

# Generador de Grafos Combinados de Expresiones Reducidas usando SAGE y Python

Gonzalo Jiménez

December 27, 2023

## Abstract

En este documento se indica cómo usar los scripts desarrollados para generar grafos combinados de expresiones reducidas para elementos de cualquier grupo simétrico  $S_n$ . Se asume que se dispone del software y hardware necesario para ejecutar código SAGE y Python.

## 1 Preparando el ambiente en SAGE

1. En SAGE, con la instrucción

---

```
# W es un nombre arbitrario para el grupo. 's' es  
# un nombre arbitrario para los generadores  
# 'A' es el tipo de Grupo de Weyl y 4 es el  
# tamaño del conjunto generador. 0 sea, en este  
# caso se está generando el grupo simétrico  
# $S_3$,  
# pues los generadores serían s_1, s_2 y s_3 donde  
# s_i corresponde a la transposición (i, i+1).  
W = WeylGroup(['A', 3], prefix='s');
```

---

Asignamos a  $W$  el objeto  $S_4$ . Es posible verificar esto, dando a SAGE la instrucción

---

```
W.list()
```

---

Que devolverá lo siguiente

---

```
(1,  
s3,  
s3*s2,  
s3*s2*s1,
```

```

s2,
s2*s3,
s2*s3*s2,
s2*s3*s2*s1,
s2*s1,
s2*s3*s1,
s2*s3*s1*s2,
s2*s3*s1*s2*s1,
s1,
s3*s1,
s3*s1*s2,
s3*s1*s2*s1,
s1*s2,
s1*s2*s3,
s1*s2*s3*s2,
s1*s2*s3*s2*s1,
s1*s2*s1,
s1*s2*s3*s1,
s1*s2*s3*s1*s2,
s1*s2*s3*s1*s2*s1)

```

---

2. La lista anterior es una tupla ordenada. Luego, es posible rescatar un elemento del grupo, referenciando a su posición en la lista. Por ejemplo, podemos asignar el elemento más largo a “w0” mediante la instrucción

---

```
w0=W[23]
```

---

*Remark 1.1.* Notar que, como ocurre habitualmente en programación, la primera posición en la lista corresponde a la posición 0. En consecuencia, el “elemento 24” se encuentra en la posición 23.

3. Ahora, es posible generar el RexGraph del elemento seleccionado. Siguiendo con el ejemplo, la instrucción sería la siguiente

---

```

#nuevamente G es un nombre arbitrario
G=w0.reduced_word_graph();

```

---

4. Acá estamos asignando a  $G$  un grafo generado por el software, pero está de alguna manera “protegido” (es un objeto *immutable*). Como queremos alterarlo, simplemente hacemos una copia y cambiamos su condición de inmutabilidad con la siguiente instrucción

---

```

#nuevamente H es un nombre arbitrario
H=G.copy(immutable=False);

```

---

5. De acá, obtenemos un grafo  $H$  que podemos manipular. En particular, queremos identificar algunos vértices. Para esto, generamos la lista de vértices del grafo  $H$  y posteriormente tratamos esos elementos en Python, para saber qué qué código ingresar a SAGE. La lista de vértices se obtiene con la instrucción

---

```
H.vertices()
```

---

En nuestro ejemplo, entrega el siguiente arreglo

---

```
[(1, 2, 1, 3, 2, 1),  
(1, 2, 3, 1, 2, 1),  
(1, 2, 3, 2, 1, 2),  
(1, 3, 2, 1, 3, 2),  
(1, 3, 2, 3, 1, 2),  
(2, 1, 2, 3, 2, 1),  
(2, 1, 3, 2, 1, 3),  
(2, 1, 3, 2, 3, 1),  
(2, 3, 1, 2, 1, 3),  
(2, 3, 1, 2, 3, 1),  
(2, 3, 2, 1, 2, 3),  
(3, 1, 2, 1, 3, 2),  
(3, 1, 2, 3, 1, 2),  
(3, 2, 1, 2, 3, 2),  
(3, 2, 1, 3, 2, 3),  
(3, 2, 3, 1, 2, 3)]
```

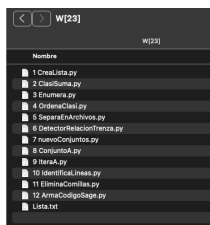
---

## 2 Tratamiento en Python

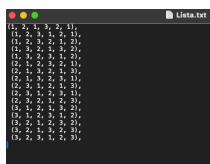
1. Recomiendo la creación de una carpeta de nombre  $W[X]$ , donde  $X$  corresponda a la posición del elemento a estudiar. En nuestro ejemplo, nuestra carpeta se llamará  $W[23]$ .

