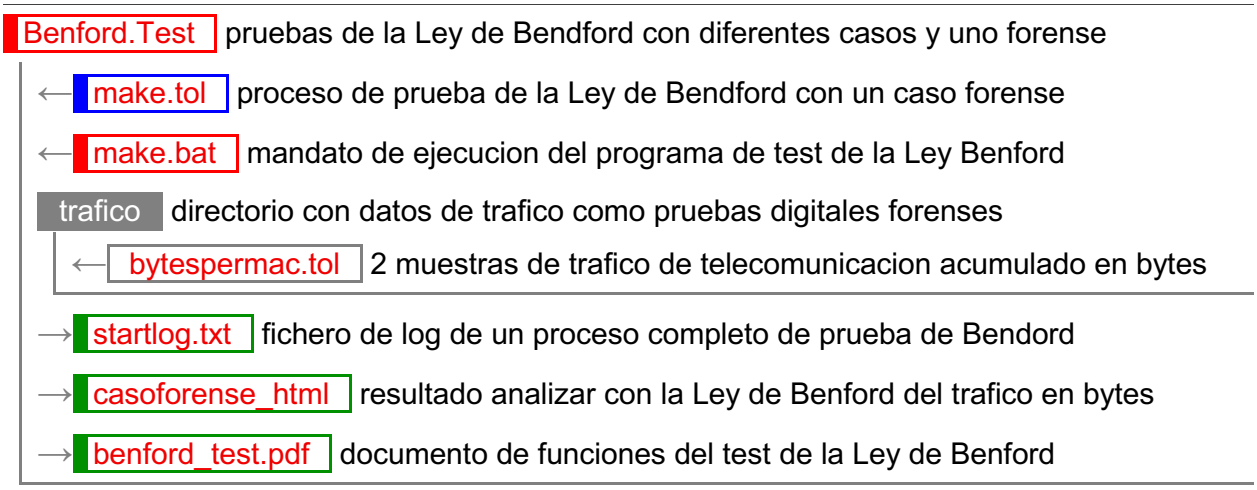
make.tol de Benford.Test

Este programa realiza diversos experimentos sobre el cumplimiento de la Ley de Benford en direrentes areas. Benford.Test consiste en una serie de pruebas que se realizan una tras otra, dependiento de su primera funcion If() de control y contiene algoritmos recursivos para la creacion de secuencias de operaciones basicas y aleatorias. La ley de Benford, tambien conocida como ley del primer digito, enuncia que, en los numeros de la vida real, el primer digito mas frecuente es el 1, luego el 2, luego el 3, ..., luego el 8 y finalmente el menos frecuente es el 9. Esta Ley de Benford parece comprobarse tanto en hechos relacionados con el mundo natural o con el social, por ejemplo, la longitud de rios, los totales de facturas, los precios de acciones, el numero de habitantes de las poblaciones, las distancias estelares, en las cifras resultantes de la realizacion de calculos con otras cifras, etc.

Antes que Frank Benford, Simon Newcomb, astronomo y matematico, en 1881 observo que las primeras paginas de las tablas de logaritmos estaban mas usadas que las finales, de lo que dedujo que los digitos iniciales de los numeros no son equiprobables sino que el 1 era el digito inicial mas frecuente, luego 2 y asi hasta el 9 que es el menos frecuente y establecio una primera aproximacion a la funcion de distribucion de esta probabilidad. Frank Benford, fisico, en 1938 establecio esta funcion de distribucion con mas precision y la comprobo en diferentes dominios. La ley de Benford establece que la primera cifra no nula n (n = 1, 2, 3, 4, ..., 9) ocurre con una probabilidad igual a (Log10(n + 1) - Log10(n)).

Ademas de otros experimentos, se dispone de un banco de 2000 datos que corresponden al acumulado del trafico de subida mas el trafico de bajada en bytes para un periodo de dias para un conjunto de 2000 direcciones MAC, Media Access Control address. La hipotesis es que este banco de 2000 datos esta manipulado y no corresponde a la realidad del trafico. Puede plantearse el siguiente experimento, seleccionar varios bancos de 2000 datos reales de trafico por MAC y para un periodo, registrados en una condiciones muy similares a las del banco de datos del que se sospecha y aplicar a estos bancos de prueba y al banco sospechoso la Ley de Benford. Se pueden establecer los 3 posibles resultados siguientes: a) Si ninguno de los bancos sigue la Ley de Benford, entonces podria concluirse que el trafico por MAC no sigue esta ley y no podria concluirse nada sobre el banco sospechoso con este experimento. b) Si todos los bancos siguen la Ley de Benford, entonces podria concluirse que el trafico por MAC si sigue esta ley y tampoco podria concluirse nada sobre el banco sospechoso con este experimento, pudiera ser autentico o falseado de tal forma que cumpliera la Ley de Benford. c) Si todos los bancos de prueba siguen la Ley de Benford pero el sospecho no la sigue entonces se podria concluir que el banco de datos de trafico por MAC efectivamente ha sido manipulado o generado de alguna forma para ser aportado como una prueba digital falsa y que no corresponde a la realidad. Cientificamente el experimento hay que hacerlo con varios bancos de trafico real. En este caso, por simplicidad, se ha seleccionado solo 1 banco de 2000 datos de trafico por MAC captados de la realidad, trafico de subida mas trafico de bajada. La conclusion puede deducirse claramente al final de este programa y existiran otras formas de analisis, pero el objetivo de este programa es realizarlo con la Ley de Benford.

Árbol de ficheros



Declaraciones

Inclusiones

- `Set datInc`
  Inclusion de 2 muestras de trafico en bytes.

Constantes

- `Text makSep`
  Separador de trazas de ejecucion.
- `Set OpeBas`
  Conjunto con las 4 operaciones basicas +, -, * y /.
- `Set BenDig`
  Digitos del 1 al 9.
- `Set BenLaw`
  Probabilidad del 1,2,...9 segun la Ley de Benford.
- `Text BenFmt`
  Formato general para sacar los resultados.

Funciones de nombre corto

- `Text F(Real numDat)`
  Retorna un numero como texto con 4 decimales, si es entero sin decimales. Es una version reducida para este programa de la funcion F() mas usual.

Funciones: Comunes a todos los experimentos

- `Real BenFstDig(Real numDat)`
  Retorna el primer digito, de 1 a 9, de un numero.
- `Set BenEval(Real numCic, Code funCod)`
  Retorna un conjunto con la frecuencia de los digitos del 1 al 9 resultado de haber evaluado la funcion funCod numCic veces.
- `Set BenView(Text expTit, Real numCic, Code funCod)`
  Retorna un conjunto con la frecuencia de los digitos del 1 al 9 resultado de haber evaluado la funcion funCod numCic veces y, como efecto lateral, a) visualiza por pantalla la comparativa del resultado con la Ley de Benford y b) el error.

Funciones: Para cada experimento

- `Real BenOpeSecRnd(Real numUno, Real numDos, Real numOpe)`
  Retorna el resultado de realizar a partir de 2 numeros de entrada, numUno y numDos, una secuencia de numOpe operaciones de suma, resta, multiplicacion y division aleatorias y equiprobables. La secuencia se realiza como una recursion lineal.
- `Real BenOpeTreRnd(Real ranMin, Real ranMax, Real numOpe)`
  Retorna el resultado de realizar a partir de 1 rango de entrada, determinado por ranMin y ranMax, una secuencia de numOpe operaciones de suma, resta, multiplicacion y division aleatorias y equiprobables. La construccion se realiza por la expansion de un arbol binario con tantos nodos de operacion como determine numOpe.

