A GCD based Hermite Normal Form algorithm, stabilized in the Chou Collins or Lenstra Lenstra Lovász manner.

We present a rather simple code to illustrate that the principles that prevent coefficient explosion in the LLL algorithm also suffice in a naive GCD based Hermite Normal Form algorithm.This should not come as a surprise, because Chou and Collins have already shown this long ago, at least for an input matrix of full rank.

Query: Does anyone know of a more recent treatment, not restricted to input of full rank?

A GCD based operation amounts to a sequence of ordinary row operations. We know from LLL that one does not need to length reduce everything all the time. From my analysis of an extended LLL algorithm it is clear that it suffices to do the following. Each time before a new row is allowed to enter the fray one must bring the matrix above it into (lower triangular) Hermite Normal Form. So one should not be satisfied with a lower triangular form. To avoid coefficient explosion one should go all the way to Hermite Normal Form. One can prove an (extended) LLL like estimate for the given code in the manner of my analysis of **extendedlll**. Thus the analysis of Chou and Collins is very much related to that of LLL and **extendedlll**. In fact Chou and Collins covered the case that corresponds with the extended hnf call **hnf[matrix,symbol]** as introduced below. They did not discuss the case of the call **hnf[matrix]**, which is faster and allows dependent vectors. (They added an identity matrix to make the vectors independent). The dependent case is an easy corollary, because it traces the other case as far as the original matrix is concerned. Thus one may forget about our analysis and just use theirs. Our main point is that it helps to think of their Hermite Normal Form algorithm as belonging to the same family as the LLL algorithm, or vice versa.

We also like to stress that our simple code describes an algorithm that often outperforms the algorithm of P. Domich, R. Kannan and L. Trotter, as implemented by Victor Shoup in NTL. Recall that this algorithm

counters coefficient explosion by using modular arithmetic. When no small multiple of the pivots exists (or when none is known), it seems much better to prevent the explosion than to try and cure it by computing modulo some large number.

Recall that one sees terrible explosions when the intermediate stabilizing steps are left out.

Wilberd van der Kallen , November 2003

The hnf code

In[14]:=

```
SetAttributes[hnf, HoldAll](* We pass some arguments by name, not by value.
   The corresponding C++ source used pointers. *)

hnf::usage := "{newmatrix,rank}=hnf[mat] gives a lower triangular Hermite
   Normal Form of the integral matrix mat, together with its rank.\n
{newmatrix,rank}=hnf[matrix,symbol] also assigns to symbol
   the matrix trans so that trans.mat=newmatrix."

hnf[bb_, uu_: False] :=
 Module[
  {u, m = Length[bb], n = Length[bb[[1]]],
   i, j, l, pp, bb0, rank = 0, dimker = 0, k = 1, maxk = 0, ans}
  ,
  pp = 0 Range[m]; (* pp[[i]] will record how far row i sticks out. We
    initialize pp as a row of zeroes. Any List of length m would do. *)
  If[uu =!= False
   , (* If the transformation is requested,
    put an identity matrix in front, to keep track of the transformation. *)
   bb0 = Transpose[Join[IdentityMatrix[m], Transpose[bb]]];
   n = n + m
   , (* else *)
   bb0 = bb
   ];
  For[i = 1, i ≤ m, i++
   , (* Record how far row i sticks out. *)
   j = n;
   While[(j > 0) && (bb0[[i, j]] == 0), j--];
   pp[[i]] = j;
  ];
  While[k ≤ m
   , (* main loop *)
   maxk = k; (* number of new row that is to be processed *)
   While[(k > dimker + 1) && (pp[[k]] ≤ pp[[k - 1]])
    ,
    swap[k, bb0, pp]; (* This subroutine uses extended Euclid *)
    k--;
   ];
   If[pp[[k]] == 0, dimker++, rank++];
   For[Null, k ≤ maxk, k++
    , (* Reduce the columns, as required by Hermite. *)
    For[l = k - 1, l > dimker, l--
     , (* We use the order of
       reduction that is recommended by LLL or Chou Collins *)
     j = pp[[l]];
     u = Quotient[bb0[[k, j]], bb0[[l, j]]];
     bb0[[k]] = bb0[[k]] - u bb0[[l]]
    ]
   ];
   k = maxk + 1 (* Proceed to next row, now that part above it is in HNF form. *)
  ]; (* end of main loop *)
  If[uu =!= False
   , (* Extract new matrix, transformation and rank. *)
   n = n - m;
   uu = Take[#, m] & /@ bb0;
   ans = Take[#, -n] & /@ bb0;
   rank = 0;
   For[i = 1, i ≤ m, i++, If[pp[[i]] > m, rank++]]
   ,
   ans = bb0
  ];
  {ans, rank}
 ]
```