*Remark 2.1.* El elemento  $W[23]$  en el contexto  $S_4$  probablemente sea distinto del elemento  $W[23]$  en el contexto  $S_5$ . Si se prefiere, se puede agregar un identificador del grupo ambiente también. Acá se omitirá para no saturar la notación.

2. En esta carpeta se deben agregar los 12 archivos .py y un archivo Lista.txt, cuyo contenido será la lista generada en la sección anterior. La carpeta debería verse así.



El archivo *Lista.txt* se verá de la siguiente forma



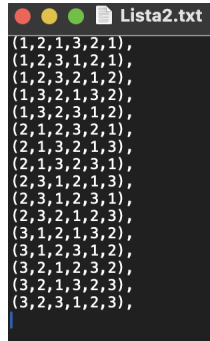
*Remark 2.2.* Nótese que para el correcto funcionamiento de los scripts, es necesario lo siguiente

- (a) copiar la lista generada por SAGE sin los corchetes de los extremos.  
[ ]
- (b) agregar una coma a la tupla final
- (c) agregar un salto de línea.

Hecho esto, podemos pasar a ejecutar el script1.

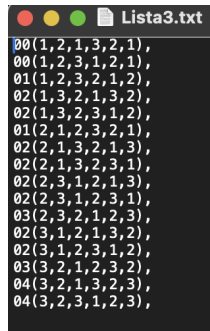
*Remark 2.3.* Para esto, yo uso el software Visual Studio Code. Es bueno recordar que se debe hacer referencia a la carpeta  $W[X]$  donde se está trabajando.

3. Se ejecuta el script1 *1CreaLista.py*. Esto genera al archivo *Lista2.txt*, que tiene el siguiente aspecto



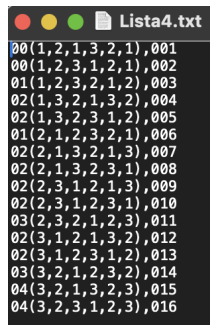
```
(1,2,1,3,2,1),
(1,2,3,1,2,1),
(1,2,3,2,1,2),
(1,3,2,1,3,2),
(1,3,2,3,1,2),
(2,1,2,3,2,1),
(2,1,3,2,1,3),
(2,1,3,2,3,1),
(2,3,1,2,1,3),
(2,3,1,2,3,1),
(2,3,2,1,2,3),
(2,3,2,1,3,1),
(2,3,2,1,3,2),
(3,1,2,1,3,2),
(3,1,2,3,1,2),
(3,2,1,2,3,2),
(3,2,1,3,2,3),
(3,2,3,1,2,3),
```

4. Se ejecuta el script2 *2ClasiSuma.py*. Esto genera al archivo *Lista3.txt*, que tiene el siguiente aspecto



```
00(1,2,1,3,2,1),
00(1,2,3,1,2,1),
01(1,2,3,2,1,2),
02(1,3,2,1,3,2),
02(1,3,2,3,1,2),
01(2,1,2,3,2,1),
02(2,1,3,2,1,3),
02(2,1,3,2,3,1),
02(2,3,1,2,1,3),
02(2,3,1,2,3,1),
02(2,3,2,1,2,3),
03(2,3,2,1,3,1),
02(2,3,2,1,3,2),
02(3,1,2,1,3,2),
02(3,1,2,3,1,2),
03(3,2,1,2,3,2),
04(3,2,1,3,2,3),
04(3,2,3,1,2,3),
```

