Universidad Nacional de Colombia

Algebra abstracta y computacional

Taller 2 Semana 2 - Grupo 9

Paola Andrea Gallegos - 1005257935 - pgallegos@unal.edu.co
Oscar Julian Rodriguez - 1022445731- osrodriguezc@unal.edu.co
Santiago Diaz Gonzalez - 1007244241 - sdiazgo@unal.edu.co

August 24, 2022

1. Muestre una forma de multiplicar números complejos $a + bi$ y $c + di$ usando unicamente tres multiplicaciones de numeros reales. El algoritmo debe arrojar por separado las componentes reales $ac - bd$ y $ad + bc$.
   Desarrollo: La multiplicación de numeros complejos normalmente implica cuatro multiplicaciones y dos sumas.

   $$(a + bi)(c + di) = (ac - bd) + (bc + ad)i.$$

   con *Gauss algorithm for complex multiplication* Gauss descubrió una manera de hacer esto con solo tres multiplicaciones, usando esencialmente el mismo cálculo que el algoritmo de Karatsuba:

   $$\begin{gathered} k_1 = c * (a + b) \\ k_2 = a * (d - c) \\ k_3 = b * (c + d) \\ \text{Real part} = k_1 - k_3 \\ \text{Imaginary part} = k_1 + k_2 \end{gathered}$$

   Se puede contar los símbolos $*, +$y $-$ de arriba y ver que el primer algoritmo usa 4 multiplicaciones y 2 sumas/restas. El segundo algoritmo utiliza 3 multiplicaciones y 5 sumas/restas.

   El algoritmo de Gauss sería más rápido que el algoritmo directo si los componentes fueran números enteros muy grandes o flotantes de muy alta precisión. El tiempo requerido para sumar números enteros de n dígitos es $O(n)$, pero el tiempo requerido para multiplicar números de n dígitos es al menos $O(nlogn)$. Entonces, para n lo suficientemente grande, vale la pena hacer una suma adicional para ahorrar una multiplicación.

2. Implemente dos funciones *MultEscuela[a,b]* y *Karatsuba[a,b]* en $Mathematica$ con los algoritmos de multiplicacion de la escuela y el de Karatsuba, respectivamente. Considere dos enteros.

   $$\begin{gathered} a = Random[Integer, \{10, 10^{500}\}] \\ b = Random[Integer, \{10, 10^{500}\}] \end{gathered}$$

   Multipliquelos con las dos funciones y con la multiplicacion que ya esta implementada en $Mathematica$. Compare los tiempos de ejecucion. Repita esto con los siguientes enteros:

   $$\begin{gathered} a = Random[Integer, \{10^{99}, 10^{100}\}] \\ b = Random[Integer, \{10^{00}, 10^{100}\}] \end{gathered}$$

