



INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

Práctica #1

SIGMA

Axel Treviño Palacios

2CM5

31 de Octubre de 2020

1. Objetivo

Programar el universo de las cadenas binarias Σ^n . Dada una 'n' que introduzca el usuario o que el programa lo determine automáticamente. El rango de 'n' debe de estar en el intervalo de [0,1000].

1. El programa debe de preguntar si quiere calcular otra 'n' o no.
2. La salida, expresada en notación de conjunto, debe ir a un archivo de texto.
3. Una segunda salida (archivo de texto) debe concatenar todas las cadenas calculadas en una sola cadena, quitar las llaves, comas y cualquier otro símbolo que no sean 0s y 1s.
4. Del primer archivo de salida, graficar el número de 1s de cada cadena. El eje de las x es la cadena y el eje de las y el número de 1s que tiene esa cadena. Específicamente, calcular y graficar cuando n=23. Al mismo tiempo, calcular la gráfica pero calculando su logaritmo en base 2 y 10 respectivamente.
5. Del segundo archivo de salida, particionar la cadena en subcadenas de longitud 32 y graficar la cantidad de unos de esas subcadenas. Al mismo tiempo, calcular la gráfica pero calculando su logaritmo en base 2 y 10 respectivamente.

2. Codigo

```
1  #include <time.h>
2
3  #include <algorithm>
4  #include <cmath>
5  #include <fstream>
6  #include <iostream>
7  #include <string>
8  #include <vector>
9
10 using namespace std;
11
12 // CONSTANTES
13 const string sigmaFileName = "sigma.txt";
14 const string binarioFileName = "binario.txt";
15 const string graficaFileName = "grafica.dat";
16 const string divisionFileName = "binario32.txt";
17 const string graficaDivisionFileName = "grafica32.dat";
18
19 // VARIABLES
20 ofstream sigmaFile;
21 ofstream binarioFile;
22 ofstream graficaFile;
23 ofstream divisionFile;
24 ofstream graficaDivisionFile;
25
26 // FUNCIONES AUXILIARES
27 int menu() {
28     int respuesta;
29
30     cout << "_____ " << endl;
31     cout << "— Menu —" << endl;
32     cout << "0. Salir" << endl;
33     cout << "1. Insertar limite" << endl;
34     cout << "2. Limite automatico" << endl;
35     cout << "Inserte la opcion: ";
36     cin >> respuesta;
```

```

37
38     return respuesta;
39 }
40
41 // FUNCIONES AUXILIARES
42 vector<bool> dec2bin(long dec, int len) {
43     vector<bool> bits;
44     for (int i = 0; i < len; ++i) {
45         bits.push_back(dec & 1);
46         dec /= 2;
47     }
48     reverse(bits.begin(), bits.end());
49     return bits;
50 }
51 int calcularUnos(vector<bool> binario) {
52     int unos, len;
53     len = binario.size();
54     for (int i = 0; i < len; ++i) {
55         unos += binario[i];
56     }
57     return unos;
58 }
59 void dividirBinario() {
60     ifstream file(binarioFileName);
61     char c;
62     int i = 1;
63
64     while (file.get(c)) {
65         if (i % 33 == 0) {
66             i = 1;
67             divisionFile << endl;
68         }
69         i++;
70         divisionFile << c;
71     }
72 }
73
74 // FUNCIONES DE ARCHIVOS
75 void escribirSigma(vector<bool> binario) {
76     int len = binario.size();
77     for (int i = 0; i < len; ++i) {
78         sigmaFile << binario[i] ? '1' : '0';
79         binarioFile << binario[i] ? '1' : '0';
80     }
81     sigmaFile << ',' << endl;
82 }
83 void escribirGrafica(int num, int unos) {
84     graficaFile << num << '\t';
85     graficaFile << unos << '\t';
86     graficaFile << log2(unos) << '\t';
87     graficaFile << log10(unos) << endl;
88 }
89 void escribirGraficaDividida() {
90     ifstream file(divisionFileName);
91     string line;
92     char c;

