Lattices and Coxeter Diagrams DZG

October 3, 2023

[47]:
```
from IPython.display import Math
import numpy as np
import pandas as pd
from IPython.display import HTML

H = IntegralLattice("H")
E8 = IntegralLattice("E8").twist(-1)
E82 = E8.twist(2)
H2 = H.twist(2)

S22 = SymmetricGroup(22)
rho = S22("(5, 9, 13, 1)(6, 10, 14, 2)(7, 11, 15, 3)(8, 12, 16, 4)(18, 19, 20,␣
↪17)(21, 22)")
s = S22("(1, 9)(2, 8)(3, 7)(4, 6)(10, 16)(11, 15)(12, 14)(17, 19)")
r = rho * rho
d = s
h = rho * s
v = s * rho
display(v)
```

$(1, 13)(2, 12)(3, 11)(4, 10)(5, 9)(6, 8)(14, 16)(17, 20)(18, 19)(21, 22)$

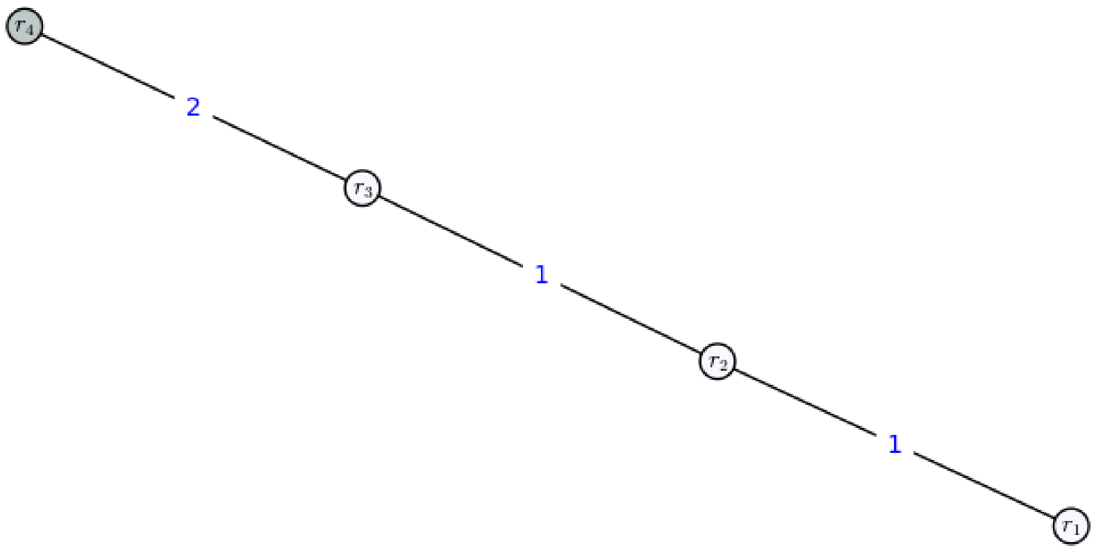
[22]:
```
# Starting new indexing
# l = [(i+1, i) for i in range(22) ]
# d = dict(l)
# H = PermutationGroup([[d[i] for i in g.tuple()] for g in S22.gens()],␣
↪domain=d.values() )
# rho = H("(4,8,12,0)(5,9,13,1)(6,10,14,2)(7,11,15,3)(17,18,19,16)(20,21)")
# s = H("(0, 8)(1, 7)(2, 6)(3, 5)(9, 15)(10, 14)(11, 13)(16, 18)")
# r = rho * rho
# d = s
# h = rho * s
# v = s * rho
```

[23]:
```
# Build a Coxeter diagram from a Coxeter matrix

def Coxeter_Diagram(M):
    nverts = M.ncols()
```

```
    # print(str(nverts) + " vertices")
    G = Graph()
    vertex_labels = dict();
# plot_coxeter_diagram(G)

    vertex_colors = {
        '#F8F9FE': [], # white
        '#BFC9CA': [], # black
    }

    for i in range(nverts):
        for j in range(nverts):
            mij = M[i, j]
            if i == j:
                if mij == -2:
                    vertex_colors["#F8F9FE"].append(i) # white
                    continue
                if mij == -4:
                    vertex_colors["#BFC9CA"].append(i) # black
                    continue
                continue
            if mij != 0:
                G.add_edge(i, j, str(mij) )
                continue
    assert len( vertex_colors["#F8F9FE"]) + len( vertex_colors["#BFC9CA"]) ==␣
 ↪nverts
    G.vertex_colors = vertex_colors
    return G

def plot_coxeter_diagram(G, v_labels, pos={}):
    n = len( G.vertices() )
    vlabs = {v: k for v, k in enumerate(v_labels)}
    if pos == {}:
        display(G.plot(
            edge_labels=True,
            vertex_labels = vlabs,
            vertex_size=200,
            vertex_colors = G.vertex_colors
        ))
    else:
        display(G.plot(
            edge_labels=True,
            vertex_labels = vlabs,
            vertex_size=200,
            vertex_colors = G.vertex_colors,
            pos = pos
        ))
```

```
# Test
M = Matrix(ZZ, 4, [ [-2, 1, 0, 0], [1, -2, 1, 0], [0, 1, -2, 2], [0, 0, 2, -4]␣
↪])
display(M)
G = Coxeter_Diagram(M)
plot_coxeter_diagram(G, v_labels = [f"$r_{ {i + 1} }$" for i in range( 4 )] )
```

$$\begin{pmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 2 \\ 0 & 0 & 2 & -4 \end{pmatrix}$$



[24]:
```
# Build U(2)+E_8+E_8.
L = H2.direct_sum(E8).direct_sum(E8)
show(L.gram_matrix())
ep,fp,a1,a2,a3,a4,a5,a6,a7,a8,a1p,a2p,a3p,a4p,a5p,a6p,a7p,a8p = L.basis()
```

$$\begin{pmatrix}
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2
\end{pmatrix}$$

[25]:

```
# Luca's matrix
M=matrix([
    [0,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,-2,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,-2,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,1,0,-2,1,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,1,1,-2,1,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,1,-2,1,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,1,-2,1,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,1,-2,1,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,1,-2,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,-2,0,1,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,-2,0,1,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,1,0,-2,1,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,1,1,-2,1,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,1,-2,1,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-2,1,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-2,1],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-2]
])

show(M)

# Check to see that I recover the same matrix. Ok!
show(M - L.gram_matrix())
```

$$
\left(\begin{array}{rrrrrrrrrrrrrrrrrr}
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -2 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2
\end{array}\right)
$$

$$
\left(\begin{array}{rrrrrrrrrrrrrrrrrr}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)
$$

[26]:
```
bar_basis = IntegralLattice(block_diagonal_matrix(H2.gram_matrix(), E8.
 ↪gram_matrix().inverse(), E8.gram_matrix().inverse() ))
show( bar_basis.gram_matrix() )

_,_,a1b,a2b,a3b,a4b,a5b,a6b,a7b,a8b,a1pb,a2pb,a3pb,a4pb,a5pb,a6pb,a7pb,a8pb =␣
 ↪bar_basis.gram_matrix().columns()
```

$$
\left(\begin{array}{cccccccccccccccccc}
0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -4 & -5 & -7 & -10 & -8 & -6 & -4 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -5 & -8 & -10 & -15 & -12 & -9 & -6 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -7 & -10 & -14 & -20 & -16 & -12 & -8 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -10 & -15 & -20 & -30 & -24 & -18 & -12 & -6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -8 & -12 & -16 & -24 & -20 & -15 & -10 & -5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -6 & -9 & -12 & -18 & -15 & -12 & -8 & -4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -4 & -6 & -8 & -12 & -10 & -8 & -6 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -2 & -3 & -4 & -6 & -5 & -4 & -3 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & -5 & -7 & -10 & -8 & -6 & -4 & -2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -5 & -8 & -10 & -15 & -12 & -9 & -6 & -3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -7 & -10 & -14 & -20 & -16 & -12 & -8 & -4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -10 & -15 & -20 & -30 & -24 & -18 & -12 & -6 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -8 & -12 & -16 & -24 & -20 & -15 & -10 & -5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -6 & -9 & -12 & -18 & -15 & -12 & -8 & -4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -4 & -6 & -8 & -12 & -10 & -8 & -6 & -3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & -3 & -4 & -6 & -5 & -4 & -3 & -2
\end{array}\right)
$$