   Desarrollo: Primero creamos las dos funciones que vamos a utilizar menos la multiplicación que ya esta implementada en Mathematica

```
In[1]:= $RecursionLimit = ∞;
        límite de recursión

        Clear[Karatsuba] (*Limpiamos cualquier funcion con este nombre que haya sido creada antes *)
        borra

        Karatsuba[x_, y_] :=
         (Karatsuba[x, y] = Module[{a, b, c, d, n, m, aporc, bpord},
                            módulo
                 n = Length[IntegerDigits[x]];
                     longitud dígitos de entero
                 m = Length[IntegerDigits[x]];
                     longitud dígitos de entero
                 m = Max[n, m];
                     máximo
                 n = 1;
                 While[n < m, n = 2 * n];
                 mientras
                 b = Mod[x, 10 ^ (n / 2)];
                     operación módulo
                 a = (x - b) / 10 ^ (n / 2);
                 d = Mod[y, 10 ^ (n / 2)];
                     operación módulo
                 c = (y - d) / 10 ^ (n / 2);
                 aporc = Karatsuba[a, c];
                 bpord = Karatsuba[b, d];
                 aporc * 10 ^ n + (aporc + bpord
                     + Karatsuba[a - b, d - c]) * 10 ^ (n / 2) + bpord]) /;
          Min[Length[IntegerDigits[x]], Length[IntegerDigits[y]]] > 1
          mí··· longitud dígitos de entero  longitud dígitos de entero

        Karatsuba[x_, y_] := x * y

In[5]:= Clear[MultEscuela] (*Limpiamos cualquier funcion con este nombre que haya sido creada antes *)
        borra
        MultEscuela[x_, y_] :=
         (MultEscuela[x, y] =
             Module[{a, b, c, d, n, m},
             módulo
              n = Length[IntegerDigits[x]];
                  longitud dígitos de entero
              m = Length[IntegerDigits[x]];
                  longitud dígitos de entero
              m = Max[n, m];
                  máximo
              n = 1; While[n < m, n = 2 * n];
                     mientras
              b = Mod[x, 10 ^ (n / 2)];
                  operación módulo
              a = (x - b) / 10 ^ (n / 2);
              d = Mod[y, 10 ^ (n / 2)];
                  operación módulo
              c = (y - d) / 10 ^ (n / 2);
              MultEscuela[a, c] * 10 ^ n +
               (MultEscuela[a, d] + MultEscuela[b, c]) *
             10 ^ (n / 2) + MultEscuela[b, d]]) /;
          Min[Length[IntegerDigits[x]], Length[IntegerDigits[y]]] > 1
          mí··· longitud dígitos de entero  longitud dígitos de entero
        MultEscuela[x_, y_] := x * y
```

Definimos los números enteros a usar con Random, primero a1 y b1.

```
In[8]:= a1 = Random[Integer, {10, 10^500}]
        b1 = Random[Integer, {10, 10^500}]

Out[8]= 29 555 923 196 541 207 091 081 792 407 168 161 716 919 204 305 284 487 005 918 247 631 496 051 879 331 761 950 094 035 301 321 962 995 888 095 588 026 556 363 367 627 530 466 332 \
120 108 007 763 461 913 846 439 471 985 235 035 102 958 100 836 469 754 158 816 870 337 626 194 148 763 974 858 356 417 807 810 221 996 427 318 128 888 637 826 967 535 418 502 \
780 346 934 475 556 668 019 622 615 363 397 362 020 222 274 840 889 112 629 194 242 100 788 491 139 874 013 014 824 915 941 128 445 889 924 513 686 992 410 096 438 884 731 013 \
380 034 211 072 612 105 216 626 546 362 793 332 006 963 512 381 633 761 912 680 033 953 700 087 380 667 165 263 367 341 960 404 486 144

Out[9]= 25 766 788 098 572 230 542 861 387 948 214 252 061 472 774 647 431 254 422 977 921 870 237 832 048 232 907 413 494 073 838 624 384 468 023 051 317 060 609 882 875 807 752 874 554 \
437 512 087 974 509 330 158 948 318 024 833 296 179 857 219 876 883 109 388 780 239 878 545 210 256 643 298 370 185 149 585 148 852 186 449 465 073 592 206 361 649 156 181 955 \
407 929 345 336 562 783 290 668 220 271 692 659 181 733 114 650 971 189 115 260 474 444 378 804 564 684 953 005 221 174 664 781 069 095 708 277 029 976 505 052 741 546 929 149 \
677 429 130 827 507 020 591 708 994 133 490 830 551 526 121 691 124 875 611 256 123 645 352 238 451 109 839 537 404 472 585 742 884 818
```

