Capítulo 1

Una Primer Mirada al Álgebra Computacional

El objetivo de estas notas es para que sirvan como guía para el curso de Álgebra Abstracta y Computacional en tiempos del Covid-19 (y al parecer post-Covid). ¡Así no hay excusas para no estudiar! Como dice la *gran* politóloga ESTUDIEN...
Cualquier corrección o sugerencia me pueden escribir al correo jlramirezr@unal.edu.co

1.1. Álgebra Computacional

El álgebra computacional se puede considerar como parte de las Matemáticas y Ciencias de la Computación, la cual se dedica al diseño, análisis, implementación y aplicación de algoritmos algebraicos. En contraste con el cálculo numérico, el énfasis en el álgebra computacional está en el cálculo y manipulación de símbolos que representan conceptos matemáticos. Esto no implica que no se hagan cálculos numéricos. Por ejemplo, se pueden hacer manipulaciones con números enteros como la siguiente:

```
In[1]:= 666^10*2021 - 369^12

Out[1]= 2832573646049422792367501747833 5
```

La complejidad de las operaciones básicas como suma y multiplicación de enteros es uno de los primeros problemas que abordaremos, junto con otros algoritmos como el del máximo común divisor.

A medida que vayamos desarrollando los contenidos matemáticos necesarios para entender los diferentes algoritmos, iremos interactuando con el software *Mathematica*®, el cual es un paquete clásico de computación simbólica. Los ejemplos que desarrollemos a lo

largo de estas notas de clase pueden ser adaptados a otros paquetes como *Maple, Maxima, SageMath*, entre otros.

Adicional a la manipulación de números enteros, también se puede manipular números racionales, decimales, entre muchos otros.

In[2]:= $\frac{1}{2} + \frac{5}{3} - 4$

Out[2]= $-\frac{11}{6}$

In[3]:= $\sqrt{2} - \frac{1}{\sqrt{3}}$

Out[3]= $\sqrt{2} - \frac{1}{\sqrt{3}}$

In[4]:= $\sqrt{2} - \frac{1}{\sqrt{3}}$ `// Together`

Out[4]= $\frac{1}{3}(3\sqrt{2} - \sqrt{3})$

```
In[5]:= 5 - 0.6*3.4
```

```
Out[5]= 2.96
```

Los sistemas de álgebra computacional se pueden utilizar como potentes calculadoras. Por ejemplo, para el cálculo del máximo común divisor.

```
In[6]:= GCD[2^30 + 1, 3^15 - 1]
```

```
Out[6]= 13
```

Para calcular los primeros cien decimales de la expansión de los números π^e y e^π

```
In[7]:= N[π^E, 100]
```

```
Out[7]= 22.459157718361045473427152204543735027589315133996692249
        2030025540669260403991179123185197527271430 3
```

```
In[8]:= N[E^π, 100]
```

```
Out[8]= 23.140692632779269005729086367948547380266106242600211993
        4450464095243423506904527835169719970675492 2
```

o los primeros mil dígitos de π, lo cual se calcula en una fracción pequeña de tiempo:

```
In[9]:= Timing[N[π, 1000]]
```

Out[9]= {0.00001,3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481846766940513200056812714526356082778577134275778960917363717872146844090122495343014654958537105079227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105973173281609631859502445945534690830264252230825334468503526193118817101000313783875288658753320838142061717766914730359825349042875546873115956286388235378759375195778185778053217122680661300192787661119590921642019 9}

Hoy en día los sistemas de cómputo simbólico permiten manipular una gran variedad de objetos matemáticos como polinomios, matrices, sumas, integrales, derivadas, series formales, sistemas de ecuaciones, figuras geométricas, grafos, cadenas de texto, entre muchos otros. Dentro de estas temáticas abordaremos el estudio de la aritmética y factorización de polinomios. Por ejemplo, estudiaremos algunos algoritmos clásicos de multiplicación de polinomios que resultan ser más eficientes que el algoritmo de la escuela.

In[10]:=
```
pol1 = Sum[Random[Integer, {1, 2022}] x^k, {k, 0, 100}]
```

Out[10]= $1427 + 578\,x + 972\,x^2 + 1018\,x^3 + 1202\,x^4 + 535\,x^5 + 1344\,x^6 + 1564\,x^7 + 1144\,x^8 + 522\,x^9 + 1963\,x^{10} + 626\,x^{11} + 1763\,x^{12} + 465\,x^{13} + 564\,x^{14} + 21\,x^{15} + 1619\,x^{16} + 109\,x^{17} + 1514\,x^{18} + 1288\,x^{19} + 68\,x^{20} + 793\,x^{21} + 1924\,x^{22} + 1446\,xx^{23} + 1729\,x^{24} + 219\,x^{25} + 1135\,x^{26} + 1125\,x^{27} + 1614\,x^{28} + 1303\,x^{29} + 1685\,x^{30} + 536\,x^{31} + 1126\,x^{32} + 1080\,x^{33} + 1570\,x^{34} + 860\,x^{35} + 1105\,x^{36} + 1187\,x^{37} + 192\,x^{38} + 1589\,x^{39} + 1925\,x^{40} + 1819\,x^{41} + 316\,x^{42} + 357\,x^{43} + 2004\,x^{44} + 1885\,x^{45} + 717\,x^{46} + 1735\,x^{47} + 1578\,x^{48} + 1684\,x^{49} + 1336\,x^{50} + 843\,x^{51} + 816\,x^{52} + 931\,x^{53} + 1510\,x^{54} + 52\,x^{55} + 1001\,x^{56} + 837\,x^{57} + 15\,x^{58} + 70\,x^{59} + 42\,x^{60} + 1920\,x^{61} + 1053\,x^{62} + 14\,x^{63} + 706\,x^{64} + 1126\,x^{65} + 1520\,x^{66} + 1594\,x^{67} + 1156\,x^{68} + 1036\,x^{69} + 556\,x^{70} + 147\,x^{71} + 181\,x^{72} + 847\,x^{73} + 514\,x^{74} + 1759\,x^{75} + 1688\,x^{76} + 1921\,x^{77} + 1895\,x^{78} + 964\,x^{79} + 1775\,x^{80} + 1847\,x^{81} + 1411\,x^{82} + 577\,x^{83} + 59\,x^{84} + 966\,x^{85} + 1077\,x^{86} + 1120\,x^{87} + 1848\,x^{88} + 1234\,x^{89} + 287\,x^{90} + 1358\,x^{91} + 743\,x^{92} + 84\,x^{93} + 1901\,x^{94} + 1708\,x^{95} + 534\,x^{96} + 1664\,x^{97} + 1452\,x^{98} + 1136\,x^{99} + 1652\,x^{100}$

In[11]:=
```
pol2 = Sum[Random[Integer, {1, 2022}] x^k, {k, 0, 100}]
```

Out[11]= $1817 + 1160\,x + 280\,x^2 + 154\,x^3 + 1621\,x^4 + 72\,x^5 + 284\,x^6 + 1700\,x^7 + 830\,x^8 + 1599\,x^9 + 1703\,x^{10} + 1823\,x^{11} + 607\,x^{12} + 1099\,x^{13} + 1357\,x^{14} + 1960\,x^{15} + 597\,x^{16} + 180\,x^{17} + 1348\,x^{18} + 1978\,x^{19} + 895\,x^{20} + 812\,x^{21} +$

$$350\,x^{22} + 1014\,x^{23} + 365\,x^{24} + 559\,x^{25} + 1124\,x^{26} + 309\,x^{27} + 1956\,x^{28} +$$
$$95\,x^{29} + 827\,x^{30} + 1062\,x^{31} + 2004\,x^{32} + 156\,x^{33} + 606\,x^{34} + 291\,x^{35} +$$
$$988\,x^{36} + 498\,x^{37} + 1705\,x^{38} + 1738\,x^{39} + 1017\,x^{40} + 1148\,x^{41} + 1508\,x^{42} +$$
$$466\,x^{43} + 1643\,x^{44} + 494\,x^{45} + 368\,x^{46} + 579\,x^{47} + 76\,x^{48} + 1332\,x^{49} +$$
$$640\,x^{50} + 1267\,x^{51} + 249\,x^{52} + 1376\,x^{53} + 1212\,x^{54} + 249\,x^{55} + 409\,x^{56} +$$
$$9\,x^{57} + 1565\,x^{58} + 934\,x^{59} + 1881\,x^{60} + 611\,x^{61} + 579\,x^{62} + 1640\,x^{63} +$$
$$993\,x^{64} + 337\,x^{65} + 333\,x^{66} + 709\,x^{67} + 708\,x^{68} + 31\,x^{69} + 386\,x^{70} +$$
$$1226\,x^{71} + 704\,x^{72} + 1655\,x^{73} + 1800\,x^{74} + 1621\,x^{75} + 1577\,x^{76} + 1149\,x^{77} +$$
$$1355\,x^{78} + 1775\,x^{79} + 1486\,x^{80} + 1619\,x^{81} + 2013\,x^{82} + 623\,x^{83} +$$
$$1103\,x^{84} + 245\,x^{85} + 1272\,x^{86} + 276\,x^{87} + 869\,x^{88} + 258\,x^{89} + 1552\,x^{90} +$$
$$1243\,x^{91} + 1202\,x^{92} + 1693\,x^{93} + 31\,x^{94} + 883\,x^{95} + 509\,x^{96} + 927\,x^{97} +$$
$$362\,x^{98} + 1609\,x^{99} + 921\,x^{100}$$

