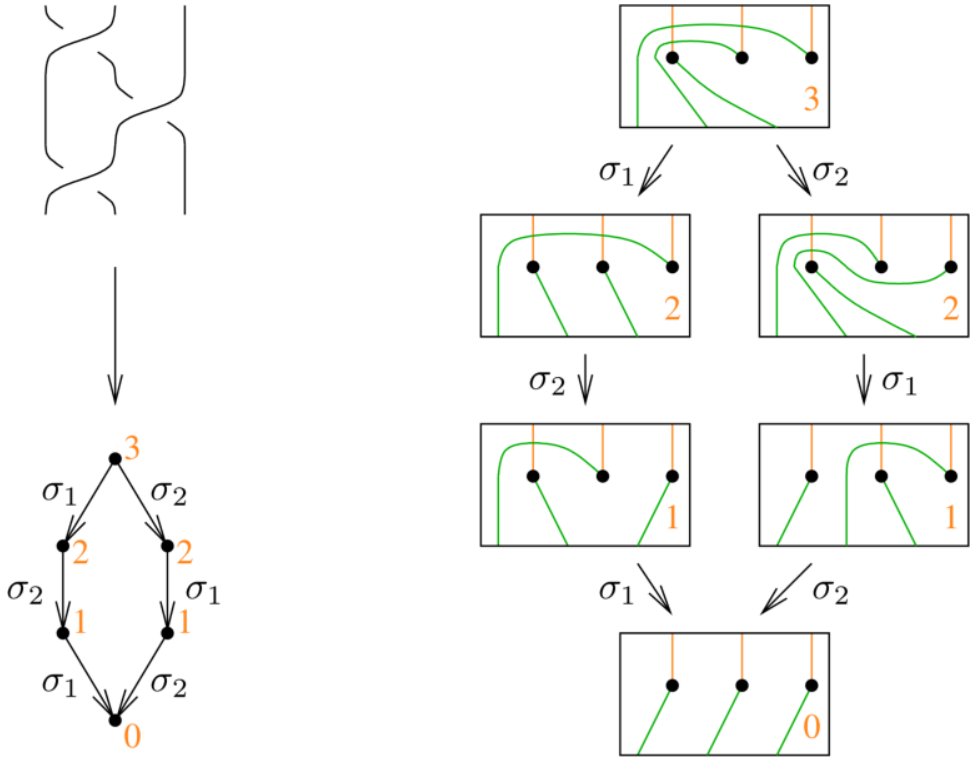
Pensieve header: Extraction graphs for braids picked by Bert Wiest.