5. Se ejecuta el script3 *3Enumera.py*. Esto genera al archivo *Lista4.txt*, que tiene el siguiente aspecto



```
00(1,2,1,3,2,1),001
00(1,2,3,1,2,1),002
01(1,2,3,2,1,2),003
02(1,3,2,1,3,2),004
02(1,3,2,3,1,2),005
01(2,1,2,3,2,1),006
02(2,1,3,2,1,3),007
02(2,1,3,2,3,1),008
02(2,3,1,2,1,3),009
02(2,3,1,2,3,1),010
03(2,3,2,1,2,3),011
02(3,1,2,1,3,2),012
02(3,1,2,3,1,2),013
03(3,2,1,2,3,2),014
04(3,2,1,3,2,3),015
04(3,2,3,1,2,3),016
```

*Remark 2.4.* Notar que el script podría necesitar modificarse dependiendo de cuántos elementos se deben numerar. La modificación requiere que la enumeración final tenga el mismo largo (en caracteres) para todos los elementos. De ahí que se agregan ceros según corresponda.

- 00 (1, 2, 1, 3, 2, 1), #001  
01 (1, 2, 3, 1, 2, 1), #002  
02 (1, 2, 3, 2, 1, 2), #003  
03 (2, 1, 2, 3, 2, 1), #006  
04 (1, 3, 2, 1, 3, 2), #004  
05 (1, 3, 2, 3, 1, 2), #005  
06 (2, 1, 3, 2, 1, 3), #007  
07 (2, 1, 3, 2, 3, 1), #008  
08 (2, 3, 1, 2, 3, 1), #009  
09 (2, 3, 1, 2, 1, 3), #010  
10 (3, 1, 2, 1, 3, 2), #012  
11 (3, 1, 2, 1, 3, 2), #013  
12 (3, 2, 3, 1, 2, 3), #011  
13 (3, 2, 1, 2, 3, 2), #014  
14 (3, 2, 1, 3, 2, 3), #015  
15 (3, 2, 3, 1, 2, 3), #016

7. Antes de ejecutar el script5, se debe adaptar dependiendo del mayor índice prefijo en el archivo anterior. En nuestro caso es 04, por lo que se debe usar 5 (4+1) en el range del script. Lo mismo sucede en los scripts que sigan. Ver la siguiente imagen.

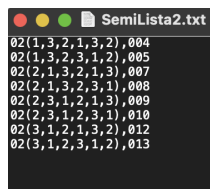
```

5 dependiente de python 2.7
1 1. Listado a semilla x
2 2. Pasara los datos en archivos distintos segun prefijos
3 3. Atención al rango 11, para los prefijos llegan hasta 10
4 suma=0
5
6 for i in range(11):
7
8     read(i)
9
10     g=open('Semilla'+str(i)+'.txt','w')
11     f=open('Lista'+str(i)+'.r')
12
13     for lines in f:
14
15         suma= int(lines[:2])
16         if suma == 1:
17             g.write(lines[:3]+'n')
18
19     g.close()
20     f.close()

```

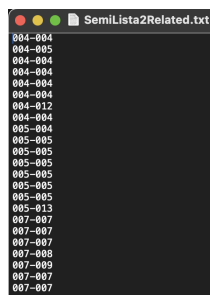
- 11 EliminaCornillas.py
- 12 ArmaCodigoSage.py
- Lista.txt
- Lista2.txt
- Lista3.txt
- Lista4.txt
- Lista5.txt
- SemiLista0.txt
- SemiLista1.txt
- SemiLista2.txt
- SemiLista3.txt
- SemiLista4.txt

los archivos tienen el siguiente aspecto



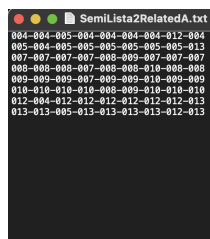
```
02(1,3,2,1,3,2),004
02(1,3,2,3,1,2),005
02(2,1,3,2,1,3),007
02(2,1,3,2,3,1),008
02(2,3,1,2,1,3),009
02(2,3,1,2,3,1),010
02(3,1,2,1,3,2),012
02(3,1,2,3,1,2),013
```

8. Se ajusta el range en el script6 si es necesario y se ejecuta. Una vez ejecutado el script6 *6DetectorRelacionTrenza.py* se generan los archivos *SemiListaXRelated.txt*, cuyo contenido se ve así.