El producto de los dos polinomios anteriores se hace a penas en una fracción de segundos:

In[12]:=
```
Timing[pol1*pol2 // Expand]
```

Out[12]=
$$\{0.000429,$$
$$2592859 + 2705546\,x + 2836164\,x^2 + 3358824\,x^3 + 6039253\,x^4 +$$
$$3840825\,x^5 + 5578476\,x^6 + 9045950\,x^7 + 8816394\,x^8 +$$
$$8577214\,x^9 + 13183742\,x^{10} + 14549133\,x^{11} + 14018428\,x^{12} +$$
$$14815850\,x^{13} + 17393005\,x^{14} + 16160176\,x^{15} + 19410092\,x^{16} +$$
$$19474749\,x^{17} + 20748007\,x^{18} + 24105861\,x^{19} + 22310979\,x^{20} +$$
$$23058559\,x^{21} + 25700321\,x^{22} + 28108885\,x^{23} + 22112580\,x^{24} +$$
$$27084298\,x^{25} + 28380693\,x^{26} + 27453944\,x^{27} + 30186765\,x^{28} +$$
$$29796168\,x^{29} + 32402532\,x^{30} + 34063678\,x^{31} + 34572396\,x^{32} +$$
$$35319919\,x^{33} + 36945002\,x^{34} + 35385828\,x^{35} + 37268865\,x^{36} +$$
$$41148286\,x^{37} + 42994935\,x^{38} + 41153565\,x^{39} + 44516635\,x^{40} +$$
$$47038480\,x^{41} + 48109383\,x^{42} + 44049486\,x^{43} + 53290106\,x^{44} +$$
$$48690330\,x^{45} + 49078678\,x^{46} + 50203134\,x^{47} + 56930853\,x^{48} +$$
$$55217375\,x^{49} + 56919085\,x^{50} + 56347233\,x^{51} + 54705155\,x^{52} +$$
$$52629663\,x^{53} + 64157585\,x^{54} + 59061072\,x^{55} + 59080145\,x^{56} +$$
$$55266887\,x^{57} + 61936194\,x^{58} + 58373395\,x^{59} + 67020704\,x^{60} +$$
$$63026733\,x^{61} + 63884412\,x^{62} + 65105326\,x^{63} + 62032766\,x^{64} +$$
$$66446207\,x^{65} + 65585950\,x^{66} + 66226283\,x^{67} + 69421024\,x^{68} +$$
$$62994772\,x^{69} + 64670599\,x^{70} + 66322740\,x^{71} + 70347698\,x^{72} +$$
$$67164369\,x^{73} + 63961394\,x^{74} + 74021310\,x^{75} + 79218361\,x^{76} +$$
$$79115782\,x^{77} + 74598062\,x^{78} + 81205532\,x^{79} + 83800419\,x^{80} +$$
$$86172217\,x^{81} + 89809118\,x^{82} + 85637180\,x^{83} + 88664264\,x^{84} +$$
$$90492672\,x^{85} + 95284356\,x^{86} + 91777736\,x^{87} + 95941915\,x^{88} +$$
$$96248694\,x^{89} + 94345056\,x^{90} + 90969013\,x^{91} + 98494281\,x^{92} +$$
$$95811458\,x^{93} + 96129194\,x^{94} + 99541662\,x^{95} + 97455485\,x^{96} +$$
$$97591647\,x^{97} + 105737793\,x^{98} + 102053079\,x^{99} + 105393607\,x^{100} +$$
$$101315351\,x^{101} + 97538183\,x^{102} + 103961511\,x^{103} + 102643866\,x^{104} +$$
$$102840189\,x^{105} + 100257092\,x^{106} + 104668952\,x^{107} + 102688606\,x^{108} +$$
$$99742407\,x^{109} + 95175226\,x^{110} + 90493593\,x^{111} + 91200697\,x^{112} +$$
$$88239927\,x^{113} + 91957626\,x^{114} + 90479327\,x^{115} + 87049248\,x^{116} +$$
$$87133821\,x^{117} + 86391461\,x^{118} + 90555862\,x^{119} + 89392578\,x^{120} +$$
$$85192369\,x^{121} + 81622123\,x^{122} + 80754065\,x^{123} + 77419774\,x^{124} +$$
$$81670126\,x^{125} + 82187020\,x^{126} + 83021242\,x^{127} + 76036573\,x^{128} +$$
$$78395770\,x^{129} + 74462042\,x^{130} + 70774963\,x^{131} + 69173723\,x^{132} +$$
$$66221084\,x^{133} + 62658957\,x^{134} + 65607565\,x^{135} + 68622793\,x^{136} +$$

$$62938861\,x^{137} + 67515454\,x^{138} + 66602726\,x^{139} + 68152800\,x^{140} + 63003819\,x^{141} + 57965919\,x^{142} + 56064108\,x^{143} + 56685846\,x^{144} + 51865219\,x^{145} + 51375674\,x^{146} + 53043705\,x^{147} + 53453276\,x^{148} + 54232594\,x^{149} + 47336793\,x^{150} + 50431913\,x^{151} + 49312635\,x^{152} + 49559526\,x^{153} + 49947818\,x^{154} + 47819993\,x^{155} + 47605083\,x^{156} + 49755349\,x^{157} + 50036905\,x^{158} + 48486298\,x^{159} + 47713925\,x^{160} + 45964787\,x^{161} + 43323837\,x^{162} + 40087156\,x^{163} + 38012431\,x^{164} + 38591473\,x^{165} + 39051456\,x^{166} + 39526200\,x^{167} + 41769627\,x^{168} + 37861904\,x^{169} + 39201246\,x^{170} + 38222005\,x^{171} + 36893911\,x^{172} + 39053638\,x^{173} + 35736461\,x^{174} + 34045403\,x^{175} + 32745401\,x^{176} + 30959492\,x^{177} + 29321493\,x^{178} + 28047230\,x^{179} + 25363750\,x^{180} + 23582763\,x^{181} + 18881208\,x^{182} + 14263307\,x^{183} + 16634236\,x^{184} + 15871638\,x^{185} + 15460127\,x^{186} + 16403055\,x^{187} + 15145608\,x^{188} + 11451255\,x^{189} + 13826918\,x^{190} + 11508462\,x^{191} + 8786019\,x^{192} + 9210531\,x^{193} + 8028197\,x^{194} + 6417586\,x^{195} + 5588754\,x^{196} + 5811448\,x^{197} + 3763140\,x^{198} + 3704324\,x^{199} + 1521492\,x^{200}\}$$

Otro de los problemas que abordaremos es el de la factorización de polinomios.