Pruebas

- `Set tst001`
  Numeros aleatorios frente a operaciones aleatorias.
- `Set tst002`
  Comparar nº de ciclos contra nº de operaciones.
- `Set tst003`
  Compararcion de la generacion en secuencia de operaciones, en arbol binario de operaciones y la exponencial de numeros aleatorios.
- `Set tstFor`
  Analiza el caso de informatica forense.

Set datInc

```
////////////////////////////////////////////////////////////////////////////
Set  datInc = Include("trafico/bytespermac.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inclusion de 2 muestras de trafico en bytes.", datInc);
////////////////////////////////////////////////////////////////////////////
```

Constantes

Text makSep

```
////////////////////////////////////////////////////////////////////////////
Text makSep = "\n"+Repeat("_", 72)+"\n";
////////////////////////////////////////////////////////////////////////////
PutDescription("Separador de trazas de ejecucion.", makSep);
////////////////////////////////////////////////////////////////////////////
```

Set OpeBas

```
////////////////////////////////////////////////////////////////////////////
Set OpeBas = SetOfText("+", "-", "*", "/");
////////////////////////////////////////////////////////////////////////////
PutDescription("Conjunto con las 4 operaciones basicas +, -, * y /.", OpeBas);
////////////////////////////////////////////////////////////////////////////
```

Set BenDig

```
////////////////////////////////////////////////////////////////////////////
Set BenDig = SetOfReal(1, 2, 3, 4, 5, 6, 7, 8, 9);
////////////////////////////////////////////////////////////////////////////
PutDescription("Digitos del 1 al 9.", BenDig);
////////////////////////////////////////////////////////////////////////////
```

Set BenLaw

```
////////////////////////////////////////////////////////////////////////////
Set BenLaw = EvalSet(BenDig, Real (Real fstDig)
                       { Log10(fstDig+1) - Log10(fstDig) });
////////////////////////////////////////////////////////////////////////////
PutDescription("Probabilidad del 1,2,...9 segun la Ley de Benford.", BenDig);
////////////////////////////////////////////////////////////////////////////
```

Text BenFmt

```
////////////////////////////////////////////////////////////////////////////
Text BenFmt = "%.5lf";
////////////////////////////////////////////////////////////////////////////
PutDescription("Formato general para sacar los resultados.", BenFmt);
////////////////////////////////////////////////////////////////////////////
```

Funciones de nombre corto

Text F()

```
////////////////////////////////////////////////////////////////////////////
Text F(Real numDat) // Numero de entrada
////////////////////////////////////////////////////////////////////////////
{
  Case(
    numDat <: BenDig,          FormatReal(numDat, "%1.0lf"), // Un solo digito
    EQ(numDat, Round(numDat)), FormatReal(numDat, "%7.0lf"), // Un entero
    TRUE,                      FormatReal(numDat, BenFmt))   // Con decimales
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un numero como texto con 4 decimales, si es entero sin decimales.
Es una version reducida para este programa de la funcion F() mas usual.",
F);
////////////////////////////////////////////////////////////////////////////
```

Real BenFstDig()

```
////////////////////////////////////////////////////////////////////////////
Real BenFstDig(Real numDat) // Un numero
////////////////////////////////////////////////////////////////////////////
{
  Real numAbs = Abs(numDat); // Valor Absoluto
  Floor(numAbs / (10^Floor(Log10(numAbs))))
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el primer digito, de 1 a 9, de un numero.",
BenFstDig);
////////////////////////////////////////////////////////////////////////////
```

Set BenEval()

```
////////////////////////////////////////////////////////////////////////////
Set BenEval(Real numCic, // Numero de veces que se ha de evaluar
            Code funCod) // Funcion que hay que evaluar
////////////////////////////////////////////////////////////////////////////
{
  Set  resSet = For(1, numCic, funCod);  // Evalua funCod numCic veces

  Set  fstSet = EvalSet(resSet, BenFstDig); // Extrae el primer digito

  // Selecciona del 1 al 9, por si la funcion retornara algun error (?).
  Set  fstChk = Select(fstSet, Real(Real fstDig) { fstDig <: BenDig });

  // Añadir 1 representante de cada numero para que salgan todos
  Set  fstCat = BenDig << fstChk;

  // Clasificar, la de los 1, la de los 2, ..., la de los 9
  Set  fstCla = Classify(fstCat, Real(Real a, Real b) { Compare(a, b) });

  // Ordenar las clases 1º el 1, luego el 2, ..., finalmente el 9
  // con comparar el primer elemento de cada clase basta
  Set  fstSrt = Sort(fstCla, Real(Set a, Set b) { Compare(a[1], b[1]) });

  // Numero de ocurrencias de cada clase, recordar que hay que quitar una
  Set  fstCrd = EvalSet(fstSrt, Real(Set claSet) { Card(claSet)-1 });


  Real numTot = SetSum(fstCrd); // Numero total de ejecuciones validas,

  // Si la funcion funCod no ha dado errores tendria que ser igual a numCic
  Real chkTot = If(numTot == numCic, TRUE, // Todo correcto
  {
    Text WriteLn("De "+F(numCic)+" han sido validas "+F(numTot));
    FALSE // Error en alguna
  });

  // Calculo de la frecuencia
  Set  fstFrq = EvalSet(fstCrd, Real(Real digCrd) { digCrd / numTot })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto con la frecuencia de los digitos del 1 al 9 resultado
de haber evaluado la funcion funCod numCic veces.",
BenEval);
////////////////////////////////////////////////////////////////////////////
```

Set BenView()

```
////////////////////////////////////////////////////////////////////////////
Set BenView(Text expTit, // Titulo del experimento
            Real numCic, // Numero de veces que se ha de evaluar
            Code funCod) // Funcion que hay que evaluar
////////////////////////////////////////////////////////////////////////////
{
  Set  expFrq = BenEval(numCic, funCod); // Evalua funCod numCic veces

  Text tabTit = "D | Benford | "+Compact(F(numCic))+" ciclos de "+expTit;
  Real lenTit = TextLength(tabTit);
  Text linTit = Repeat("_", lenTit);

  Text WriteLn("\n"+linTit+"\n"+tabTit+"\n"+linTit);

  Set  expTab = EvalSet(BenDig, Set(Real fstDig)
  {
    Real benTeo = BenLaw[fstDig]; // Valor teorico
    Real fstFrq = expFrq[fstDig]; // Frecuencia resultante
    Text WriteLn(F(fstDig)+" | "+F(benTeo)+" | "+F(fstFrq));
```

```
    Real errQua = ((fstFrq - benTeo) / benTeo) ^ 2; // Calculo del error

    [[ fstDig, benTeo, fstFrq, errQua]]
  });
  Text WriteLn(linTit);

  Real benSum = SetSum(BenLaw); // Total de frecuencias de Benford
  Real expSum = SetSum(expFrq); // Total de frecuencias del experimento

  Set  expErr = Traspose(expTab)[4]; // Conjunto de errores
  Real errSum = SetSum(expErr);      // Error del experimento

  Text WriteLn("Sum "+FormatReal(benSum, BenFmt)+" | "+
                      FormatReal(expSum, BenFmt)+" | error "+F(errSum));
//Text WriteLn(linTit);

  Extract(expTab, 1, 2, 3) // Tabla digito, Benford, experimento
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un conjunto con la frecuencia de los digitos del 1 al 9 resultado
de haber evaluado la funcion funCod numCic veces y, como efecto lateral,
a) visualiza por pantalla la comparativa del resultado con la Ley de Benford
y b) el error.",
BenView);
////////////////////////////////////////////////////////////////////////////
```