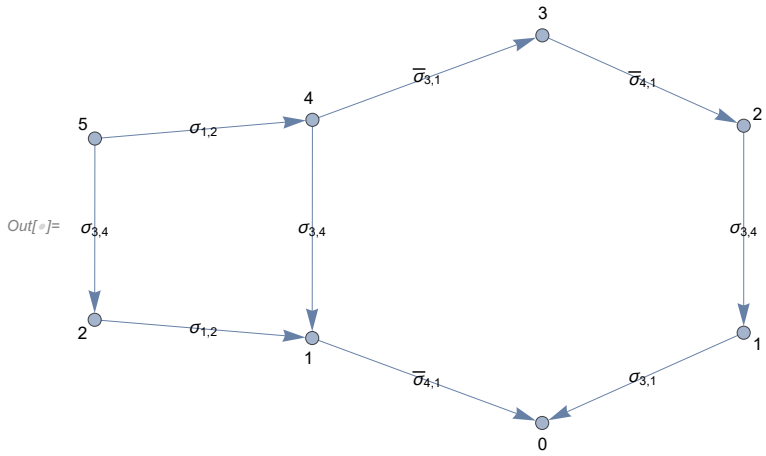


Braids for Wiest

$(1\bar{3}2)^k$

In[•]:= `BR[4, {1, 3, -2}] // ExtractionGraph`

Out[•]=

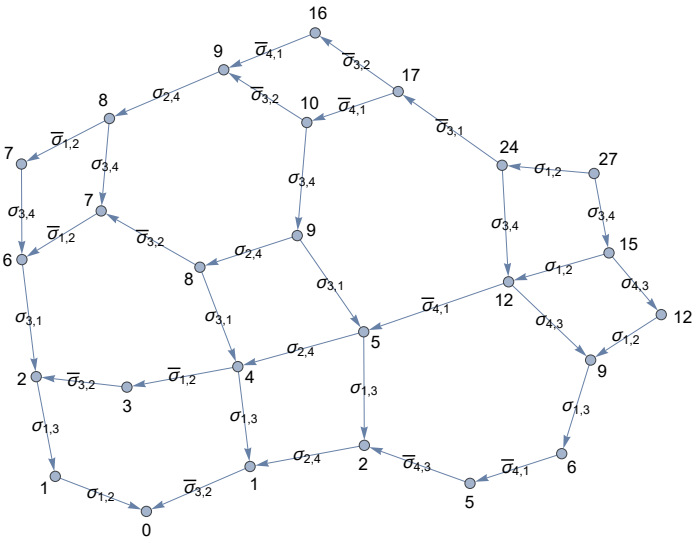


In[•]:= `BR[4, {1, 3, -2}] // ExtractionGraph // Bend`

Out[•]= `{3, 5, 1.66667}`

In[•]:= **BR[4, {1, 3, -2, 1, 3, -2}] // ExtractionGraph**

Out[•]=

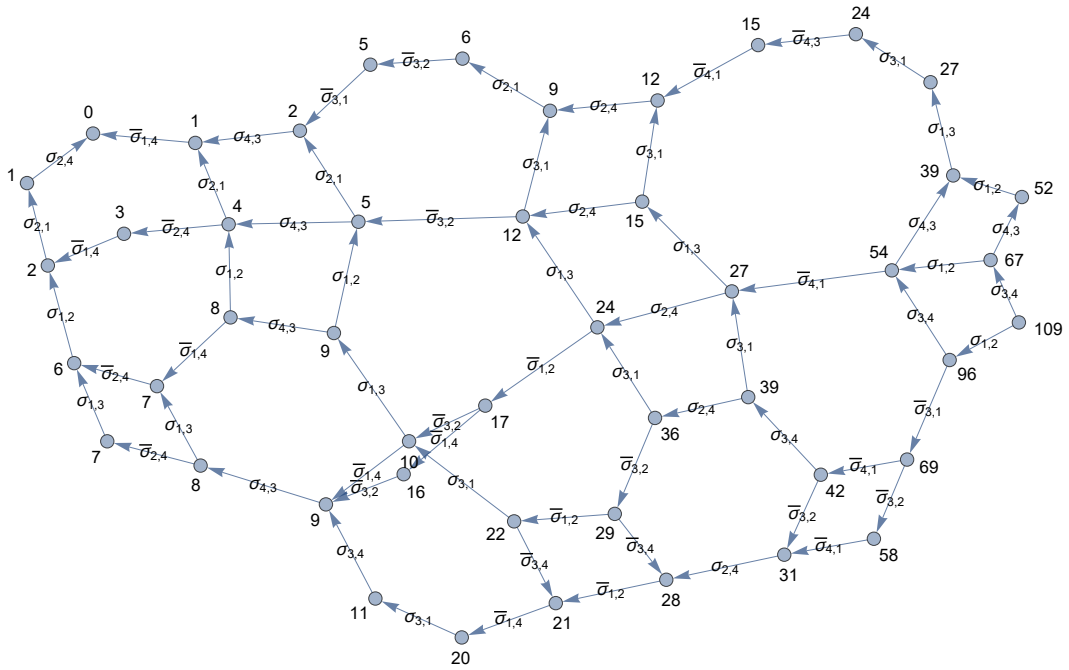


In[•]:= **BR[4, {1, 3, -2, 1, 3, -2}] // ExtractionGraph // Bend**

Out[•]= {6, 10, 1.66667}

In[•]:= **BR[4, {1, 3, -2, 1, 3, -2, 1, 3, -2}] // ExtractionGraph**

Out[•]=

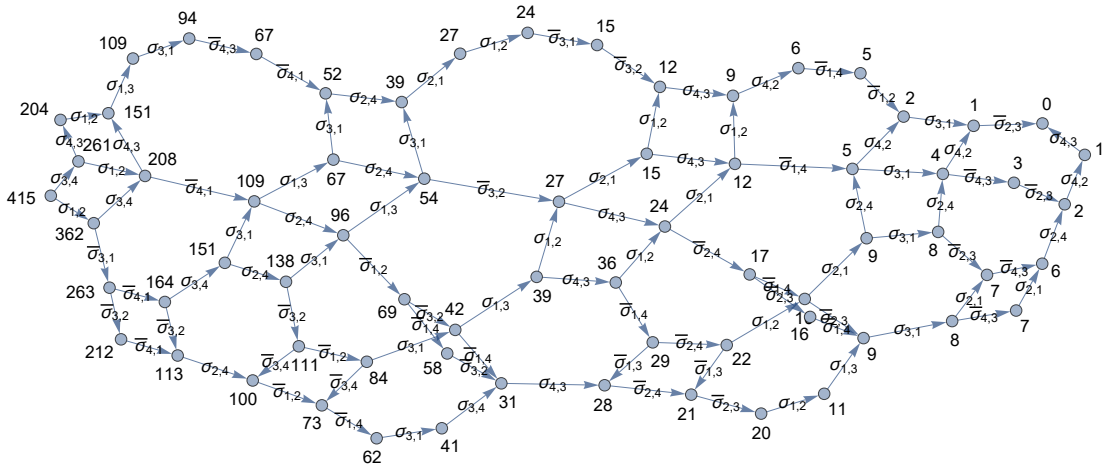


In[•]:= **BR[4, {1, 3, -2, 1, 3, -2, 1, 3, -2}] // ExtractionGraph // Bend**

Out[•]= {9, 17, 1.88889}

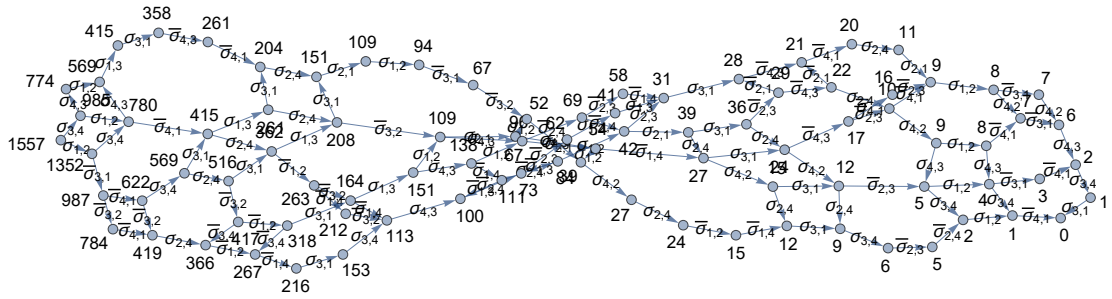
In[•]:=
```
BR[4, {1, 3, -2, 1, 3, -2, 1, 3, -2, 1, 3, -2}] // ExtractionGraph
```

Out[•]=



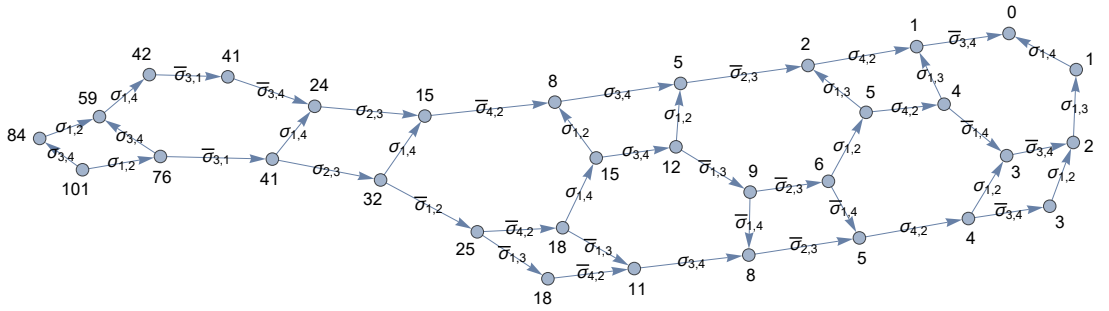
In[•]:=
```
BR[4, {1, 3, -2, 1, 3, -2, 1, 3, -2, 1, 3, -2, 1, 3, -2}] // ExtractionGraph
```

