Construcción del juego *Dobble*

Análisis y diseño de algoritmos

Juan Cisneros Resendiz

9 de diciembre de 2020

¿Que es el juego *Dobble*?

- El juego *Dobble* consiste en un mazo de **55 cartas**, cada carta tiene **8 símbolos**, y dos cartas diferentes tienen exáctamente un símbolo en común. Hay **57 símbolos diferentes** total.
- Este juego se juega sacando dos cartas del mazo, y el primer jugador que encuentre el símbolo en común gana.

¿En que estamos interesados acerca del juego?

- **¿Como podemos construir ese juego *Dobble*?**
- **¿Existen juegos tipo *Dobble* para un número diferente de símbolos o cartas? ¿El número de símbolos determina el número de cartas o viceversa?**
- **¿Como podemos contruirlos?**
- **¿Podemos construir esos juegos de manera *óptima* (maximizando el número de cartas dado un número de símbolos)?** *spoiler:* No lo sé, pero encontré una clase infinita de juegos de *Dobble* que puedo generar.

Definición de un juego *Dobble*

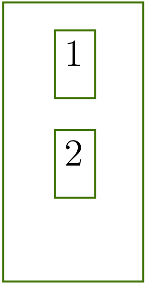
Antes de poder responder esas preguntas, necesitamos definir que es un juego *Dobble*.

Definición

Un juego *Dobble* D de orden n es un conjunto de símbolos Σ, y un conjunto C de subconjuntos de tamaño n de Σ llamados cartas, con las siguientes propiedades.

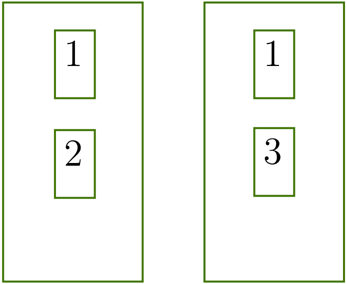
- $D1$: **Cada par de cartas tiene únicamente un símbolo en común**, es decir, para dos cartas diferentes $C_1 \neq C_2$, tenemos que $C_1 \cap C_2 = s$, con $s \in \Sigma$.
- $D2$: **No existe un símbolo en común para todas las cartas**, es decir $\bigcap_{k \in C} = \emptyset$ (¿Porque es necesaria esta condición?)

Ejemplo de un juego *Dobble* de orden 2



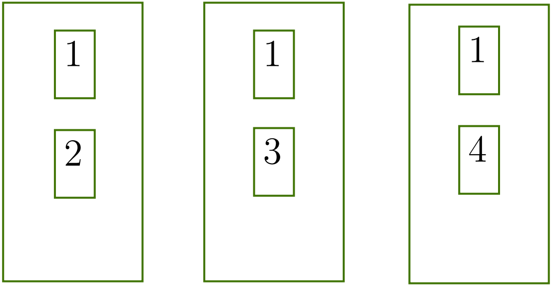
Ejemplo de un juego *Dobble* de orden 2

Ahora hay dos casos, agregar una carta con el número 1 o sin el



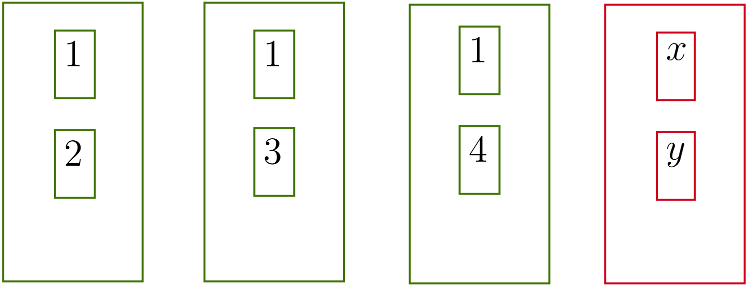
Ejemplo de un juego *Dobble* de orden 2

Caso uno, agregamos otra carta con 1



Ejemplo de un juego *Dobble* de orden 2

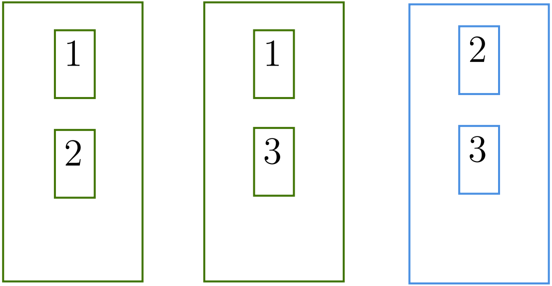
Ahora, como no podemos tener para todas las cartas un símbolo en común, necesitamos poder agregar una carta sin un 1.



Pero vemos que como no podemos usar el 1 en la carta roja, necesitamos tres símbolos diferentes en la carta roja (el $2, 3$ y 4), pero solo tenemos dos espacios, así que este caso no es posible.

Ejemplo de un juego *Dobble* de orden 2

Caso 2: agregamos una carta sin el número 1.



- Por inspección podemos ver que ya no podemos agregar más cartas al juego con los símbolos actuales (1,2,3),
- Usando un argumento similar al caso 1 podemos ver que tampoco no podemos agregar otra carta con un símbolo nuevo.