In[13]:=
```
pol1 = x^365 + 1;
```

In[14]:=
```
Factor[pol1]
```

Out[14]=
$$(1 + x)(1 - x + x^2 - x^3 + x^4)(1 - x + x^2 - x^3 + x^4 - x^5 + x^6 - x^7 + x^8 - x^9 + x^{10} - x^{11} + x^{12} - x^{13} + x^{14} - x^{15} + x^{16} - x^{17} + x^{18} - x^{19} + x^{20} - x^{21} + x^{22} - x^{23} + x^{24} - x^{25} + x^{26} - x^{27} + x^{28} - x^{29} + x^{30} - x^{31} + x^{32} - x^{33} + x^{34} - x^{35} + x^{36} - x^{37} + x^{38} - x^{39} + x^{40} - x^{41} + x^{42} - x^{43} + x^{44} - x^{45} + x^{46} - x^{47} + x^{48} - x^{49} + x^{50} - x^{51} + x^{52} - x^{53} + x^{54} - x^{55} + x^{56} - x^{57} + x^{58} - x^{59} + x^{60} - x^{61} + x^{62} - x^{63} + x^{64} - x^{65} + x^{66} - x^{67} + x^{68} - x^{69} + x^{70} - x^{71} + x^{72})$$
$$(1 + x - x^5 - x^6 + x^{10} + x^{11} - x^{15} - x^{16} + x^{20} + x^{21} - x^{25} - x^{26} + x^{30} + x^{31} - x^{35} - x^{36} + x^{40} + x^{41} - x^{45} - x^{46} + x^{50} + x^{51} - x^{55} - x^{56} + x^{60} + x^{61} - x^{65} - x^{66} + x^{70} + x^{71} - x^{73} - x^{74} - x^{75} - x^{76} + x^{78} + x^{79} + x^{80} + x^{81} - x^{83} - x^{84} - x^{85} - x^{86} + x^{88} + x^{89} + x^{90} + x^{91} - x^{93} - x^{94} - x^{95} - x^{96} + x^{98} + x^{99} + x^{100} + x^{101} - x^{103} - x^{104} - x^{105} - x^{106} + x^{108} + x^{109} + x^{110} + x^{111} - x^{113} - x^{114} - x^{115} - x^{116} + x^{118} + x^{119} + x^{120} + x^{121} - x^{123} - x^{124} - x^{125} - x^{126} + x^{128} + x^{129} + x^{130} + x^{131} - x^{133} - x^{134} - x^{135} - x^{136} + x^{138} + x^{139} + x^{140} + x^{141} - x^{143} - x^{144} - x^{145} + x^{147} + x^{148} + x^{149} + x^{150} - x^{152} - x^{153} - x^{154} - x^{155} + x^{157} + x^{158} + x^{159} + x^{160} - x^{162} - x^{163} - x^{164} - x^{165} + x^{167} + x^{168} + x^{169} + x^{170} - x^{172} - x^{173} - x^{174} - x^{175} + x^{177} + x^{178} + x^{179} + x^{180} - x^{182} - x^{183} - x^{184} - x^{185} + x^{187} + x^{188} + x^{189} + x^{190} - x^{192} - x^{193} - x^{194} - x^{195} + x^{197} + x^{198} + x^{199} + x^{200} - x^{202} - x^{203} - x^{204} - x^{205} + x^{207} + x^{208} + x^{209} + x^{210} - x^{212} - x^{213} - x^{214} - x^{215} + x^{217} + x^{218} - x^{222} - x^{223} + x^{227} + x^{228} - x^{232} - x^{233} + x^{237} + x^{238} - x^{242} - x^{243} + x^{247} + x^{248} - x^{252} - x^{253} + x^{257} + x^{258} - x^{262} - x^{263} + x^{267} + x^{268} - x^{272} - x^{273} + x^{277} + x^{278} - x^{282} - x^{283} + x^{287} + x^{288})$$

In[15]:=
```
Factor[pol1, Modulus -> 5]
```

```
Out[15]= (1 + x)^5 (1 + 4 x + x^2 + 4 x^3 + x^4 + 4 x^5 + x^6 + 4 x^7 + x^8 +
     4 x^9 + x^10 + 4 x^11 + x^12 + 4 x^13 + x^14 + 4 x^15 + x^16 +
     4 x^17 + x^18 + 4 x^19 + x^20 + 4 x^21 + x^22 + 4 x^23 + x^24 +
     4 x^25 + x^26 + 4 x^27 + x^28 + 4 x^29 + x^30 + 4 x^31 + x^32 +
     4 x^33 + x^34 + 4 x^35 + x^36 + 4 x^37 + x^38 + 4 x^39 + x^40 +
     4 x^41 + x^42 + 4 x^43 + x^44 + 4 x^45 + x^46 + 4 x^47 + x^48 +
     4 x^49 + x^50 + 4 x^51 + x^52 + 4 x^53 + x^54 + 4 x^55 + x^56 +
     4 x^57 + x^58 + 4 x^59 + x^60 + 4 x^61 + x^62 + 4 x^63 + x^64 +
     4 x^65 + x^66 + 4 x^67 + x^68 + 4 x^69 + x^70 + 4 x^71 + x^72)^5
```

Relacionado al tema de polinomios, otro problema interesante es el de la solución de sistemas de ecuaciones polinomiales de la forma:

$$\begin{cases} f_1(x_1, \ldots, x_n) & = 0, \\ f_2(x_1, \ldots, x_n) & = 0, \\ & \vdots \\ f_m(x_1, \ldots, x_n) & = 0, \end{cases}$$

donde $f_i(x_1, \ldots, x_m)$ son polinomios en n variables con coeficientes en un cuerpo. Observe que si el número de variables es 1 $(n = 1)$ entonces se debe calcular y factorizar el máximo común divisor de cada uno de los polinomios f_i. El cálculo del máximo común divisor se puede abordar con el Algoritmo de Euclides, el cual desarrollaremos con detalle. Por ejemplo, considere los siguientes polinomios f_1, f_2 y f_3:

```
In[16]:= f1 = 1 - 3 x/4 - 21 x^2/4 - x^3/4 + 33 x^4/4 + 6 x^5 + x^6;
         f2 = -2 + 11 x - 113 x^2/8 - 69 x^3/4 + 163 x^4/4 + 5 x^5/2 - 249 x^6/8
         + 11 x^7/4 + 13 x^8/2 + x^9;
         f3 = -2 + 9 x/2 + 21 x^2/4 - 10 x^3 - 27 x^4/4 + 3 x^5 + x^6;
```

```
In[17]:= Solve[{f1 == 0 && f2 == 0 && f3 == 0}]
```

```
Out[17]= {{x -> -4}, {x -> -1}, {x -> -1}, {x -> 1/2}, {x -> 1/2}}
```

En este caso el máximo común divisor de los polinomios f_1, f_2 y f_3 es

```
In[18]:= Factor[PolynomialGCD[f1, f2, f3]]
```

```
Out[18]= 1/8 (1 + x)^2 (4 + x) (-1 + 2 x)^2
```

Si los polinomios son de grado 1, entonces este problema se puede abordar desde el álgebra lineal por medio de la Eliminación de Gauss. Por ejemplo, considere los siguientes polinomios f_1, f_2 y f_3

```
In[19]:= f1 = 3 x - 2 y + z + 1/3;
         f2 = x + y - 4 z - 1;
         f3 = 2 x - 5 z;
```

En este caso estamos interesados en resolver el sistema de ecuaciones lineales $f_1 = 0, f_2 = 0, f_3 = 0$. Consideremos la matriz aumentada del sistema de ecuaciones:

```
In[20]:= A = {{3, -2, 1}, {1, 1, -4}, {2, 0, -5}};
         b = {-1/3, 1, 0};
```

```
In[21]:= MatrixForm[B = Transpose[Join[Transpose[A], {b}]]]
```

$$\text{Out[21]=} \begin{pmatrix} 3 & -2 & 1 & -\frac{1}{3} \\ 1 & 1 & -4 & 1 \\ 2 & 0 & -5 & 0 \end{pmatrix}$$

La forma escalonada reducida de la matriz aumentada es

```
In[22]:= MatrixForm[r = RowReduce[B]]
```

$$\text{Out[22]=} \begin{pmatrix} 1 & 0 & 0 & \frac{25}{33} \\ 0 & 1 & 0 & \frac{16}{11} \\ 0 & 0 & 1 & \frac{10}{33} \end{pmatrix}$$

En este caso la última columna es la solución del sistema

```
In[23]:= sol = r[[All, -1]]
```

```
Out[23]= {25/33, 16/11, 10/33}
```

```
In[24]:= A . sol == b
```

```
Out[24]= True
```

El sistema se puede solucionar directamente con el comando `Solve`.