A la hora de probar la multiplicacion de cada uno de las funciones y el de la multiplicacion por defecto en mathematicas, el algortimo de karatsuba su eficiencia no puedo ser alcanzada por ningún programa en el lenguaje de alto nivel de Mathematica .

```
In[10]:= Mult1 = a1 * b1

Out[10]= 761 561 210 062 952 891 616 565 794 040 437 985 194 758 267 706 665 660 600 174 999 801 231 011 645 813 782 920 972 319 078 245 283 880 225 720 745 144 806 662 945 197 073 278 \
303 620 485 998 237 679 316 471 120 467 651 477 922 512 296 506 995 323 457 258 484 951 383 381 799 168 555 265 345 958 025 554 675 273 703 967 084 970 176 112 121 942 865 872 \
846 067 790 593 313 758 469 116 356 166 066 753 668 997 227 083 584 874 722 394 545 710 439 993 749 708 266 062 604 518 347 239 210 202 447 872 833 084 441 975 089 728 463 064 \
564 013 264 381 770 178 514 016 986 223 707 120 384 285 157 745 113 060 234 391 815 901 949 244 313 892 626 147 378 470 669 819 801 721 106 640 364 935 386 167 101 332 830 222 \
144 655 328 279 565 779 060 093 674 311 401 765 973 490 735 940 839 570 100 776 390 521 264 992 505 824 355 178 938 932 511 207 683 882 719 631 309 315 301 774 410 752 717 473 \
158 956 051 498 485 056 715 176 019 841 799 820 866 224 432 926 325 052 501 674 920 563 380 330 546 964 057 695 892 367 119 642 583 967 351 010 235 467 184 031 739 965 677 233 \
699 792 690 682 900 390 915 084 875 691 120 124 637 932 949 478 796 802 847 806 113 236 367 718 170 691 414 587 370 417 731 457 974 638 342 082 908 977 907 101 517 898 335 151 \
445 791 909 970 406 354 161 212 303 982 816 843 513 732 442 427 336 215 383 659 212 135 468 961 792

In[11]:= Mult2 = MultEscuela[a1, b1]

Out[11]= 761 561 210 062 952 891 616 565 794 040 437 985 194 758 267 706 665 660 600 174 999 801 231 011 645 813 782 920 972 319 078 245 283 880 225 720 745 144 806 662 945 197 073 278 \
303 620 485 998 237 679 316 471 120 467 651 477 922 512 296 506 995 323 457 258 484 951 383 381 799 168 555 265 345 958 025 554 675 273 703 967 084 970 176 112 121 942 865 872 \
846 067 790 593 313 758 469 116 356 166 066 753 668 997 227 083 584 874 722 394 545 710 439 993 749 708 266 062 604 518 347 239 210 202 447 872 833 084 441 975 089 728 463 064 \
564 013 264 381 770 178 514 016 986 223 707 120 384 285 157 745 113 060 234 391 815 901 949 244 313 892 626 147 378 470 669 819 801 721 106 640 364 935 386 167 101 332 830 222 \
144 655 328 279 565 779 060 093 674 311 401 765 973 490 735 940 839 570 100 776 390 521 264 992 505 824 355 178 938 932 511 207 683 882 719 631 309 315 301 774 410 752 717 473 \
158 956 051 498 485 056 715 176 019 841 799 820 866 224 432 926 325 052 501 674 920 563 380 330 546 964 057 695 892 367 119 642 583 967 351 010 235 467 184 031 739 965 677 233 \
699 792 690 682 900 390 915 084 875 691 120 124 637 932 949 478 796 802 847 806 113 236 367 718 170 691 414 587 370 417 731 457 974 638 342 082 908 977 907 101 517 898 335 151 \
445 791 909 970 406 354 161 212 303 982 816 843 513 732 442 427 336 215 383 659 212 135 468 961 792

In[12]:= (*Mult3= Karatsuba[a1,b1]*)(*Tal eficiencia no puede ser alcanzada por ningún programa en el lenguaje de alto nivel de Mathematica.*)
```

