progressive transmission

signed binary of integers

In[40]:=
```
IntegerDigits[5, 2, 6]
```

Out[40]=
```
{0, 0, 0, 1, 0, 1}
```

In[41]:=
```
tosignedbinary[int_, n_] :=
 Module[{s},
  If[Abs[int] ≥ 2^n, Throw["input too big"]];
  s = If[int ≥ 0, 0, 1];
  Flatten[{s, IntegerDigits[int, 2, n]}]
 ]
```

In[42]:=
```
tosignedbinary[5, 4]
tosignedbinary[5, 5]
tosignedbinary[5, 2]
```

Out[42]=
```
{0, 0, 1, 0, 1}
```

Out[43]=
```
{0, 0, 0, 1, 0, 1}
```

… Throw: Uncaught Throw[input too big] returned to top level.

Out[44]=
```
Hold[Throw[input too big]]
```

lists of signed binary numbers

In[45]:=
```
convertlist[list_] := Module[{lmax, n},
  lmax = Max[Abs[list]];
  n = IntegerLength[lmax, 2];
  {n, Map[tosignedbinary[#, n] &, list]}
 ]
```

In[46]:=
```
convertlist[{-12, 3, 4}] // MatrixForm
```

Out[46]//MatrixForm=

$$\begin{pmatrix} 4 \\ \{\{1,1,1,0,0\},\{0,0,0,1,1\},\{0,0,1,0,0\}\} \end{pmatrix}$$

In[47]:=
```
convertlist[{3, -12, 7, 9, -7, 0, 1, -5, 8, 0}] // TableForm
```

Out[47]//TableForm=

4									
0	1	0	0	1	0	0	1	0	0
0	1	0	1	0	0	0	0	1	0
0	1	1	0	1	0	0	1	0	0
1	0	1	0	1	0	0	0	0	0
1	0	1	1	1	0	1	1	0	0

reordering lists according to bitlength

In[48]:=
```
reorder[list_] := Module[{cvl, in, out},
  cvl = convertlist[list];
  in = Table[
    Append[cvl[[2, i]], i], {i, 1, Length[cvl[[2]]]}];
  out = {};
  For[i = 1, i ≤ cvl[[1]], i++,
   out = Join[out, Select[in, #[[i + 1]] == 1 &]];
   in = Select[in, #[[i + 1]] == 0 &]];
  Join[out, in]
 ]
```

In[49]:=
```
reorder[{3, -12, 4, 9, -8, 0, -1, -8, 8, 0}] // TableForm
```

Out[49]//TableForm=

1	1	1	0	0	2
0	1	0	0	1	4
1	1	0	0	0	5
1	1	0	0	0	8
0	1	0	0	0	9
0	0	1	0	0	3
0	0	0	1	1	1
1	0	0	0	1	7
0	0	0	0	0	6
0	0	0	0	0	10

progressive transmission

In[50]:=
```
progtrans[list_, K_] :=
 Module[{len, cvl, n, in, out, newout, sgnout, posout, bitout},
  len = Length[list];
  cvl = convertlist[list];
  n = cvl[[1]];
  If[K > n, Print["order too big"]];
  in = Table[
    Prepend[cvl[[2, i]], i], {i, 1, len}];
  out = {};
  For[j = 1, j ≤ Min[n, K], j++,
   newout = Select[in, #[[j + 2]] == 1 &];
   in = Select[in, #[[j + 2]] == 0 &];
   sgnout = newout[[All, 2]];
   posout = newout[[All, 1]];
   bitout = If[j == 1, {}, out[[All, j + 2]]];
   out = Join[out, newout];
   Print["round ", j];
   Print[{posout, sgnout, bitout}];
  ];
  Print["remaining ", in // MatrixForm];
  Join[out, in] // MatrixForm
 ]
```

In[51]:=
```
progtrans[{3, -12, 4, 9, -5, 0, -1, -8, 8, 0}, 4]
```

```
round 1
{{2, 4, 8, 9}, {1, 0, 1, 0}, {}}
round 2
{{3, 5}, {0, 1}, {1, 0, 0, 0}}
round 3
{{1}, {0}, {0, 0, 0, 0, 0, 0}}
round 4
{{7}, {1}, {0, 1, 0, 0, 0, 1, 1}}
remaining ⎛ 6  0 0 0 0 0 ⎞
          ⎝ 10 0 0 0 0 0 ⎠
```