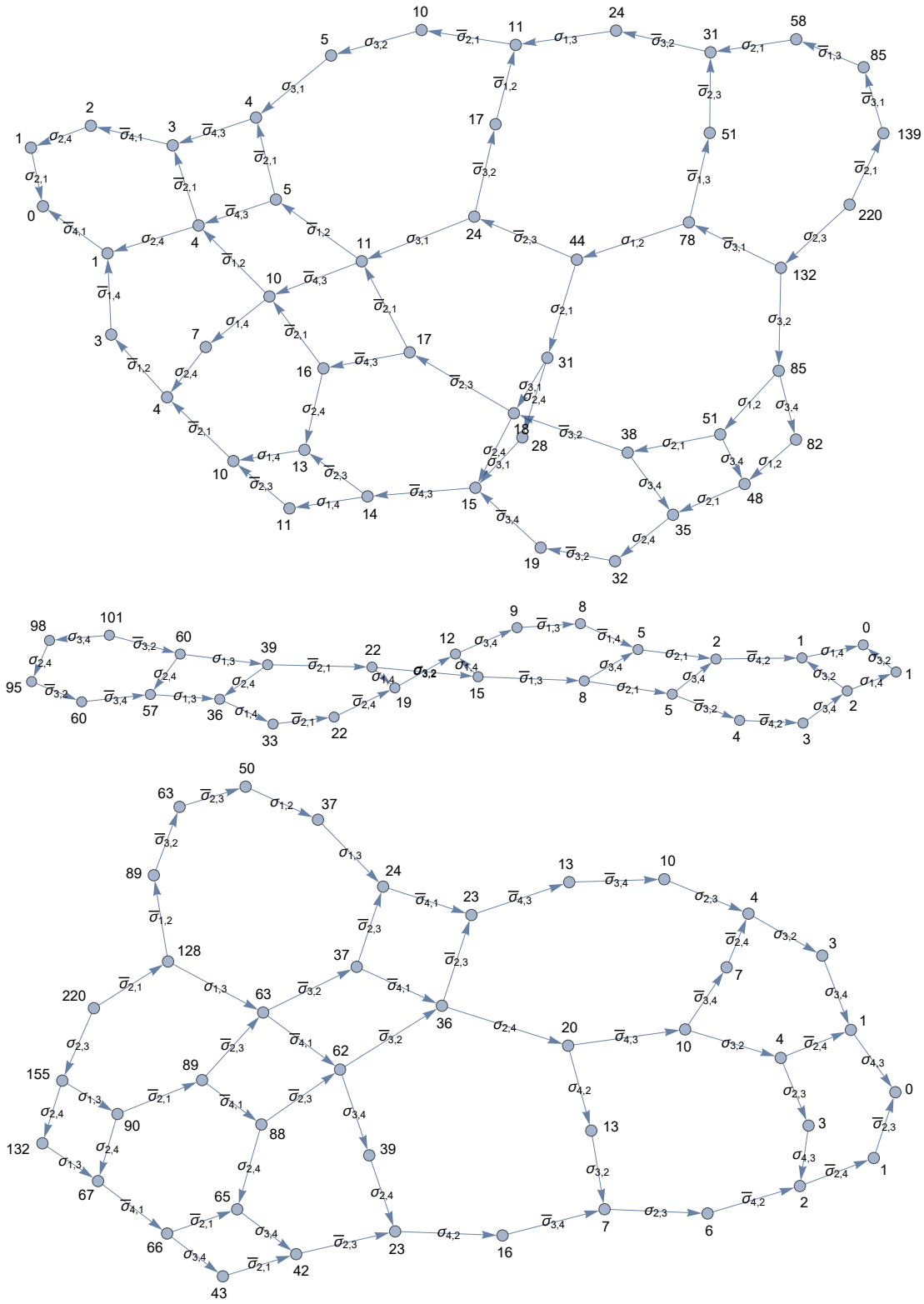
Out[•]=



$\beta = 1\,\overline{2}\,3\,(1\,\overline{2})^k$, its inverse, transpose, and inverse-transpose

In[•]:=
```
β = {1, -2, 3, 1, -2, 1, -2, 1, -2};
Print[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β};
```