Ya teniendo ambas funciones implementadas podemos comparar la eficiencia de ambos algoritmos bajo consideración. Para esto, multiplicamos dos enteros a1 y b1 con las funciones MulEscuela y la multiplicación implementada en Mathematica.

```
In[35]:= Timing[Mult1 = a1 * b1] (*Multiplicacion de mathematica*)
         Timing[Mult2 = MultEscuela[a1, b1]] (*Multiplicacion de MultEscuela*)
         Mult2 - Mult1

Out[35]= {0.000012,
 761 561 210 062 952 891 616 565 794 040 437 985 194 758 267 706 665 660 600 174 999 801 231 011 645 813 782 920 972 319 078 245 283 880 225 720 745 144 806 662 945 197 073 278 \
303 620 485 998 237 679 316 471 120 467 651 477 922 512 296 506 995 323 457 258 484 951 383 381 799 168 555 265 345 958 025 554 675 273 703 967 084 970 176 112 121 942 865 872 \
846 067 790 593 313 758 469 116 356 166 066 753 668 997 227 083 584 874 722 394 545 710 439 993 749 708 266 062 604 518 347 239 210 202 447 872 833 084 441 975 089 728 463 064 \
564 013 264 381 770 178 514 016 986 223 707 120 384 285 157 745 113 060 234 391 815 901 949 244 313 892 626 147 378 470 669 819 801 721 106 640 364 935 386 167 101 332 830 222 \
144 655 328 279 565 779 060 093 674 311 401 765 973 490 735 940 839 570 100 776 390 521 264 992 505 824 355 178 938 932 511 207 683 882 719 631 309 315 301 774 410 752 717 473 \
158 956 051 498 485 056 715 176 019 841 799 820 866 224 432 926 325 052 501 674 920 563 380 330 546 964 057 695 892 367 119 642 583 967 351 010 235 467 184 031 739 965 677 233 \
699 792 690 682 900 390 915 084 875 691 120 124 637 932 949 478 796 802 847 806 113 236 367 718 170 691 414 587 370 417 731 457 974 638 342 082 908 977 907 101 517 898 335 151 \
445 791 909 970 406 354 161 212 303 982 816 843 513 732 442 427 336 215 383 659 212 135 468 961 792}

Out[36]= {8. × 10^-6,
 761 561 210 062 952 891 616 565 794 040 437 985 194 758 267 706 665 660 600 174 999 801 231 011 645 813 782 920 972 319 078 245 283 880 225 720 745 144 806 662 945 197 073 278 \
303 620 485 998 237 679 316 471 120 467 651 477 922 512 296 506 995 323 457 258 484 951 383 381 799 168 555 265 345 958 025 554 675 273 703 967 084 970 176 112 121 942 865 872 \
846 067 790 593 313 758 469 116 356 166 066 753 668 997 227 083 584 874 722 394 545 710 439 993 749 708 266 062 604 518 347 239 210 202 447 872 833 084 441 975 089 728 463 064 \
564 013 264 381 770 178 514 016 986 223 707 120 384 285 157 745 113 060 234 391 815 901 949 244 313 892 626 147 378 470 669 819 801 721 106 640 364 935 386 167 101 332 830 222 \
144 655 328 279 565 779 060 093 674 311 401 765 973 490 735 940 839 570 100 776 390 521 264 992 505 824 355 178 938 932 511 207 683 882 719 631 309 315 301 774 410 752 717 473 \
158 956 051 498 485 056 715 176 019 841 799 820 866 224 432 926 325 052 501 674 920 563 380 330 546 964 057 695 892 367 119 642 583 967 351 010 235 467 184 031 739 965 677 233 \
699 792 690 682 900 390 915 084 875 691 120 124 637 932 949 478 796 802 847 806 113 236 367 718 170 691 414 587 370 417 731 457 974 638 342 082 908 977 907 101 517 898 335 151 \
445 791 909 970 406 354 161 212 303 982 816 843 513 732 442 427 336 215 383 659 212 135 468 961 792}

Out[37]= 0
```

Ahora utilizaremos los enteros x2 y y2 y haremos lo mismo que hicimos con los enteros anteriores.

```
In[15]:= x2 = Random[Integer, {10^99, 10^100}]
y2 = Random[Integer, {10^99, 10^100}]
Out[15]= 4144282183963007200139920410832675810023481581690213405793256594604815436954910807093671889426827659
Out[16]= 1553493443181948347360683254705783735821636628521290089785529184725550643159248012924162842641314625
In[17]:= Mult4 = x2*y2
Out[17]= 6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875
In[18]:= Mult5 = MultEscuela[x2, y2]
Out[18]= 6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875
In[19]:= Mult3 = Karatsuba[x2, y2]
Out[19]= 6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875
```