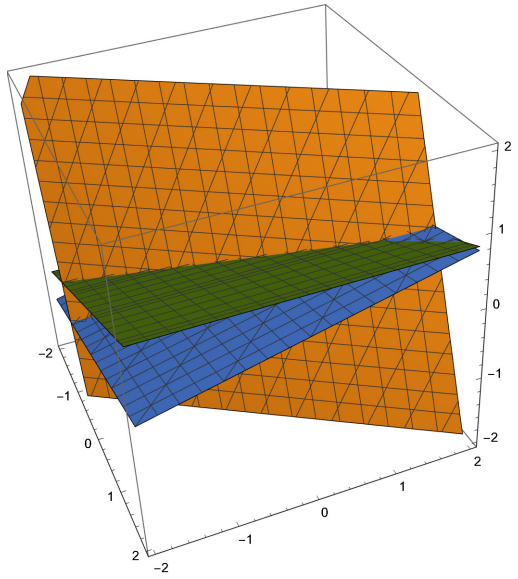
```
In[25]:= Solve[{f1 == 0 && f2 == 0 && f3 == 0}]
```

```
Out[25]= {{x -> 25/33, y -> 16/11, z -> 10/33}}
```

Adicionalmente se puede tener la interpretación geométrica:

```
In[26]:= ContourPlot3D[{f1 == 0, f2 == 0, f3 == 0}, {x, -2, 2},
         {y, -2, 2}, {z, -2, 2}]
```

Out[26]=



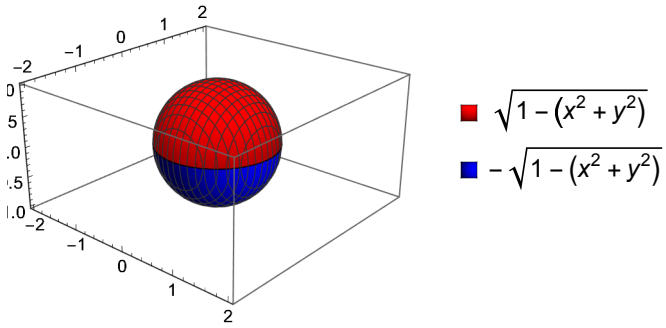
Cuando el grado de los polinomios es mayor que uno el problema lo abordaremos mediante el resultante y bases de Gröbner. Una primera aplicación de este tema consiste en determinar la intersección de curvas. Por ejemplo, cuál es la intersección de la curvas definidas por las ecuaciones

$$x^2 + y^2 + z^2 = 1 \quad \text{y} \quad z = x^2 + y^2 - 1.$$

Las gráficas definidas por estas ecuaciones son:

In[27]:=
```
Plot3D[{Sqrt[1 - (x^2 + y^2)], -Sqrt[1 - (x^2 + y^2)]},
{x, -2,  2}, {y, -2, 2}, PlotPoints -> 80,
BoxRatios -> Automatic, PlotLegends -> "Expressions",
PlotStyle -> {Red, Blue}]
```

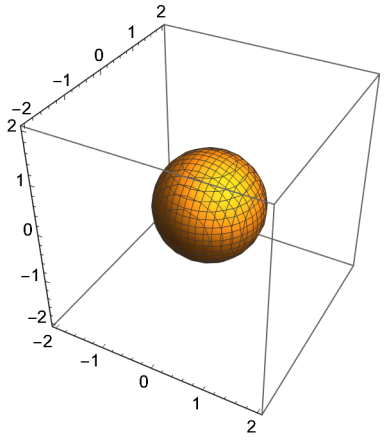
Out[27]=



Otra forma de hacer la gráfica de la esfera es con el comando `ContourPlot3D`.

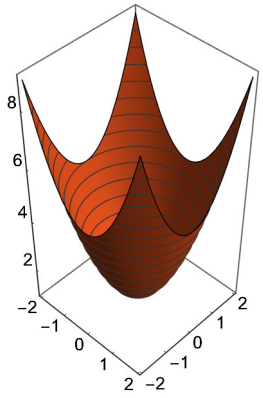
In[28]:=
```
ContourPlot3D[{z^2 + x^2 + y^2 == 1},
{x, -2, 2}, {y, -2, 2}, {z, -2, 2}]
```

Out[28]=



In[29]:=
```
Plot3D[x^2 + y^2 + 1, {x, -2, 2}, {y, -2, 2},
PlotPoints -> 100, BoxRatios -> Automatic,
PlotTheme -> "Web"]
```

Out[29]=



La intersección de estas curvas se puede determinar mediante bases de Gröbner. Básicamente se determina un sistema de ecuaciones equivalente al original.

In[30]:=
```
h = x^2 + y^2 + z^2 - 1;
g = z - (x^2 + y^2 - 1);
```

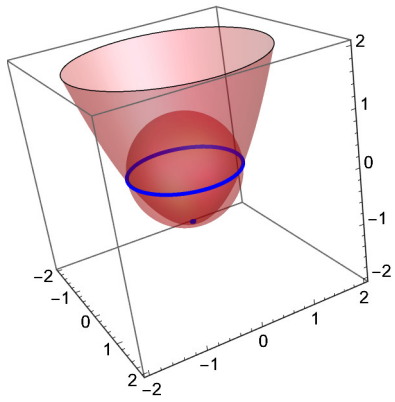
In[31]:=
```
GroebnerBasis[{h, g}, {x, y, z}]
```

Out[31]=
```
{z + z^2, -1 + x^2 + y^2 - z}
```

Observe que las soluciones de la ecuación $z+z^2=0$ son $z=0$ y $z=-1$. Si $z=0$ entonces la segunda ecuación se reduce a $x^2+y^2=1$ lo cual representa una circunferencia de radio 1. Si $z=-1$, entonces $x^2+y^2=0$, es decir $(x,y)=(0,0)$. En conclusión la intersección es una circunferencia sobre el plano $z=0$ y el punto $(0,0,-1)$.

In[32]:=
```
ContourPlot3D[{h == 0, g == 0}, {x, -2, 2}, {y, -2, 2},
{z, -2, 2},  MeshFunctions -> {Function[{x, y, z, f}, h - g]},
MeshStyle -> {{Thick, Blue}}, Mesh -> {{0}},
ContourStyle -> Directive[Red, Opacity[0.3],
Specularity[White, 10]],  PlotPoints -> 120]
```

Out[32]=



El último problema que abordaremos es el del cálculo de sumas combinatorias. El objetivo es estudiar algunos algoritmos sobre sumas hipergeométricas que permiten obtener de manera automática y simbólica sumas que involcuran términos como factoriales y coeficientes binomiales.

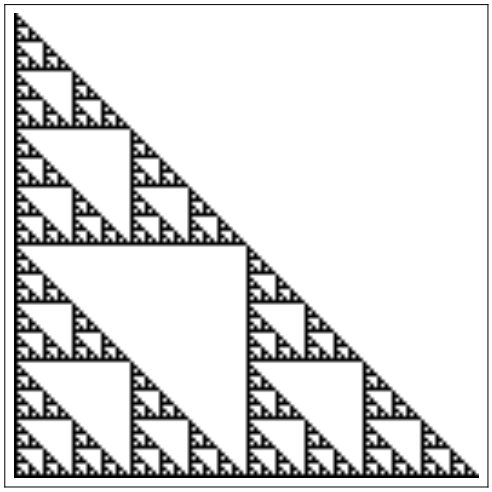
In[33]:=
```
Sum[Binomial[n, i], {i, 0, n}]
```

Out[33]= 2^{n}

Cabe resaltar que los sistemas de computo simbólico cuentan con una gran capacidad para graficar y manipular objetos en el plano y el espacio.

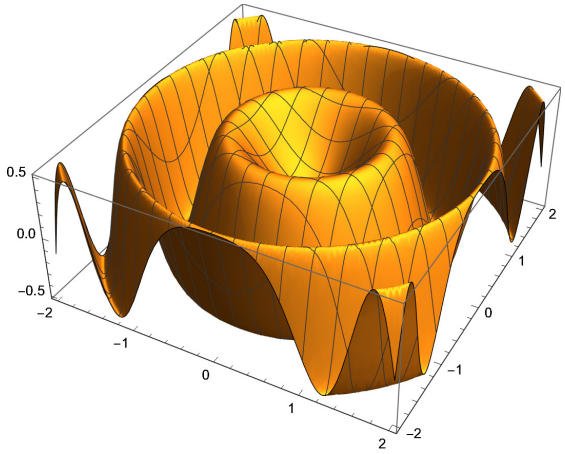
In[34]:=
```
ArrayPlot[Table[Mod[Binomial[n, k], 2], {n, 0, 127},
{k, 0, 127}]]
```

Out[34]=



In[35]:=
```
Plot3D[Sin[x^2 + y^2]*Cos[x^2 + y^2], {x, -2, 2}, {y, -2, 2},
PlotPoints -> 50]
```

Out[35]=



1.1.1. Resolución de problemas con *Mathematica*®

Con la ayuda de las funciones de *Mathematica*® se pueden resolver o encontrar soluciones particulares de diferentes problemas, lo cual permite experimentar, conjeturar y de paso entender mejor los detalles del problema. A lo largo de estas notas de clase iremos proponiendo problemas para ser abordados desde un enfoque simbólico.