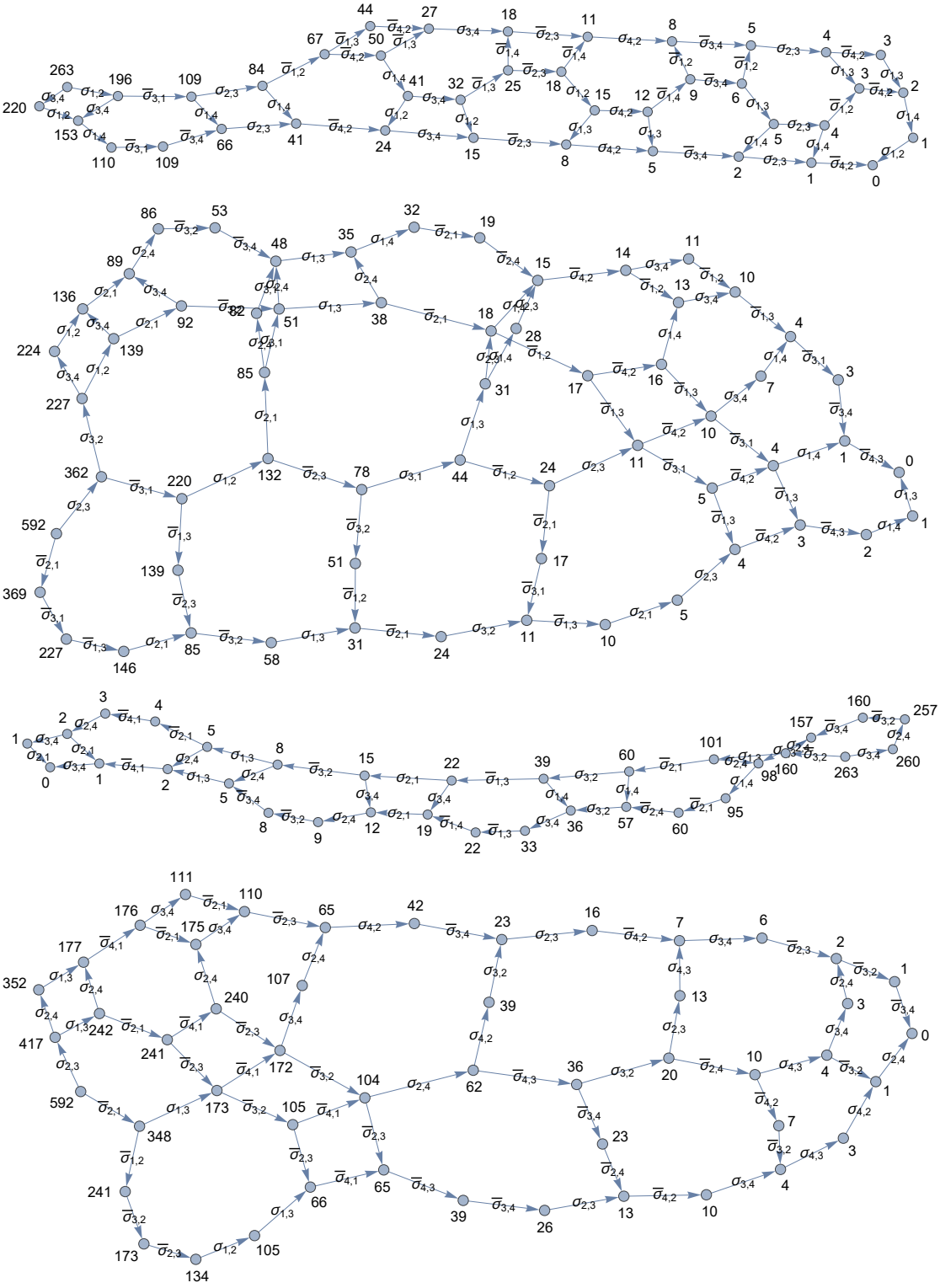




In[•]:=
```
β = {1, -2, 3, 1, -2, 1, -2, 1, -2};
Bend[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β}
```

Out[•]=
```
{{9, 15, 1.66667}, {9, 15, 1.66667}, {9, 15, 1.66667}, {9, 13, 1.44444}}
```

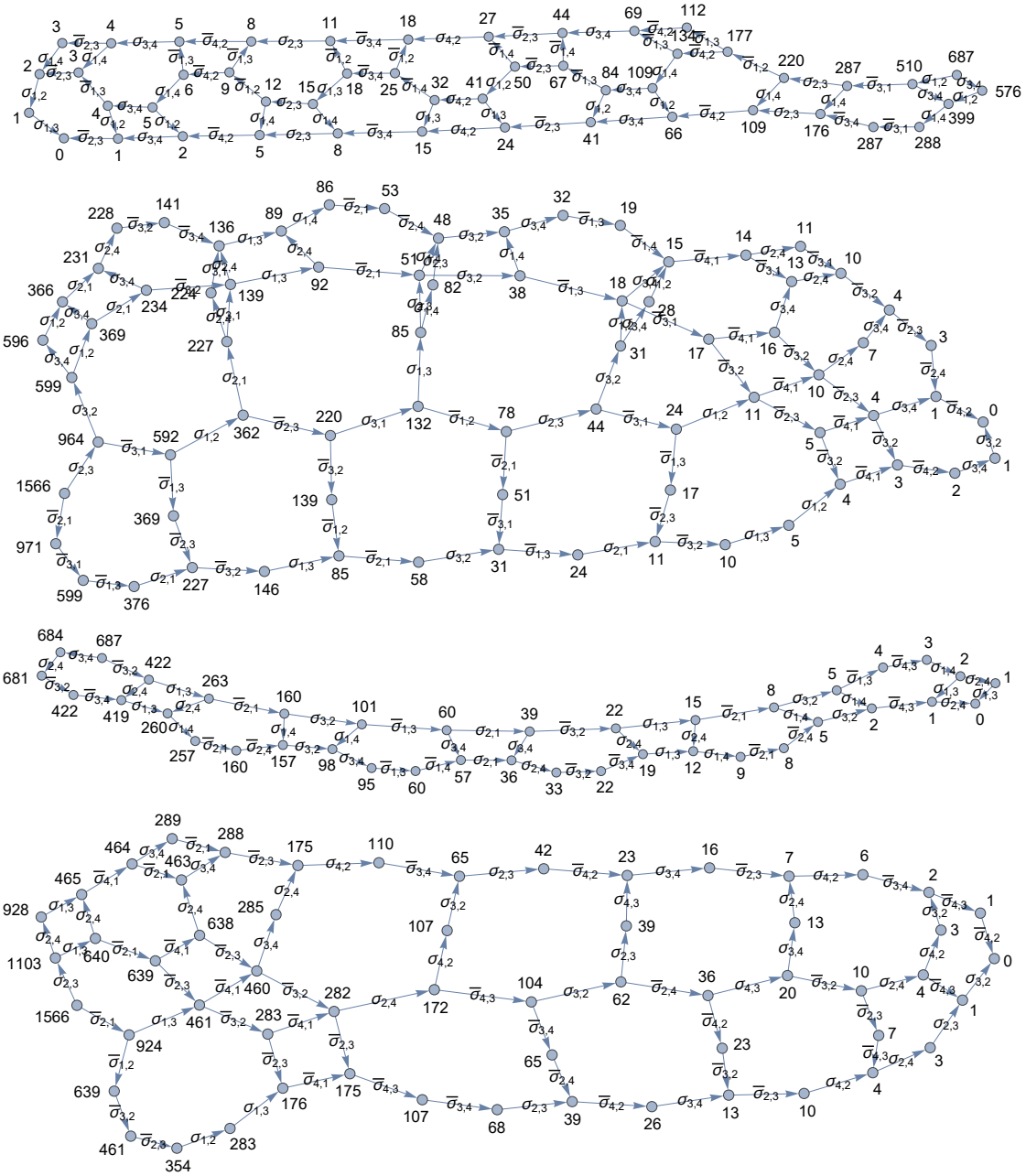
```
In[•]:= β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2};
        Print[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β};
```



In[•]:=
```
β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2};
Bend[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β}
```

Out[•]=
```
{{11, 19, 1.72727}, {11, 19, 1.72727}, {11, 19, 1.72727}, {11, 15, 1.36364}}
```

In[•]:=
```
β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2};
Print[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β};
```