La función incorporada es considerablemente mejor ya que la aritmética de enteros, como el módulo más importante del álgebra combinacional, está completamente implementada en el lenguaje de programación C y reside en el kernel de Mathematica. Los programas que están escritos en el lenguaje de alto nivel de Mathematica no pueden tener la misma eficiencia que las funciones del kernel, encontramos que en el lenguaje de alto nivel de Mathematica, el algoritmo de Karatsuba ya es más rápido para números enteros de 100 dígitos.

```
In[20]:= Timing[Mult4 = x2*y2] (*Multiplicacion de mathematica*)
Timing[Mult5 = MultEscuela[x2, y2]] (*Multiplicacion de MultEscuela*)
Timing[Mult6 = Karatsuba[x2, y2]]
(*Multiplicacion de Karatsuba*)
Out[20]= {0.000014,
6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875}
Out[21]= {8.×10^-6,
6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875}
Out[22]= {5.×10^-6,
6438115199482296734347547296515955743147116684306679943427908010986629362555343158680690459706276356506688329379847444374293270145193\
160459206129680912571554029144094292200870296236026454323071212875}
```

3. Lea la Sección 4.1.4 de la referencia [2] y realice el Ejercicio 3-c) (a mano!).
Desarrollo:

Se tienen los polinomios:

$$u(x) = 5x^5 + 4x^4 + 3x^3 + 2x^2 + x + 1$$
$$v(x) = 5x^5 - 4x^4 + 3x^3 - 2x^2 + x - 1$$

Las listas correspondientes a los polinomios $u(x)$ y $v(x)$ son:

$$U = [(5,5),(4,4),(3,3),(2,2),(1,1),(0,1)]$$
$$V = [(5,5),(4,-4),(3,3),(2,-2),(1,1),(0,-1)]$$

Ahora se construyen dos listas S_k correspondientes a los prodcutos de cada término de $v(x)$ con el polinomio $u(x)$:

$$S_1 = [(10,25),(9,20),(8,15),(7,10),(6,5),(5,5)]$$
$$S_2 = [(9,-20),(8,-16),(7,-12),(6,-8),(5,-4),(4,-4)]$$

$$S_3 = [(8,15),(7,12),(6,9),(5,6),(4,3),(3,3)]$$
$$S_4 = [(7,-10),(6,-8),(5,-6),(4,-4),(3,-2),(2,-2)]$$
$$S_5 = [(6,5),(5,4),(4,3),(3,2),(2,1),(1,1)]$$
$$S_6 = [(5,-5),(4,-4),(3,-3),(2,-2),(1,-1),(0,-1)]$$

Ahora se combinan S_1 con S_2, S_3 con S_4 y S_5 con S_6:

$$T_1 = [(10,25),(9,0),(8,-1),(7,-2),(6,-3),(5,1),(4,-4)]$$
$$T_2 = [(8,15),(7,2),(6,1),(5,0),(4,-1),(3,1),(2,-2)]$$
$$T_3 = [(6,5),(5,-1),(4,-1),(3,-1),(2,-1),(1,0),(0,-1)]$$

Ahora se combinan T_1 con T_2 en T_4 y este se combinará con T_3 en W:

$$T_4 = [(10,25),(9,0),(8,14),(7,0),(6-2),(5,1),(4,-5),(3,1),(2,-2)]$$

$$W = [(10,25),(9,0),(8,14),(7,0),(6,3),(5,0),(4,-6),(3,0),(2,-3),(1,0),(0,-1)]$$

Finalmente:

$$W(x) = 25x^{10} + 14x^8 + 3x^6 - 6x^4 - 3x^2 - 1$$

4. Sean a y b enteros y de tamaño $n = 3^t$, es decir que tanto a como b se pueden escribir como $a = A_2X^2 + A_1X + A_0 := a(X)$ y $b = B_2X^2 + B_1X + B_0 := b(X)$ con $A_2, A_1, A_0, B_2, B_1, B_0$ listas de $n/3$ digitos y $X = \beta^{n/3}$.El producto $c = ab$ se puede pensar como un polinomio de grado 4 en la indeterminada X,

$$\begin{aligned} c = ab &= (A_2X^2 + A_1X + A_0)(B_2X^2 + B_1X + B_0) \\ &= A_2B_2X^4 + A_2B_1X^3 + A_2B_0X^2 + A_1B_2X^3 + A_1B_1X^2 + A_1B_0X + A_0B_2X^2 + A_0B_1X + A_0B_0 \\ &= A_2B_2X^4 + (A_2B_1 + A_1B_2)X^3 + (A_2B_0 + A_1B_1 + A_0B_2)X^2 + (A_1B_0 + A_0B_1)X + A_0B_0 \\ &= C_4X^4 + C_3X^3 + C_2X^2 + C_1X + C_0 \end{aligned}$$