Ejemplo 1.1 El número 1634 tiene una propiedad interesante. Este número de 4 dígitos satisface que la suma de las potencias cuartas de sus dígitos da el mismo número. Es decir $1^4 + 6^4 + 3^4 + 4^4 = 1634$.

In[36]:=
```
1^4 + 6^4 + 3^4 + 4^4 == 1634
```

Out[36]=
```
True
```

¿Cuántos enteros de m dígitos son iguales a la suma de las m-ésimas potencias de sus dígitos en el intervalo $[1, \ldots, 10^7]$?
Con *Mathematica*® encontramos los números que satisfacen dicha propiedad, que en este intervalo son en total 24.

In[37]:=
```
Select[Range[10^7], # ==
Total[IntegerDigits[#]^IntegerLength[#]] &]
```

Out[37]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, 8208,
9474, 54748, 92727, 93084, 548834, 1741725, 4210818,
9800817, 9926315}
```

In[38]:=
```
Length[%]
```

Out[38]=
```
24
```

Esta sucesión de números se conocen como los números narcisistas. En realidad esta sucesión es finita ya que la suma máxima de las k-ésimas potencias de un número de

k-dígitos en base 10 es $k(10-1)^k$ y el número más pequeño de k dígitos es 10^{k-1}. Esta nos permite acotar el valor de k:

```
In[39]:= N[Reduce[10^(k - 1) < k 9^k, k]]

Out[39]= 0.101071 < k < 60.8479
```

Es decir que un número en base 10 es narcisista si es menor que 10^{60}. Se puede verificar que sólo existen 88 números narcisistas en base 10. El mayor de los números narcisistas es

$$115132219018763992565095597973971522401$$

con 39 dígitos.

En el siguiente ejemplo mostramos un problema que puede ser abordado utilizando diferentes conceptos matemáticos y diferentes funciones de *Mathematica*® .

Ejemplo 1.2 (Números de Fibonacci) La sucesión de Fibonacci F_n se define como $F_n = F_{n-1} + F_{n-2}$ para $n \geq 2$, con los valores iniciales $F_0 = 0$ y $F_1 = 1$. ¿Cómo se pueden calcular los números de Fibonacci con *Mathematica*® ? Existen varias formas de llevar a cabo esta tarea. Una primera opción es definir la recursión:

```
In[40]:= Fibo1[0]=0;
         Fibo1[1]=1;
         Fibo1[n_]:=Fibo1[n-1]+Fibo1[n-2];
```

Los primeros 10 valores son

```
In[41]:= Table[Fibo1[n], {n, 1, 10}]

Out[41]= {1, 1, 2, 3, 5, 8, 13, 21, 34, 55}
```

El tiempo para calcular los números de Fibonacci con esta función va aumentando rápidamente (¿por qué?)

```
In[42]:= Timing[Fibo1[40]]

Out[42]= {197.029, 102334155}
```

Una forma de mejorar este tiempo es guardando los valores ya calculados:

```
In[43]:= $RecursionLimit=Infinity;
         Fibo2[0]=0;
         Fibo2[1]=0;
         Fibo2[n_]:=Fibo2[n]=Fibo2[n-1]+Fibo2[n-2];

In[44]:= Timing[Fibo2[40]]
```

```
Out[44]= {0.000227, 102334155}

In[45]:= Timing[Fibo2[40]]

Out[45]= {7.*10^-6, 102334155}
```

Utilizando las identidades de los números de Fibonacci

$$F_{2n} = F_n(F_n + 2F_{n-1}) \quad \text{y} \quad F_{2n+1} = F_{n+1}^2 + F_n^2, \quad n \geq 0 \tag{1.1}$$

se puede escribir una nueva función bajo el paradigma de *divide y vencerás*.

```
In[46]:= Fibo3[0]=0;
         Fibo3[1]=1;
         Fibo3[n_]:=Fibo3[n]=If[Mod[n, 2] == 0,
         Fibo3[n/2] (Fibo3[n/2] + 2 Fibo3[n/2 - 1]),
          Fibo3[(n + 1)/2]^2 + Fibo3[(n - 1)/2]^2]

In[47]:= Timing[Fibo3[40]]

Out[47]= {0.000128, 102334155}
```

Dada una sucesión $\{a_n\}$ la serie formal, denominada función generatriz de la sucesión $\{a_n\}$, está definida por $A(x) = \sum_{n\geq 0} a_n x^n$. Para el caso de la sucesión de Fibonacci, su función generatriz está dada por

$$\begin{aligned} F(x) &= \sum_{n\geq 0} F_n x^n = F_0 + F_1 x + \sum_{n\geq 2} F_n x^n \\ &= x + \sum_{n\geq 2} (F_{n-1} + F_{n-2}) x^n = x + \sum_{n\geq 2} F_{n-1} x^n + \sum_{n\geq 2} F_{n-2} x^n \\ &= x + x \sum_{n\geq 2} F_{n-1} x^{n-1} + x^2 \sum_{n\geq 2} F_{n-2} x^{n-2} = x + x \sum_{n\geq 1} F_n x^n + x^2 \sum_{n\geq 0} F_n x^n \\ &= x + xF(x) + x^2 F(x). \end{aligned}$$

Resolviendo esta última ecuación se obtiene la función racional

$$F(x) = \frac{x}{1 - x - x^2}.$$

La función generatriz es otra forma de calcular los números de Fibonacci, haciendo el desarrollo en series

```
In[48]:= Series[x/(1 - x - x^2), {x, 0, 10}]

Out[48]= x + x^2 + 2 x^3 + 3 x^4 + 5 x^5 + 8 x^6 + 13 x^7 + 21 x^8 + 34 x^9
         + 55x^10 +O[x]^11
```

```
In[49]:= Fibo4[n_] := Fibo4[n] =
         Coefficient[Series[x/(1 - x - x^2), {x, 0, n}], x, n]

In[50]:= Timing[Fibo4[40]]

Out[50]= {0.000858, 102334155}
```

A partir de la función generatriz podemos deducir una forma adicional para encontrar los números de Fibonacci. Por ejemplo, usando fracciones parciales se sabe que

$$F(x) = \frac{x}{1-x-x^2} = \frac{C}{1-\alpha x} + \frac{D}{1-\beta x},$$

con C, D números reales. Puesto que

$$1 - x - x^2 = (1-\alpha x)(1-\beta x) = 1 - (\alpha+\beta)x + \alpha\beta x^2$$

al comparar los coeficientes obtenemos el sistema de ecuaciones:

$$\begin{cases} \alpha + \beta & = 1 \\ \alpha\beta & = -1. \end{cases}$$

Resolviendo el sistema obtenemos que una solución es $\alpha = \frac{1-\sqrt{5}}{2}$ y $\beta = \frac{1+\sqrt{5}}{2}$. Con el comando `Apart` podemos encontrar la descomposición en fracciones parciales.

```
In[51]:= Apart[x/(1 - x - x^2)]

Out[51]= -(x/(-1 + x + x^2))

In[52]:= Apart[x/Factor[(1-x-x^2), Extension -> Sqrt[5]]] // Cancel

Out[52]= (-1 + Sqrt[5])/(Sqrt[5] (-1+Sqrt[5]-2 x)) - (1 + Sqrt[5])/(Sqrt[5] (1+Sqrt[5]+2 x))
```

Como $x = (1-\beta x)C + (1-\alpha x)D = C + D - (\beta C + \alpha D)x$, entonces

$$\begin{cases} C + D & = 0 \\ \beta C + \alpha D & = -1. \end{cases}$$

Resolviéndolo el sistema se deduce que $C = -\frac{1}{\sqrt{5}}$ y $D = \frac{1}{\sqrt{5}}$.