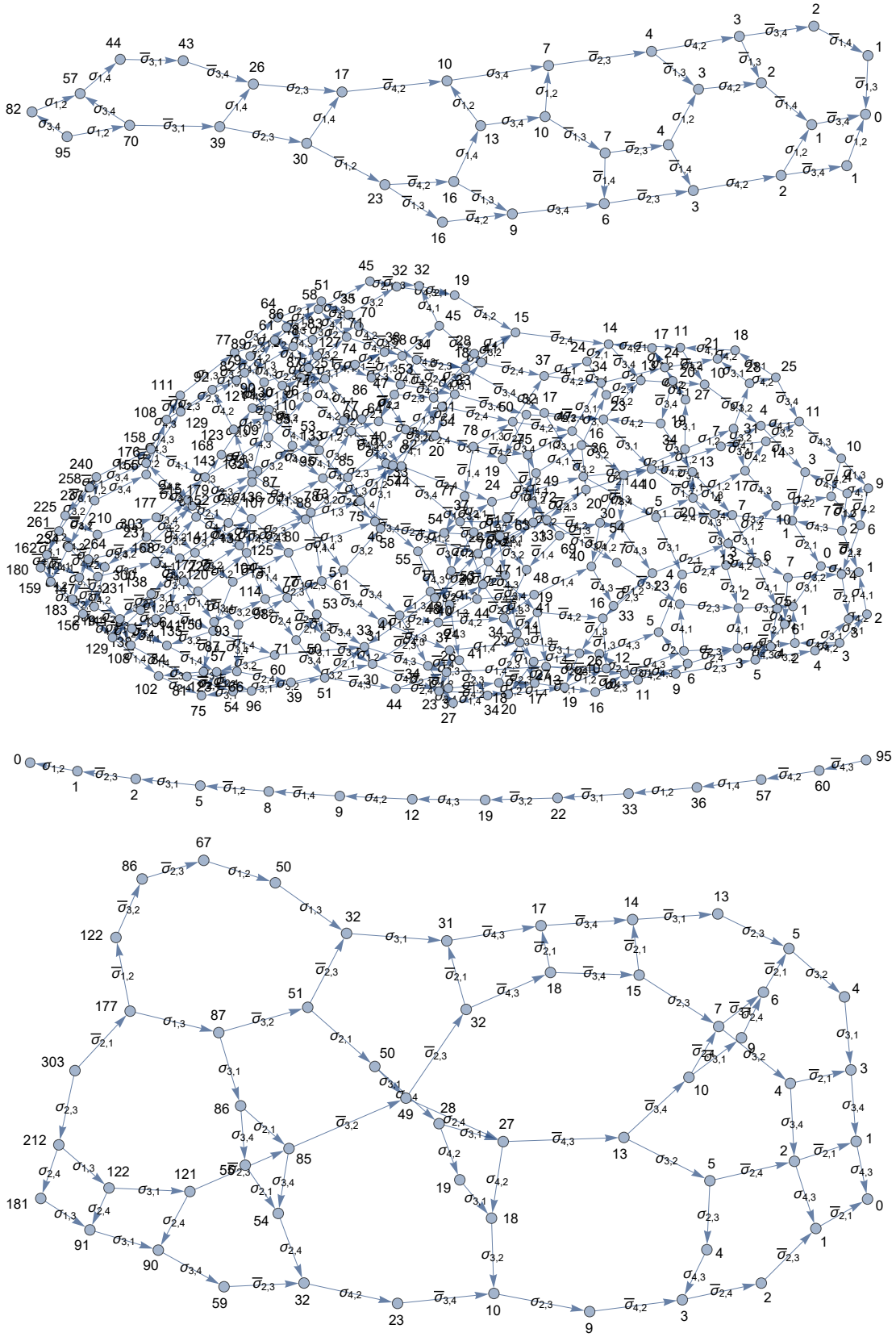


In[•]:=
```
β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2};
Bend[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β}
```

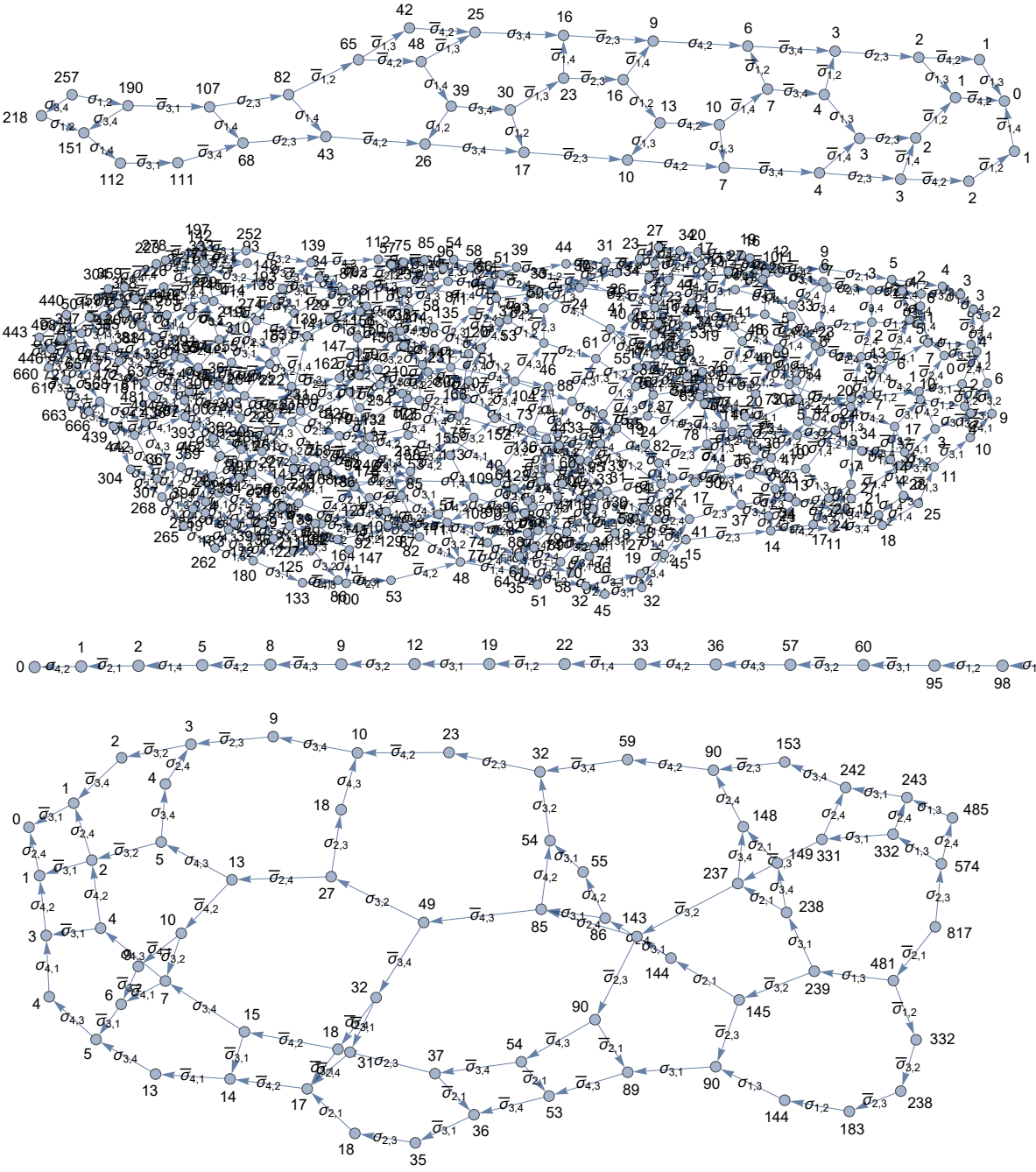
Out[•]=
```
{{13, 23, 1.76923}, {13, 23, 1.76923}, {13, 23, 1.76923}, {13, 17, 1.30769}}
```