[27]:

```python
# Check all of the vectors we have

def namestr(obj):
    namespace = globals()
    return [name for name in namespace if namespace[name] is obj][0]

for l in␣
 ↪[ep,fp,a1,a2,a3,a4,a5,a6,a7,a8,a1p,a2p,a3p,a4p,a5p,a6p,a7p,a8p,a1b,a2b,a3b,a4b,a5b,a6b,a7b,
 ↪
    print(namestr(l), "=", l)
```

```
l = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
```

```
l = (0, 0, -4, -5, -7, -10, -8, -6, -4, -2, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -5, -8, -10, -15, -12, -9, -6, -3, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -7, -10, -14, -20, -16, -12, -8, -4, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -10, -15, -20, -30, -24, -18, -12, -6, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -8, -12, -16, -24, -20, -15, -10, -5, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -6, -9, -12, -18, -15, -12, -8, -4, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -4, -6, -8, -12, -10, -8, -6, -3, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -2, -3, -4, -6, -5, -4, -3, -2, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -4, -5, -7, -10, -8, -6, -4, -2)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -5, -8, -10, -15, -12, -9, -6, -3)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -7, -10, -14, -20, -16, -12, -8, -4)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -10, -15, -20, -30, -24, -18, -12, -6)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -8, -12, -16, -24, -20, -15, -10, -5)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -6, -9, -12, -18, -15, -12, -8, -4)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -4, -6, -8, -12, -10, -8, -6, -3)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, -3, -4, -6, -5, -4, -3, -2)
```

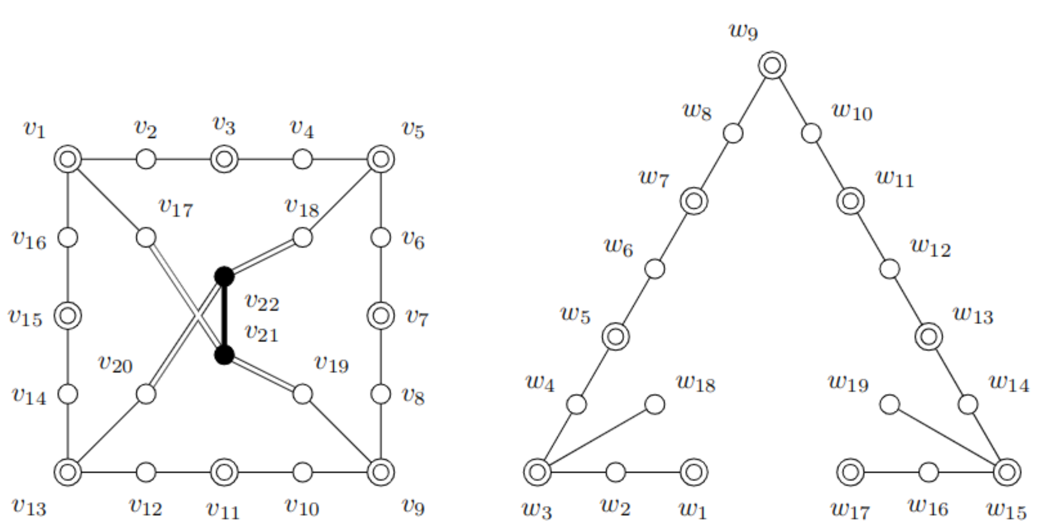
[28]:
```
dot = lambda x,y : x * L.gram_matrix() * y
nm = lambda x: dot(x, x)

show(dot(a1, a1b))
show(dot(a1, a2b))
show(nm(a1))
```

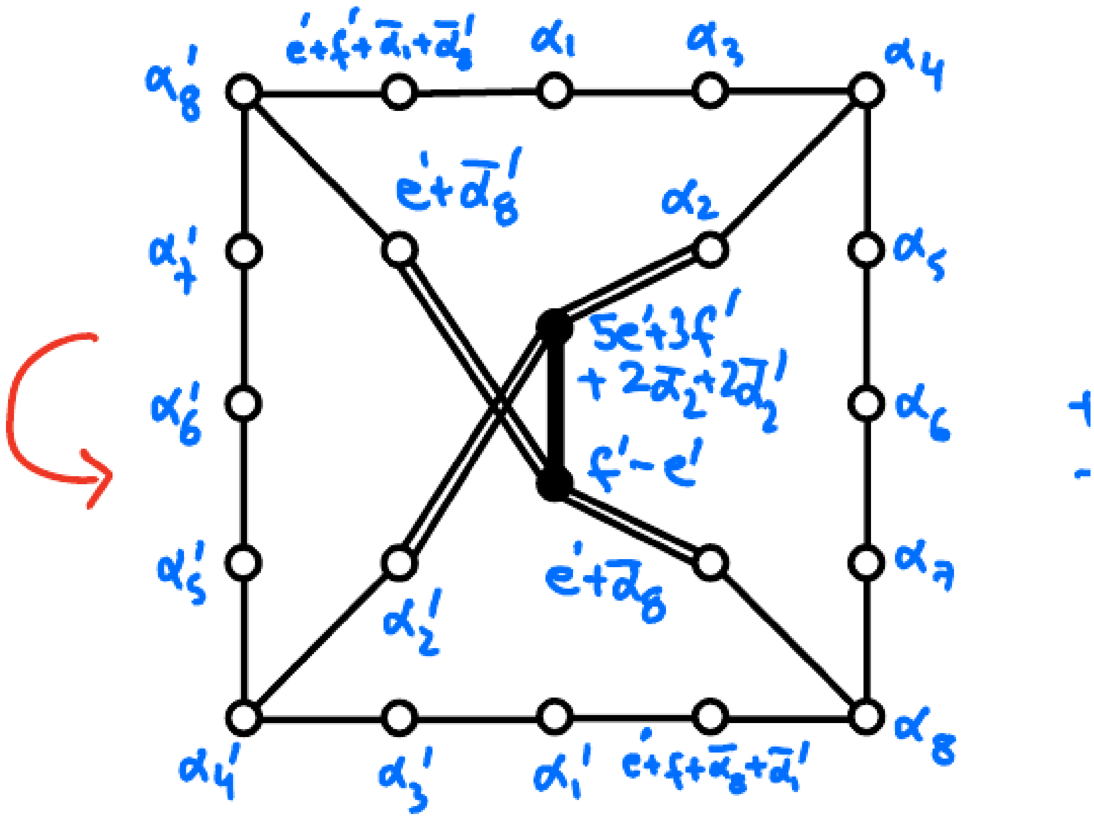
1

0

-2

1 Coxeter diagram and roots for $(18, 2, 0)_1 = U(2) + E_8^2$





[29]:
```
# Root vectors for (18, 2, 0), roots taken from above, v_i are according to numerical labeling above

v1 = a8p
v2 = ep + fp + a1b + a8pb
v3 = a1
v4 = a3
v5 = a4
v6 = a5
v7 = a6
v8 = a7
v9 = a8
v10 = ep + fp + a8b + a1pb
v11 = a1p
v12 = a3p
v13 = a4p
v14 = a5p
v15 = a6p
v16 = a7p
```

```
v17 = ep + a8pb
v18 = a2
v19 = ep + a8b
v20 = a2p

v21 = fp-ep
v22 = 5ep + 3fp + 2a2b + 2a2pb

V = [v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, v16,␣
↪v17, v18, v19, v20, v21, v22]
for v in V:
    display(v)
```

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)$

$(1, 1, -4, -5, -7, -10, -8, -6, -4, -2, -2, -3, -4, -6, -5, -4, -3, -2)$

$(0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$

$(1, 1, -2, -3, -4, -6, -5, -4, -3, -2, -4, -5, -7, -10, -8, -6, -4, -2)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)$

$(1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, -3, -4, -6, -5, -4, -3, -2)$

$(0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(1, 0, -2, -3, -4, -6, -5, -4, -3, -2, 0, 0, 0, 0, 0, 0, 0, 0)$

$(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)$

$(-1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$

$(5, 3, -10, -16, -20, -30, -24, -18, -12, -6, -10, -16, -20, -30, -24, -18, -12, -6)$

[30]:

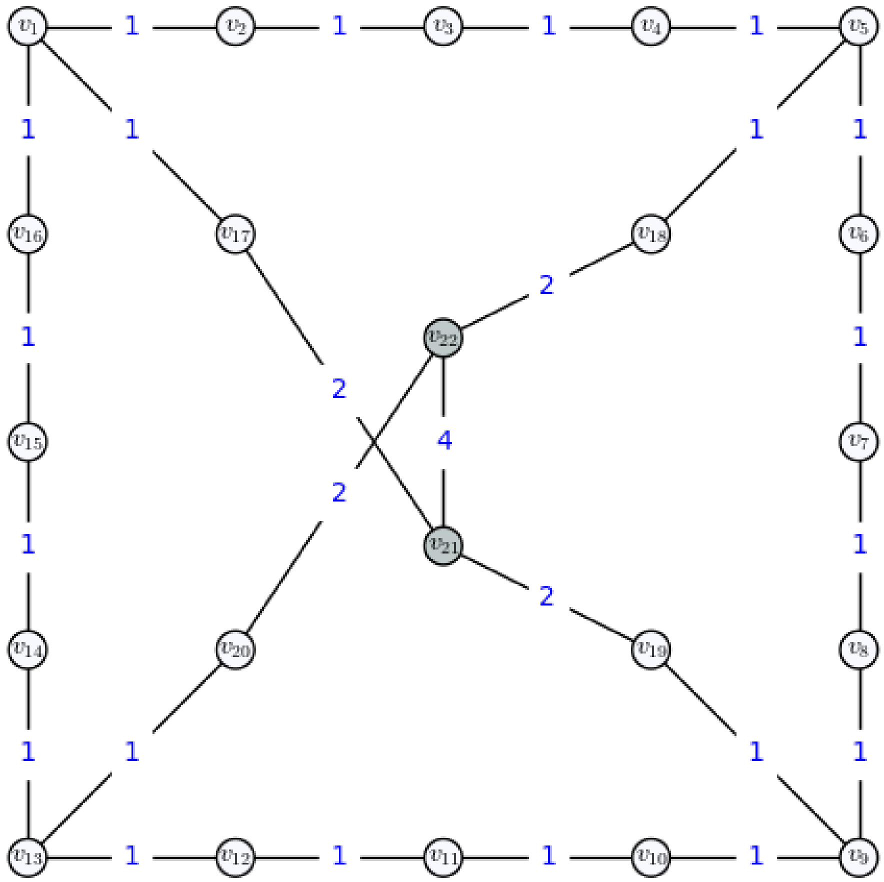
```
# Verify our choices of roots by checking all of the mutual intersections

def root_intersection_matrix(vectors, labels, bil_form):
    n = len(vectors)
    M = zero_matrix(ZZ, n)
    nums = Set(range(n))
    for i in range(n):
        for j in range(n):
            M[i, j] = bil_form( vectors[i], vectors[j] )

    print("Diagonal entries/square norms: ")
    display(M.diagonal())

    # Labels!


    df = pd.DataFrame(M, columns=labels, index=labels)
    display(HTML(df.to_html()))

    # Must be symmetric
    assert M.is_symmetric()

    # Must have -2 or -4 on the diagonal
    s = Set( M.diagonal() )
    assert s in Subsets( Set( [-2, -4] ) )

    # Diagonals should be square norms of vectors
    for i in range(n):
        assert M[i, i] == bil_form(vectors[i], vectors[i])



    return M

MV = root_intersection_matrix(V, labels = [f"$v_{ {r + 1} }$" for r in range(␣
↪len(V) )], bil_form=dot)


# MV = zero_matrix(QQ, 22)
# nums = Set(range(22))

# for i in range(22):
#     for j in range(22):
#         MV[i, j] = dot( V[i], V[j] )
# MV
```

Diagonal entries/square norms:

$[-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -4, -4]$

<IPython.core.display.HTML object>

[31]:
```
G = Coxeter_Diagram(MV)
plot_coxeter_diagram(
    G,
    v_labels = [f"$v_{ {i + 1} }$" for i in range( 22 )],
    pos = {
        0: [0, 0],
        1: [4, 0],
        2: [8, 0],
        3: [12, 0],
        4: [16, 0],
        5: [16, -4],
        6: [16, -8],
        7: [16, -12],
        8: [16, -16],
        9: [12, -16],
        10: [8, -16],
        11: [4, -16],
        12: [0, -16],
        13: [0, -12],
        14: [0, -8],
        15: [0, -4],
        16: [4, -4],
        17: [12, -4],
        18: [12, -12],
        19: [4, -12],
        20: [8, -10],
        21: [8, -6],
    }
)
```

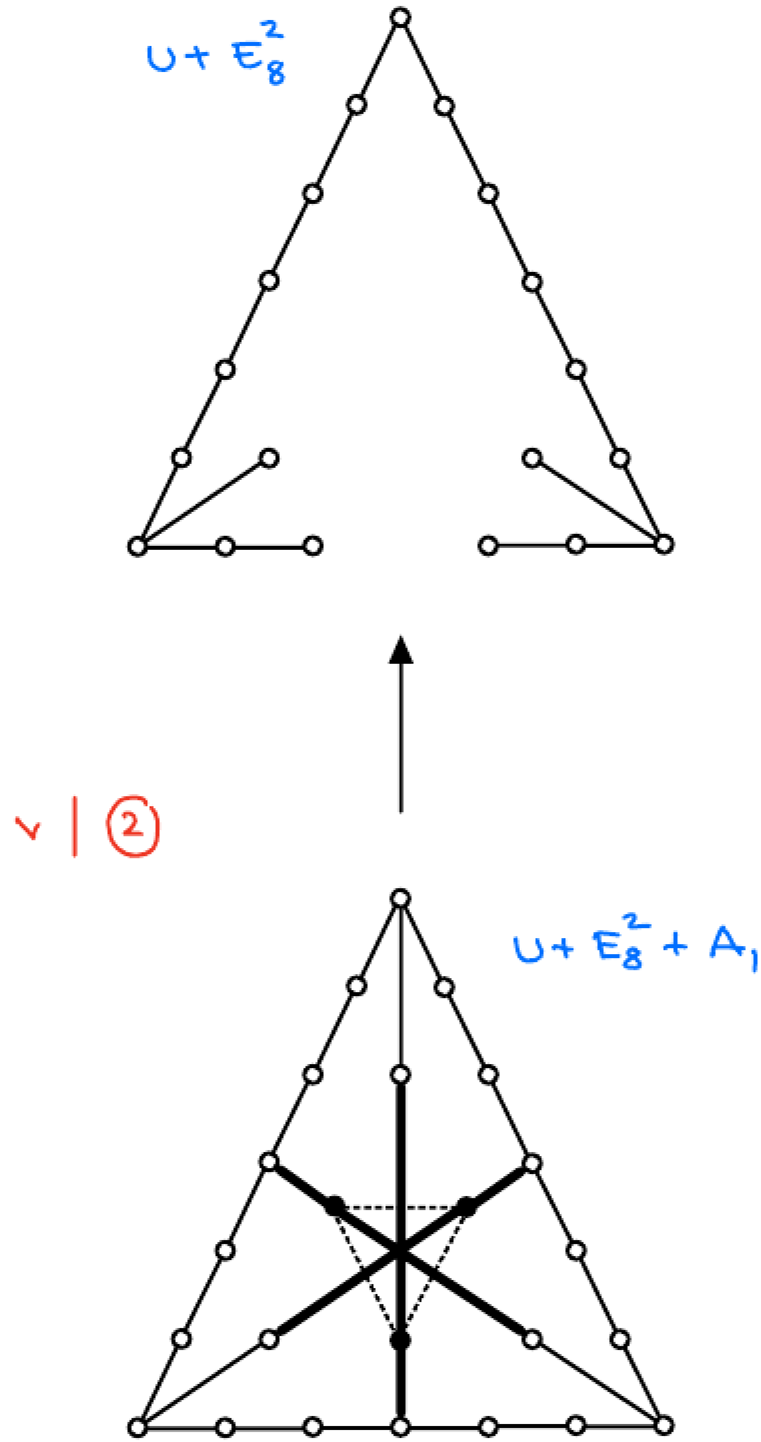


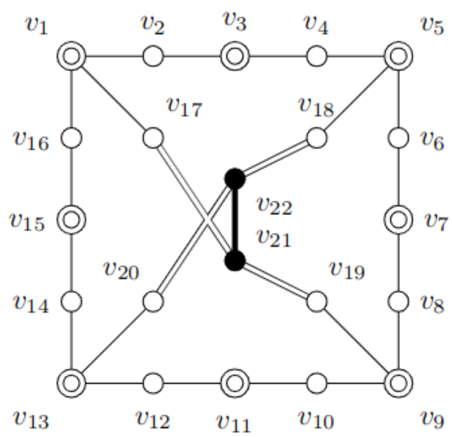
[32]:
```
## Build U+E_8+E_8.

L_18_2_0 = H.direct_sum(E8).direct_sum(E8)
#Math("(18, 2, 0) =")
#display(L_18_2_0.gram_matrix())
e,f,a1,a2,a3,a4,a5,a6,a7,a8,a1p,a2p,a3p,a4p,a5p,a6p,a7p,a8p = L_18_2_0.basis()

barbasis_18_2_0 = IntegralLattice(block_diagonal_matrix(H.gram_matrix(), E8.
 ↪gram_matrix().inverse(), E8.gram_matrix().inverse() ))

#show( barbasis_18_2_0.gram_matrix() )

_,_,a1b,a2b,a3b,a4b,a5b,a6b,a7b,a8b,a1pb,a2pb,a3pb,a4pb,a5pb,a6pb,a7pb,a8pb =␣
 ↪barbasis_18_2_0.gram_matrix().columns()

dot2 = lambda x,y : x * L_18_2_0.gram_matrix() * y
nm2 = lambda x: dot(x, x)

# Check all of the vectors we have
```