In[17]:=
```
SetAttributes[swap, HoldAll];

swap[kk_, bb_, pp_] := (* Swaps row kk with row kk-1,
  using extended GCD if needed. Changes bb and pp. *)
 Module[
  {i, j, l, t1, t2, e, x, y, k = kk}(* So kk is passed by value. *)
  ,
  If[pp[[k]] < pp[[k - 1]]
   , (* Row k-1 sticks out further. *)
   {bb[[k - 1]], bb[[k]]} = {bb[[k]], bb[[k - 1]]};
   {pp[[k - 1]], pp[[k]]} = {pp[[k]], pp[[k - 1]]};
   , (* They stick out equally. *)
   j = pp[[k]];
   {e, {x, y}} = ExtendedGCD[bb[[k - 1, j]], bb[[k, j]]];
   t1 = Quotient[bb[[k, j]], e];
   t2 = -Quotient[bb[[k - 1, j]], e];
   (* This is a doubly extended GCD, so that Det[{{t1,t2},{x,y}}]==1. *)
   {bb[[k - 1]], bb[[k]]} = {{t1, t2}, {x, y}}.{bb[[k - 1]], bb[[k]]};
   If[(j > 0) && (bb[[k, j]] < 0), bb[[k]] = -bb[[k]]];
   (* Keep pseudo-diagonal positive. *)
   k--; (* Look again how far it sticks out. *)
   While[(j > 0) && (bb[[k, j]] == 0), j--];
   pp[[k]] = j;
   If[(j > 0) && (bb[[k, j]] < 0), bb[[k]] = -bb[[k]]];
   (* Keep pseudo-diagonal positive. *)
  ]
 ]
```

Examples

In[19]:=
```
? hnf
```

```
{newmatrix,rank}=hnf[mat] gives a lower triangular Hermite Normal
  Form of the integral matrix mat, together with its rank.
 {newmatrix,rank}=hnf[matrix,symbol] also assigns to
  symbol the matrix trans so that trans.mat=newmatrix.
```

In[20]:=
```
ma = {{990519, -4051508, -3802647, -43245, -369337, 1967100},
   {1598245, -8116013, -6067138, -3323116, -2053036, 3161362},
   {1786244, 1069431, -465693, -227421, -2029817, -542818},
   {1084810, 1537944, 260218, -1057178, -1016978, -362744},
   {824048, -2096515, -2199740, -427760, -447464, 1118810},
   {-90848, 5193882, 1635132, 6421956, 2423012, -1147532}};
```

In[21]:=
```
{new, rank} = hnf[ma, u];
```

In[22]:=
```
u.ma == new
```

Out[22]=
```
True
```

In[23]:=
```
hnf[ma][[1]] == new
```

Out[23]=
```
True
```

In[24]:=
```
MatrixForm[ma]
```

Out[24]//MatrixForm=

$$\begin{pmatrix} 990519 & -4051508 & -3802647 & -43245 & -369337 & 1967100 \\ 1598245 & -8116013 & -6067138 & -3323116 & -2053036 & 3161362 \\ 1786244 & 1069431 & -465693 & -227421 & -2029817 & -542818 \\ 1084810 & 1537944 & 260218 & -1057178 & -1016978 & -362744 \\ 824048 & -2096515 & -2199740 & -427760 & -447464 & 1118810 \\ -90848 & 5193882 & 1635132 & 6421956 & 2423012 & -1147532 \end{pmatrix}$$

In[25]:=
```
MatrixForm[new]
```

Out[25]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 7957707264 & 15915414528 & 0 & 0 & 0 & 0 \\ 617299785 & 1234599394 & -154 & 440 & 0 & 0 \\ 1606404357 & 3212808586 & -113 & 325 & 1 & 0 \\ 4612883786 & 9225767525 & -40 & 104 & 0 & 2 \end{pmatrix}$$