```

```

93     int i = 1;
94     int unos, len;
95
96     while (getline(file, line)) {
97         unos = 0;
98         len = line.size();
99         for (int i = 0; i < len; ++i) {
100             unos += line[i] == '1' ? 1 : 0;
101         }
102
103         graficaDivisionFile << i << '\t';
104         graficaDivisionFile << unos << '\t';
105         graficaDivisionFile << log2(unos) << '\t';
106         graficaDivisionFile << log10(unos) << endl;
107         ++i;
108     }
109 }
110 void abrirArchivos() {
111     sigmaFile.open(sigmaFileName);
112     binarioFile.open(binarioFileName);
113     graficaFile.open(graficaFileName);
114     divisionFile.open(divisionFileName);
115     graficaDivisionFile.open(graficaDivisionFileName);
116 }
117 void cerrarArchivos() {
118     sigmaFile.close();
119     binarioFile.close();
120     graficaFile.close();
121     divisionFile.close();
122     graficaDivisionFile.close();
123 }
124
125 // FUNCION PRINCIPAL
126 void procesar(int k) {
127     int unos, limite;
128     bool final = false;
129
130     sigmaFile << "{e," << endl;
131
132     cout << "Iniciando...";
133
134     // Calcular las permutaciones
135     for (int i = 0; i <= k; ++i) {
136         cout << endl
137             << i << ":" << endl;
138         limite = pow(2, i);
139         final = i == k;
140
141         for (long j = 0; j < limite; ++j) {
142             cout << "\r\t" << j;
143             vector<bool> binario = dec2bin(j, i + 1);
144
145             escribirSigma(binario);
146
147             if (final) {
148                 unos = calcularUnos(binario);

```

```

149             escribirGrafica(j, unos);
150         }
151     }
152 }
153
154 dividirBinario();
155 escribirGraficaDividida();
156 }
157
158 // MAIN
159 int main() {
160     int k;
161     string resultado;
162     int respuesta;
163     bool continuar = true;
164
165     while (true) {
166         respuesta = menu();
167
168         switch (respuesta) {
169             case 0:
170                 continuar = false;
171                 break;
172             case 1:
173                 cout << "Inserte k" << endl;
174                 cin >> k;
175                 break;
176             case 2:
177                 srand(time(NULL));
178                 k = rand() % 1000;
179                 cout << "K = " << k;
180                 break;
181         }
182         if (!continuar) {
183             break;
184         }
185
186         abrirArchivos();
187         procesar(k);
188         cerrarArchivos();
189
190         cout << endl
191             << "El resultado esta en los archivos c:";
192     }
193 }

```

3. Conclusiones

Las computadoras, aunque increíblemente potentes, aun necesitan de nuestra ayuda para no matarse a sí mismas.