Los coeficientes $C_0, ..., C_4$ se pueden determinar evaluando $c(X)$ en cinco valores diferentes.Consideremos $x = 0, 1, -1, 2, \infty$:

- $c(\infty) = C_4 = a(\infty)b(\infty) = A_2B_2$
- $c(0) = C_0 = a(0)b(0) = A_0B_0$
- $c(-1) = C_4 - C_3 + C_2 - C_1 + C_0 = a(-1)b(-1) = A_2B_2 - A_2B_1 - A_1B_2 + A_2B_0 + A_1B_1 + A_0B_2 - A_1B_0 - A_0B_1 + A_0B_0$

  Que puede ser factorizado como $(B_0 + B_2 - B_1)(A_2 - A_1 + A_0)$.
- $c(1) = C_4 + C_3 + C_2 + C_1 + C_0 = a(1)b(1) = A_2B_2 + A_2B_1 + A_1B_2 + A_2B_0 + A_1B_1 + A_0B_2 + A_1B_0 + A_0B_1 + A_0B_0$

  Que puede ser factorizado como $(B_0 + B_2 + B_1)(A_2 + A_1 + A_0)$.

Explicitamente se tiene que $C_0 = A_0B_0$ y $C_4 = A_2B_2$, para encontrar los demas coeficientes del sistema se utilizará Mathematica:

```
Solve[c4 == a2*b2 && c0 == a0*b0 && c4 - c3 + c2 - c1 + c0 == (b0 + b2 - b1) (a2 - a1 + a0) &&
resuelve
  c4 + c3 + c2 + c1 + c0 == (b0 + b2 + b1) (a2 + a1 + a0) &&
  16 c4 + 8 c3 + 4 c2 + 2 c1 + c0 == 16*a2*b2 + 8*a2*b1 + 8*a1*b2 + 4*a2*b0 + 4*a1*b1 + 4*
     a0*b2 + 2*a1*b0 + 2*a0*b1 + a0*b0, {c1, c2, c3, c4, c0}]

Out[3]= {{c1 → a1 b0 + a0 b1, c2 → a2 b0 + a1 b1 + a0 b2, c3 → a2 b1 + a1 b2, c4 → a2 b2, c0 → a0 b0}}
```

De esta manera:

- $C_0 = A_0B_0$
- $C_1 = A_1B_0 + A_0B1$
  Utilizando el truco de Karatsuba se puede reescribir como:

  $A_0B_0 + A_1B_1 + (A_0 - A_1)(B_1 - B_0)$
- $C_2 = A_2B_0 + A_1B_1 + A_0B_2$
  Utilizando el truco de Karatsuba se puede reescribir como: $(A_0+A_1+A_2)(B_0+B_1+B_2)-A_0B_0-A_2B_2-(A_1B_0+A_0B_1)-(A_2B_1+A_1B_2)$
- $C_3 = A_2B_1 + A_1B_2$
  Utilizando el truco de Karatsuba se puede reescribir como: $A_1B_1 + A_2B_2 + (A_1 - A_2)(B_2 - B_1)$
- $C_4 = A_2B_2$

Por lo tanto el producto de a por b se puede escribir como:

$a \cdot b = A_0B_0 + (A_0B_0 + A_1B_1 + (A_0 - A_1)(B_1 - B_0))X + ((A_0 + A_1 + A_2)(B_0 + B_1 + B_2) - A_0B_0 - A_2B_2 - (A_1B_0 + A_0B_1) - (A_2B_1 + A_1B_2))X^2 + (A_1B_1 + A_2B_2 + (A_1 - A_2)(B_2 - B_1))X^3 + (A_2B_2)C^4$

De esta manera se tienen 5 productos y 12 sumas-restas.