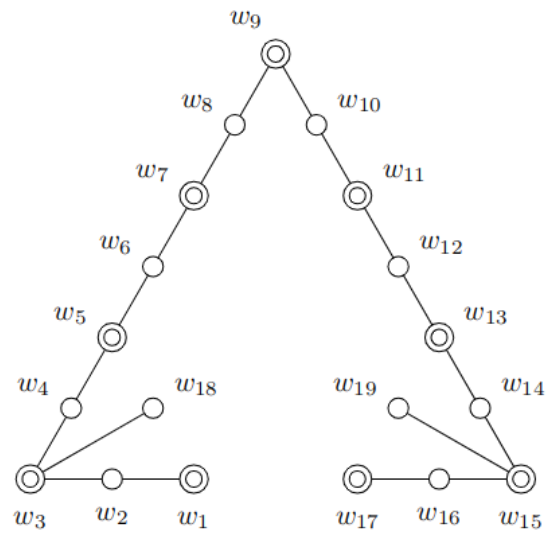
```
def namestr(obj):
    namespace = globals()
    return [name for name in namespace if namespace[name] is obj][0]

for l in␣
  ↪[e,f,a1,a2,a3,a4,a5,a6,a7,a8,a1p,a2p,a3p,a4p,a5p,a6p,a7p,a8p,a1b,a2b,a3b,a4b,a5b,a6b,a7b,a8
  ↪
    print(namestr(l), "=", l)
```

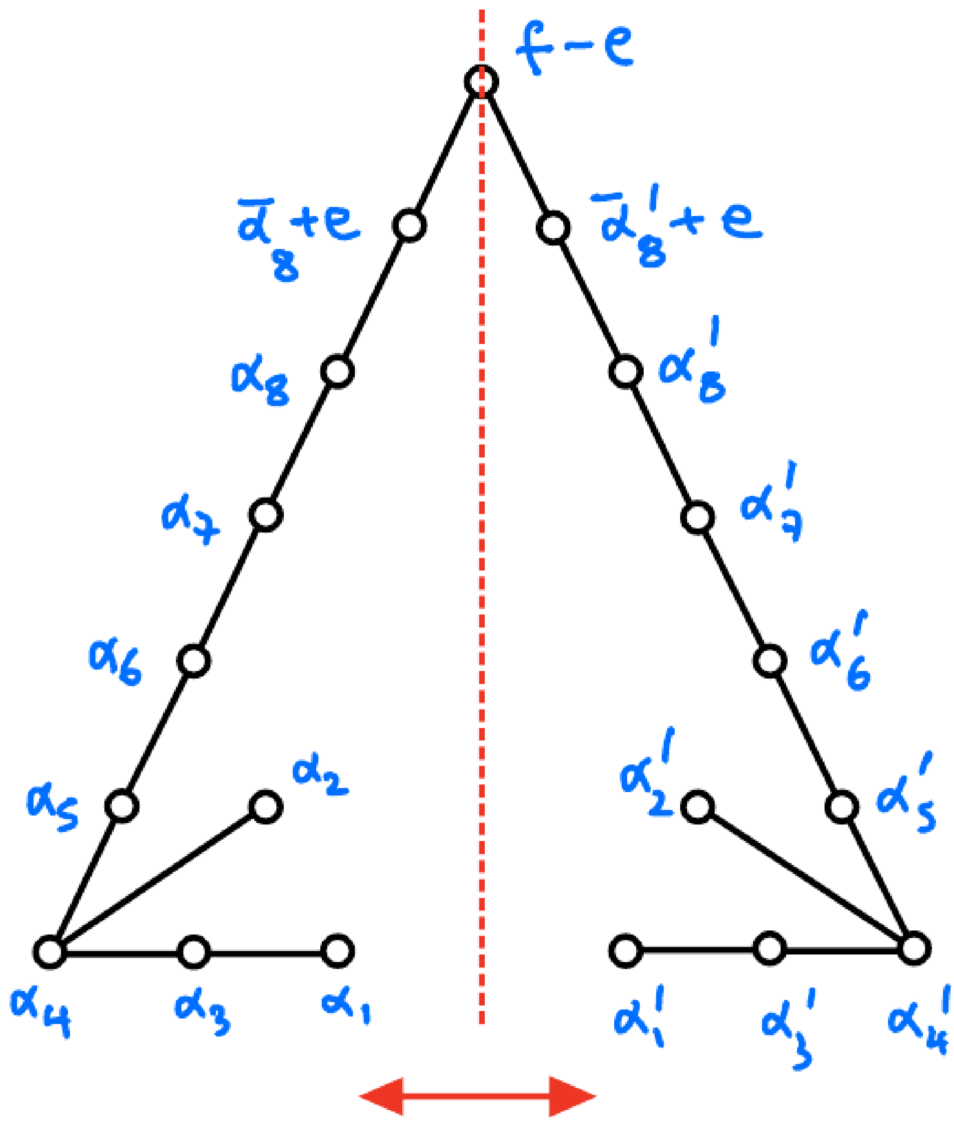
```
e = (1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1)
l = (0, 0, -4, -5, -7, -10, -8, -6, -4, -2, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -5, -8, -10, -15, -12, -9, -6, -3, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -7, -10, -14, -20, -16, -12, -8, -4, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -10, -15, -20, -30, -24, -18, -12, -6, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -8, -12, -16, -24, -20, -15, -10, -5, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -6, -9, -12, -18, -15, -12, -8, -4, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -4, -6, -8, -12, -10, -8, -6, -3, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, -2, -3, -4, -6, -5, -4, -3, -2, 0, 0, 0, 0, 0, 0, 0, 0)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -4, -5, -7, -10, -8, -6, -4, -2)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -5, -8, -10, -15, -12, -9, -6, -3)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -7, -10, -14, -20, -16, -12, -8, -4)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -10, -15, -20, -30, -24, -18, -12, -6)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -8, -12, -16, -24, -20, -15, -10, -5)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -6, -9, -12, -18, -15, -12, -8, -4)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -4, -6, -8, -12, -10, -8, -6, -3)
l = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, -3, -4, -6, -5, -4, -3, -2)
```

2 Coxeter diagram and roots for $(18, 0, 0)_1 = U + E_8^2$, coming from $U + E_8^2 + A_1$




v_1
v_2
v_3
v_4
v_5
v_{16}
v_{17}
v_{18}
v_6
v_{22}
v_{15}
v_{21}
v_7
v_{20}
v_{19}
v_{14}
v_8
v_{13}
v_{12}
v_{11}
v_{10}
v_9


w_9
w_8
w_{10}
w_7
w_{11}
w_6
w_{12}
w_5
w_{13}
w_4
w_{18}
w_{19}
w_{14}
w_3
w_2
w_1
w_{17}
w_{16}
w_{15}



[33]:

```
# Root vectors for (18, 0, 0), roots taken from above, w_i are according to
↪numerical labeling above

w1 = a1
w2 = a3
w3 = a4
w4 = a5
w5 = a6
w6 = a7
w7 = a8
w8 = a8b + e
w9 = f-e
```

```
w10 = a8pb + e
w11 = a8p
w12 = a7p
w13 = a6p
w14 = a5p
w15 = a4p
w16 = a3p
w17 = a1p
w18 = a2
w19 = a2p

W = [w1, w2, w3, w4, w5, w6, w7, w8, w9, w10, w11, w12, w13, w14, w15, w16,␣
 ↪w17, w18, w19]

MW = root_intersection_matrix(W, labels = [f"$w_{ {r + 1} }$" for r in range(␣
 ↪len(W) )], bil_form=dot2)
```

