Hasse Diagrams

Naman Taggar | https://github.com/zplus11/hasse

Definitions

In[1]:=
```
niceblue = RGBColor["#4A84FF"];
nicered = RGBColor["#FF634A"];
pairQ[{_, _}] := True (* defining the pairs *)
pairQ[{___}] := False
relationQ[{___?pairQ}] := True (* defining the relation *)
relationQ[{___}] := False
```

Defining a divisibility relation:

In[7]:=
```
divRel[A : {__Integer}] := Select[Tuples[A, 2], Divisible[#[[2]], #[[1]]] &]
```

Similar relation $a \leq b$

In[8]:=
```
ltRel[n_Integer] := Select[Tuples[Range[n], 2], #[[1]] ≤ #[[2]] &]
ltRel[5]
```

Out[9]=
```
{{1, 1}, {1, 2}, {1, 3}, {1, 4}, {1, 5}, {2, 2}, {2, 3},
 {2, 4}, {2, 5}, {3, 3}, {3, 4}, {3, 5}, {4, 4}, {4, 5}, {5, 5}}
```

Relation with $a \bmod n = b \bmod n$ where entries *a*, *b* come from input list, and $n \in \mathbb{Z}$ is also input.

In[10]:=
```
modnRel[list_List, n_Integer] :=
 Select[Tuples[list, 2], Mod[#[[1]], n ] == Mod[#[[2]], n] &]
modnRel[{2, 7, 4, 6, 0, 9, 8, 3}, 5]
```

Out[11]=
```
{{2, 2}, {2, 7}, {7, 2}, {7, 7}, {4, 4}, {4, 9},
 {6, 6}, {0, 0}, {9, 4}, {9, 9}, {8, 8}, {8, 3}, {3, 8}, {3, 3}}
```

Relation with set inclusion

In[12]:=
```
subsetRel[list_List] := Select[Tuples[Subsets[list], 2], SubsetQ[#[[2]], #[[1]]] &]
subsetRel[{"a", "b", "c"}]
```

Out[13]=
```
{{{}, {}}, {{}, {a}}, {{}, {b}}, {{}, {c}}, {{}, {a, b}}, {{}, {a, c}}, {{}, {b, c}},
 {{}, {a, b, c}}, {{a}, {a}}, {{a}, {a, b}}, {{a}, {a, c}}, {{a}, {a, b, c}},
 {{b}, {b}}, {{b}, {a, b}}, {{b}, {b, c}}, {{b}, {a, b, c}}, {{c}, {c}}, {{c}, {a, c}},
 {{c}, {b, c}}, {{c}, {a, b, c}}, {{a, b}, {a, b}}, {{a, b}, {a, b, c}}, {{a, c}, {a, c}},
 {{a, c}, {a, b, c}}, {{b, c}, {b, c}}, {{b, c}, {a, b, c}}, {{a, b, c}, {a, b, c}}}
```

Inverse Relation

In[14]:=
```
inverseRel[R_?relationQ] := Reverse[R, 2]
```

Finding domain of relation:

In[15]:=
```
findDomain[R_?relationQ] := Union[Flatten[R, 1]]
findDomain[divRel[Range[10]]]
```

Out[16]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

Checking types of relations

R is said to be reflexive if $(a, a) \in R \,\forall\, a \in S$.

In[17]:=
```
reflexiveQ[R_?relationQ] := Module[{a, domain}, domain = findDomain[R];
  Catch[Do[If[! MemberQ[R, {a, a}], Throw[Flase]], {a, domain}];
   Throw[True]]]
```

R is said to be symmetric if $(a, b) \in R \implies (b, a) \in R \,\forall\, a, b \in S$.

In[18]:=
```
symmetricQ[R_?relationQ] :=
 Module[{u}, Catch[Do[If[! MemberQ[R, Reverse[u]], Throw[False]], {u, R}];
   Throw[True]]]
```

R is said to be anti-symmetric if $(a, b), (b, a) \in R \implies a = b \,\forall\, a, b \in S$.

In[19]:=
```
antisymmetricQ[R_?relationQ] := Module[{u},
  Catch[Do[If[MemberQ[R, Reverse[u]] && u[[1]] ≠ u[[2]], Throw[False]], {u, R}];
   Throw[True]]]
```

R is said to be transitive if $(a, b), (b, c) \in R \implies (a, c) \in R \,\forall\, a, b, c \in S$.