Out[51]//MatrixForm=

$$\begin{pmatrix} 2 & 1 & 1 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 \\ 8 & 1 & 1 & 0 & 0 & 0 \\ 9 & 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 0 \\ 5 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 7 & 1 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

coding and exploring trees (forests)

In[52]:=
```
exploretree[T_, σ_] :=
 Module[{n, ttest, edges, roots, properedges, succ, delta, lambda},
  n = Length[T];
  If[! (n == Length[σ]), Throw["input error"]];
  ttest = ReplacePart[Table[T[[i]] ≤ i, {i, 1, n}], 0 → And];
  If[! ttest, Throw["T not a tree"]];
  edges = Table[{T[[j]], j}, {j, 1, n}];
  roots = Flatten[Map[DeleteDuplicates[#] &, Select[edges, #[[1]] == #[[2]] &]]];
  properedges = Select[edges, #[[1]] < #[[2]] &];
  succ = Table[Map[#[[2]] &, Select[properedges, #[[1]] == k &]], {k, 1, n}];
  For[j = n, j > 0, j--,
   If[succ[[j]] == {}, δj = 0; λj = 0; Continue[]];
   λj = Max[Map[δ# &, succ[[j]]]];
   δj = Max[Join[Map[σ[[#]] &, succ[[j]]], {λj}]]
  ];
  delta = Table[δj, {j, 1, n}];
  lambda = Table[λj, {j, 1, n}];
  {roots, succ, delta, lambda}
 ]
```

the forest T

In[53]:=
```
T = {1, 2, 3, 1, 4, 2, 3, 3, 5, 6, 2, 4, 7, 7, 10, 7};
```

In[54]:=
```
TT = Table[{T[[a]], a}, {a, 1, Length[T]}]
```

Out[54]=
```
{{1, 1}, {2, 2}, {3, 3}, {1, 4}, {4, 5}, {2, 6}, {3, 7}, {3, 8},
 {5, 9}, {6, 10}, {2, 11}, {4, 12}, {7, 13}, {7, 14}, {10, 15}, {7, 16}}
```

the edges of T

In[55]:=
```
edges = Select[TT, #[[1]] < #[[2]] &]
```

Out[55]=
```
{{1, 4}, {4, 5}, {2, 6}, {3, 7}, {3, 8}, {5, 9},
 {6, 10}, {2, 11}, {4, 12}, {7, 13}, {7, 14}, {10, 15}, {7, 16}}
```

the successor map of T

In[56]:=
```
succ = Map[#[[1]] ↦ #[[2]] &, edges]
```

Out[56]=
```
{1 ↦ 4, 4 ↦ 5, 2 ↦ 6, 3 ↦ 7, 3 ↦ 8, 5 ↦ 9,
 6 ↦ 10, 2 ↦ 11, 4 ↦ 12, 7 ↦ 13, 7 ↦ 14, 10 ↦ 15, 7 ↦ 16}
```

A valuation σ

In[57]:=
```
σ = {0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1};
```

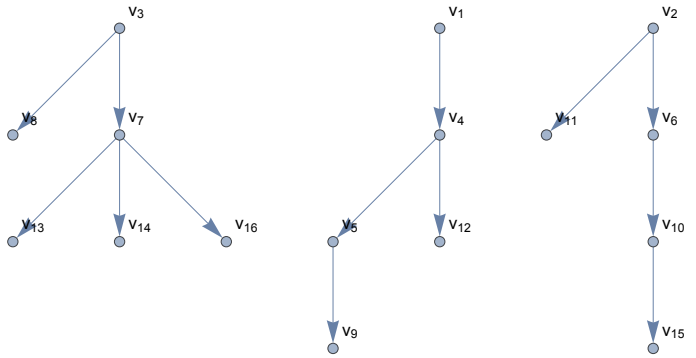
In[58]:=
```
σv = Table[a → σ[[a]], {a, 1, 16}]
```