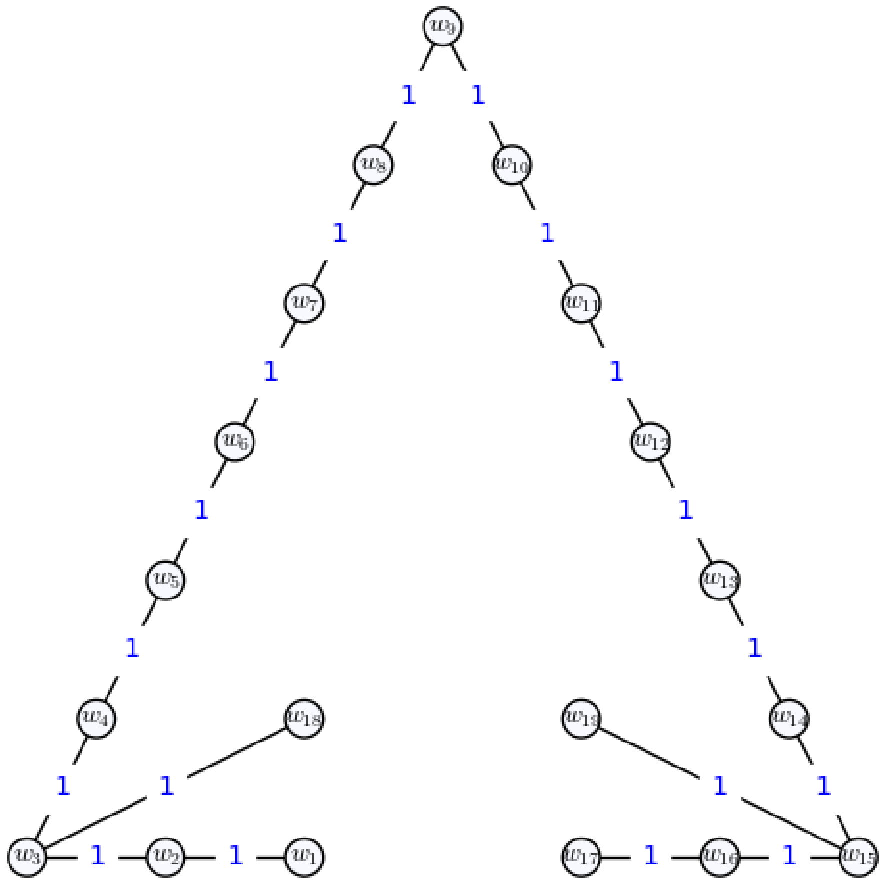
Diagonal entries/square norms:

$[-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2]$

<IPython.core.display.HTML object>

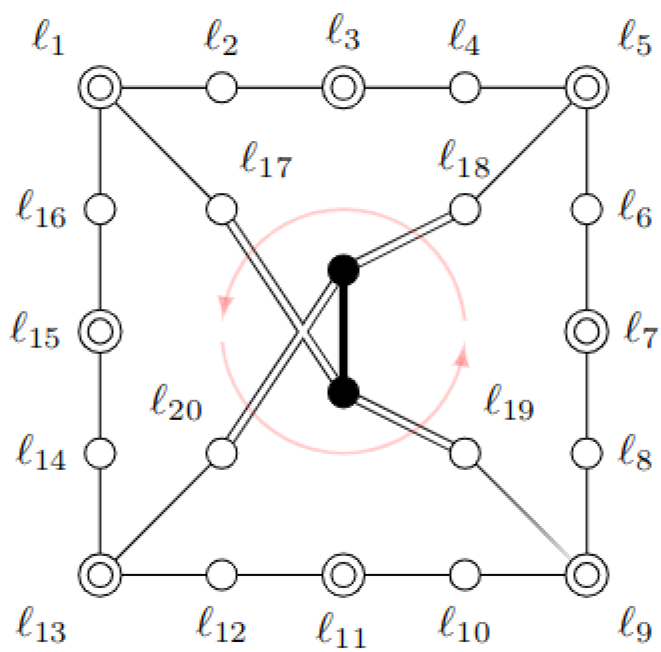
[34]:
```
G = Coxeter_Diagram(MW)
plot_coxeter_diagram(
    G,
    v_labels = [f"$w_{ {i + 1} }$" for i in range( 19 )],
    pos = {
        0: [-4, 0],
        1: [-8, 0],
        2: [-12, 0],
        3: [-10, 4],
        4: [-8, 8],
        5: [-6, 12],
        6: [-4, 16],
        7: [-2, 20],
        8: [0, 24],
        9: [2, 20],
        10: [4, 16],
        11: [6, 12],
        12: [8, 8],
        13: [10, 4],
        14: [12, 0],
        15: [8, 0],
        16: [4, 0],
        17: [-4, 4],
        18: [4, 4]
    }
```

```
)
```



w_9
w_8
w_{10}
w_7
w_{11}
w_6
w_{12}
w_5
w_{13}
w_4
w_{18}
w_{19}
w_{14}
w_3
w_2
w_1
w_{17}
w_{16}
w_{15}
1

3 Sterk 1



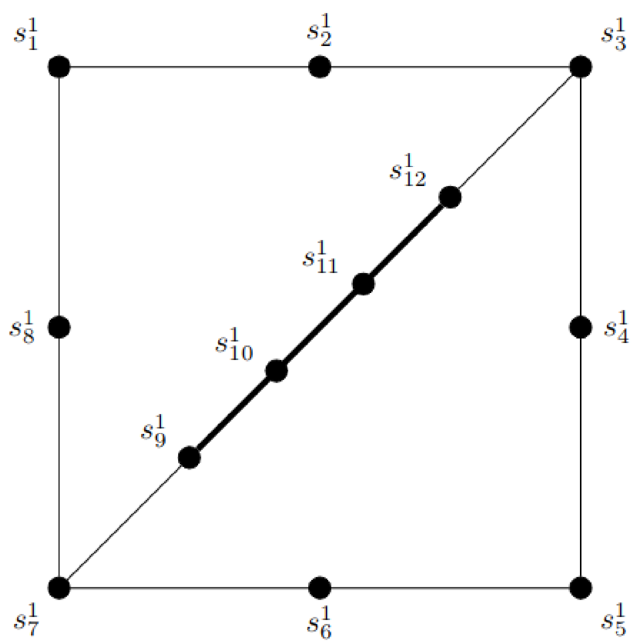


FIGURE 16. Sterk's Coxeter diagram for the 0-cusps #1 corresponding to e with $e^{\perp}/e \cong U(2) \oplus E_8(2)$.

[35]:
```
# Sterk 1

display(r.cycle_tuples(singletons=True))

s1_1 = v3 + v11
s1_2 = v4 + v12
s1_3 = v5 + v13
s1_4 = v6 + v14
s1_5 = v7 + v15
s1_6 = v8 + v16
s1_7 = v9 + v1
s1_8 = v10 + v2
s1_9 = v17 + v19
s1_10 = v21
s1_11 = v22
s1_12 = v18 + v20

# S1 = [s1_1, s1_2, s1_3, s1_4, s1_5, s1_6, s1_7, s1_8, s1_9, s1_10, s1_11,␣
↪s1_12]
# MS1 = root_intersection_matrix(S1, labels = [f"$s^1_{ {r + 1} }$" for r in␣
↪range( len(S1) )], bil_form=dot)
```

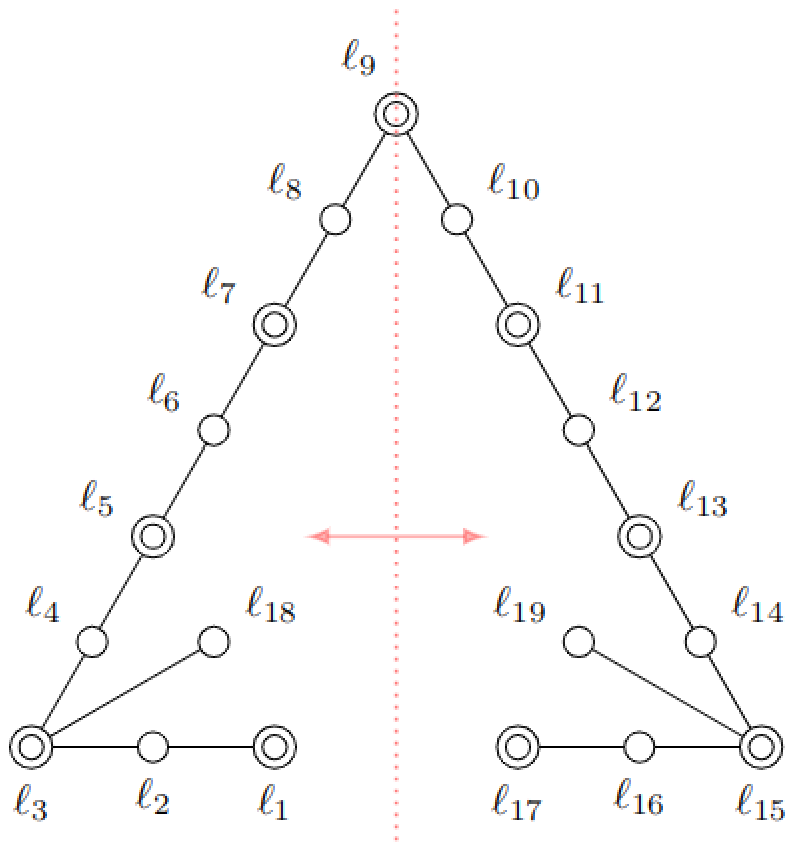
$[(1,9), (2,10), (3,11), (4,12), (5,13), (6,14), (7,15), (8,16), (17,19), (18,20), (21), (22)]$