In[20]:=
```
transitiveQ[R_?relationQ] := Module[{domain, a, b, c}, domain = findDomain[R];
  Catch[Do[If[MemberQ[R, {a, b}] && MemberQ[R, {b, c}] && ! MemberQ[R, {a, c}],
     Throw[False]], {a, domain}, {b, domain}, {c, domain}];
   Throw[
    True]]]
```

In[21]:=
```
checkRelations[x_?relationQ] := Text@Grid[Table[{k, k[x]},
    {k, {reflexiveQ, symmetricQ, antisymmetricQ, transitiveQ}}], Dividers → niceblue]
```

In[22]:=
```
checkRelations[{{1, 1}, {2, 2}, {3, 3}, {1, 2}, {2, 1}, {2, 3}, {3, 2}, {1, 3}, {3, 1}}]
```

Out[22]=

reflexiveQ	True
symmetricQ	True
antisymmetricQ	False
transitiveQ	True

In[23]:=
```
checkRelations[{{1, 1}}]
```

Out[23]=

reflexiveQ	True
symmetricQ	True
antisymmetricQ	True
transitiveQ	True

In[24]:=
```
checkRelations[divRel[Range[10]]]
```

Out[24]=

reflexiveQ	True
symmetricQ	False
antisymmetricQ	True
transitiveQ	True

In[25]:=
```
checkRelations[subsetRel[{1, 2, 3}]]
```

Out[25]=

reflexiveQ	True
symmetricQ	False
antisymmetricQ	True
transitiveQ	True

Checking if a relation is POSET:

In[26]:=
```
partialOrderQ[R_?relationQ] := reflexiveQ[R] && antisymmetricQ[R] && transitiveQ[R]
```

In[27]:=
```
div2to17 = divRel[Range[2, 17]]
partialOrderQ[div2to17]
```

Out[27]=
```
{{2, 2}, {2, 4}, {2, 6}, {2, 8}, {2, 10}, {2, 12}, {2, 14}, {2, 16}, {3, 3},
 {3, 6}, {3, 9}, {3, 12}, {3, 15}, {4, 4}, {4, 8}, {4, 12}, {4, 16}, {5, 5},
 {5, 10}, {5, 15}, {6, 6}, {6, 12}, {7, 7}, {7, 14}, {8, 8}, {8, 16}, {9, 9},
 {10, 10}, {11, 11}, {12, 12}, {13, 13}, {14, 14}, {15, 15}, {16, 16}, {17, 17}}
```

Out[28]= True

Constructing duals:

In[29]:=
```
dual[R_?partialOrderQ] := Reverse[R, 2]
```

In[30]:=
```
dual[divRel[Range[4]]]
```

Out[30]=
```
{{1, 1}, {2, 1}, {3, 1}, {4, 1}, {2, 2}, {4, 2}, {3, 3}, {4, 4}}
```

Other settings

Covering relations

If P is a poset under the relation $\leq$ and x, y are two elements in P, then x is said to be covered by y (or y covers x), and we write $x \longrightarrow\!\!< y$, if $x < y$ and $x \leq z < y \implies x = z$. Below we will create a function to find the elements that cover each other, and hence the set of all covering relations.

In[31]:=
```
coversQ[R_?partialOrderQ, {x_, y_}] :=
 Module[{z, checkSet}, Catch[If[x == y || ! MemberQ[R, {x, y}], Throw[False]];
   checkSet = Complement[findDomain[R], {x, y}];
   Do[If[MemberQ[R, {x, z}] && MemberQ[R, {z, y}], Throw[False]], {z, checkSet}];
   Throw[True]]]
```

In[32]:=
```
coveringRelation[R_?partialOrderQ] := Select[R, coversQ[R, #] &]
```