Real BenOpeSecRnd()

```
////////////////////////////////////////////////////////////////////////////
Real BenOpeSecRnd(Real numUno, // Primer numero de entrada
                  Real numDos, // Segundo numero de entrada
                  Real numOpe) // Numero de operaciones a realizar
////////////////////////////////////////////////////////////////////////////
{
  If(numOpe <= 0, numDos, // No hay ya mas operaciones que hacer
  {
    Real numRnd = Floor(Rand(1, Card(OpeBas)+1)); // Numero aleatorio
    Text opeTxt = OpeBas[numRnd];                 // Operador como texto
    Code opeCod = FindCode("Real", opeTxt);       // Operador Tol de Real
    Real newVal = opeCod(numUno, numDos);         // Realiza la operacion

//  Text WriteLn(F(numUno)+" "+opeTxt+" "+F(numDos)+" = "+F(newVal));

    BenOpeSecRnd(numDos, newVal, numOpe-1)
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el resultado de realizar a partir de 2 numeros de entrada, numUno y
numDos, una secuencia de numOpe operaciones de suma, resta, multiplicacion y
division aleatorias y equiprobables.
La secuencia se realiza como una recursion lineal.",
BenOpeSecRnd);
////////////////////////////////////////////////////////////////////////////
```

Real BenOpeTreRnd()

```
////////////////////////////////////////////////////////////////////////////
Real BenOpeTreRnd(Real ranMin, // Minimo del rango de aletorios
                  Real ranMax, // Maximo del rango de aletorios
                  Real numOpe) // Numero de operaciones a realizar
////////////////////////////////////////////////////////////////////////////
{
  Real numRnd = Floor(Rand(1, Card(OpeBas)+1)); // Numero aleatorio
  Text opeTxt = OpeBas[numRnd];                 // Operador como texto
  Code opeCod = FindCode("Real", opeTxt);       // Operador Tol de Real
```

```
  If(numOpe <= 1, opeCod(Rand(ranMin, ranMax),
                         Rand(ranMin, ranMax)), // Operacion final
  {
    Real numLft = Floor(numOpe/2); // La mitad para la izquierda
    Real numRht = numOpe - numLft; // La otra mitad para la derecha

    opeCod(BenOpeTreRnd(ranMin, ranMax, numLft), // Izquierda
           BenOpeTreRnd(ranMin, ranMax, numRht)) // Derecha
  })
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el resultado de realizar a partir de 1 rango de entrada, determinado
por ranMin y ranMax, una secuencia de numOpe operaciones de suma, resta,
multiplicacion y division aleatorias y equiprobables.
La construccion se realiza por la expansion de un arbol binario con tantos
nodos de operacion como determine numOpe.",
BenOpeTreRnd);
////////////////////////////////////////////////////////////////////////
```

Set tst001

```
////////////////////////////////////////////////////////////////////////
Set tst001 = If(FALSE, FALSE, // Poner a true/false para ejecutar o no
{
  Set tstR01 = BenView(
               "numeros aleatorios de distribucion uniforme", 1000,
               Real(Real posCic) { Rand(0,1) });

  Set tstS01 = BenView(
               " 4 operaciones secuenciales basicas + - * /", 1000,
               Real(Real posCic) { BenOpeSecRnd(Rand(0,1),Rand(0,1),  4)})
});
////////////////////////////////////////////////////////////////////////
PutDescription("Numeros aleatorios frente a operaciones aleatorias.", tst001);
////////////////////////////////////////////////////////////////////////
```

Set tst002

```
////////////////////////////////////////////////////////////////////////
Set tst002 = If(FALSE, FALSE, // Poner a true/false para ejecutar o no
{
  Set tstS02 = BenView(
               "10 operaciones secuenciales basicas + - * /", 2000,
               Real(Real posCic) { BenOpeSecRnd(Rand(0,1),Rand(0,1), 10)});

  Set tstS04 = BenView(
               " 5 operaciones secuenciales basicas + - * /", 4000,
               Real(Real posCic) { BenOpeSecRnd(Rand(0,1),Rand(0,1),  5)})
});
////////////////////////////////////////////////////////////////////////
PutDescription("Comparar nº de ciclos contra nº de operaciones.", tst002);
////////////////////////////////////////////////////////////////////////
```

Set tst003

```
////////////////////////////////////////////////////////////////////////
Set tst003 = If(FALSE, FALSE, // Poner a true/false para ejecutar o no
{
  Set tstS03 = BenView(
               " 7 operaciones secuenciales basicas + - * /", 3000,
               Real(Real posCic) { BenOpeSecRnd(Rand(0,1),Rand(0,1),  7)});

  Set tstT03 = BenView(
               " 7 operaciones en arbol basicas + - * /", 3000,
```

```
                Real(Real posCic) { BenOpeTreRnd(0, 1, 7)});

  Set tstE03 = BenView(
                "exponencial de numeros aleatorios entre 0 y 1", 3000,
                Real(Real posCic) { Exp(Rand(-10,10)) })
});
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Compararcion de la generacion en secuencia de operaciones, en arbol binario
de operaciones y la exponencial de numeros aleatorios.",
tst003);
////////////////////////////////////////////////////////////////////////////
```

Set tstFor

```
////////////////////////////////////////////////////////////////////////////
Set tstFor = If(FALSE, FALSE, // Poner a true/false para ejecutar o no
{
  Set tstD02 = BenView(
                "trafico dudoso, acumulado en bytes", Card(TraDud),
                Real(Real posCic) { TraDud[posCic] });

  Set tstC02 = BenView(
                "trafico controlado, acumulado en bytes", Card(TraChk),
                Real(Real posCic) { TraChk[posCic] })
});
////////////////////////////////////////////////////////////////////////////
PutDescription("Analiza el caso de informatica forense.", tstFor);
////////////////////////////////////////////////////////////////////////////
```

bytespermac.tol de Benford.Test

Muestras de trafico de subida mas bajada acumulado para varios dias para un conjunto de direciones MAC, Media Access Control address.

Hay 2 bancos, el primero ofrece dudas de si ha sido manipulado, el segundo esta extraido en condiciones garantizadas.

En la realidad, este experimento se tendria que hacer con varios bancos de datos de control, pero por simplicidad se comparan solo 2, el que plantea dudas contra 1 solo banco de control.

En lenguaje Tol estos datos se podrian haber guardado de multiples formas, por ejemplo, en un fichero Bst, pero en este caso para pruebas se han declarado 2 grandes conjuntos de 2000 elementos cada uno.