In[26]:=
```
MatrixForm[u]
```

Out[26]//MatrixForm=

$$\begin{pmatrix} -56489124892 & -2642575114 & 6570408110 & -31201137677 & 99858246196 & 0 \\ -51427413573 & -2405786993 & 5981666301 & -28405357902 & 90910442188 & 12 \\ -171531944 & -8028396 & 19948520 & -94735486 & 303237226 & 1 \\ -24510371581 & -1146601504 & 2850869649 & -13538029544 & 43328034909 & 3 \\ -48579325552 & -2272553565 & 5650396645 & -26832246603 & 85875757576 & 11 \\ -47995909587 & -2245262773 & 5582536831 & -26510000186 & 84844433049 & 9 \end{pmatrix}$$

The above matrix "new" is in lower triangular HNF form. A matrix is in lower triangular HNF form if it is either zero, or there is a "first pivot", in the example the entry 15915414528, which

is positive

has zero rows above it

has zeroes to the right of it in its row

has nonnegative entries in its column, all other entries being smaller

has to its right a matrix in lower triangular HNF form. In the example this submatrix is:

In[27]:=
```
MatrixForm[Take[#, -4] & /@ new]
```

Out[27]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -154 & 440 & 0 & 0 \\ -113 & 325 & 1 & 0 \\ -40 & 104 & 0 & 2 \end{pmatrix}$$

If you prefer a more traditional upper triangular (echelon) Hermite Normal Form, then you may proceed like this:

In[28]:=
```
{new, rank} = hnf[Reverse[Reverse /@ ma], u];
new = Reverse[Reverse /@ new]; u = Reverse[Reverse /@ u]; u.ma == new
```

Out[28]=
```
True
```

In[29]:=
```
MatrixForm[ma]
```

Out[29]//MatrixForm=

$$\begin{pmatrix} 990519 & -4051508 & -3802647 & -43245 & -369337 & 1967100 \\ 1598245 & -8116013 & -6067138 & -3323116 & -2053036 & 3161362 \\ 1786244 & 1069431 & -465693 & -227421 & -2029817 & -542818 \\ 1084810 & 1537944 & 260218 & -1057178 & -1016978 & -362744 \\ 824048 & -2096515 & -2199740 & -427760 & -447464 & 1118810 \\ -90848 & 5193882 & 1635132 & 6421956 & 2423012 & -1147532 \end{pmatrix}$$

In[30]:=
```
MatrixForm[new]
```

Out[30]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 8943649870818 & -1277664267262 & -2271403141796 \\ 0 & 1 & 0 & 8121001086444 & -1160143012348 & -2062476466398 \\ 0 & 0 & 1 & 1304088368557 & -186298338367 & -331197045984 \\ 0 & 0 & 0 & 11029382267904 & -1575626038272 & -2801112956928 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[31]:=
```
MatrixForm[u]
```

Out[31]//MatrixForm=

$$\begin{pmatrix} 0 & 1160661967383 & -402774640356 & 958706785419 & -2577000444333 & 57250624 \\ 0 & 1159207084284 & -402270191576 & 957505649050 & -2573770053475 & 57178857 \\ 0 & 401563846232 & -139352224409 & 331693054390 & -891585915583 & 19807465 \\ 0 & 103903639810 & -36051413906 & 85816880019 & -230697410462 & 5125177 \\ 1 & 858445509387 & -297902516315 & 709080858371 & -1905992647328 & 42343516 \\ 0 & 1374269492566 & -476906612538 & 1135154393337 & -3051268274612 & 67786949 \end{pmatrix}$$

Of course you could simply use Developer‘HermiteNormalForm. But we want to illustrate our lightweight code.

Let us try a few random cases.

In[32]:=
```
mat[n_, m_, d_] := Table[Random[Integer, {-d, d}], {i, m}, {j, n}]
```

First an example where the normal form itself is not so favorable. Then of course the algorithm can’t keep the integers smaller than that.