In[33]:=
```
coveringRelation[ltRel[4]]
```

Out[33]=
```
{{1, 2}, {2, 3}, {3, 4}}
```

In[34]:=
```
divisorLattice[n_Integer] := divRel[Divisors[n]]
```

In[35]:=
```
coveringRelation[divisorLattice[30]]
```

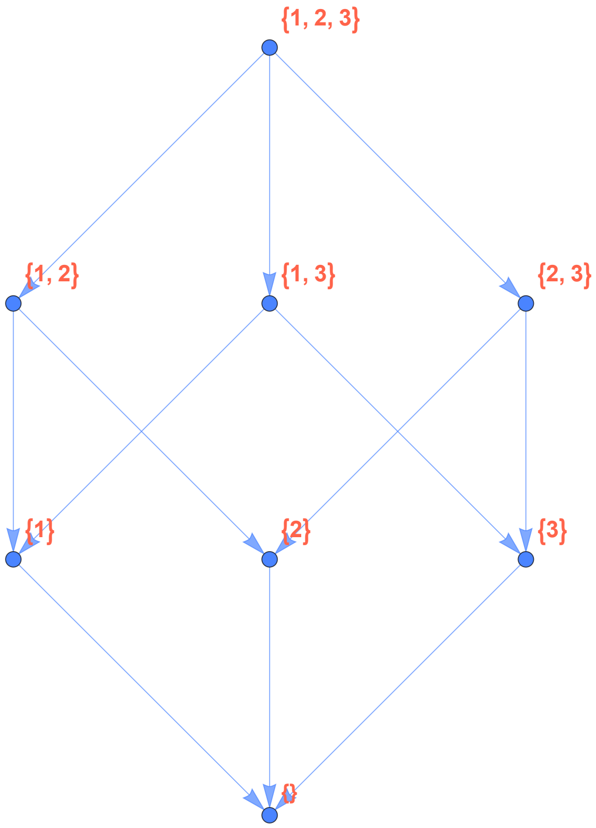
Out[35]=
```
{{1, 2}, {1, 3}, {1, 5}, {2, 6}, {2, 10}, {3, 6},
 {3, 15}, {5, 10}, {5, 15}, {6, 30}, {10, 30}, {15, 30}}
```

Hasse Diagrams

In[36]:=
```
hasseDiagram[R_?partialOrderQ] := Module[{edges},
  edges = coveringRelation[R] /. {a_, b_} → Rule[b, a];
  LayeredGraphPlot[edges, VertexLabels → Automatic,
   VertexLabelStyle → Directive[nicered, Bold, 12], PlotStyle → niceblue]]
```

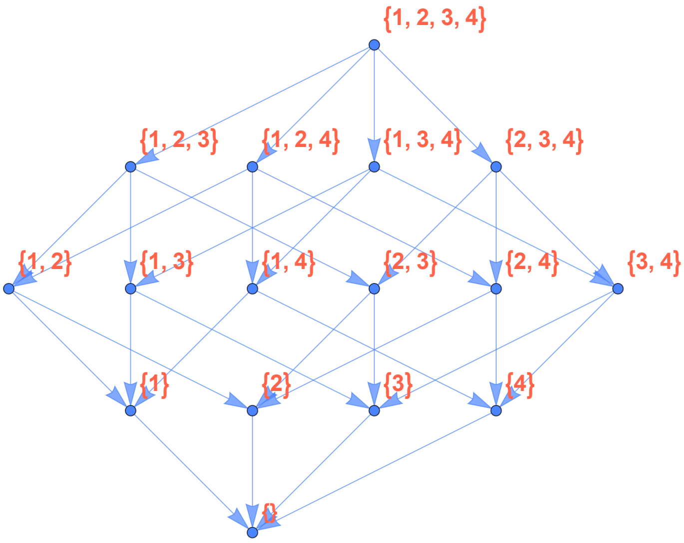
In[37]:=
```
hasseDiagram[subsetRel[Range[3]]]
```

Out[37]=



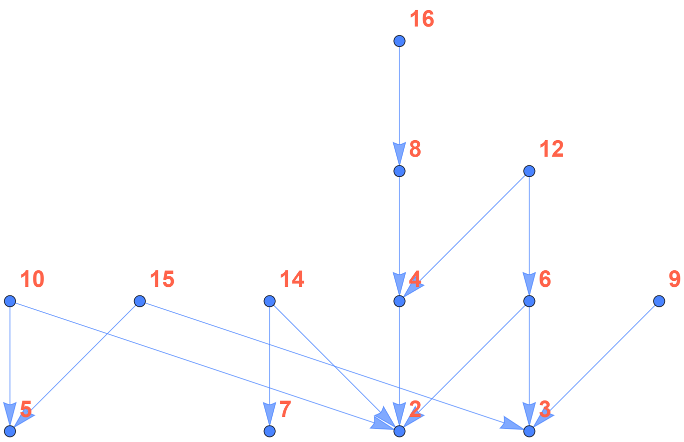
In[38]:= `hasseDiagram[subsetRel[Range[4]]]`