Declaraciones

Constantes

- `Set TraDud`
  Banco de 2000 datos que corresponden al acumulado del trafico de subida mas el tráfico de bajada en bytes para un periodo de dias para un conjunto de 2000 direcciones MAC. Se tienen dudas de si este banco de datos ha sido manipulado.
- `Set TraChk`
  Banco de 2000 datos comprobados y obtenidos bajo supervision que corresponden al acumulado del trafico de subida mas el tráfico de bajada en bytes para un periodo de dias para un conjunto de 2000 direcciones MAC. Este es el banco de control.

Constantes

Set TraDud

```
////////////////////////////////////////////////////////////////////////////
Set TraDud = [[
  9, 627, 295, 653, 58, 842, 9452, 4797, 9505, 97862, 1, 19781, 5171, 3, 9832,
  34, 7345, 491, 572, 8488, 771, 1714, 85453, 4566, 4573, 2442, 75664, 19262,
  72, 631, 291, 9, 1, 6964, 17, 8, 5, 64242, 31, 16652, 92, 66281, 131, 3602,
  718, 173, 788, 1, 71, 918, 1772, 588, 82, 74866783, 2192, 3601, 71654, 322,
  103, 641, 672, 122, 85601, 836, 5801, 3501, 136, 95, 3, 288, 69, 21, 8, 258,
  241, 91, 56, 769069, 62, 6393, 8253, 74, 61, 7834, 981, 13, 491, 61, 1391,
  673, 411, 72, 12, 7604, 33, 2, 118, 31621, 1502, 2074, 297, 3644, 458636,
  533, 5934, 417, 67941, 40282, 439493, 47441, 95, 9195, 2, 777, 95, 85852,
  5741, 501, 325, 5617, 768, 7606, 127, 59531, 1, 26, 7768, 804, 52, 2, 415,
  957, 8863, 13, 6777, 21, 16172, 482, 105, 464, 5, 2, 30171, 35, 7776, 823,
  55, 3, 609, 867, 1656, 211, 8644998, 7781, 9391, 87, 38, 474, 5, 429, 22706,
  771, 62187, 1, 9571, 473, 609, 6665, 7471, 2, 2742, 10101, 81, 1568, 7521,
  7711, 5514, 62, 52, 4, 31, 24, 56721, 8631, 50486, 53303, 1, 897, 337, 23,
  74376, 914922, 23, 71, 733, 425, 62, 4501, 40431, 1, 98, 19861, 19, 3331,
  77492, 55501, 7622, 7312, 86311, 1, 5271, 11, 1637, 311, 461, 634, 191,
  9337211, 191, 861, 95, 4, 17851, 46, 761, 991255, 55, 14628, 3641, 65922,
  47076, 11, 70445, 5615, 49339, 5332, 7501, 4302, 7132, 702, 172, 74064, 658,
  3681, 81, 251, 38, 1314, 1783, 7957041, 2178, 9431, 112, 121, 3, 7, 874, 7,
  24, 8226248, 34632, 41, 622, 49, 9762, 734, 64, 869, 2976, 682, 28, 324,
  7749, 5499, 861, 4624, 5081, 93, 18, 647, 1792, 95, 1284, 3461, 791, 516,
  432, 2971, 24591, 65641, 377, 1751, 5085, 484, 2, 4141, 673632, 3198, 86,
  294. 5687. 296781. 5783. 4154. 3. 3. 24361676. 839271. 14. 701. 4931. 951.
```