$\beta = 1\,\overline{2}\,3\,(1\,\overline{2})^k\,\overline{3}\,\overline{2}$, its inverse, transpose, and inverse-transpose

```
In[•]:= BR[4, {2, 3, -3, 2, -1}] // ExtractionGraph // Bend
Out[•]= {3, 5, 1.66667}

In[•]:= BR[4, {2, 3, 2, -1, -3, 2, -1}] // ExtractionGraph // Bend
Out[•]= {5, 13, 2.6}

In[•]:= BR[4, {2, 3, 2, -1, 2, -1, -3, 2, -1}] // ExtractionGraph // Bend
Out[•]= {9, 25, 2.77778}

In[•]:= β = {1, -2, 3, 1, -2, 1, -2, 1, -2, -3, -2};
        Print[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β};
```



```
In[•]:= β = {1, -2, 3, 1, -2, 1, -2, 1, -2, -3, -2};
        Bend[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β}
Out[•]= {{11, 13, 1.18182}, {11, 37, 3.36364}, {13, 13, 1.}, {11, 15, 1.36364}}

In[•]:= BR[4, {2, 3, 2, -1, 2, -1, 2, -1, -3, 2, -1}] // ExtractionGraph // Bend
Out[•]= {11, 37, 3.36364}

In[•]:= β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2, -3, -2};
        Print[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β};
```



```
In[•]:= β = {1, -2, 3, 1, -2, 1, -2, 1, -2, 1, -2, -3, -2};
        Bend[ExtractionGraph[BR[4, #]]] & /@ {β, -Reverse@β, Reverse@β, -β}
Out[•]= {{13, 17, 1.30769}, {13, 49, 3.76923}, {17, 17, 1.}, {13, 19, 1.46154}}

In[•]:= BR[4, {2, 3, 2, -1, 2, -1, 2, -1, 2, -1, -3, 2, -1}] // ExtractionGraph // Bend
Out[•]= {13, 49, 3.76923}

(Alt) In[•]:= Timing@Monitor[
                BR[4, {2, 3, 2, -1, 2, -1, 2, -1, 2, -1, 2, -1, -3, 2, -1}] // ExtractionGraph // Bend,
                $mon
              ]
(Alt) Out[•]= {404 631., {15, 61, 4.06667}}

(Alt) In[•]:= BR[4, {2, 3, 2, -1, 2, -1, 2, -1, 2, -1, 2, -1, -3, 2, -1}] // ExtractionGraph // VertexCount
(Alt) Out[•]= 513

(Alt) In[•]:= Timing@Monitor[
                BR[4, {2, 3, 2, -1, 2, -1, 2, -1, 2, -1}] // ExtractionGraph // Bend,
                $mon
              ]
(Alt) Out[•]= {95.7188, {10, 20, 2.}}
```

The Programs

```
(Alt) In[•]:= SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\OU"];
              << KnotTheory` (* can be commented out with little loss *)
```

```
Loading KnotTheory` version of February 2, 2020, 10:53:45.2097.
Read more at http://katlas.org/wiki/KnotTheory.
```

```
(Alt) In[•]:= SetAttributes[VD, Orderless]

(Alt) In[•]:= bp[i_, j_] := σ_{i,j}; bm[i_, j_] := σ̄_{i,j};

(Alt) In[•]:= Tidy[vd_VD] := Module[{ps = Union @@ (List @@@ vd)},
                Replace[vd, Thread[ps → Range@Length@ps], {2}] ]

(Alt) In[•]:= R12Reduce1[vd_VD] := Tidy@Module[{R2s, R2}, Which[
                  Length[R2s = Cases[vd, X_s_[i_, j_] :> X_-s[i + 1, j + 1]] ⋂ (List @@ vd)] > 0,
                  Complement[vd, VD[R2 = First@R2s, R2 /. X_s_[i_, j_] :> X_-s[i - 1, j - 1]]],
                  Length[R2s = Cases[vd, X_s_[i_, j_] :> X_-s[i + 1, j - 1]] ⋂ (List @@ vd)] > 0,
                  Complement[vd, VD[R2 = First@R2s, R2 /. X_s_[i_, j_] :> X_-s[i - 1, j + 1]]],
                  True, DeleteCases[vd, X_[i_, j_] /; Abs[i - j] == 1] ]]
```

(Alt) In[•]:=
```
R12Reduce[vd_VD] := FixedPoint[R12Reduce1, vd]
```

(Alt) In[•]:=
```
γ[vd_VD] := Module[{js, s1, i1, j1, s2, i2, j2},
  js = Cases[vd, X_[_, j_] :> j] ⋂ Cases[vd, X_[i_, _] :> i - 1];
  If[Length[js] == 0, vd,
   j1 = RandomChoice[js]; i2 = j1 + 1;
   Cases[vd, X_s_[i_, j1] :> (s1 = s; i1 = i)];
   Cases[vd, X_s_[i2, j_] :> (s2 = s; j2 = j)];
   Tidy@Join[Complement[vd, VD[X_s1[i1, j1], X_s2[i2, j2]]],
     VD[X_s2[j1, j2], X_s1[i1, i2], X_s1s2[i1 - s1/3, j2 + s2/3], X_-s1s2[i1 + s1/3, j2 - s2/3]]
   ]]]
```