Out[58]=
```
{1 → 0, 2 → 0, 3 → 0, 4 → 1, 5 → 1, 6 → 0, 7 → 1, 8 → 0,
 9 → 1, 10 → 0, 11 → 1, 12 → 1, 13 → 0, 14 → 0, 15 → 0, 16 → 1}
```

the forest seen by the successor map

In[59]:=
```
Graph[Range[16], succ, VertexLabels → Table[a → "v"ₐ, {a, 1, 16}]]
```

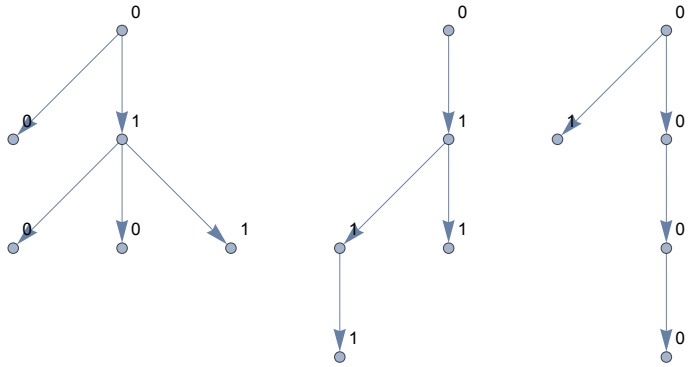
Out[59]=



the valuation σ

In[60]:=
```
Graph[Range[16], succ, VertexLabels → σv]
```

Out[60]=



computing the δ and λ functions

In[61]:=
```
t = exploretree[T, σ];
```

the roots of the forest T

In[62]:=
```
t[[1]]
```

Out[62]=
```
{1, 2, 3}
```

the successor map succ

In[63]:=
```
t[[2]]
```

Out[63]=
```
{{4}, {6, 11}, {7, 8}, {5, 12}, {9}, {10},
 {13, 14, 16}, {}, {}, {15}, {}, {}, {}, {}, {}, {}}
```

the δ-function of (T,σ)

In[64]:=
```
δ = t[[3]]
```

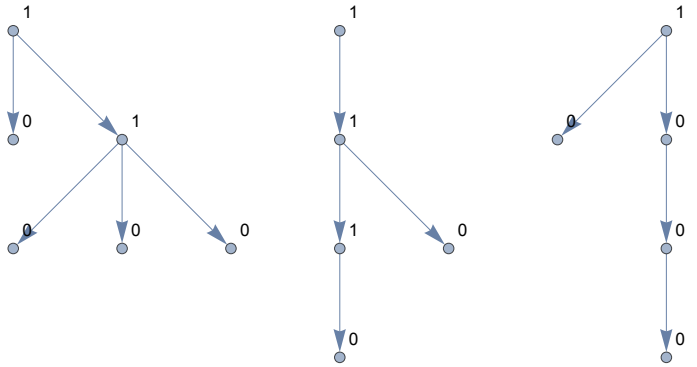
Out[64]=
```
{1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

In[65]:=
```
δv = Table[a → δ[[a]], {a, 1, 16}]
```

Out[65]=
```
{1 → 1, 2 → 1, 3 → 1, 4 → 1, 5 → 1, 6 → 0, 7 → 1, 8 → 0,
 9 → 0, 10 → 0, 11 → 0, 12 → 0, 13 → 0, 14 → 0, 15 → 0, 16 → 0}
```

In[66]:=
```
Graph[succ, VertexLabels → δv]
```

Out[66]=



the λ-function of (T,σ)

In[67]:=
```
λ = t[[4]]
```

Out[67]=
```
{1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

In[68]:=
```
λv = Table[a → λ[[a]], {a, 1, 16}]
```

Out[68]=
```
{1 → 1, 2 → 0, 3 → 1, 4 → 1, 5 → 0, 6 → 0, 7 → 0, 8 → 0,
 9 → 0, 10 → 0, 11 → 0, 12 → 0, 13 → 0, 14 → 0, 15 → 0, 16 → 0}
```

In[69]:=
```
Graph[succ, VertexLabels → λv]
```

Out[69]=