4. Resultado

```
Windows PowerShell
1. Insertar limite
2. Limite automatico
Inserte la opcion: 1
Inserte k
23
Iniciando...
0:
1: 0
2: 1
3: 3
4: 7
5: 15
6: 31
7: 63
8: 127
9: 255
10: 511
11: 1023
12: 2047
13: 4095
14: 8191
15: 16383
16: 32767
17: 65535
18: 131071
19: 262143
20: 524287
21: 1048575
22: 2097151
23: 4194303
24: 8388607
25: 16777215
26: 33554431
27: 67108863
28: 134217727
29: 268435455
30: 536870911
31: 1073741823
32: 2147483647
33: 4294967295
34: 8589934591
35: 17179869183
36: 34359738367
37: 68719476735
38: 137438953471
39: 274877906943
40: 549755813887
41: 1099511627775
42: 2199023255551
43: 4398046511103
44: 8796093022207
45: 17592186044415
46: 35184372088831
47: 70368744177663
48: 140737488355327
49: 281474976710655
50: 562949953421311
51: 1125899906842623
52: 2251799813685247
53: 4503599627370495
54: 9007199254740991
55: 18014398509481983
56: 36028797018963967
57: 72057594037927935
58: 144115188075855871
59: 288230376151711743
60: 576460752303423487
61: 1152921504606846975
62: 2305843009213693951
63: 4611686018427387903
64: 9223372036854775807
65: 18446744073709551615
66: 36893488147419103231
67: 73786976294838206463
68: 147573952589676412927
69: 295147905179352825855
70: 590295810358705651711
71: 1180591620717411303423
72: 2361183241434822606847
73: 4722366482869645213695
74: 9444732965739290427391
75: 18889465931478580854783
76: 37778931862957161709567
77: 75557863725914323419135
78: 151115727451828646838271
79: 302231454903657293676543
80: 604462909807314587353087
81: 1208925819614629174706175
82: 2417851639229258349412351
83: 4835703278458516698824703
84: 9671406556917033397649407
85: 19342813113834066795298815
86: 38685626227668133590597631
87: 77371252455336267181195263
88: 154742504910672534362390527
89: 309485009821345068724781055
90: 618970019642690137449562111
91: 1237940039285380274899124223
92: 2475880078570760549798248447
93: 4951760157141521099596496895
94: 9903520314283042199192993791
95: 19807040628566084398385987583
96: 39614081257132168796771975167
97: 79228162514264337593543950335
98: 158456325028528675187087900671
99: 316912650057057350374175801343
100: 633825300114114700748351602687
101: 1267650600228229401496703205375
102: 2535301200456458802993406410751
103: 5070602400912917605986812821503
104: 10141204801825835211973625643007
105: 20282409603651670423947251286015
106: 40564819207303340847894502572031
107: 81129638414606681695789005144063
108: 162259276829213363391578010288127
109: 324518553658426726783156020576255
110: 649037107316853453566312041152511
111: 1298074214633706907132624082305023
112: 2596148429267413814265248164610047
113: 5192296858534827628530496329220095
114: 10384593717069655257060992658440191
115: 20769187434139310514121985316880383
116: 41538374868278621028243970633760767
117: 83076749736557242056487941267521535
118: 166153499473114484112975882535043071
119: 332306998946228968225951765070086143
120: 664613997892457936451903530140172287
121: 1329227995784915872903807060280344575
122: 2658455991569831745807614120560689151
123: 5316911983139663491615228241121378303
124: 10633823966279326983230456482242756607
125: 21267647932558653966460912964485513215
126: 42535295865117307932921825928971026431
127: 85070591730234615865843651857942052863
128: 170141183460469231731687303715884105727
129: 340282366920938463463374607431768211455
130: 680564733841876926926749214863536422911
131: 1361129467683753853853498429727072845823
132: 2722258935367507707706996859454145691647
133: 5444517870735015415413993718908291383295
134: 10889035741470030830827987437816582766591
135: 21778071482940061661655974875633165533183
136: 43556142965880123323311949751266331066367
137: 87112285931760246646623899502532662132735
138: 174224571863520493293247799005065324265471
139: 348449143727040986586495598010130648530943
140: 696898287454081973172991196020261297061887
141: 1393796574908163946345982392040522594123775
142: 2787593149816327892691964784081045188247551
143: 5575186299632655785383929568162090376495103
144: 11150372599265311570767859136324180752990207
145: 22300745198530623141535718272648361505980415
146: 44601490397061246283071436545296723011960831
147: 89202980794122492566142873090593446023921663
148: 178405961588244985132285746181186892047843327
149: 356811923176489970264571492362373784095686655
150: 713623846352979940529142984724747568191373311
151: 1427247692705959881058285969449495136382746623
152: 2854495385411919762116571938898990272765493247
153: 5708990770823839524233143877797980545530986495
154: 11417981541647679048466287755595961091061972991
155: 22835963083295358096932575511191922182123945983
156: 45671926166590716193865151022383844364247891967
157: 91343852333181432387730302044767688728495783935
158: 182687704666362864775460604089535377456991567871
159: 365375409332725729550921208179070754913983135743
160: 730750818665451459101842416358141509827966271487
161: 1461501637330902918203684832716283019655932542975
162: 2923003274661805836407369665432566039311865085951
163: 5846006549323611672814739330865132078623730171903
164: 11692013098647223345629478661730264157247460343807
165: 23384026197294446691258957323460528314494920687615
166: 46768052394588893382517914646921056628989841375231
167: 93536104789177786765035829293842113257979682750463
168: 187072209578355573530071658587684226515959365500927
169: 374144419156711147060143317175368453031918731001855
170: 748288838313422294120286634350736906063837462003711
171: 1496577676626844588240573268701473812127674924007423
172: 2993155353253689176481146537402947624255349848014847
173: 5986310706507378352962293074805895248510699696029695
174: 11972621413014756705924586149611790497021399392059391
175: 23945242826029513411849172299223580994042798784118783
176: 47890485652059026823698344598447161988085597568237567
177: 95780971304118053647396689196894323976171195136475135
178: 191561942608236107294793378393788647952342390272950271
179: 383123885216472214589586756787577295904684780545900543
180: 766247770432944429179173513575154591809369561091801087
181: 1532495540865888858358347027150309183618739122183602175
182: 3064991081731777716716694054300618367237478244367204351
183: 6129982163463555433433388108601236734474956488734408703