(Alt) In[•]:=
```
Γ[vd_VD] := FixedPoint[γ, vd, 2^16]
```

(Alt) In[•]:=
```
Γ[T_] /; Head[T] =!= VD := Γ[VD[T]]
```

(Alt) In[•]:=
```
Γ̄[vd_VD] := FixedPoint[γ@*R12Reduce, vd, 2^16]
```

(Alt) In[•]:=
```
Γ̄[T_] /; Head[T] =!= VD := Γ̄[VD[T]]
```

In[•]:=
```
BR[3, {2, -1, -1, -1, -1}] // Γ̄
```
Out[•]=
```
VD[BR[1, 2]]
```

(Alt) In[•]:=
```
VPB[n_, {σs___}] := VPB[n, σs];
```

(Alt) In[•]:=
```
VD /: vd1_VD ** vd2_VD := Module[{es1, es2, m2},
  es1 = Cases[vd1, EOS[i_] :> i];
  m2 = Max[es2 = Cases[vd2, EOS[i_] :> i]];
  Tidy[vd1 ⋃ Replace[DeleteCases[vd2, _EOS],
     i_ :> i/m2 - 1 + es1[[1 + Count[es2, e_ /; i > e]]], {2}]]
 ]
```

(Alt) In[•]:=
```
VD[VPB[n_]] := VD @@ (EOS /@ Range[n]);
VD[VPB[n_, σ_i_,j_]] := Tidy@Append[VD @@ (EOS /@ Range[n]), X_+1[i - 0.5, j - 0.5]];
VD[VPB[n_, σ̄_i_,j_]] := Tidy@Append[VD @@ (EOS /@ Range[n]), X_-1[i - 0.5, j - 0.5]];
VD[VPB[n_, σ_, σs__]] := VD[VPB[n, σ]] ** VD[VPB[n, σs]]
```

(Alt) In[•]:=
```
VPBGenerators[n_] :=
  VPBGenerators[n] = Flatten@Table[{σ_i,j, σ̄_i,j}, {i, n}, {j, DeleteCases[Range@n, i]}];
```

(Alt) In[•]:=
```
ProudFollowers[n_, σ_i_,j_] := ProudFollowers[n, σ_i,j] = Module[{p, q, s},
    Flatten@{σ_i,j, σ_j,i, σ̄_j,i,
      Table[{σ_p,q, σ_q,p, σ̄_p,q, σ̄_q,p}, {p, {i, j}}, {q, Complement[Range[n], {i, j}]}],
      Table[{σ_p,q, σ̄_p,q},
       {p, Complement[Range[i + 1, n], {j}]}, {q, Complement[Range[n], {i, j, p}]}]
     }];
ProudFollowers[n_, σ̄_i_,j_] :=
 ProudFollowers[n, σ̄_i,j] = ProudFollowers[n, σ_i,j] /. σ_i,j → σ̄_i,j
```

(Alt) In[•]:=
```
ProudFollowers[n_, i_Integer] :=
  DeleteCases[Range[Max[Abs[i] - 1, 1], n - 1] ⋃ (-Range[Max[Abs[i] - 1, 1], n - 1]), -i];
```

In[•]:=
```
ProudFollowers[5, 3]
```

Out[•]=
```
{-4, -2, 2, 3, 4}
```

(Alt) In[•]:=
```
ProudVPBs[n_, 0] := {VPB[n]};
ProudVPBs[n_, 1] := VPB[n, #] & /@ VPBGenerators[n];
ProudVPBs[n_, m_] /; m > 1 := Flatten[
  ProudVPBs[n, m - 1] /. VPB[n, σs___, σ_] :↦ (VPB[n, σs, σ, #] & /@ ProudFollowers[n, σ])]
```

(Alt) In[•]:=
```
CountOUForms[n_, m_] := Module[{k},
  Length@Union@Flatten@Table[Γ̄@vpb, {k, 0, m}, {vpb, ProudVPBs[n, k]}]]
```

(Alt) In[•]:=
```
ProudBs[n_, 0] := {BR[n, {}]};
ProudBs[n_, 1] := BR[n, {#}] & /@ (Range[n - 1] ⋃ (-Range[n - 1]));
ProudBs[n_, m_] /; m > 1 := Flatten[
  ProudBs[n, m - 1] /. BR[n, {σs___, σ_}] :↦ (BR[n, {σs, σ, #}] & /@ ProudFollowers[n, σ])]
```

In[•]:=
```
ProudBs[3, 3]
```

Out[•]=
```
{BR[3, {-2, -2, -2}], BR[3, {-2, -2, -1}], BR[3, {-2, -2, 1}], BR[3, {-2, -1, -2}],
 BR[3, {-2, -1, -1}], BR[3, {-2, -1, 2}], BR[3, {-2, 1, -2}], BR[3, {-2, 1, 1}],
 BR[3, {-2, 1, 2}], BR[3, {-1, -2, -2}], BR[3, {-1, -2, -1}], BR[3, {-1, -2, 1}],
 BR[3, {-1, -1, -2}], BR[3, {-1, -1, -1}], BR[3, {-1, -1, 2}], BR[3, {-1, 2, -1}],
 BR[3, {-1, 2, 1}], BR[3, {-1, 2, 2}], BR[3, {1, -2, -2}], BR[3, {1, -2, -1}],
 BR[3, {1, -2, 1}], BR[3, {1, 1, -2}], BR[3, {1, 1, 1}], BR[3, {1, 1, 2}],
 BR[3, {1, 2, -1}], BR[3, {1, 2, 1}], BR[3, {1, 2, 2}], BR[3, {2, -1, -2}],
 BR[3, {2, -1, -1}], BR[3, {2, -1, 2}], BR[3, {2, 1, -2}], BR[3, {2, 1, 1}],
 BR[3, {2, 1, 2}], BR[3, {2, 2, -1}], BR[3, {2, 2, 1}], BR[3, {2, 2, 2}]}
```

(Alt) In[•]:=
```
CountBs[n_, m_] := Module[{k},
  Length@Union@Flatten@Table[Γ̄@b, {k, 0, m}, {b, ProudBs[n, k]}]]
```