Out[38]=



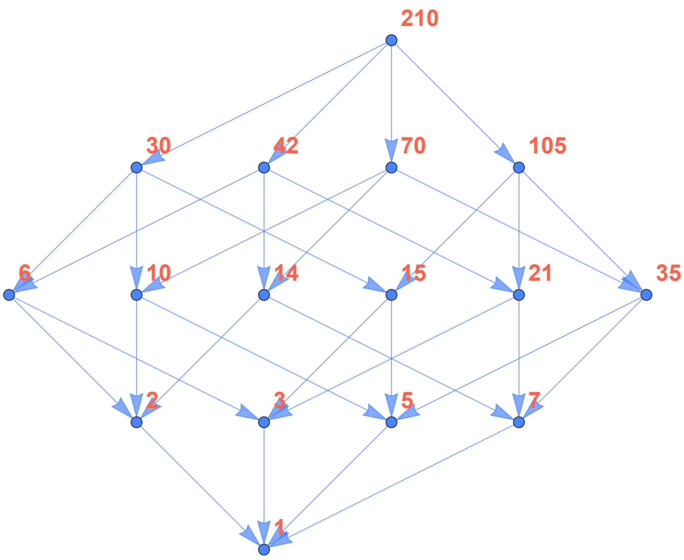
In[39]:= `hasseDiagram[div2to17]`

Out[39]=



In[40]:= `hasseDiagram[divisorLattice[2 * 3 * 5 * 7]]`

Out[40]=



Chains

Chains are defined as a relation P on $\{1, 2, \ldots, n\}$ such that $(a, b) \in P$ if and only if $a \leq b$ or $b \leq a$.

In[41]:=
```
chainRel[n_Integer] := Select[Tuples[Range[n], 2], #⟦1⟧ ≤ #⟦2⟧ &] (* wlog *)
```

In[42]:=
```
hasseDiagram[chainRel[4]]
```

Out[42]=

Anti-chain

In anti-chains, $(a, b) \in P$ if and only if $a = b$.

In[43]:=
```
antichainRel[n_Integer] := Select[Tuples[Range[n], 2], #⟦1⟧ == #⟦2⟧ &]
antichainRel[3]
```

Out[44]=
```
{{1, 1}, {2, 2}, {3, 3}}
```

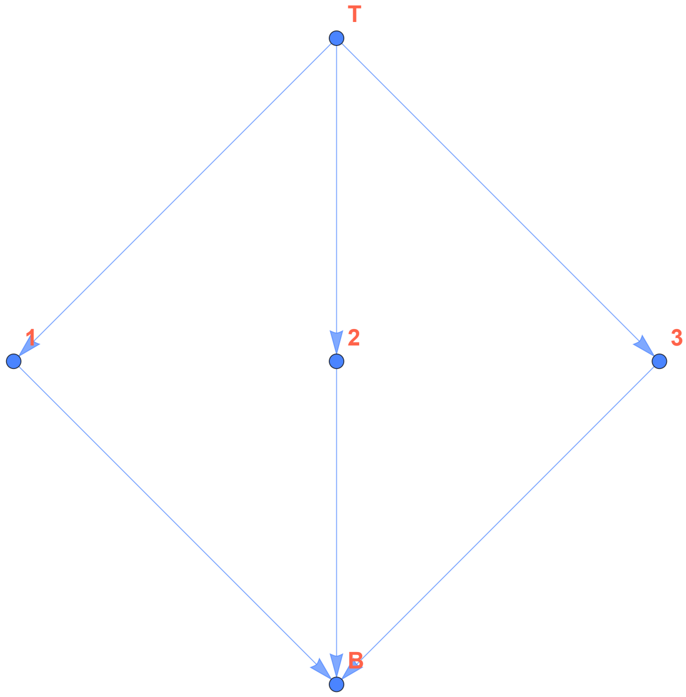
M_n

M_n is the linear sum of 1, anti chain(n), and 1.