184: 12259964326927110866866776217202473468949912977468817407
185: 24519928653854221733733552434404946937899825954937634815
186: 49039857307708443467467104868809893875799651909875269631
187: 98079714615416886934934209737619787751599303819750539263
188: 196159429230833773869868419475239575503198607639501078527
189: 392318858461667547739736838950479151006397215279002157055
190: 784637716923335095479473677900958302012794430558004314111
191: 1569275433846670190958947355801916604025588861116008628223
192: 3138550867693340381917894711603833208051177722232017256447
193: 6277101735386680763835789423207666416102355444464034512895
194: 12554203470773361527671578846415332832204710888928069025791
195: 25108406941546723055343157692830665664409421777856138051583
196: 50216813883093446110686315385661331328818843555712276103167
197: 100433627766186892221372630771322662657637687111424552206335
198: 200867255532373784442745261542645325315275374222849104412671
199: 401734511064747568885490523085290650630550748445698208825343
200: 803469022129495137770981046170581301261101496891396417650687
201: 1606938044258990275541962092341162602522202993782792835301375
202: 3213876088517980551083924184682325205044405987565585670602751
203: 6427752177035961102167848369364650410088811975131171341205503
204: 12855504354071922204335696738729300820177623950262342682411007
205: 25711008708143844408671393477458601640355247900524685364822015
206: 51422017416287688817342786954917203280710495801049370729644031
207: 102844034832575377634685573909834406561420991602098741459288063
208: 205688069665150755269371147819668813122841983204197482918576127
209: 411376139330301510538742295639337626245683966408394965837152255
210: 822752278660603021077484591278675252491367932816789931674304511
211: 1645504557321206042154969182557350504982735865633579863348609023
212: 3291009114642412084309938365114701009965471731267159726697218047
213: 6582018229284824168619876730229402019930943462534319453394436095
214: 13164036458569648337239753460458804039861886925068638906788872191
215: 26328072917139296674479506920917608079723773850137277813577744383
216: 52656145834278593348959013841835216159447547700274555627155488767
217: 105312291668557186697918027683670432318895095400549111254310977535
218: 210624583337114373395836055367340864637790190801098222508621955071
219: 421249166674228746791672110734681729275580381602196445017243910143
220: 842498333348457493583344221469363458551160763204392890034487820287
221: 1684996666696914987166688442938726917102321526408785780068975640575
222: 3369993333393829974333376885877453834204643052817571560137951281151
223: 6739986666787659948666753771754907668409286105635143120275902562303
224: 13479973333575319897333507543509815336818572211270286240551805124607
225: 26959946667150639794667015087019630673637144422540572481103610249215
226: 53919893334301279589334030174039261347274288845081144962207220498431
227: 107839786668602559178668060348078522694548577690162289924414440996863
228: 215679573337205118357336120696157045389097155380324579848828881993727
229: 431359146674410236714672241392314090778194310760649159697657763987455
230: 862718293348820473429344482784628181556388621521298319395315527974911
231: 1725436586697640946858688965569256363112777243042596638790631055949823
232: 3450873173395281893717377931138512726225554486085193277581262111899647
233: 6901746346790563787434755862277025452451108972170386555162524223799295
234: 13803492693581127574869511724554050904902217944340773110325048447598591
235: 27606985387162255149739023449108101809804435888681546220650096895197183
236: 55213970774324510299478046898216203619608871777363092441300193790394367
237: 110427941548649020598956093796432407239217743554726184882600387580788735
238: 220855883097298041197912187592864814478435487109452369765200775161577471
239: 441711766194596082395824375185729628956870974218904739530401550323154943
240: 883423532389192164791648750371459257913741948437809479060803100646309887
241: 1766847064778384329583297500742918515827483896875618958121606201292619775
242: 3533694129556768659166595001485837031654967793751237916243212402585239551
243: 7067388259113537318333190002971674063309935587502475832486424805170479103
244: 14134776518227074636666380005943348126619871175004951664972849610340958207
245: 28269553036454149273332760011886696253239742350009903329945699220681916415
246: 56539106072908298546665520023773392506479484700019806659891398441363832831
247: 113078212145816597093331040047546785012958969400039613319782796882727665663
248: 226156424291633194186662080095093570025917938800079226639565593765455331327
249: 452312848583266388373324160190187140051835877600158453279131187530910662655
250: 904625697166532776746648320380374280103671755200316906558262375061821325311
251: 1809251394333065553493296640760748560207343510400633813116524750123642650623
252: 3618502788666131106986593281521497120414687020801267626233049500247285301247
253: 7237005577332262213973186563042994240829374041602535252466099000494570602495
254: 14474011154664524427946373126085988481658748083205070504932198000989141204991
255: 28948022309329048855892746252171976963317496166410141009864396001978282409983
256: 57896044618658097711785492504343953926634992332820282019728792003956564819967
257: 115792089237316195423570985008687907853269984665640564039457584007913129639935
258: 231584178474632390847141970017375815706539969331281128078915168015826259279871
259: 463168356949264781694283940034751631413079938662562256157830336031652518559743
260: 926336713898529563388567880069503262826159877325124512315660672063305037119487
261: 1852673427797059126777135760139006525652319754650249024631321344126610074238975
262: 3705346855594118253554271520278013051304639509300498049262642688253220148477951
263: 7410693711188236507108543040556026102609279018600996098525285376506440296955903
264: 14821387422376473014217086081112052205218558037201992197050570753012880593911807
265: 29642774844752946028434172162224104410437116074403984394101141506025761187823615
266: 59285549689505892056868344324448208820874232148807968788202283012051522375647231
267: 11857109937901178411373668864889641764174
```