472, 317, 48513, 2057, 21, 7177, 635479, 6, 411, 3824, 226, 592, 755, 34,
53091, 14022, 844, 8336, 1, 4, 11, 102, 1, 52472, 5263, 971, 39, 581, 8961,
3, 7236, 23, 11, 432, 13, 711, 96631, 841, 81, 122, 151, 7254283, 9887,
102010944, 72786, 4891, 986, 4081, 31, 9, 542, 178, 461, 661, 653, 4342,
741, 9244, 3, 247, 89174, 48556, 65, 1151, 972, 2231, 14, 91963, 7941, 4,
2224, 1, 49127704, 932, 11116, 9321, 2991, 865904, 37, 668, 4411, 817651,
535, 425, 93, 2712, 4, 580361, 687, 81, 78, 61011, 5, 775, 302, 8325, 5849,
4, 48141, 1386, 65996604, 23562, 21, 37, 2817, 931, 252, 6, 1314, 95, 1401,
8201, 1, 765, 1986, 7004, 208, 36228, 3241, 6426, 16, 566692, 5, 838, 93071,
4275, 8651, 64983, 91, 71, 4805, 482, 396, 26, 995, 8481, 561, 1, 557, 76,
895, 8451, 462291, 1, 117, 1485, 1251, 11, 69294, 522, 71, 813, 101, 26,
32362, 9841, 62, 8224, 7, 311, 3534, 6078413, 71, 363, 4741, 375, 4, 2, 11,
48, 233, 1283, 171, 4195, 35, 632, 12, 4569, 8413, 421, 810598, 8756, 1, 2,
4942, 9753, 889, 685, 691, 501, 1, 5831, 531, 347, 73, 262, 8841, 8668,
4169, 51, 55, 16126, 736, 3121, 731, 2, 2687, 23, 7668, 1724, 2, 233, 2805,
571681, 5, 55076, 62, 228, 57, 11, 22, 2341, 92022, 4682, 3135, 6, 185, 971,
9085, 5834, 12968, 40863, 529, 68654, 552, 5054, 11, 23, 81, 6, 692, 7341,
66, 491, 7832, 291, 1001, 1995, 3133, 9152, 6346, 9122, 42, 262, 8981, 523,
49692, 461, 266, 6799, 1794, 4511, 8853, 673138, 4313, 115, 71, 468, 336,
5674, 933, 47, 63466, 1, 74315, 21181, 81, 351, 741, 933, 952, 47, 12167,
461, 28649, 15, 99, 8721, 7902, 81, 496532, 19, 1871, 2, 9813, 722, 3, 5215,
1785, 25614, 291721643, 1, 5212, 91, 7406, 143, 567, 83, 207, 78531, 65706,
3, 9869, 3387, 3788, 7406, 2823, 2065, 7803, 8001, 318, 31, 2161, 6, 21325,
51, 781, 971, 45514, 21, 44193, 313675, 5761, 1758, 393, 5622, 37, 311,
9292, 2762, 1, 51, 6455, 2461, 6202, 659, 2103, 33815, 9953, 15442, 201,
47092, 9892, 45652, 33, 39, 599, 685, 51, 34, 4, 1783, 2, 2191, 2385,
594463, 755, 6452, 1546, 4265, 64, 63, 85031, 241241, 15, 94, 5876, 7486,
6374, 7201, 2, 7, 1, 20275, 652, 8243, 67321, 1, 6709, 82, 942, 7855, 42,
934, 876, 3316335, 7341, 5842, 704, 57, 4116, 564, 17711, 142, 36, 51,
93261, 1996, 251, 97302, 21, 50461, 81, 9725, 1, 62, 44, 3331, 12, 1291,
6511, 511, 7, 4167, 23, 251, 25561, 3358, 6872, 217, 126853, 912, 62832,
3052, 7861, 5711, 5, 9682, 1271, 2, 1, 1451527, 688163, 7141, 4682, 127, 24,
56601, 33, 484, 906, 373, 397, 6363, 681, 341604, 881, 434, 1929, 102, 1594,
3, 4914, 7401, 23372, 71, 35, 2, 7862, 37, 718, 14061, 37, 812, 21, 301,
38751, 8033, 954, 71, 1197, 189, 1, 354, 95, 9841, 2973, 392, 9052, 1, 641,
412, 436, 7458, 8592, 221, 76055, 6051, 43, 224, 641, 4071, 7361, 337, 23,
4, 26, 8511, 82, 4, 623, 2, 11, 1, 7, 42064, 373, 192, 281, 65, 3, 33185,
491738071, 3, 1293, 961, 6769, 91, 6904, 5011, 842, 21, 47, 4036, 3071,
6062, 10581, 9782, 843, 45, 19064, 231, 321, 91, 2, 7961, 41, 95, 336, 63,
49261, 352, 343, 85, 45, 655, 985, 725, 71, 2684, 216742, 508, 2, 8165, 592,
76501, 864, 34613, 35, 74243, 99, 152, 3786, 706, 456, 96351, 4971, 227284,
4681, 2782, 7, 22462, 4067762, 84, 6, 4375, 4032, 5355, 5749, 71, 56, 703,
3084, 5, 8769, 1, 411, 52, 76702, 4, 411, 408, 6, 9037, 52802, 7, 7893, 66,
585, 18164, 4, 3702, 13791, 895, 355182, 41, 2034, 3424, 144, 5645, 7318,
335, 46661, 173, 7341, 28785, 896472, 73, 1142, 9194, 653, 43, 7725, 43302,
384, 442, 682, 5, 48, 30924, 641, 13, 4192, 838, 66, 9971, 73, 467, 1, 258,
992, 7724, 9011, 121, 4321, 66679, 53, 67, 3, 2346, 361, 4895, 803, 8756,
43374, 7459, 587, 4, 7059, 53, 408027, 27126, 4, 744, 513, 11, 729, 24, 84,
5079, 40351, 63501, 18262, 21, 8557, 871, 9156, 15542, 4, 5, 9522, 5732,
243, 3385, 71, 2485, 1513, 354, 53, 5223, 43762, 1409431, 68452, 2, 8487,
3501, 1, 6421, 991, 25, 1, 81, 7, 48, 3913, 4763, 5662, 261, 6, 1241, 354,
193729, 393, 642, 2, 38, 6991, 2, 57, 43382, 6235, 229, 709, 711, 65861,
431, 944, 301, 942, 15, 876, 82, 92647, 635, 2462, 731346, 475, 901, 423,
562, 52, 606, 32, 6287, 4108, 67, 141, 8042, 342, 66032, 1, 692, 4, 975,
472, 985, 673, 42, 981, 3, 9563, 204, 9205, 181, 64, 2336, 6723, 2, 457,
7317, 31, 3216, 3171, 322, 917, 27, 217, 80512, 91, 73, 27094, 2229, 5317,
15201, 871, 152, 69, 6508, 462, 77553, 84, 2, 4971, 609, 5386, 141, 43, 617,
31, 2311, 55621, 2611, 9241, 4, 428, 51, 8211, 8281, 9714, 51, 699, 797,
5415, 641, 6228, 821, 9895, 3661, 91, 985, 56, 1, 11, 4322, 41, 21, 909946,
4, 73, 23, 1522, 516, 133, 391, 5, 701, 61, 3977, 6631, 153, 15678, 872,
414, 2564, 71, 9513, 1456, 8283, 508, 7, 209, 51, 536, 365, 83, 173, 1, 2,
4307, 23, 1, 34162, 1, 67381, 1971, 4562, 36, 8595, 6, 4, 101, 11, 7998, 72,
2862, 495585, 55, 9651, 12, 4492, 99, 711, 1, 5949, 4661, 280301772, 85214,
171, 23, 31, 55601, 6453, 41651, 7, 182, 125, 7367, 5291, 497, 45, 2936, 63,
75, 11, 11, 41, 213, 67407, 382, 2405, 58, 1722, 13, 1581, 3263, 226972,
4181, 61, 8572, 517, 331, 509, 3, 113, 5467, 952, 601, 4677, 2, 467, 1215,
92, 70272, 3732, 4466, 3, 35461, 253, 264809, 2771, 755, 108, 94, 4841, 802,
47055, 311, 2, 6587, 31, 30901, 5641, 35, 488, 9151, 12383, 1, 697, 542, 18,
981, 9795, 4, 552, 714, 22, 1, 335, 93, 5, 74, 42901, 27206, 7941, 3261, 2,
13848, 2321, 811, 501, 757811, 87, 305, 504, 643, 8134, 31, 1, 508, 2, 2245,
7232, 28864, 27592, 5, 26, 632, 31, 2302, 1, 32107, 141, 3602, 5014, 962,
1385, 452, 158, 251, 2, 3, 547, 95635, 322, 6316, 646003, 40685, 1, 794, 7,
1, 319, 11, 251, 4, 3, 3936, 707, 511, 699, 556, 75132, 81, 8341, 4013,
78811, 94, 388, 59587, 11, 52, 2377, 29, 11, 8844, 13, 303631, 751, 943,
3333, 631, 232, 33, 362, 4275, 495, 917, 6, 171, 844, 9608, 5, 715, 4, 997,
4511, 591, 4261, 292732, 20677, 20432, 52, 8596, 2756, 691, 172, 71, 4761,

```
  6745, 7, 4684, 34, 4, 61, 669, 463, 102, 3193, 7453, 302, 881, 756, 676,
  233, 96, 2444, 8114, 4041, 46, 383539, 23003, 12433, 39985, 613552, 21,
  47793, 31, 58, 174, 27, 9848, 421, 35761, 828, 67681, 173, 5582, 431, 781,
  402, 7331, 2, 978, 51, 52, 265, 5163, 948292, 71, 3735, 82681, 6771, 49534,
  1, 3787163, 5451, 82, 249, 9785, 20121, 946, 2223, 446, 73, 13, 44893, 1,
  4659, 81, 689, 772337, 61, 98, 862, 762, 33737, 655391, 82, 1, 4, 3451, 1,
  4406, 8521, 9661504, 869, 1699, 352, 17782, 4, 534, 22072, 7756, 9221, 7731,
  5723, 3, 17531, 6823, 21, 68, 32, 15308, 271, 786, 8085, 87, 2, 952663,
  9571, 409033, 6901, 1813, 77312, 138, 30182, 1752, 51163, 33564, 42, 7,
  1542, 2, 35, 5444, 9378, 88251, 5181, 71, 97274, 1, 57951, 1708, 1, 7076,
  6242, 23, 3, 13884, 1, 5504, 6713, 264, 73261, 91, 8, 48, 6524, 22352, 4611,
  3, 5903, 11, 8957, 5191, 471, 45, 5, 62, 622, 648, 2571, 6897, 43, 191, 191,
  24, 31, 8776, 70912, 2724, 665, 444, 1312, 61, 174, 8227, 13, 6774, 56073,
  41, 4715, 483, 1, 4741, 78, 44, 56864, 372, 67813, 11, 25, 35, 4541, 79871,
  222, 8, 16744, 6681, 66, 4828, 9472, 51, 21927, 2953, 5, 43821, 5835, 35,
  199297, 6312, 1, 2881, 72, 22462, 34, 318, 585, 6, 88111, 4522, 936882,
  3071, 372, 4158, 101, 5, 4621, 863, 94802, 744, 678084, 56001, 1, 12, 561,
  561, 75, 9751, 54, 2, 77, 1871, 152, 281, 578, 56252, 39142, 451, 72, 3704,
  67, 8305, 13, 101, 7121, 332, 171, 8031, 2, 3936, 1, 1105, 3825, 254, 23,
  92, 13, 9291, 5851, 61, 2, 9909, 3292, 25, 27, 47, 9282, 46, 42698, 7115,
  8825, 631, 262, 6153, 694, 242, 1, 10335, 7801, 772, 221, 1, 542, 35695, 63,
  1, 865, 2, 3094, 488, 8431, 391, 86, 63, 13, 561, 44, 62, 495, 23, 9, 3, 4,
  792, 3696, 3502, 9351, 4541, 7, 5549, 22, 5, 56, 16302, 74, 978, 10883, 2,
  225, 4286, 392, 903, 45, 773, 44614, 679, 175, 3461, 5555, 5261, 912, 21,
  9341, 67, 995, 6, 16, 11, 2, 6392, 7358, 809, 203, 74, 495, 4313, 29173,
  381, 3053, 66451, 1, 463771, 1131, 3, 1826, 3, 173, 2, 39997, 15341, 712,
  315831, 2182, 2378, 3517, 813, 73, 2441, 222, 8939, 7135, 23, 74, 26, 7204,
  5771, 525, 256512, 342451, 1, 7461, 99211, 928324, 65, 7772, 37, 14, 1977,
  788321, 21, 9277, 11, 9273, 361, 94, 6, 247, 224, 1866, 1332, 45054, 2091,
  12, 6562, 452, 6343, 334, 861, 5163, 1164, 13763, 711, 1, 182, 81, 275,
  8629, 4163, 377481, 4, 2287, 3, 4408, 501, 9761, 27938, 471, 9943, 36, 4,
  7089, 3, 81555, 71, 11, 631, 6724, 1232, 54, 31, 41225, 22, 8313, 25, 55,
  535, 836, 3, 861, 3321, 6101, 2596, 91, 53, 6676, 81, 842, 5086, 7, 67592,
  553, 89633, 2, 31, 2, 1, 51404, 862, 4812, 5321, 29255, 5155925, 611, 1721,
  40751, 129211, 441, 11, 5, 542, 533, 9405, 7393, 3809, 1768, 17592, 6904361,
  5, 287, 51, 587, 296, 775, 681, 41081, 14, 459, 55, 8883, 2631, 1, 7494,
  998, 34789, 413, 2012, 73693, 663732, 818791, 502, 71, 69, 1, 41492, 861,
  9331, 2, 2365, 467224, 6, 526, 14, 8417, 3634, 285, 636, 483, 96995, 194768,
  892, 701, 724, 1, 52917, 3552, 32, 4859, 9163, 142, 3681, 1011, 7441, 19352,
  14697, 23001, 7, 7, 75, 33, 161, 209, 3238, 61, 611, 411, 4681, 25, 52, 471,
  31, 1992454, 3512, 54, 8608, 587, 88, 157, 385, 9909, 3, 2539, 3087, 6391,
  488, 667, 21, 622, 61478, 75831, 2555, 7751, 61793, 862, 823, 5082, 1, 1771
]];
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Banco de 2000 datos que corresponden al acumulado del trafico de subida mas
el tráfico de bajada en bytes para un periodo de dias para un conjunto de
2000 direcciones MAC.
Se tienen dudas de si este banco de datos ha sido manipulado.",
TraDud);
////////////////////////////////////////////////////////////////////////////
```