In[33]:=
```
{new, rank} = hnf[ma = mat[60, 40, 1000], u];
```

In[34]:=
```
Max[Abs[new]]
```

Out[34]=
```
49457306956141535546186114932430509816732871847639933934119565200896
 38142270017780491242648182932396133684078514115339311928517505828
```

In[35]:=
```
Max[Abs[u]]
```

Out[35]=
```
17912678666057497574978120640494902214640702682910798775995991617524
 25328904241961792002717808443509098540147537404910189255501135
```

Next an example where the normal form itself is fine.

In[36]:=
```
{new, rank} = hnf[ma = mat[40, 60, 1000], u];
```

In[37]:=
```
Max[Abs[new]]
```

Out[37]=
```
1
```

In[38]:=
```
Max[Abs[u]]
```

Out[38]=
```
3058152376185370830653374863886928843896301170922521338922120783829501
 90103408247903668605726351231581363349478519543286099056764658799 5
```

You see that the entries of the transformation matrix get as bad as in the previous case. One has an estimate for the bitlength of entries of the transformation matrix that is roughly the number of rows of the input matrix times the bitlength of its largest entry .

In[39]:=
```
Log[N[Max[u]]]
```

Out[39]=
```
309.637
```

In[40]:=
```
Log[N[Max[ma]]]
```

Out[40]=
```
6.90575
```

In[41]:=
```
Length[ma] Log[Length[ma] N[Max[ma]]]
```

Out[41]=
```
660.006
```

To see how bad it gets when you are stupid and wait with the reductions until the end, let us consider the variant where one does not do any of these reductions.

In[42]:=
```
SetAttributes[hnfstupid, HoldAll]

hnfstupid::usage := "{newmatrix,rank}=hnf[mat]  gives a lower triangular
   Normal Form of the integral matrix mat, together with its rank.
{newmatrix,rank}=hnf[matrix,symbol] also assigns to symbol
   the matrix trans so that trans.mat=newmatrix."

hnfstupid[bb_, uu_: False] :=
 Module[{u, m = Length[bb], n = Length[bb[[1]]], i, j, l, pp,
   bb0, rank = 0, dimker = 0, k = 1, maxk = 0, ans}, pp = 0 Range[m];
  If[uu =!= False, bb0 = Transpose[Join[IdentityMatrix[m], Transpose[bb]]];
   n = n + m, bb0 = bb];
  For[i = 1, i ≤ m, i++, j = n;
   While[(j > 0) && (bb0[[i, j]] == 0), j--];
   pp[[i]] = j;];
  While[k ≤ m, maxk = k;
   While[(k > dimker + 1) && (pp[[k]] ≤ pp[[k - 1]]), swap[k, bb0, pp];
    k--;];
   If[pp[[k]] == 0, dimker++, rank++];
   For[Null, k ≤ maxk, k++, For[l = k - 1, l > dimker, l--, j = pp[[l]];
     u = Quotient[bb0[[k, j]], bb0[[l, j]]];
     (* bb0[[k]]=bb0[[k]]-u bb0[[l]] *)
    ] ];
   k = maxk + 1];
  If[uu =!= False, n = n - m;
   uu = Take[#, m] & /@ bb0;
   ans = Take[#, -n] & /@ bb0;
   rank = 0;
   For[i = 1, i ≤ m, i++, If[pp[[i]] > m, rank++]], ans = bb0];
  {ans, rank}]
```

Now let us see how bad it gets.