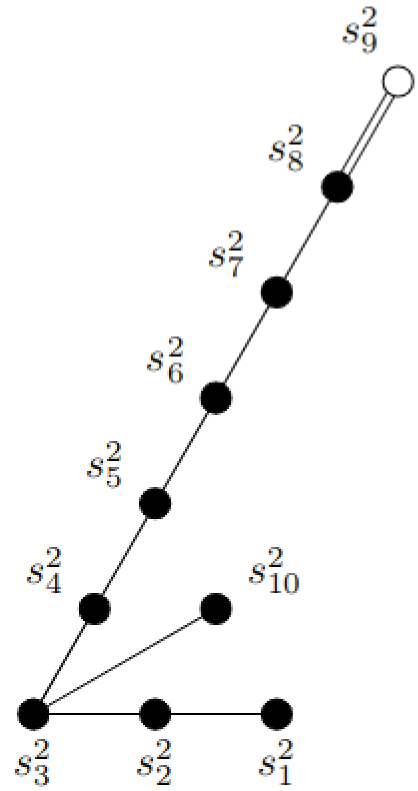
[36]:
```
G = Coxeter_Diagram(MS1)
plot_coxeter_diagram(
    G,
    v_labels = [f"$s^1_{ {i + 1} }$" for i in range( 22 )],
    pos = {
        0: [0, 0],
        1: [4, 0],
        2: [8, 0],
        3: [8, -4],
        4: [8, -8],
        5: [4, -8],
        6: [0, -8],
        7: [0, -4],
        8: [0, -8],
        9: [2, -6],
        10: [4, -4],
        11: [6, -2]
    }
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[36], line 1
----> 1 G = Coxeter_Diagram(MS1)
```

```
plot_coxeter_diagram(
    G,
    v_labels = [f"$s^1_{ {i + Integer(1)} }$" for i in range(␣
↪Integer(22) )],
(…)
    }
)

NameError: name 'MS1' is not defined
```

4 Sterk 2





[37]:
```
# Sterk 2

s2_1 = w1 + w17
s2_2 = w2 + w16
s2_3 = w3 + w15
s2_4 = w4 + w14
s2_5 = w5 + w13
s2_6 = w6 + w12
s2_7 = w7 + w11
s2_8 = w8 + w10
s2_9 = w9
s2_10 = w18 + w19

S2 = [s2_1, s2_2, s2_3, s2_4, s2_5, s2_6, s2_7, s2_8, s2_9, s2_10]
MS2 = root_intersection_matrix(S2, labels = [f"$s^2_{ {r + 1} }$" for r in␣
↪range( len(S2) )], bil_form=dot2 )
```

Diagonal entries/square norms:

$[-4, -4, -4, -4, -4, -4, -4, -4, -2, -4]$

<IPython.core.display.HTML object>

[38]:
```
from sage.modules.free_module_integer import IntegerLattice

n = len(S2)
M = zero_matrix(QQ, n)
nums = Set(range(n))
for i in range(n):
    for j in range(n):
        M[i, j] = dot( S2[i], S2[j] )

LS2 = IntegralLattice(M)
LS2p = IntegerLattice(M)

display( LS2.signature_pair() )
display( LS2.is_even() )
display( LS2p.is_unimodular() )
M.rational_form()
```

$(1, 9)$

True

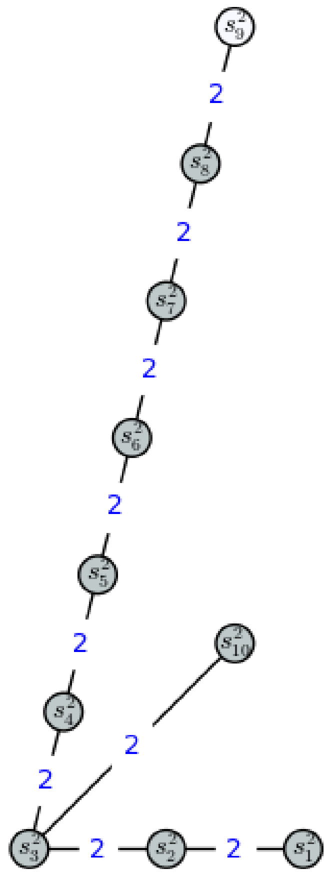
False

[38]:

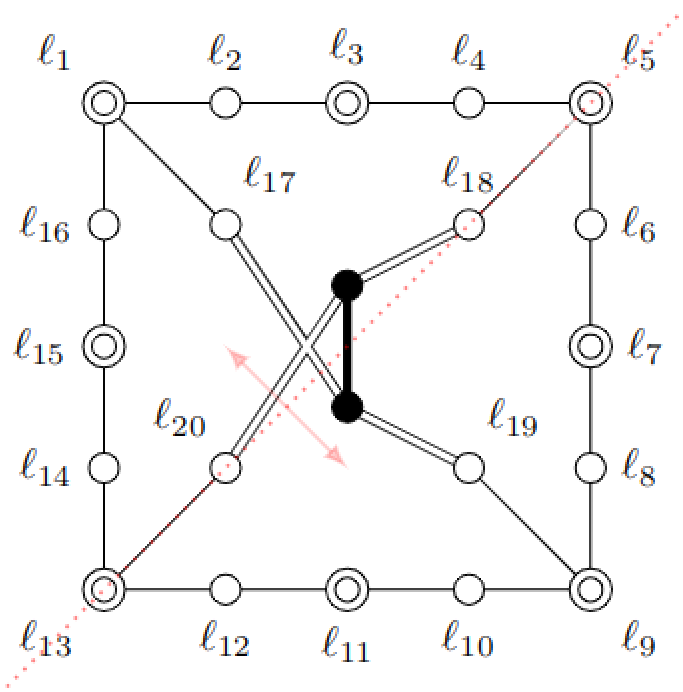
$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4096 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 73728 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 46080 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -121856 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -179648 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -104448 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -33024 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -6144 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -672 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -40 \end{pmatrix}$$

[39]:
```
G = Coxeter_Diagram(MS2)
plot_coxeter_diagram(
    G,
    v_labels = [f"$s^2_{ {i + 1} }$" for i in range( 22 )],
    pos = {
        0: [0, 0],
        1: [-4, 0],
        2: [-8, 0],
        3: [-7, 4],
        4: [-6, 8],
        5: [-5, 12],
        6: [-4, 16],
        7: [-3, 20],
        8: [-2, 24],
```

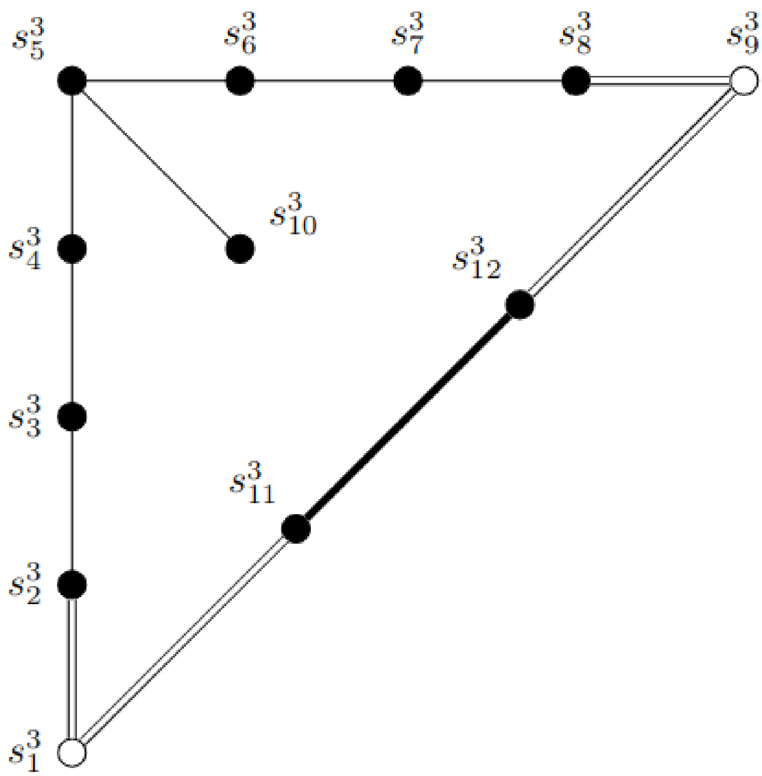
```
        9: [-2, 6]
    }
)
```



4.1 Sterk 3



ℓ_1
ℓ_2
ℓ_3
ℓ_4
ℓ_5
ℓ_{17}
ℓ_{18}
ℓ_{16}
ℓ_6
ℓ_{15}
ℓ_7
ℓ_{20}
ℓ_{19}
ℓ_{14}
ℓ_8
ℓ_{13}
ℓ_{12}
ℓ_{11}
ℓ_{10}
ℓ_9