Set TraChk

```
////////////////////////////////////////////////////////////////////////////
Set TraChk = [[
  303, 5177, 1, 1508, 282, 386, 26179, 3775, 1255, 57583, 16147, 8062, 22,
  1066, 748, 8639, 1308, 7253, 9, 3990, 938, 515, 78, 858, 7860, 85, 4351,
  2199245, 43, 1047, 15, 22, 5468, 1041, 161, 106, 1323, 820, 916, 13, 37, 5,
  7, 72300, 11469, 71935, 2744, 1101, 1368, 114, 2916, 3485, 93, 20355, 25291,
  7, 172, 4070, 189, 219476, 89699, 548, 363, 628, 5363, 4841, 71, 452, 6,
  646722, 4236, 5, 2933, 186, 14149, 2, 16, 97, 150, 281879, 166373, 27369,
  3201, 241, 33478, 999, 8749, 4, 1263, 1888, 305, 545, 91, 44108, 1068, 177,
  529, 658, 75, 128, 7, 5690436, 11759, 2176, 8380, 9739, 3940, 553, 354, 2,
  51, 144, 112921, 14075, 1172, 161, 10515592, 52925, 5532, 1481, 286, 25140,
  4, 10, 23, 1, 28963, 3055, 36759, 20, 7508, 684, 28, 1667, 65, 39, 100259,
  6610, 10332, 186, 1, 383, 653, 55, 52, 5831, 32, 24122, 53, 15, 4123, 2672,
  463, 11, 107, 18155, 5, 30708, 9, 43, 6994, 347, 12793, 8976, 150, 1440, 80,
  3228, 7, 437748, 1416, 3862, 927, 58, 118, 20, 171, 11376, 3116, 4516,
  30086, 133, 4634, 345, 2656, 21, 20209, 14505, 4133, 2186, 622, 424, 7, 609,
  436, 1927, 31, 10927, 72, 178832, 1197, 71, 43, 7777, 26, 592832, 49921,
  10746, 1, 134245, 125651, 252, 50577, 1720, 42, 607, 42, 3713, 5893, 922,
```