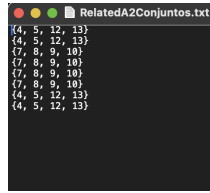
```
004-004
004-005
004-004
004-004
004-004
004-012
004-004
005-004
005-005
005-005
005-005
005-005
005-005
005-005
005-013
007-007
007-007
007-007
007-008
007-009
007-007
007-007
```

9. Se ajusta el range en el script7 si es necesario y se ejecuta. Una vez ejecutado el script7 *7nuevoConjuntos.py* se generan los archivos *SemiListaXRelatedA.txt*, cuyo contenido se ve así.



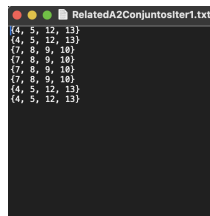
```
004-004-005-004-004-004-012-004
005-004-005-005-005-005-005-013
007-007-007-007-008-009-007-007-007
009-009-009-007-008-009-010-008-008
009-009-009-009-009-009-010-009-009
010-010-010-010-008-009-010-010-010
012-004-012-012-012-012-012-013
013-013-005-013-013-013-013-012-013
```

10. Se ajusta el range en el script8 si es necesario y se ejecuta. Una vez ejecutado el script8 *8ConjuntoA.py* se generan los archivos *RelatedAXConjuntos.txt*, cuyo contenido se ve así.



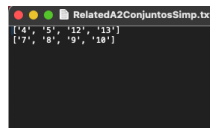
```
{4, 5, 12, 13}
{4, 5, 12, 13}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{4, 5, 12, 13}
{4, 5, 12, 13}
```

11. Se ajusta el range en el script9 si es necesario y se ejecuta. Una vez ejecutado el script9 *9IteraA.py* se generan los archivos *RelatedAXConjuntosIter1.txt*, cuyo contenido se ve así.



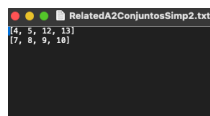
```
{4, 5, 12, 13}
{4, 5, 12, 13}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{7, 8, 9, 10}
{4, 5, 12, 13}
{4, 5, 12, 13}
```

12. Se ajusta el range en el script10 si es necesario y se ejecuta. Una vez ejecutado el script10 *10IdentificaLineas.py* se generan los archivos *RelatedAXConjuntosSimp.txt*, cuyo contenido se ve así.



```
{4, 5, 12, 13}
{7, 8, 9, 10}
```

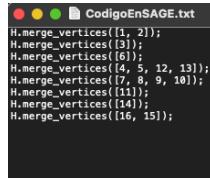
13. Se ajusta el range en el script11 si es necesario y se ejecuta. Una vez ejecutado el script11 *11EliminaComillas.py* se generan los archivos *RelatedAXConjuntosSimp2.txt*, cuyo contenido se ve así.



```
{4, 5, 12, 13}
{7, 8, 9, 10}
```



14. Se ajusta el range en el script12 si es necesario y se ejecuta. Una vez ejecutado el script12 *12ArmaCodigoSage.py* se genera el archivo *CodigoEnSAGE.txt*, cuyo contenido se ve así.



### 3 De vuelta a SAGE

Ya se tiene todo el código necesario. Para generar el grafo en 3 dimensiones, basta con ejecutar el siguiente código.

---

```
W = WeylGroup(['A',3], prefix='s');
w0=W.long_element();
G=w0.reduced_word_graph();
H=G.copy(immutable=False);
H.relabel(range(1,769),return_map=True);
#Acá va el c\ódigo generado en python
H.show3d()
```

---

*Remark 3.1.* El rango  $X$  de la re etiquetación de  $H$  ( $relabel(1, X)$ ) depende del tamaño del grafo con el que se está trabajando. Se puede jugar sobre seguro introduciendo un número muy grande, y si no es suficiente, se puede subir la apuesta.

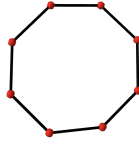
En nuestro ejemplo, el código a introducir en SAGE es el siguiente.

---

```
W = WeylGroup(['A',3], prefix='s');
w0=W.long_element();
G=w0.reduced_word_graph();
H=G.copy(immutable=False);
H.relabel(range(1,769),return_map=True);
H.merge_vertices([1, 2]);
H.merge_vertices([3]);
H.merge_vertices([6]);
H.merge_vertices([4, 5, 12, 13]);
H.merge_vertices([7, 8, 9, 10]);
H.merge_vertices([11]);
H.merge_vertices([14]);
H.merge_vertices([16, 15]);
H.show3d()
```

---

Esto genera el siguiente grafo.