s_5^3
s_6^3
s_7^3
s_8^3
s_9^3
s_{10}^3
s_4^3
s_{12}^3
s_3^3
s_{11}^3
s_2^3
s_1^3

[40]:
```
# Sterk 3

display(d.cycle_tuples(singletons=True))

s3_1 = v13
s3_2 = v14 + v12
s3_3 = v15 + v11
s3_4 = v16 + v10
s3_5 = v1 + v9
s3_6 = v2 + v8
s3_7 = v3 + v7
s3_8 = v4 + v6
s3_9 = v5
s3_10 = v17 + v19
s3_11 = v20
s3_12 = v18
s3_13 = v21
s3_14 = v22

S3 = [s3_1, s3_2, s3_3, s3_4, s3_5, s3_6, s3_7, s3_8, s3_9, s3_10, s3_11,␣
↪s3_12, s3_13, s3_14]

MS3 = root_intersection_matrix(S3, labels = [f"$s^2_{ {r + 1} }$" for r in␣
↪range( len(S3) )], bil_form=dot )
```

$[(1,9), (2,8), (3,7), (4,6), (5), (10,16), (11,15), (12,14), (13), (17,19), (18), (20), (21), (22)]$

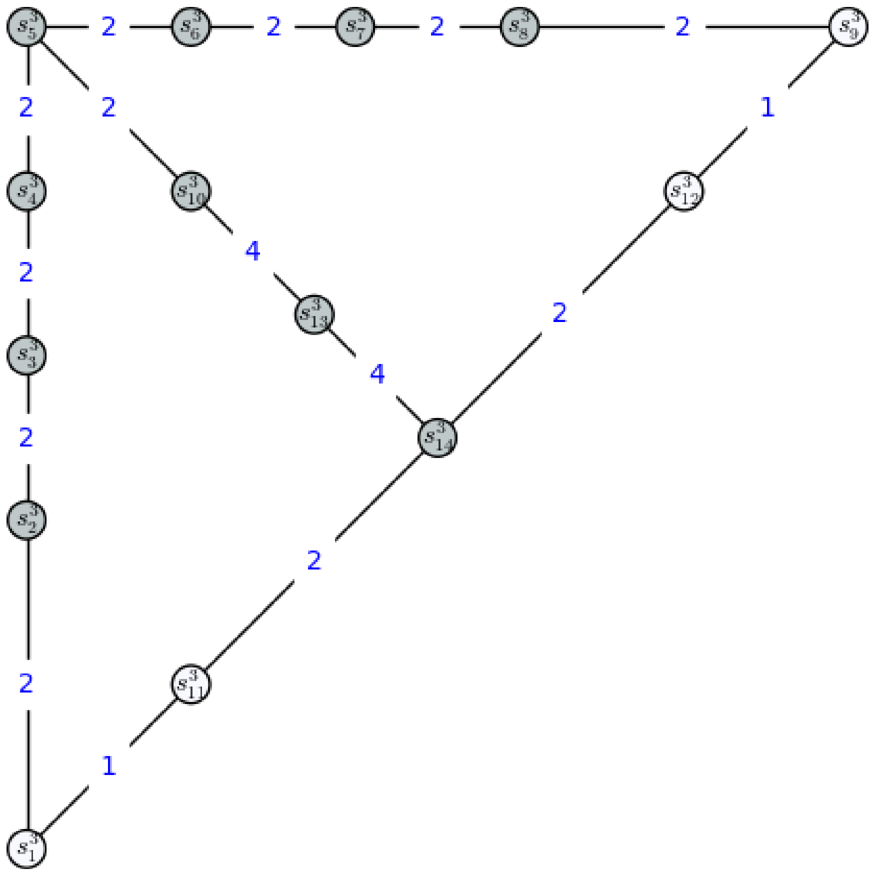
Diagonal entries/square norms:

$[-2, -4, -4, -4, -4, -4, -4, -4, -2, -4, -2, -2, -4, -4]$

<IPython.core.display.HTML object>

[41]:
```
G = Coxeter_Diagram(MS3)
pos_dict = {
        0: [0, -4],
        1: [0, 4],
        2: [0, 8],
        3: [0, 12],
        4: [0, 16],
        5: [4, 16],
        6: [8, 16],
        7: [12, 16],
        8: [20, 16],
        9: [4, 12],
        10: [4, 0],
        11: [16, 12],
        12: [7, 9],
        13: [10, 6],
    }
plot_coxeter_diagram(
```

```
    G,
    v_labels = [f"$s^3_{ {i + 1} }$" for i in range( len(S3) )],
    pos = pos_dict
)

pos_dict
```

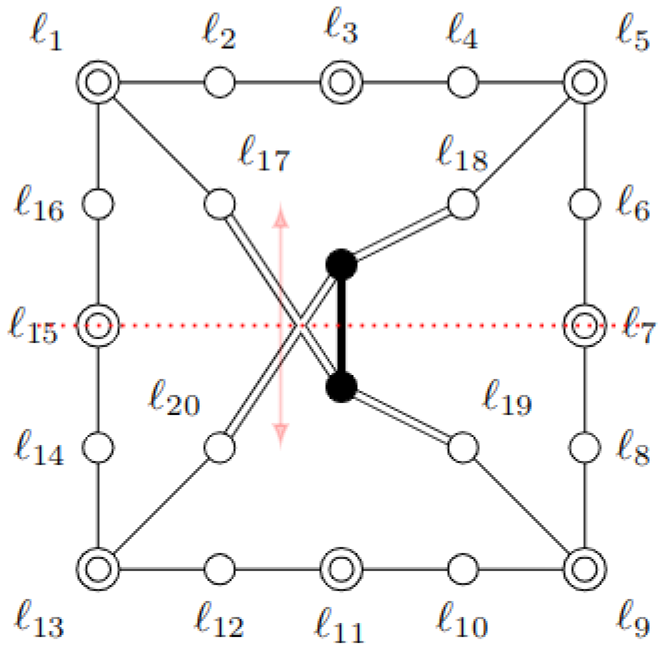


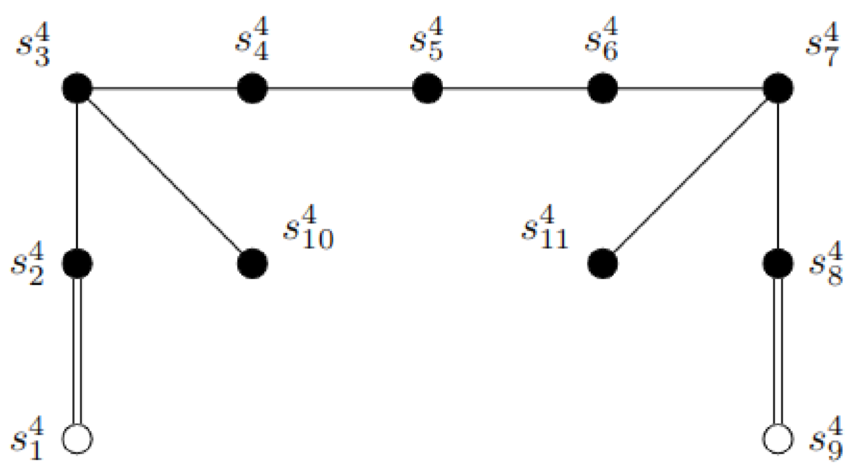
[41]: $\{0:[0,-4], 1:[0,4], 2:[0,8], 3:[0,12], 4:[0,16], 5:[4,16], 6:[8,16], 7:[12,16], 8:[20,16], 9:[4,12], 10:[4,0],$