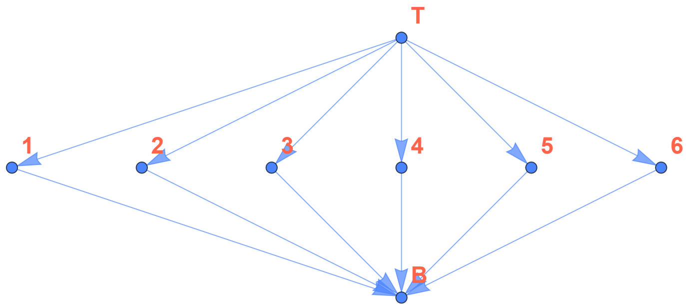
In[45]:=
```
mnRel[n_Integer] := Select[Tuples[Union[Range[n], {"T", "B"}], 2],
  #⟦1⟧ == #⟦2⟧ || #⟦1⟧ == "B" || #⟦2⟧ == "T" &]
mnRel[
 3]
```

Out[46]=
```
{{1, 1}, {1, T}, {2, 2}, {2, T}, {3, 3},
 {3, T}, {B, 1}, {B, 2}, {B, 3}, {B, B}, {B, T}, {T, T}}
```

In[47]:= `hasseDiagram[mnRel[3]]`

Out[47]=


In[48]:= `hasseDiagram[mnRel[6]]`

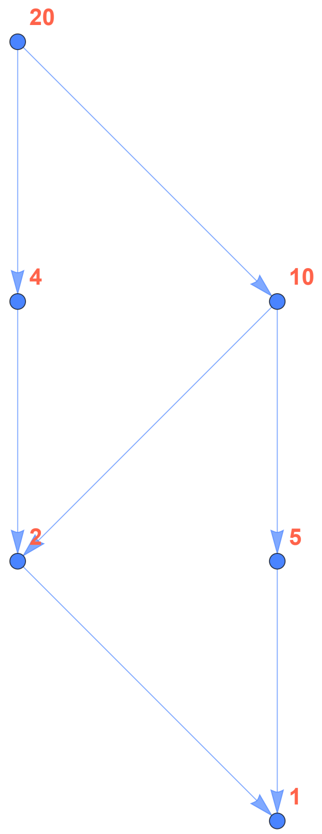
Out[48]=


Duals

The dual P^{∂} of P is a poset such that $x \leq y$ in P^{∂} if and only if $y \leq x$ in P.

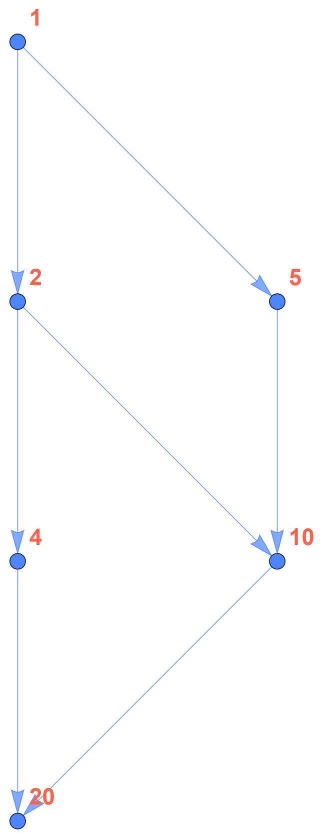
In[49]:= `hasseDiagram[divisorLattice[20]]`

Out[49]=



In[50]:= `hasseDiagram[dual[divisorLattice[20]]]`

Out[50]=



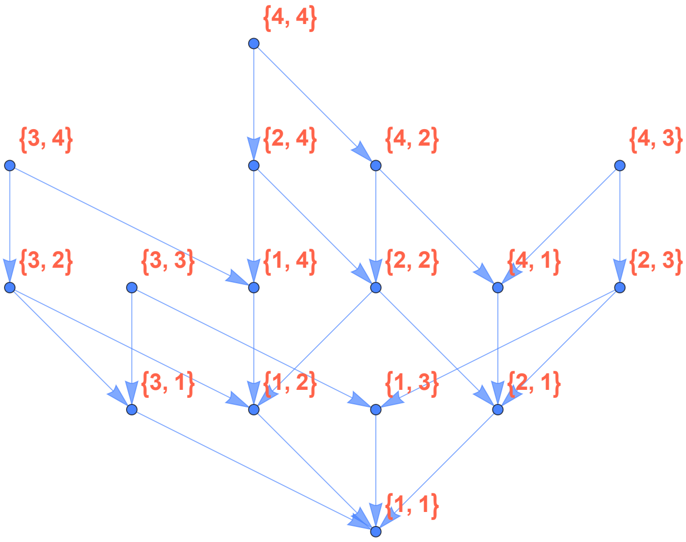
One step further: Product Posets

The cartesian product $P_1 \times P_2 \times \cdots \times P_n$ of posets can be a poset by defining $(x_1, x_2, \ldots, x_n) \leq (y_1, y_2, \ldots, y_n)$ if and only if $x_i \leq y_i \;\forall\; i = 1, 2, \ldots, n$.