In[45]:=
```
{new, rank} = hnfstupid[ma = mat[10, 20, 1000], u];
```

In[46]:=
```
Max[Abs[new]]
```

Out[46]=
```
91282105759176311800179721821790945245048314045661729213923151008242 5
30733126780133363379750775523236433325494615497654319344766097229446 5
19161272991264518363996353105636269408265507933698818368994855547205 5
10140532980489298387488365383905849659397819362968467544286615924325 5
24434483233811720089355166008140514863937204148812586685024010184654 5
37796583363610369030565932082794415764537040389020476916431215076658 5
66738820647349068609667136523996908740523294450463286825232639900668 5
05925566836168311019037565602235732110667994999444070329177437895707 5
62757829729653769330367130964977672438337034515770310857657405194386 5
84639236467964564623436306447931299195478375120974626103031811467256 5
35877183637464055335409759670880570782457072054804336087880930734587 5
98427110292961022797913678421916484483703170409527076264510799393985 5
95069304010908479388615983029083934779576846453347355767828018646106 5
07955503057286648931417683756121154576184049474537073896526791497811 5
33260099414885874788291608265413437884757432528357722776415413787735 5
08998990517959148869417643778684650998850434719023432653626963806017 5
40808222385295438982621333426984797784871510871425015857844966449153 5
20288399862989482009952097628295717645390502958547621417366401102133 5
22503751481891889869179028237671910313232855036885946997340994782347 5
54348960635320984477938323029035764631098973728483571931118147645316 5
61493707602986325632900903556 82
```

In[47]:=
```
Max[Abs[u]]
```

Out[47]=
```
636924943993161418378781742289687811826476723753294506649816856217895
 268783132940666783006791624489188179195575071215311843833899578764 31
 256179124557177907315046960313151000727635669367573457632046231542 00
 176171878035642036382346154883563902834553684620966265476171512726 76
 937583485184604623597317561034395073756993928204772944845263850892 51
 414942508276469741138567831527159986836903409772129445675224696839 36
 168554544114237171189447468944765757790963194089549620154457344427 24
 393720083008543183541113762030241806567488451296529798681831837880 25
 438922279292686141069481213236581719472104568210044980172093510901 82
 938395827374960403381115833939068721342582971612075113414656003513 29
 999276891618550550057282004497720736258034720613568519918703992928 33
 507430843926063409121632857095885753227146572678173651840337509320 98
 438154961003661419473066230820534371670397705184793790698357080418 50
 979003887396858597766880273990914745946757636703260047975601107001 11
 429305232847167802673461983553485825973563376345379576498936599796 37
 049903002886479211499318411095434076469745316399444074420833778348 98
 261038202612253349302329942473647203858959478069433561169987617198 00
 707262814791961217220038877849991876799339191136666644549987164755 56
 435607907097910541005421720614127110642138704576729724043059951371 61
 182461867693986655262835250717493271244943070836708741279734798747 28
 260629385797721657350600637956575704283941191401719194565553766198 49
 102512093594687912319731338986506451040238280825104439721512334538 97
 439924129218229722177320134990752599468217895718382182408283135641 26
 598637107892298355969515080690631388273480700818078276619375227976 75
 914540295939744888171286980552488815021122547161948690027618015543 79
 701193136699754797540835677425090503470114087014000042662541772558 60
 606069718791638568188199513696900175644480037261334948262618401374 57
 945106391508046566555866371152793363726822027917534555266015799049 33
 305659759805340496953291600055844162317953475029100905781555913877 30
 835994895945147926696257246925962588191677331707517023862590028097 83
 187178903055097509523607064239394917362115681572706974707497161557 25
 268759230571189679407479259547874239005569092157652964151583198374 44
 111471946800638252531343944344029766085815838821838707803406845010 99
 798401743233458152348655971697898356004590571123125780083871061732 66
 988403927305263765060698233120814252962974245274086749171557629623 78
 393509348234313419741842728445574122381214389480024468215499812012 65
 131641458301190463404880737483824952183444013030997086333312938363 43
 336512818670642596054236682392750980175459579112029074069811623399 22
 844204680672016478617648529987993284317377477824024794939788184250 27
 552512004820983725458277306776810962916064788307759063531366128344 83
 556591086784786580633628593932082391750361193292655839419429629 11
```

Compare that with

In[48]:=
```
{new, rank} = hnf[ma, u];
```

In[49]:=
```
Max[Abs[new]]
```

Out[49]=
```
1
```

In[50]:=
```
Max[Abs[u]]
```

Out[50]=
```
4565973050135281941340805202 3
```

We leave it to you to explore larger cases.

Better transformation matrices.

Following Havas–Majewski–Matthews we can use LLL to improve on the transformation matrix when there are zero rows in the normal form. The difference with their approach is that we postpone this improving until our lightweight `hnf` has run its course. We believe this often makes sense. But we did not run tests to compare the two approaches. Note that we will be mixing the compiled code LatticeReduce with uncompiled code.