2244, 17, 381, 3351, 7237, 8218, 231583, 1071, 21534, 13999, 70, 217, 3,
3182, 6, 3113, 146377, 1, 16645, 1305, 3038, 12917, 3431, 349, 57, 420, 380,
9735, 8639, 2, 21, 11, 6, 699, 56, 7934, 12, 1091, 2526, 1555, 5346, 117,
567, 94461, 477, 34, 590, 339, 2428, 622, 5, 21088, 397, 87, 41630, 25, 6,
52, 2554, 90, 7454, 1935, 1350, 2369, 679, 2, 4, 3, 902, 349, 56, 24, 456,
11, 36, 38, 639, 10843, 148, 8309, 4, 286, 50, 16, 33569, 554, 2, 122, 177,
2780, 11033, 5, 124, 269, 4615, 326, 263, 1882, 5711, 54269, 84, 6928, 24,
72, 72, 5329, 2990, 9, 206, 1585, 1929, 11, 39, 22, 325, 4382, 5110, 5,
1393, 6, 2803, 117, 133, 2712, 110, 11, 3830, 506, 7370, 47, 245, 13914,
25625, 257, 828, 115, 2187, 17, 70, 25, 4975, 17, 556, 156, 101, 2, 15600,
167808, 474, 49480, 286, 3462, 1, 510, 1415, 837, 2307, 193688, 2452, 28811,
1, 60058, 531, 83, 42246, 207, 2288, 1, 1631, 219929, 7583, 54382, 1, 5295,
32192, 75, 1947, 2482, 86, 60668, 11674, 4, 822, 27987, 1454, 103, 52, 51,
16781, 337, 25686, 44, 47, 8474, 1, 148, 3471, 54, 15607, 3677, 41, 3822,
717, 406, 1131, 244, 466, 5272, 47091, 2877, 63, 12, 419, 119, 14, 3689,
7257, 164, 862, 206, 2, 54, 547, 10519, 1895, 71, 1590, 3, 3461, 12235,
2652, 44, 8, 89, 17326, 126, 4671, 3550, 40, 11388, 4, 32, 3624, 2707, 6,
10170, 85795, 1, 19727, 47, 1518, 18825, 1937, 8544, 43, 5, 2154, 20, 74,
23356, 45116, 3175, 23018, 213, 87731, 2181, 3690, 140903, 3957, 1095266, 3,
28, 7808, 578, 627, 11530, 83, 2006, 8002, 1682, 31753, 1, 3572, 3773, 1922,
1775, 62207, 253, 4, 41778, 235, 2169, 986, 9966150, 4852, 1440, 6, 1345,
100, 40, 18, 265539, 13373, 776, 286, 29, 86, 177233, 768, 98, 10465, 9,
14489, 31, 37, 3044, 6222, 394, 62012, 1978, 524, 908, 20545, 1378716, 3,
14953, 4677, 18268, 1373, 57, 194829, 2516, 3026, 505, 25, 197, 8, 2733,
140, 2078, 143, 1558, 217, 36768, 182, 83576, 142, 1984, 82, 717, 45, 83,
14779, 32, 161355, 23998, 51243, 32300, 338353, 94, 6404, 10811, 4244, 49,
623, 367, 675, 6088, 1030, 2745, 3319, 310, 246, 366, 96, 1000, 43, 4468, 4,
7674, 5476, 107, 117, 269, 1275, 6859, 65, 22043, 22067, 729273, 2426, 159,
1451, 199, 7, 4071, 5, 5960, 884, 417, 1, 691, 7049, 5427, 536, 203, 323,
263, 1333, 394, 375, 130363, 11, 3884, 41, 12, 9237, 705, 21, 15958, 738,
89, 47881, 1401, 3834, 108, 107513, 255, 669, 951, 170, 7393, 6, 3, 425, 11,
1031, 9, 1, 779, 4, 14068, 5064600, 3631, 632, 29563, 554, 7, 30, 225, 115,
845, 2138, 596, 2501, 4360, 214, 13210, 7, 1230, 203, 163, 22, 60, 52759,
22886, 4723, 2224, 5, 250, 8, 13, 1813, 464, 350, 430, 58, 775781, 150,
1081, 1232, 11384, 8, 2326, 10794, 12720, 64290, 17, 4, 59, 333, 5, 12, 271,
455, 2, 4979, 598, 11, 854, 269, 7, 11238, 3915, 1048, 83, 5564, 13090, 13,
1658, 7, 2031, 14705, 580, 2484, 19, 410, 875, 5277, 126480, 925, 33546, 10,
3446, 6373, 27027, 29, 2121, 546, 7, 2467, 760, 1095, 2546, 711, 3, 350,
933, 151, 7857, 26, 1418, 122697, 2326, 3158, 11, 3172, 25, 8240, 538,
26740, 721, 34, 11, 11, 17, 56, 3293, 64, 549, 7529, 1736, 712, 518, 20,
74165, 10645, 35560, 13, 102, 3017, 18521, 428030177, 2466, 1594, 9, 171,
19, 9449, 21, 2965, 105, 549558, 50286, 27, 2799, 801, 110, 1, 4, 6859, 53,
6456, 2197, 16738, 1, 13416, 2, 12, 3430, 7, 2, 117, 38, 336, 553, 65, 1020,
9, 750, 8828, 3145, 6951, 37, 1256, 441, 487, 21567, 815, 3663, 1397, 76,
170, 1, 539, 100013, 351, 6152, 22, 37, 3, 490994, 602, 104, 1432, 21, 10,
15, 698, 59, 1366, 1989, 582, 1622, 864, 10541, 579, 769, 905, 10971, 4828,
1821, 105, 1042, 8, 4924, 1261, 105562, 1231, 13, 161, 74, 314, 10538, 660,
9497, 1, 208, 47755, 346, 2650, 86, 915, 287, 11520, 1531, 7222, 92709, 47,
39, 18051, 2021, 72, 7091, 732, 2317, 1321, 603, 10731, 745, 7, 2672, 3233,
66, 418, 1920, 520, 4956, 3, 3098, 14, 267, 3098, 547, 297, 5, 469, 2,
16603, 234, 2804, 214, 16, 7410, 8628, 73, 260, 65, 2056, 242, 3090, 147,
573134, 6246, 263, 59264, 78, 287, 61, 594, 230, 194, 4043, 106586, 171,
1070, 922, 9623, 54338, 25, 7, 2699, 13, 8, 264, 239, 319372, 9035, 4124, 1,
626, 1566, 2476, 3391, 34, 8, 78, 100, 12, 599, 1642, 1, 1350, 1848, 7,
26845, 2140943, 14376, 2522, 35, 335, 4151, 3248, 57, 1338, 524, 3573, 2952,
20, 50, 41554, 2915, 687, 1855, 72, 11826, 26350, 14035, 1507, 98, 42, 4072,
901, 1051, 374, 4, 42712, 640802, 75, 3705, 9, 458, 7745, 94337, 4875, 680,
3489, 536, 1234, 6, 36, 75, 36667, 9432, 112, 1901, 1772, 499, 225, 8, 1046,
77, 3997, 106, 683, 8419, 21, 364, 13092, 404, 8, 568, 244, 238, 44330,
2772, 54, 365, 3919, 4114, 27, 389647, 22878, 5734, 117, 34666, 133, 5731,
8564, 514, 4342, 4203, 44, 135518, 289, 51379, 1370, 2, 41816, 458, 56,
6789, 3, 75280, 2903, 7, 640, 8041, 1, 47670, 205, 241, 1, 1876, 9, 5308,
470, 35, 67, 1574, 9535, 5403, 2437, 5, 68, 4406776, 22246, 2, 7278, 2468,
122728, 4497, 19635, 145, 670, 6378, 615, 209, 97424, 303, 53461, 386,
47650, 159, 2816, 5, 2050, 821674, 2268, 47, 172, 598, 565, 54, 58, 534,
335, 24926, 106, 333, 69, 54, 1796, 1000, 29, 132, 123, 11906, 40004, 5084,
3978, 21058, 1606, 2307, 1403, 64, 702, 184, 2501, 1690, 49, 1676, 996,