In[51]:=
```
productQ[{l___?partialOrderQ}] := Select[Tuples[Tuples[Map[findDomain, {l}]], 2],
  And @@ Table[MemberQ[{l}〚k〛, {#〚1, k〛, #〚2, k〛}], {k, 1, Length[{l}]}] &]
```

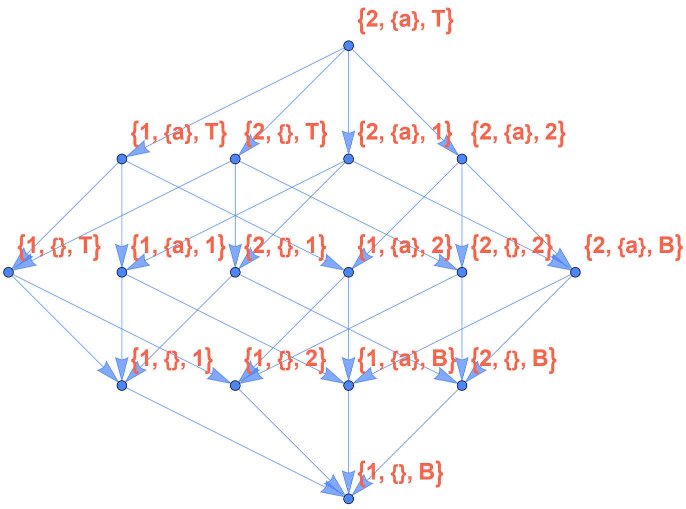
Product lattice of ({1, 2, 3, 4}, |) × ({1, 2, 3, 4}, |)

In[52]:=
```
hasseDiagram[productQ[{divRel[Range[4]], divRel[Range[4]]}]]
```

Out[52]=


Product lattice of ({1, 2}, |) × ({a}, ⊆) × M_2

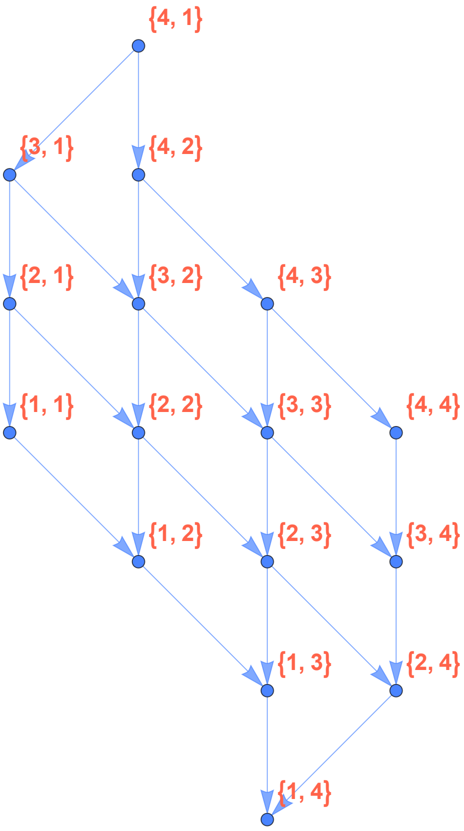
In[53]:=
```
hasseDiagram[productQ[{divRel[{1, 2}], subsetRel[{"a"}] , mnRel[2]}]]
```

Out[53]=


Product lattice of ({1, 2, 3, 4}, ≤) and its dual

In[54]:= `hasseDiagram[productQ[{ltRel[4], dual[ltRel[4]]}]]`

Out[54]=



Product lattice of ({1, 2, 3, 4}, |) and its dual

In[55]:= `hasseDiagram[productQ[{divRel[Range[4]], dual[divRel[Range[4]]]}]]`

Out[55]=