The hnfLLL code

In[51]:=

```
SetAttributes[hnfLLL, HoldAll];

hnfLLL::usage := "{newmatrix,rank}=hnfLLL[mat,symbol] gives a lower triangular
    Hermite Normal Form of the integral matrix mat, together with its rank.\n
It also assigns to symbol an optimized matrix trans
    so that trans.mat=newmatrix.";

hnfLLL[bb_, uu_] :=
 Module[{u, m = Length[bb], i, j, k, l, dd, lam, rank, dimker, ans},
  dd = Range[m + 1];
  lam = 0 IdentityMatrix[m];
  {ans, rank} = hnf[bb, uu];
  dimker = m - rank;
  If[dimker > 0, (* otherwise there is no point in calling hnfLLL *)
   uu = Join[LatticeReduce[Take[uu, dimker]], Take[uu, -rank]]
   (* LatticeReduce the nullspace. *)
  ];
  For[k = 1, k ≤ dimker, k++, (* Build the GramSchmidt table. *)
   For[i = 1, i ≤ k, i++, gramS[uu, dd, lam, i, k]];
   For[l = k - 1, l > 0, l--, reduce[k, l, dd, lam, uu]]
   (* This should not be necessary. *)
  ];
  For[k = dimker + 1, k ≤ m, k++, (* Build more of the Gram-Schmidt table. *)
   For[l = 1, l ≤ dimker, l++, gramS[uu, dd, lam, l, k]];
   For[l = dimker, l > 0, l--, reduce[k, l, dd, lam, uu]]
   (* This should help a lot. *)
  ];
  {ans, rank}
 ]

SetAttributes[gramS, HoldAll];

gramS[uu_, dd_, lam_, low_, up_] :=
 Module[{u, t1, t2, i}, (* Compute entry of Gram-Schmidt table. *)
  u = uu[[low]].uu[[up]];
  For[i = 1, i < low, i++,
   t1 = u dd[[i + 1]];
   t2 = lam[[up, i]] lam[[low, i]];
   t1 = t1 - t2;
   u = Quotient[t1, dd[[i]]];
  ];
  If[low < up, lam[[up, low]] = u, dd[[up + 1]] = u]
 ]

SetAttributes[reduce, HoldAll];

reduce[k_, l_, dd_, lam_, uu_] :=
 Module[{t1, t2, u, i}, (* Subtract multiple of row l from row k. *)
  t2 = dd[[l + 1]];
  t1 = Abs[2 lam[[k, l]]];
  If[t1 > t2,
   u = Quotient[2 lam[[k, l]] + t2, 2 t2];
   uu[[k]] = uu[[k]] - u uu[[l]];
   lam[[k, l]] = lam[[k, l]] - t2 u;
   For[i = 1, i < l, i++,
    lam[[k, i]] = lam[[k, i]] - u lam[[l, i]]
   ]
  ]
 ]
```

Examples

In[58]:=
```
? hnfLLL
```

```
{newmatrix,rank}=hnfLLL[mat,symbol] gives a lower triangular Hermite
  Normal Form of the integral matrix mat, together with its rank.
 It also assigns to symbol an optimized matrix
  trans so that trans.mat=newmatrix.
```

In[59]:=
```
ma = {{51, 78, -17}, {-75, -63, 48}, {-92, 87, -66}, {-96, 7, 73},
   {41, 17, 3}, {-78, 6, 58}, {47, -34, -71}, {34, 28, -11}, {13, -59, -94},
   {90, -32, -33}, {76, -62, 48}, {-20, 16, -87}, {2, 92, 29}, {14, -89, -32}};
```

In[60]:=
```
hnf[ma, uu]; Max[Abs[uu]]
```

Out[60]=
```
1234483
```

In[61]:=
```
hnfLLL[ma, uuLLL]; Max[Abs[uuLLL]]
```

Out[61]=
```
4
```

In[62]:=
```
uuLLL.ma == uu.ma
```

Out[62]=
```
True
```

In[63]:=
```
Det[uuLLL]
```

Out[63]=
```
1
```

In[64]:=
```
MatrixForm[ma]
```

Out[64]//MatrixForm=

$$\begin{pmatrix} 51 & 78 & -17 \\ -75 & -63 & 48 \\ -92 & 87 & -66 \\ -96 & 7 & 73 \\ 41 & 17 & 3 \\ -78 & 6 & 58 \\ 47 & -34 & -71 \\ 34 & 28 & -11 \\ 13 & -59 & -94 \\ 90 & -32 & -33 \\ 76 & -62 & 48 \\ -20 & 16 & -87 \\ 2 & 92 & 29 \\ 14 & -89 & -32 \end{pmatrix}$$