[42]:
```
MS3.rank()
```

[42]: 11

5 Sterk 4





[43]:
```
# Sterk 4

display(v.cycle_tuples(singletons=True))

s4_1 = v15
s4_2 = v16 + v14
s4_3 = v1 + v13
```

```
s4_4 = v2 + v12
s4_5 = v3 + v11
s4_6 = v4 + v10
s4_7 = v5 + v9
s4_8 = v6 + v8
s4_9 = v7
s4_10 = v17 + v20
s4_11 = v18 + v19
s4_12 = v22 + v21

# Although s412 is an invariant vector, it is not a root:
# from IPython.display import Math
# Math('(s^4_{12})^2=' + str( nm(s4_12)))

S4 = [s4_1, s4_2, s4_3, s4_4, s4_5, s4_6, s4_7, s4_8, s4_9, s4_10, s4_11]
MS4 = root_intersection_matrix(S4, labels = [f"$s^4_{ {r + 1} }$" for r in␣
↪range( len(S4) )], bil_form=dot)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[43], line 3
      1 # Sterk 4
----> 3 display(v.cycle_tuples(singletons=True))
      5 s4_1 = v15
      6 s4_2 = v16 + v14

File /usr/lib/python3.11/site-packages/sage/structure/element.pyx:488, in sage.
↪structure.element.Element.__getattr__ (build/cythonized/sage/structure/element.
↪c:4860)()
    486         AttributeError: 'LeftZeroSemigroup_with_category.element_class'␣
↪object has no attribute 'blah_blah'
    487     """
--> 488     return self.getattr_from_category(name)
    489
    490 cdef getattr_from_category(self, name):

File /usr/lib/python3.11/site-packages/sage/structure/element.pyx:501, in sage.
↪structure.element.Element.getattr_from_category (build/cythonized/sage/
↪structure/element.c:4972)()
    499     else:
    500         cls = P._abstract_element_class
--> 501     return getattr_from_other_class(self, cls, name)
    502
    503 def __dir__(self):

File /usr/lib/python3.11/site-packages/sage/cpython/getattr.pyx:362, in sage.
↪cpython.getattr.getattr_from_other_class (build/cythonized/sage/cpython/
↪getattr.c:2786)()
```

```
    360     dummy_error_message.cls = type(self)
    361     dummy_error_message.name = name
--> 362     raise AttributeError(dummy_error_message)
    363 attribute = <object>attr
    364 # Check for a descriptor (__get__ in Python)

AttributeError: 'sage.modules.vector_integer_dense.Vector_integer_dense' object
 ↪has no attribute 'cycle_tuples'
```

[44]:

```
G = Coxeter_Diagram(MS4)
plot_coxeter_diagram(
    G,
    v_labels = [f"$s^4_{ {i + 1} }$" for i in range( 11 )],
    pos = {
        0: [0, 0],
        1: [0, 4],
        2: [0, 8],
        3: [4, 8],
        4: [8, 8],
        5: [12, 8],
        6: [16, 8],
        7: [16, 4],
        8: [16, 0],
        9: [4, 4],
        10: [12, 4]
}
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[44], line 1
----> 1 G = Coxeter_Diagram(MS4)
      2 plot_coxeter_diagram(
      3     G,
      4     v_labels = [f"$s^4_{ {i + Integer(1)} }$" for i in range(
 ↪Integer(11) )],
   (…)
     17 }
     18 )

NameError: name 'MS4' is not defined
```

6 Sterk 5

Involution: On the boundary:

$e_{2i+1} \longrightarrow -e_{2i+1}$ $\qquad (1+\iota)e_{2i+1} = 0$

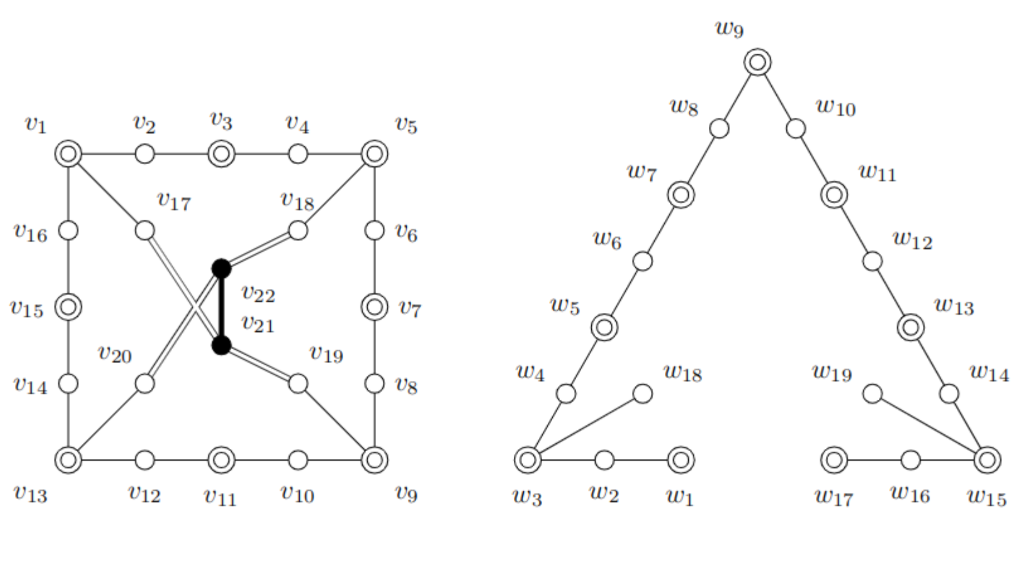
$e_{2i} \longrightarrow e_{2i-1} + e_{2i} + e_{2i+1}$ $\qquad (1+\iota)e_{2i} = e_{2i-1} + 2e_{2i} + e_{2i+1}$

In the middle: $e_k \to e_k$.

This is a composition of 8 reflections in the vecs e_{2i+1}.

The original vectors e_i and the reflected vecs ιe_i lie in two different chambers which share a 10-dim'l face.

In the other 4 cases e_i and ιe_i belong to the same Weyl chamber.



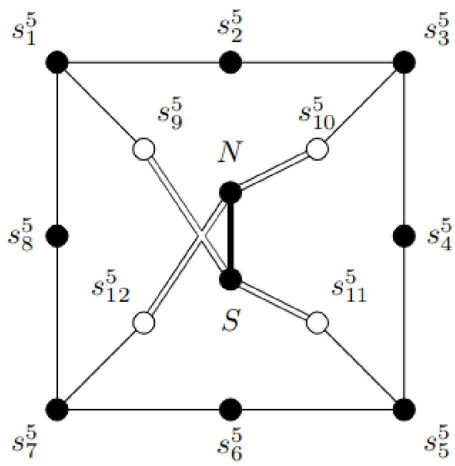


FIGURE 20. Sterk's Coxeter diagram for the 0-cusps #5 corresponding to $v := 2e + 2f + \bar{\alpha}_1$ with $v^{\perp}/v \cong U \oplus E_8(2)$.

[45]:

```
# Sterk 5

s5_1 = v1 + v2 + v3
s5_2 = v3 + v4 + v5
s5_3 = v5 + v6 + v7
s5_4 = v7 + v8 + v9
s5_5 = v9 + v10 + v11
s5_6 = v11 + v12 + v13
s5_7 = v13 + v14 + v15
s5_8 = v15 + v16 + v1
s5_9 = v9
s5_10 = v10
s5_11 = v11
s5_12 = v12

S5 = [s5_1, s5_2, s5_3, s5_4, s5_5, s5_6, s5_7, s5_8, s5_9, s5_10, s5_11, s5_12]
MS5 = root_intersection_matrix(S5, labels = [f"$s^5_{ {r + 1} }$" for r in␣
↪range( len(S5) )], bil_form=dot)

## ISSUE: this is not the right folded diagram....
```

Diagonal entries/square norms:

$[-2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2]$

<IPython.core.display.HTML object>

[46]:
```
# G = Coxeter_Diagram(MS5)
# plot_coxeter_diagram(G, v_labels = [f"$s^5_{ {i + 1} }$" for i in range( 22␣
↪)] )
```

[ ]:
```
# Maybe I messed up the parity. Let's try rotating the outer cycle by one vertex

s5_1 = v6 + v1 + v2
s5_2 = v2 + v3 + v4
s5_3 = v4 + v5 + v6
s5_4 = v6 + v7 + v8
s5_5 = v8 + v9 + v10
s5_6 = v10 + v11 + v12
s5_7 = v12 + v13 + v14
s5_8 = v14 + v15 + v16
s5_9 = v9
s5_10 = v10
s5_11 = v11
s5_12 = v12

S5 = [s5_1, s5_2, s5_3, s5_4, s5_5, s5_6, s5_7, s5_8, s5_9, s5_10, s5_11, s5_12]
MS5 = root_intersection_matrix(S5, labels = [f"$s^5_{ {r + 1} }$" for r in␣
↪range( len(S5) )], bil_form=dot)

# Nope....still issues with negative intersections..
```

[ ]:
```
```