```
In[53]:= α = (1 - Sqrt[5])/2;
         β = (1 + Sqrt[5])/2;
         Solve[{C + D == 0, β C + α D == -1}, {C, D}]

Out[53]= {{C -> -1/Sqrt[5], D -> 1/Sqrt[5])}}
```

Por lo tanto

$$F(x) = \frac{\left(-\frac{1}{\sqrt{5}}\right)}{1-\alpha x} + \frac{\left(\frac{1}{\sqrt{5}}\right)}{1-\beta x} = \frac{1}{\sqrt{5}} \sum_{n\geq 0} (\beta^n - \alpha^n) x^n.$$

Observe que en el último paso utilizamos el hecho que (serie geométrica formal)

$$\sum_{n\geq 0} x^n = \frac{1}{1-x}.$$

Comparando los coeficientes de $F(x)$ con la serie que obtuvimos se concluye que

$$F_n = \frac{1}{\sqrt{5}}(\beta^n - \alpha^n), \quad n \geq 0.$$

Es decir que ya tenemos una fórmula cerrada. Este tipo de fórmulas se conocen como fórmulas tipo Binet.

```
In[54]:= Fibo5[n_]:=Fibo5[n]=1/Sqrt[5](β^n-α^n)// Simplify

In[55]:= Timing[Fibo5[40]]

Out[55]= {0.000113, 102334155}
```

Puesto que $|\alpha| < 1$ y $|\beta| > 1$, entonces $\frac{|\alpha^n|}{\sqrt{5}} < \frac{1}{\sqrt{5}} < \frac{1}{2}$, entonces se tiene que

$$F_n = \left\lfloor \frac{1}{\sqrt{5}}\beta^n + \frac{1}{2} \right\rfloor.$$

Adicionalmente *Mathematica*® tiene implementada una función propia, `Fibonacci`, para calcular el n-ésimo número de Fibonacci la cual es muy eficiente.

```
In[56]:= Timing[Fibonacci[40]]

Out[56]= {0.000501, 102334155}
```

Comparando el tiempo de ejecución de las anteriores funciones para valores de $n = 10$ a 1000, tomados de diez en diez se obtiene la Gráfica 1.1.

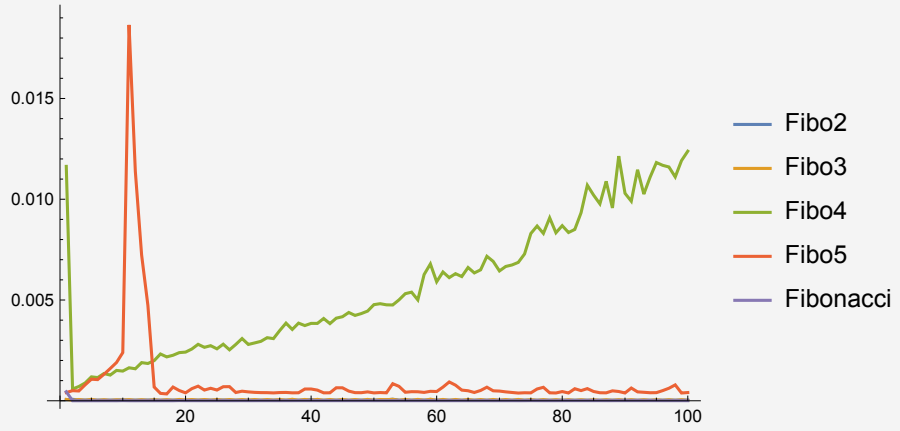


Figura 1.1: Tiempo de ejecución de las funciones para calcular los números de Fibonacci

Si ejecutamos por segunda vez las funciones se obtiene los tiempos que se muestran en la Gráfica 1.2.

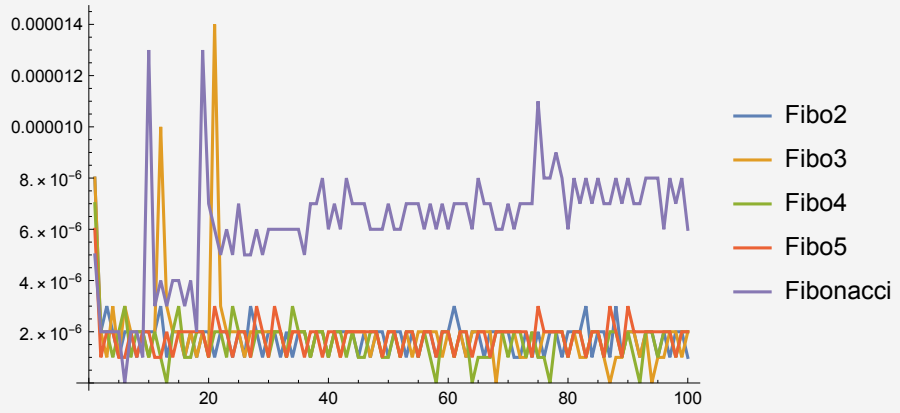


Figura 1.2: Tiempo de ejecución de las funciones para calcular los números de Fibonacci

1.1.2. Computación Simbólica al Servicio de la Combinatoria

Una de los intereses personales, como autor de estas notas de clase, es la profunda interacción que hay entre la Computación Simbólica y la Combinatoria Enumerativa. El desarrollo de la computación simbólica y de diferentes algoritmos combinatorios a abierto nuevas oportunidades para comprender con mayor profundidad el estudio de diferentes estructuras discretas. El poder hacer experimentaciones y *manipular* de manera simbólica objetos combinatorios permite conjeturar y revisar ejemplos que de otra manera sería muy difícil de llevar a cabo. Por ejemplo, estos paquetes de cómputo simbólico permiten trabajar con funciones generatrices, extraer y estimar el comportamiento asintótico de sus

coeficientes.

Ejemplo 1.3 El tema de *pattern avoidance*, debido a Donal Knuth, consiste en enumerar permutaciones que evitan una determina subsucesión. Consideremos una permutación π, la normalización de π es una nueva permutación, denotada por $n(\pi)$, obtenida a partir de π en la que el elemento más pequeño de la permutación se reemplaza por 1, el segundo más pequeño por 2 y así sucesivamente. Por ejemplo, si $\sigma = 8165$ entonces $n(\sigma) = 4132$. Una subsucesión de una permutación π es una palabra obtenida de eliminar algunos (posiblemente ninguno) de los elementos en π. Por ejemplo, 256 es una subsucesión de la permutación $713\mathbf{2}489\mathbf{56}$. Denotaremos por S_n el conjunto de todas las permutaciones de $[n]$. Considere las permutaciones $\pi \in S_n$ y $\sigma \in S_k$. La permutación π contiene una copia de σ si existe una subsucesión σ' de π tal que $n(\sigma') = \sigma$. En este caso decimos que σ es un *pattern* de π. Por ejemplo la permutación $\pi = 713248956$ contiene una copia de 123, ya que $n(256) = 123$ y 256 es una subsucesión de π. Una permutación π evita σ si no existe una subsucesión σ' tal que $n(\sigma') = \sigma$. Por ejemplo, la permutación $\pi = 425613$ evita a 4321. Denotaremos por $\mathrm{Av}_n(\pi)$ el conjunto de permutaciones en S_n que evitan π y por $\mathrm{av}_n(\pi)$ su cardinal.

Una de las primeras cosas que nos permite hacer un paquete de cómputo simbólico es explorar los primeros elementos del conjunto. Por ejemplo, con el siguiente comando encontramos los conjuntos $\mathrm{Av}_3(132)$ y $\mathrm{Av}_4(132)$. Se verifica que $\mathrm{av}_3(132) = 5$ y $\mathrm{av}_4(132) = 14$.

```
In[57]:= DeleteCases[Permutations[Range[3]],
         {___, x_, ___, y_, ___, z_, ___} /; x < z && z < y]

Out[57]= {{1, 2, 3}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}

In[58]:= DeleteCases[Permutations[Range[4]],
         {___, x_, ___, y_, ___, z_, ___} /; x < z && z < y]

Out[58]= {{1, 2, 3, 4}, {2, 1, 3, 4}, {2, 3, 1, 4}, {2, 3, 4, 1},
          {3, 1, 2,  4}, {3, 2, 1, 4}, {3, 2, 4, 1}, {3, 4, 1, 2},
          {3, 4, 2, 1}, {4, 1,  2, 3}, {4, 2, 1, 3}, {4, 2, 3, 1},
          {4, 3, 1, 2}, {4, 3, 2, 1}}
```

Calculando los primeros valores de la sucesión $\mathrm{av}_n(132)$, se puede conjeturar con ayuda de la OEIS (The On-Line Encyclopedia of Integer Sequences) que la sucesión $\mathrm{av}_n(132)$ coincide con los números de Catalan $C_n = \frac{1}{n+1}\binom{2n}{n}$.