In[65]:=
```
MatrixForm[uu]
```

Out[65]//MatrixForm=

$$\begin{pmatrix} -963267 & -1234483 & -72318 & 522009 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -614329 & -787298 & -46121 & 332914 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -21658 & -27756 & -1626 & 11736 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -721535 & -924690 & -54170 & 391012 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -266625 & -341695 & -20017 & 144488 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -102343 & -131160 & -7684 & 55463 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -49034 & -62840 & -3681 & 26573 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ -212489 & -272317 & -15952 & 115151 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -32531 & -41691 & -2443 & 17630 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -421083 & -539641 & -31613 & 228190 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -272031 & -348625 & -20423 & 147419 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -441141 & -565348 & -33119 & 239061 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -603910 & -773946 & -45339 & 327268 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -297899 & -381775 & -22365 & 161436 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

In[66]:=
```
MatrixForm[uuLLL]
```

Out[66]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & -1 & 1 & 1 & 0 & 0 & 2 & 1 & -1 & 0 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 & -2 & 0 & -1 & 1 & 0 & 2 \\ -2 & -1 & -2 & 0 & -1 & -2 & -1 & 1 & -1 & -1 & -2 & 1 & 1 & 1 \\ -2 & -1 & -1 & 1 & -2 & 1 & 0 & 3 & 1 & 2 & -1 & 0 & 0 & -2 \\ -1 & -1 & 0 & -2 & 1 & 2 & 0 & -1 & -3 & 0 & 0 & 2 & -1 & 1 \\ 2 & -1 & 0 & 0 & 0 & 2 & -1 & -2 & 2 & 1 & -1 & -2 & 0 & 1 \\ -1 & 0 & 0 & -2 & -4 & 0 & 0 & 2 & -3 & 0 & 0 & 1 & 0 & 1 \\ -1 & 3 & -1 & 0 & 1 & -1 & 2 & 0 & 1 & 0 & 0 & 1 & 3 & -2 \\ 1 & 0 & 2 & 0 & 0 & 2 & -1 & 1 & -2 & 4 & -1 & -1 & -2 & 2 \\ 0 & 0 & -1 & 3 & -2 & -1 & 0 & 0 & 0 & 2 & 1 & 3 & 2 & 0 \\ -3 & -1 & 0 & 1 & 1 & 0 & 2 & 1 & 1 & -1 & 1 & 0 & 3 & 0 \\ 1 & -1 & 1 & 0 & 0 & 2 & -2 & -1 & 1 & 2 & 1 & 0 & -2 & -1 \\ 0 & 0 & 1 & -2 & 0 & 1 & -1 & -2 & 1 & -1 & 2 & 0 & 2 & 1 \\ -1 & -1 & -1 & 2 & -1 & -1 & 3 & 0 & 0 & -1 & 0 & 0 & 1 & -1 \end{pmatrix}$$

In[67]:=
```
ma = mat[1, 18, 100]
```

Out[67]=
```
{{-8}, {-34}, {-85}, {-35}, {-18}, {-63}, {3}, {90}, {31},
 {-20}, {-5}, {-42}, {-44}, {85}, {-37}, {-75}, {96}, {6}}
```

In[68]:=
```
hnf[ma, uu]
```

Out[68]=
```
{{{0}, {0}, {0}, {0}, {0}, {0}, {0}, {0},
  {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {1}}, 1}
```

In[69]:=
```
Max[Abs[uu]]
```

Out[69]=
```
34
```

In[70]:=
```
hnfLLL[ma, uuLLL]
```

Out[70]=
```
{{{0}, {0}, {0}, {0}, {0}, {0}, {0}, {0},
  {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {0}, {1}}, 1}
```

In[71]:=
```
Max[Abs[uuLLL]]
```

Out[71]=
```
2
```