Sea $T(n)$ el costo de la complejidad del producto de dos n-dígitos con $n = 3^t$.Teniendo en cuenta que el algoritmo de Karatsuba utilizaba 5 productos y 12 sumas-restas, entonces se obtiene que $T(n) = 5T(n/2) + Cn$ con C una constante positiva.

Si $a = 5$ y $b = 3$ entonces $log_3(5) - \epsilon = 1$ luego $\epsilon = log_3(5) - 1 = 0.46497 > 0$.Por lo tanto por el teorema maestro $T(n) = O(n^{log_3(5)}) = O(n^{1.46497})$

5. **(Mathematica)** Un entero n cumple la propiedad AUV-2 si n es producto de tres primos impares diferentes y además la ecuación diofántica $x^2 + y^2 = n$ tiene solución. Por ejemplo, 1087985 cumple esta propiedad, en efecto:

   $1087985 = 5 * 37 * 5881$

   Además, $x^2 + y^2 = 1087985$ tiene solución, por ejemplo (92, 1039) es una de ellas. Encuentre todos los enteros que satisfacen la propiedad AUV-2 en el intervalo $[2, 10^4]$

   Desarrollo:

   Para determinar los números que satisfacen la propiedad AUV-2, es necesario reducir la lista de posibles números primos que se pueden multiplicar en ternas para generar números entre 2 y 10000. Para esto, se tomará el producto de 3 y 5, (los dos primos impares más pequeños posibles) y se multiplicará con otros números primos hasta encontrar el máximo primo p tal que $3 * 5 * p < 10000$. Además, se guardarán en una lista para ahorrar tiempo de cómputo a la hora de realizar las multiplicaciones entre primos.

In[2]:=
```
primos = {3, 5};
n = NextPrime[5];
While[15 * n < 10 000, AppendTo[primos, n]; n = NextPrime[n]]
```
In[5]:=
```
Print[primos]
```
```
{3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113,
 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241,
 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383,
 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 521, 523,
 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 661}
```
In[6]:=
```
longprimos = Length[primos]
```

El algoritmo consiste en tres bucles anidados, el bucle más externo comienza desde el primer elemento del arreglo, mientras que el segundo y tercero inician desde el segundo y tercer elemento respectivamente, asegurando así que no haya dos primos iguales a la hora de realizar el producto. En el bucle más interno, se realiza el producto de los tres primos en las posiciones i, j, k del arreglo, a continuación, se evalúa si dicho producto es mayor a 10^4, en caso de serlo, se procede a elegir otros elementos de la lista. Si es menor a 10^4, se verifica si es solución a la ecuación diofántica con el comando "PowersRepresentation", si lo es, se añade a la lista de números que cumple con la propiedad AUV-2.

```
i = 1;
numsauv2 = {};
(*Bucle para elegir el primer elemento del producto*)While[i ≤ longprimos, j = i + 1;
  (*Bucle para elegir el segundo elemento del producto*)While[j ≤ longprimos, k = j + 1;
   (*Bucle para elegir el tercer elemento del producto*)
   While[k ≤ longprimos, producto = primos[[i]] * primos[[j]] * primos[[k]];
    (*Evalúa si el poducto es mayor a 10^4*)If[producto > 10 000, Break[]];
    (*Evalúa si hay solución para la ecuación diofántica*)r = PowersRepresentations[producto, 2, 2];
    (*En caso de haber soluciones, añade el
      producto a la lista de números AUV-2*)If[Length[r] > 0, AppendTo[numsauv2, producto]];
    k = k + 1];
   j = j + 1];
  i = i + 1];
(*Muestra la lista de números que complen con la propiedad AUV-2 entre 2 y 10^4*)
Print[numsauv2];
```
```
{1105, 1885, 2405, 2665, 3445, 3965, 4745, 5785, 6305, 6565, 7085, 7345, 8905, 9685, 2465, 3145, 3485,
 4505, 5185, 6205, 7565, 8245, 8585, 9265, 9605, 5365, 5945, 7685, 8845, 7585, 9805, 6409, 8177, 9061}
```