```
In[59]:= Table[Length[DeleteCases[
         Permutations[Range[n]], {___, x_, ___, y_, ___, z_, ___} /;
             x < z && z < y]], {n, 0, 9}]
```

```
Out[59]= {1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862}
```

Supongamos que no conocemos ninguna fórmula para la sucesión $\text{av}_n(132)$

```
In[60]:= FindSequenceFunction[{1, 1, 2, 5, 14, 42, 132, 429, 1430,
         4862}, n] // Simplify

Out[60]= 4^(-1+n) Pochhammer[1/2, -1 + n] / Pochhammer[2, -1 + n]
```

A partir de la definición del símbolo de Pochhammer se puede simplificar a la expresión que define los números de Catalan. Los algoritmos que permiten adivinar fórmulas a partir de los primeros términos de una sucesión se conocen como algoritmos hipergeométricos. Aún más con las algoritmos desarrollados por el RISC (Research Institute for Symbolic Computation) se pueden conjeturar relaciones de recurrencia polionomiales:

```
In[61]:= << RISC`Guess`

Out[61]=
```

HolonomicFunctions Package version 1.7.3 (21-Mar-2017)
written by Christoph Koutschan
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria

```
--> Type  ?HolonomicFunctions  for help.
```

Package GeneratingFunctions version 0.8 written by Christian Mallinger
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria

Guess Package version 0.52
written by Manuel Kauers
Copyright Research Institute for Symbolic Computation (RISC),
Johannes Kepler University, Linz, Austria

```
In[62]:= GuessMinRE[{1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862},a[n]]

Out[62]= (-2 - 4 n) a[n] + (2 + n) a[1 + n]
```

Esto último significa que la sucesión satisface la relación $(-2-4n)a_n+(2+n)a_{n+1}=0$. En general es posible demostrar que para todo $n \geq 0$ y π una permutación en S_3 se tiene que $\text{av}_n(\pi) = C_n$.

Capítulo 2

Aritmética de los Enteros

En este capítulo estudiamos algunas propiedades básicas de la aritmética del anillo de los enteros. Sin duda la aritmética de los enteros es uno de los puntos centrales de cualquier sistema de computo simbólico. En particular analizaremos la complejidad computacional de las operaciones básicas.

2.1. Representación de los Enteros

Denotaremos por $\mathbb{Z}$ el conjunto de los números enteros, en el que hay definidas dos operaciones adición $(+)$ y multiplicación $(\cdot)$. Se sabe que $(\mathbb{Z}, +)$ es un grupo abeliano, además, el producto de enteros es asociativo, conmutativo, el 1 es el elemento neutro y la multiplicación es distributiva con respecto a la suma. Esto nos permite concluir que la terna $(\mathbb{Z}, +, \cdot)$ es un anillo conmutativo con identidad.

Lo primero que vamos a analizar es cómo representar un entero ante un sistema de cómputo. Para ello utilizaremos un resultado bien conocido (se deduce del algoritmo de la división) y es que todo número entero a se puede representar de manera única de la forma

$$a = \pm(c_0 + c_1\beta^1 + c_2\beta^2 + \cdots + c_{n-1}\beta^{n-1}),$$

donde β es un entero mayor que 1, $n \geq 1$, $c_{n-1} \neq 0$ y $0 \leq c_i < \beta$ para todo $i = 0, 1, \ldots, n-1$. Esta es la representación en base β de a.

Por ejemplo, $2022 = 2(5)^0 + 4(5)^1 + 1(5)^3 + 3(5)^4$. En este caso se dice que la representación en *base 5* del número 2022 es $(3, 1, 0, 4, 2)_5$. Cuando se utiliza la representación en base 2, esta se conoce como la representación binaria del entero. Por ejemplo,

```
In[63]:= BaseForm[2022, 5]

Out[63]= 31042_5

In[64]:= BaseForm[2022, 2]

Out[64]= 11111100110_2
```

Cuando la base es mayor que 10 es necesario introducir símbolos para los dígitos $11, 12, \ldots, (b-1)$. Por ejemplo cuando la base es 16, sistema hexadecimal, se utilizan los siguientes símbolos $10 = A,\ 11 = B,\ 12 = C,\ 13 = D,\ 14 = E,\ 15 = F$. Por ejemplo,

```
In[65]:= BaseForm[365, 16]

Out[65]= 16d16
```

La mayor base permitida en *Mathematica*® es 36 (¿por qué?).

La representación en base β nos permite escribir un número entero como una lista. Usualmente se escoge una base β que sea potencia de 2. La razón es que cuando se llevan a cabo operaciones aritméticas en un computador estos tienen una unidad de memoria, que para un procesador de 64-bits consiste de 2^{64} estados.

Consideremos la descomposición de un entero positivo n en base β, escrita como

$$n = c_0 + c_1\beta + c_2\beta^2 + \cdots + c_\ell\beta^\ell,$$

con $c_\ell \neq 0$ y $0 \leq c_i < \beta$, $i = 1, \ldots, \ell$. Puesto que ℓ es el mayor entero tal que $\beta^\ell \leq n$, entonces $\ell \leq \log_\beta n < \ell + 1$. Así, el tamaño de n en base β está dado por

$$\lfloor \log_\beta n \rfloor + \underbrace{1}_{c_0},$$

siempre y cuando $n \neq 0$. Si $n = 0$ entonces el tamaño se define como 1. Observe que el tamaño de un entero se puede entender como los espacios de memoria (o celdas) utilizados para representarlo como una lista.

```
In[66]:= IntegerDigits[8375, 8]

Out[66]= {2, 0, 2, 6, 7}

In[67]:= Floor[Log[8, 8375]] + 1

Out[67]= 5

In[68]:= IntegerDigits[8375, 2]

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

In[69]:= Floor[Log[2, 8375]] + 1

Out[69]= 14

In[70]:= Length[IntegerDigits[8375, 2]]