10129, 30, 349173807, 13318, 5, 36, 528, 119, 237, 34206, 40, 7, 11, 1, 262,
3, 48, 2851, 12, 6, 42, 333, 7736, 1407, 164, 122, 44, 3605, 17605, 522,
1859, 2745, 843, 641, 264, 1, 1905, 2039, 2297, 3984, 109, 535, 4, 118,
9119, 77, 195, 4803, 33875, 130, 102, 181, 23, 71406, 171, 83, 7786, 2, 23,
57, 12337, 2740, 1491, 4, 3159, 410, 2192, 943, 4088, 134160, 4068, 1636,
339, 737, 390, 648, 43, 35660, 12, 412, 7468, 2714, 168816, 3145152, 7, 1,
2127, 10968, 2, 5571, 1786, 1305, 26283, 291, 2012685, 321, 7687, 2335, 562,
82556, 125, 12, 3416, 7, 751, 1651, 1129, 11, 2333, 14, 46, 2, 1972, 58,
105046, 12, 109730, 225, 10199, 69326, 15, 13, 614, 21771, 156, 4411, 65,

```
  770, 4584, 2734, 1331633, 587, 6093, 44, 2785, 594, 28, 200670, 9, 16732,
  1824, 365, 22027, 5, 10, 7, 2720, 1637, 4748, 6587, 609, 41, 524124, 18503,
  16, 36, 4426, 5154, 6645, 275, 559446, 30238, 5219, 10, 2178, 3, 43, 45,
  168, 5059, 93, 93, 34565, 2989, 24709, 2020, 11544, 2995, 33381, 5210, 365,
  9620, 2246, 1645, 55, 1, 1276, 20929, 231, 13, 7562, 239, 3175, 8576, 329,
  131367, 54419, 32, 14551, 4097, 178, 15, 12132, 5, 6216, 103, 131, 8800,
  1780, 3206, 5282, 30740, 6378, 8338, 7986, 9, 36570, 67853, 120, 78, 356,
  714, 3740, 69, 1521, 20, 129172164, 32561, 4178, 5521, 5, 372, 2981, 30,
  2187, 11, 949653, 28, 1790, 2872, 19, 901, 52864, 946, 11216, 704, 795, 293,
  374, 135, 108, 12118, 17431, 50, 16346, 64, 793, 3567, 433, 6046, 87, 111,
  5431, 367313, 8885, 3451, 1179, 4679, 926, 646, 14969, 252, 3898, 126, 24,
  2912, 2634, 6915, 2313, 3199, 5100, 129, 1783, 249, 16, 6734, 169, 2, 6008,
  12, 31, 1505, 455, 26865, 452, 94086, 31296, 8583, 4908, 597, 1018, 5, 6313,
  5468, 29202, 2234, 12, 21, 15, 722, 86, 25507, 6, 557168, 1280, 523, 30,
  2172, 4766, 82, 3268, 7, 273, 1312, 173, 61612, 65, 55, 1416, 9866, 8884,
  126, 27, 334, 753, 1583, 10, 150, 169, 168, 588, 9975, 3494, 2, 2, 1875,
  681059, 842, 1841, 3456, 91, 263, 23, 5419, 517, 1128, 323, 34, 81, 1, 2,
  437, 5474, 136, 37, 1607841, 3353, 431, 10, 7822, 46, 2984, 13236, 22, 610,
  181, 37, 152, 26929, 41, 1125, 1148, 511, 7, 146229, 1845, 189, 507, 655, 7,
  156, 1848, 199, 52, 639, 648, 2480, 57, 19, 16498, 3865, 1427, 59307, 183,
  8, 556669, 201, 6642, 6324, 13622, 820, 8700, 4198, 676, 5, 1820, 1140, 109,
  5131, 40, 625, 2093, 10281, 73, 72, 12356, 26599660, 4138, 64814, 1, 4584,
  9832, 5030, 277, 5, 56101, 17, 88, 168, 758036, 1, 40271, 29, 342, 553, 865,
  581765, 1441, 1066, 83, 786590, 4299, 1932, 11111, 693, 24912770, 4, 1222,
  4, 4794, 19196, 31, 4223, 197, 2115, 16, 54855, 68917, 424, 7, 3924, 474,
  1434, 265, 3066, 146, 117, 854, 2, 93, 10408, 198, 6489, 17278, 610201094,
  4988, 7725428, 3015, 112, 208, 184, 19663, 171, 11, 343, 20001, 12, 3723, 6,
  3896, 1058, 13, 997, 1526, 35247, 2, 1010, 2001, 1, 40, 10833, 6084, 41402,
  25309, 13, 475, 559, 222, 6382, 441, 1, 663547, 9717, 702, 1205, 74851, 331,
  747, 295, 1493, 170, 101, 483927, 12436167, 6, 3, 3415, 4578, 329678, 1568,
  729, 48, 6319, 867363, 2414, 1, 248, 4508, 5175, 137, 76564, 12459, 1297,
  143, 251, 679, 1346, 1128, 49, 5179, 264, 71, 89, 3508, 1462, 486, 1549,
  9774, 932, 42, 868, 2297, 686, 96, 473, 4976, 24, 9062, 24, 13463, 2822624,
  82, 4, 787, 40, 7, 312, 111, 2943, 1217, 8795704, 1178, 3131, 43, 825, 18,
  1368, 165, 87406, 417, 270, 2713, 2430, 2750, 1533, 24933, 9561, 57044, 501,
  14707, 66592, 2364, 11462, 85, 599125, 576, 14, 61785, 1, 49, 586, 119, 205,
  1933721, 119, 163, 446, 131, 1163, 71, 1527, 1, 18631, 1731, 2762, 25550,
  17749, 2333, 101, 91986, 19, 8, 14043, 1450, 16, 242, 573, 37, 12, 391492,
  27437, 62, 333, 789, 7, 15330, 35048, 6863, 15672, 12, 43, 1, 45, 26, 2551,
  4771, 1752, 1156, 88, 11010, 1274, 2, 2747, 1666, 560, 9047, 3957, 1, 16218,
  777, 12270, 642, 9, 547, 43, 88, 7939, 1778, 1864499, 821, 1165, 686, 760,
  9, 35, 582, 3777, 63, 53017, 1, 200, 546, 410, 548, 201617, 202, 1677, 71,
  3886, 395, 7041, 5, 25, 2080, 4776, 82, 6, 15953, 1012, 7760, 676, 8561,
  732, 550, 1574, 18585, 29, 577, 7, 2919, 59, 54744, 143949, 34028, 26794, 7,
  141, 7593, 453, 345863, 6364, 429, 7207, 4150, 23162, 111, 8, 23, 3760, 401,
  27, 241, 167, 3139, 16, 149, 11, 398, 1783, 46, 107, 4825, 3639, 36, 276906,
  95, 69, 1024, 125, 8, 82, 16, 928, 8, 39, 513, 6350, 1580, 1083, 68560, 112,
  267, 2064, 110, 332, 27165, 4360, 1219, 27486678, 308, 258, 8177, 291, 87,
  1, 178, 817, 371, 8360, 213, 16628, 19, 21665, 23, 16424, 2, 50, 81, 7696,
  4, 1, 929, 163, 17, 21926, 27566, 4244, 2457, 3456, 68545, 3171, 477, 1848,
  857, 249, 1734, 53, 4983, 2, 3517, 11978, 1, 19786, 2950, 5479, 7945, 2084

]];
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Banco de 2000 datos comprobados y obtenidos bajo supervision que corresponden
al acumulado del trafico de subida mas el tráfico de bajada en bytes para un
periodo de dias para un conjunto de 2000 direcciones MAC.
Este es el banco de control.",
TraChk);
////////////////////////////////////////////////////////////////////////////
```