Otro ejemplo. Introduciendo el código

---

```

W = WeylGroup(['A',4], prefix='s');
w0=W.long_element();
G=w0.reduced_word_graph();
H=G.copy(immutable=False);
H.relabel(range(1,769),return_map=True);
H.merge_vertices([1, 2, 3, 5, 6, 15, 16, 17, 19, 20, 34, 35]);
H.merge_vertices([169, 170, 171, 173, 174]);
H.merge_vertices([4, 36, 7, 18, 21]);
H.merge_vertices([39, 24, 25, 28]);
H.merge_vertices([37, 38, 8, 9, 45, 46, 22, 23]);
H.merge_vertices([26, 27, 29, 30, 31, 32, 40, 41, 42, 43]);
H.merge_vertices([385, 386, 71, 72, 394, 395, 398, 80, 81, 84]);
H.merge_vertices([172, 175]);
H.merge_vertices([239, 243, 244, 245, 183, 187, 188, 189]);
H.merge_vertices([128, 129, 130, 131, 132, 133, 134, 135, 136, 10,
    11, 12, 13, 14, 143, 144, 50, 51, 52, 53, 54, 55, 56, 57, 58,
    59, 66, 67, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610,
    617, 618, 127]);
H.merge_vertices([387, 388, 73, 74, 396, 397, 399, 400, 401, 82,
    83, 402, 85, 86, 87, 88]);
H.merge_vertices([33, 44]);
H.merge_vertices([256, 258, 259, 260, 261, 262, 263, 267, 294,
    295, 296, 297, 298, 299, 303, 184, 185, 186, 191, 192, 193,
    194, 195, 196, 200, 202, 203, 204, 205, 206, 207, 211, 240,
    241, 242, 247, 248, 249, 250, 251, 252]);
H.merge_vertices([190, 246]);
H.merge_vertices([441, 442]);
H.merge_vertices([176, 177]);
H.merge_vertices([531, 532, 533, 534, 535, 536, 537, 538, 539,
    540, 544, 545, 75, 76, 77, 78, 79, 90, 91, 92, 93, 94, 95, 96,
    97, 98, 99, 103, 104, 106, 107, 108, 109, 110, 111, 112, 113,
    114, 115, 119, 120, 389, 390, 391, 392, 393, 404, 405, 406,
    407, 408, 409, 410, 411, 412, 413, 417, 418, 420, 421, 422,
    423, 424, 425, 426, 427, 428, 429, 433, 434]);
H.merge_vertices([68, 137, 138, 140, 145, 611, 612, 614, 619, 60,
    61, 63]);
H.merge_vertices([48, 49]);
H.merge_vertices([89, 403]);