Out[70]= 14
```

Definición 2.1 Para un entero no negativo n, denotamos por Lon(n), el tamaño de n escrito en binario. Es decir que

$$\text{Lon}(n) = \begin{cases} \lfloor \log_2(n) \rfloor + 1, & \text{si } n \neq 0; \\ 1, & \text{si } n = 0. \end{cases}$$

2.1.1. Notación Asintótica

El análisis de la complejidad computacional de los algoritmos que analizaremos en este capítulo se hará en términos del tamaño de las entradas. Tenga en cuenta que una operación simple como la adición y multiplicación toma exactamente un paso computacional. El análisis de un algoritmo se hará para el peor de los casos, es decir, para el número máximo de pasos que requiere el algoritmo para una entrada de tamaño n. Para simplificar el análisis vamos a utilizar la notación *O grande*, la cual también es adecuada para comparar diferentes algoritmos.

Definición 2.2 Sea $g(x)$ una función de variable real definida para todo real no negativo. Denotamos por $O(g(x))$, leído como o grande de g, el conjunto

$$O(g(x)) = \{f(n) : \text{existen reales } c > 0 \text{ y } b \geq 0 \text{ tales que } |f(x)| \leq c|g(x)|, \text{ para todo } x \geq b\}.$$

Escribiremos $f(x) = O(g(x))$ para indicar que f es un elemento de $O(g(x))$. Hay que tener mucho cuidado con este abuso de notación ya que se pueden tener cosas como $n = O(n^3)$ y $n^2 = O(n^3)$, y entonces concluir de manera errada que las funciones $f(n) = n$ y $g(n) = n^2$ son iguales. Observe que para toda constante positiva k se cumple que $k = O(1)$.

Un algoritmo polinomial es uno cuyo tiempo de ejecución para una entrada de tamaño n es a lo más $an^b + c = O(n^b)$, con a, b y c constantes.

En la Figura 2.1 mostramos de manera intuitiva dos funciones $g(x)$ y $f(x)$ tales que $f(x) = O(g(x))$. Si f pertenece a $O(g)$ entonces $g(x)$ sirve como una cota superior para $|f(x)|$. Esto es $|f(x)|$ no crece más rápido que un múltiplo de $|g(x)|$.

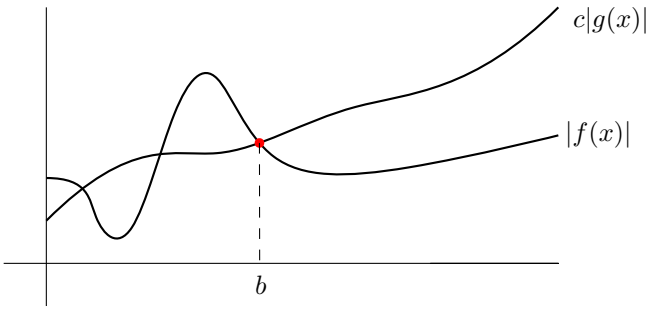


Figura 2.1: La función $f(x)$ es $O(g(x))$.

Ejemplo 2.1 Si $f(n) = n^2 + 10^{10}n$, entonces $f = O(n^2)$. En efecto,

$$\begin{aligned} |f(n)| &= |n^2 + 10^{10}n| \\ &\leq n^2 + 10^{10}n \quad (n \geq 0) \\ &\leq n^2 + 10^{10}n^2 \quad (n \geq 1) \\ &= (10^{10} + 1)n^2 = c|n^2|, \end{aligned}$$

con $c = 10^{10} + 1$.

Ejemplo 2.2 Considere el polinomio $f(x) = 19x^6 - 3x^4 + 2x^5 + 15$. En este caso se tiene que

$$|19x^6 - 3x^4 + 2x^3 + 15| \leq |19x^6| + |3x^4| + |2x^3| + |15|$$

(esto debido a las desigualdades $|x \pm y| \leq |x| + |y|$). Así, para todo real $x \geq 1$ se cumple que

$$|f(x)| \leq 19x^6 + 3x^4 + 2x^3 + 16 \leq 19x^6 + 3x^6 + 2x^6 + 15x^6 = 39x^6.$$

De modo que $f(x) = O(x^6)$.

En general, tenemos el siguiente resultado para polinomios.

Proposición 2.1 Sea $f(x) = a_0 + a_1x + \cdots + a_nx^n$ un polinomio en $\mathbb{R}[x]$ con $a_n \neq 0$. Entonces $f = O(x^\ell)$ para todo $\ell \geq n$.

Demostración. Para todo $x > 1$ se tiene que

$$|f(x)| \leq \sum_{i=0}^{n} |a_i|x^i = x^n \sum_{i=0}^{n} \frac{|a_{n-i}|}{x^i} \leq x^n(|a_n| + |a_{n-1}| + \cdots + |a_0|).$$

Quedando demostrado. □

Ejemplo 2.3 De la Proposición 2.1, podemos concluir que

$$\sum_{i=0}^{n} i = \frac{1}{2}n(n+1) = O(n^2).$$

Ejemplo 2.4 Se tiene que $\ln(n!)$ es $O(n \ln n)$. En efecto $n! \leq n^n$ para todo $n \geq 1$. Tomando logaritmo en esta desigualdad se obtiene el resultado.

Ejemplo 2.5 Se tiene que $\log_\beta(n)$ es $O(\log_2 n)$. En efecto por la fórmula de cambio de bases de los logaritmos se tiene que $\log_\beta(n) = \frac{\log_2 n}{\log_2 \beta}$. Observe que Lon$(n) =$

$O(\log_2(n))$.

Ejemplo 2.6 Si $f(x) = x^3 + x\ln x$, entonces $f = O(x^3)$. En efecto

$$\begin{aligned} |f(x)| &= |x^3 + x\ln x| \\ &= x^3 + x\ln x \quad (x \geq 1) \\ &\leq x^3 + x^2 \quad \text{(asumiendo } x \geq 1 \text{ tenemos que } x \geq \ln x.) \\ &\leq 2x^3. \end{aligned}$$

De manera similar, $f(x) = 5\log x + 3(\log x)^2 + 7x^3 + 2x^2 = O(x^3)$.

Ejemplo 2.7 La notación O grande se utilizará también en ecuaciones funcionales con sumas. Por ejemplo, si $f(x) = x^5+2x^4-3x^2+8x+1$ entonces $f(x) = x^5+O(x^4)$. Esto significa que $f(x) = x^5 + g(x)$, donde $g(x)$ es alguna función en $O(x^4)$.

```
In[71]:= Series[Exp[x], {x, 0, 5}]
```

$$\text{Out[71]= } 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + O[x]^6$$

Ejemplo 2.8 La notación $5n^2 + O(n) = O(n^2)$ significa que para cualquier función en f en $O(n)$ existe una función g en $O(n^2)$ tal que $5n^2 + f(n) = g(n)$ para todo n.

A partir de la definición de O grande se puede demostrar la siguiente proposición.

Proposición 2.2 Dada una función $f(n)$ y k un real positivo se tiene que

- $O(kf(n)) = O(f(n))$.
- $O(f(n)) + O(f(n)) = O(f(n))$.
- $O(O(f(n))) = O(f(n))$.
- $O(f_1(n)) + O(f_2(n)) = O(f_1(n) + f_2(n)) = O(\max\{f_1(n), f_2(n)\})$, donde máx es máxima punto a punto.
- $O(f_1(n))O(f_2(n)) = O(f_1(n)f_2(n)) = f_1(n)O(f_2(n))$.

2.1.2. Tiempo computacional

El tiempo computacional de un algoritmo lo definiremos como el número de operaciones aritméticas que utiliza el algoritmo para procesar una entrada de longitud n. Si el tiempo computacional se describe como una función en $O(g(n))$ entonces decimos que el algoritmo es de orden $g(n)$.

Asumiendo que una operación toma una nanosegundo (10^{-9} segundo) en el Cuadro 2.1 comparamos el tiempo computacional para diferentes ordenes de una función $f(n)$.

$f(n)$	$n = 10$	$n = 10^3$	$n = 10^5$	$n = 10^7$
$\log_2(n)$	$3.3 * 10^{-9}$ s	10^{-8} s	$1.7 * 10^{-8}$ s	$2.3 * 10^{-8}$s
n	10^{-8} s	10^{-6} s	0.0001 s	$0,01$s
$n \log_2(n)$	$3.3 * 10^{-8}$ s	0.000010 s	0.0017 s	$0,23$s
n^2	10^{-7} s	0.0010 s	10 s	27horas
n^3	10^{-6} s	1 s	11.5 días	31709 años
2^n	10^{-6} s	–	–	–

Cuadro 2.1: Comparación del tiempo computacional

Ejemplo 2.9 El Algoritmo 1 es de orden lineal. En efecto cada iteración al interior del `for` realiza una suma y una multiplicación. Estamos asumiendo que estas operaciones se hacen en un tiempo constante. Puesto que el algoritmo se ejecuta $n - 2$ veces entonces en total se llevan a cabo $2n - 2$ operaciones. Es decir el orden es lineal $O(n)$.

Algoritmo 1 Algoritmo de Ejemplo

Input: $a := 0, b := 2$

```
1: procedure AlgEje
2:     for i := 2 to n do
3:         a := (a + i) · b
4:     end for
5: end procedure
```

Ejemplo 2.10 El Algoritmo 2 es de orden cuadrático. En efecto cada iteración al interior del `for` realiza 4 operaciones básicas y de nuevo estamos asumiendo que estas operaciones se hacen en un tiempo constante. El `for` interno se ejecuta

$$1 + 2 + \cdots + n = \frac{n(n+2)}{2}$$

veces. Por lo tanto, se llevan a cabo $2n(n + 2)$ operaciones. Es decir el orden es cuadrático ($O(n^2)$).

Algoritmo 2 Algoritmo de Ejemplo

Input: $a := 0$

```
1: procedure AlgEje2
2:     for i := 1 to n do
3:         for j := 1 to i do
4:             a := a + j · (i + j + 3)
5:         end for
6:     end for
7: end procedure
```

Ejemplo 2.11 Si se utiliza la definición recursiva de la sucesión de Fibonacci F_n se puede demostrar que este algoritmo es de orden exponencial en n. En efecto si $T(n)$ denota el tiempo computacional entones se tiene que $T(n) = T(n-1) + T(n-2) + O(1)$, donde el último término denota el tiempo extra de hacer la suma. Iterando esta igualdad se puede verificar que $T(n) = aF_n + O(1)$, para a una contante. Luego $T(n) = O(F_n) = O\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$.