In[•]:=
```
CountBs[3, 3]
```

Out[•]=
```
1
```

(Alt) In[•]:=
```
AllBs[n_, m_] := DeleteDuplicatesBy[Γ̄]@Flatten@Table[b, {k, 0, m}, {b, ProudBs[n, k]}]
```

In[•]:=
```
Length /@ Table[ProudBs[3, k], {k, 0, 3}]
```

Out[•]=
```
{1, 4, 12, 36}
```

(Alt) In[•]:=
```
AllOUs[n_, 0] := {VD @@ Flatten@Table[{BT[2 i - 1], EOS[2 i]}, {i, n}]};
AllOUs[n_, m_] /; m > 0 :=
 Sort@Flatten[AllOUs[n, m - 1] /. vd_VD :> Module[{BTs, EOSs, k, maxO, s},
       BTs = Sort@Cases[vd, BT[i_] :> i];
       EOSs = Sort@Cases[vd, EOS[i_] :> i];
       maxO = Max[1, Max[Cases[vd, X_[i_, _] :> i]]];
       Table[
        Tidy[Append[vd, Xs[p - 0.5, q + 0.5]]],
        {s, {-1, 1}}, {k, Length[BTs]},
        {q, BTs[[k]], EOSs[[k]] - 1}, {p, Select[BTs, (# ≥ maxO) &]}
       ]]]
```

(Alt) In[•]:=
```
AllROUs[n_, m_] :=
 Select[AllOUs[n, m] /. vd_VD :> Tidy@DeleteCases[vd, _BT], (# === R12Reduce[#]) &]
```

(Alt) In[•]:=
```
ξ[vd_VD] := Count[Γ̄[vd], X_[_, _]]
```

(Alt) In[•]:=
```
VD /: (σ_i_,j_ | vd_VD) := Switch[Order[ξ[vd], ξ[VD[VPB[Count[vd, _EOS], σ̄_i,j]] ** vd]],
   0, Print["OMG, Trouble!"],
   1, False, -1, True];
VD /: (σ̄_i_,j_ | vd_VD) := Switch[Order[ξ[vd], ξ[VD[VPB[Count[vd, _EOS], σ_i,j]] ** vd]],
   0, Print["OMG, Trouble!"],
   1, False, -1, True];
```

(Alt) In[•]:=
```
VD /: Divisors[vd_VD] := Select[VPBGenerators[Count[vd, _EOS]], (# | vd) &];
VD /: Quotients[vd_VD] :=
  Γ̄[VD[VPB[Count[vd, _EOS], # /. {σ → σ̄, σ̄ → σ}]] ** vd] & /@ Divisors[vd];
```

(Alt) In[ ]:=
```
OUGraph[n_, m_] := Module[{gens, OUs, k, d, g, q, m1, m2},
  gens = VPBGenerators[n];
  OUs = Flatten@Table[AllROUs[n, k], {k, 0, m}];
  OURule = Dispatch@Thread[OUs → Range@Length@OUs];
  Graph[
   Range@Length@OUs,
   Union@Flatten@Table[
      m1 = Count[d, X_[_, _]];
      m2 = Count[q = Γ̄[VD[VPB[n, g]] ** d], X_[_, _]];
      If[m2 < m1, Labeled[(d ↔ q) /. OURule, g], Nothing],
      {d, OUs}, {g, gens}
     ]
   ]
  ]
```

(Alt) In[ ]:=
```
ExtractVPB[vd_VD] := Module[{n, ds, d},
   n = Count[vd, _EOS];
   If[Length[ds = Divisors[vd]] == 0, VPB[n],
    d = First@Sort[ds];
    q = Γ̄[VD[VPB[n, d /. {σ → σ̄, σ̄ → σ}]] ** vd];
    Insert[ExtractVPB[q], d, 2]
   ]];
CF[vpb_VPB] := ExtractVPB[Γ̄[vpb]];
```

(Alt) In[ ]:=
```
ExtractionGraph[O_, opts___] :=
 ExtractionGraph[O] = Module[{vd, n, gs, vs, es, p, m1, m2, g, q, k},
   gs = VPBGenerators[n = Count[vd = Γ̄[O], _EOS]];
   vs = {vd}; es = {}; p = 0;
   While[p < Length[vs],
    m1 = Count[vd = vs[[++p]], X_[_, _]];
    Do[
     m2 = Count[q = Γ̄[VD[VPB[n, g /. {σ → σ̄, σ̄ → σ}]] ** vd], X_[_, _]];
     If[m2 < m1,
      If[! MemberQ[vs, q], $mon = {p, Length@vs}; AppendTo[vs, q]];
      k = Position[vs, q][[1, 1]];
      AppendTo[es, Labeled[p ↔ k, g]]
     ],
     {g, gs}
    ]
   ];
   Graph[Table[Labeled[k, Length[vs[[k]]] - n], {k, p}],
    es, GraphLayout → "SpringElectricalEmbedding", opts]
  ]
```

In[•]:=
```
BR[3, {2, -1, -1, -1, -1}] // ExtractionGraph
```

Out[•]=

(Alt) In[•]:=
```
VPB[BR[n_, is_List]] := VPB[n, Module[{π, i},
    π = Range[n];
    Sequence @@ Table[
      If[i > 0,
       π[[{i, i + 1}]] = π[[{i + 1, i}]]; σ_{π[[i+1]],π[[i]]},
       π[[{-i, -i + 1}]] = π[[{-i + 1, -i}]]; σ̄_{π[[-i]],π[[-i+1]]}
      ],
      {i, is}
     ]]];
VD[br_BR] := VD[VPB@br]
```

(Alt) In[•]:=
```
RandomBraid[n_, m_] := BR[n, Table[RandomChoice[Range[n - 1] ⋃ (-Range[n - 1])], {m}]]
```

(Alt) In[•]:=
```
Bend[G_Graph] := Module[{A, s, t, k = 0, l},
   A = AdjacencyMatrix@G;
   While[Total[MatrixPower[A, k + 1], 2] > 0, ++k];
   {s} = FirstPosition[Plus @@ Normal@A, 0];
   {t} = FirstPosition[Plus @@@ Normal@A, 0];
   l = Length@FindShortestPath[G, s, t] - 1;
   {l, k, N[k / l]}
  ];
```