```

```

H.merge_vertices([643, 644, 645, 646, 647, 329, 330, 331, 332,
333, 218, 219, 220, 221, 222, 178, 179, 180, 181, 182]);
H.merge_vertices([275, 276, 277, 278, 279, 280, 284, 285, 286,
310, 311, 312]);
H.merge_vertices([443, 444]);
H.merge_vertices([450, 476]);
H.merge_vertices([257, 197, 198, 199, 264, 265, 266, 201, 268,
208, 209, 210, 212, 300, 301, 302, 304, 253, 254, 255]);
H.merge_vertices([64, 65, 69, 70, 139, 141, 142, 146, 147, 613,
615, 616, 620, 621, 62]);
H.merge_vertices([541, 414, 415, 416, 542, 543, 419, 546, 100,
101, 102, 105, 430, 431, 432, 435, 116, 117, 118, 121]);
H.merge_vertices([704, 705, 707, 148, 149, 153, 154, 156, 622,
623, 627, 628, 630, 699, 700]);
H.merge_vertices([320, 321, 322, 323, 324, 269, 270, 271, 272,
273, 213, 214, 215, 216, 217, 305, 306, 307, 308, 309]);
H.merge_vertices([648, 651, 652, 653, 334, 337, 338, 339, 664,
667, 668, 669, 350, 223, 353, 226, 227, 228, 354, 355]);
H.merge_vertices([281, 282, 283, 287, 288, 289, 290, 291, 292,
313, 314, 315, 316, 317, 318]);
H.merge_vertices([448, 449, 460, 461, 462, 463, 464, 552, 553,
554, 555, 556, 496, 497, 498, 499, 500, 445, 446, 447]);
H.merge_vertices([451, 452, 453, 454, 455, 456, 477, 478, 479,
480, 481, 482, 486, 487, 488]);
H.merge_vertices([293, 319]);
H.merge_vertices([587, 588, 589, 590, 591, 547, 548, 549, 550,
551, 436, 437, 438, 439, 440, 122, 123, 124, 125, 126]);
H.merge_vertices([706, 708, 709, 150, 155, 157, 158, 624, 629,
631, 632, 701]);
H.merge_vertices([335, 336, 340, 341, 342, 343, 344, 345, 346,
347, 348, 349, 351, 352, 356, 357, 358, 359, 360, 361, 362,
363, 364, 365, 376, 377, 378, 379, 380, 649, 650, 654, 655,
656, 657, 658, 659, 660, 661, 662, 663, 665, 666, 670, 671,
672, 673, 674, 675, 676, 677, 678, 679, 690, 691, 692, 693,
694, 224, 225, 229, 230, 231, 232, 233, 234, 235, 236, 237,
238]);
H.merge_vertices([325, 326]);
H.merge_vertices([457, 458, 459, 483, 484, 485, 489, 490, 491,
492, 493, 494]);
H.merge_vertices([512, 514, 515, 516, 465, 467, 468, 469, 504,
557, 559, 560, 561, 501, 503, 568, 505, 570, 571, 572]);
H.merge_vertices([680, 366]);
H.merge_vertices([720, 721]);
H.merge_vertices([640, 641, 642, 151, 152, 159, 160, 161, 162,
163, 164, 165, 166, 167, 168, 702, 703, 710, 711, 712, 713,
714, 715, 716, 717, 718, 719, 625, 626, 755, 756, 757, 758,
759, 633, 634, 635, 636, 637, 638, 639]);

```

```

H.merge_vertices([328, 327]);
H.merge_vertices([681, 682, 683, 684, 686, 367, 368, 369, 370,
687, 372, 373, 695, 696, 381, 382]);
H.merge_vertices([513, 517, 518, 519, 520, 521, 522, 527, 528,
529, 558, 562, 563, 564, 565, 566, 567, 569, 573, 574, 575,
576, 577, 578, 583, 584, 585, 466, 470, 471, 472, 473, 474,
475, 502, 506, 507, 508, 509, 510, 511]);
H.merge_vertices([523, 579]);
H.merge_vertices([592, 593]);
H.merge_vertices([736, 725]);
H.merge_vertices([384, 685, 688, 689, 371, 374, 375, 697, 698,
383]);
H.merge_vertices([580, 581, 582, 586, 524, 525, 526, 530]);
H.merge_vertices([594, 597]);
H.merge_vertices([746, 747, 723, 724, 760, 761, 731, 732]);
H.merge_vertices([726, 727, 728, 729, 737, 738, 739, 740, 742,
743]);
H.merge_vertices([595, 596, 598, 599, 600]);
H.merge_vertices([741, 744, 745, 730]);
H.merge_vertices([765, 748, 751, 762, 733]);
H.merge_vertices([768, 734, 735, 749, 750, 752, 753, 754, 763,
764, 766, 767]);
H.show3d()

```

---

Se obtiene el grafo combinado de expresiones del elemento más largo de  $S_5$ . Es decir, el elemento  $s_1s_2s_1s_3s_2s_1s_4s_3s_2s_1$

## 4 Resumen

Tenemos un método para generar Grafos combinados de expresiones reducidas, que si bien no es óptimo, tiene la ventaja de ser adaptable y claro.

Desconozco si existen comandos que generen los mismos resultados de maneras más directas o incluso otro software que sea más adecuado para esta tarea específica, pero en su momento fue útil y logró su objetivo. Libero el código con la intención de no se haga el mismo trabajo 2 veces y, si alguien quisiera, que lo mejore de alguna forma.