Ejemplo de un juego *Dobble* de orden 3

- Primero veamos cuantas veces podemos agregar el símbolo 1 en el mazo de cartas.
- Agregamos una carta con el símbolo 1

$$\boxed{\begin{array}{c} \boxed{1} \\ \boxed{2} \\ \boxed{3} \end{array}}$$

Ejemplo de un juego *Dobble* de orden 3

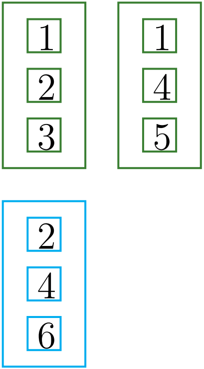
- Primero veamos cuantas veces podemos agregar el símbolo 1 en el mazo de cartas.
- Agregamos una carta con el símbolo 1
- Ya que no puede haber un símbolo en común para todas las cartas (D2), entonces tiene que existir una carta sin el símbolo 1

$$\boxed{\begin{array}{c}\boxed{1}\\ \boxed{2}\\ \boxed{3}\end{array}}$$

$$\boxed{\begin{array}{c}\boxed{2}\\ \boxed{4}\\ \boxed{6}\end{array}}$$

Ejemplo de un juego *Dobble* de orden 3

- Ahora agregamos otra carta con el símbolo 1



Ejemplo de un juego *Dobble* de orden 3

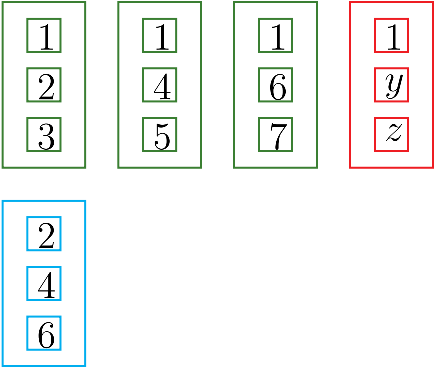
- Tambien podemos agregar una tercera carta con el símbolo 1

1	1	1
2	4	6
3	5	7

2
4
6

Ejemplo de un juego *Dobble* de orden 3

- Ahora tratemos de agregar una cuarta carta con el símbolo 1



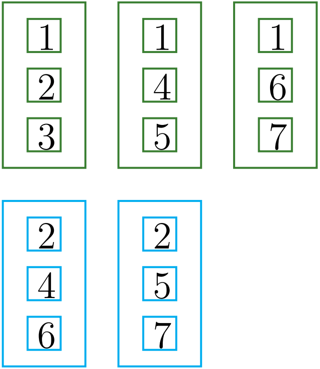
Ejemplo de un juego *Dobble* de orden 3

- Cada carta que tiene el símbolo 1 (las verdes) tambien tienen un símbolo **diferente** de la carta $C = \{2, 4, 6\}$ (la azul)
- Cada símbolo de la carta C están tambien en una carta verde.
- Por lo tanto no podemos agregar otra carta con el símbolo 1, ya que al querer relacionarla con la carta C tendriamos dos símbolos en común con una carta verde.

1	1	1	1	2
2	4	6	y	4
3	5	7	z	6

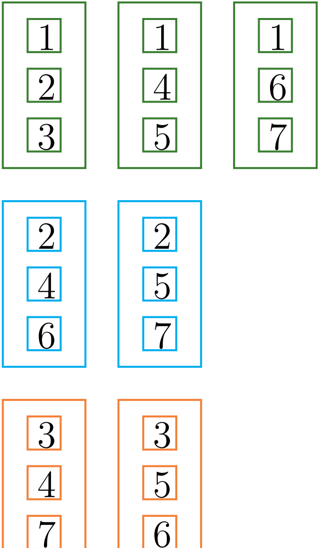
Ejemplo de un juego *Dobble* de orden 3

- Solo tenemos 3 cartas con el símbolo 1
- Ahora agregamos 2 cartas mas con el símbolo 2 para tener 3 en total con el símbolo 2



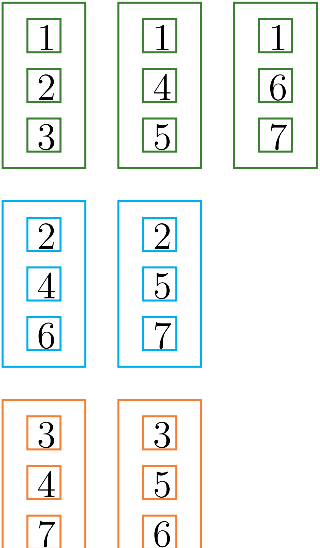
Ejemplo de un juego *Dobble* de orden 3

- Con un razonamiento similar al caso anterior no podemos agregar una cuarta carta con el símbolo 2
- Agregamos ahora dos cartas con el símbolo 3



Ejemplo de un juego *Dobble* de orden 3

- Podemos ver que hay tres repeticiones de cada símbolo
- Con un argumento similar al anterior podemos probar que no podemos agregar más cartas al mazo.



Ejemplo de un juego *Dobble* de orden 3

- Tambien lo podemos ver de la siguiente manera.
- Podemos considerar los pares de símbolos que podemos formar con las cartas. Por ejemplo la carta $\{1,2,3\}$ forma los pares $1,2$; $2,3$ y $1,3$
- Un par x, y está en a lo mucho una carta (D1)
- Así que las cartas forman pares diferentes.
- Con los símbolos $1 \ldots 7$ podemos formar $\binom{7}{2}$ pares.
- Hay 7 cartas, y podemos formar $\binom{3}{2}$ pares, en total tenemos $7\binom{3}{2} = 7(3 \cdot 2)/2 = (7 \cdot 6)/2 = \binom{7}{2} = 21$, eso es, tenemos todos los pares posibles.

Propiedades del juego *Dobble*

- En general tenemos que un juego *Dobble* de orden n un símbolo s puede aparecer en a lo más n cartas.

Propiedades del juego *Dobble*

- En general tenemos que un juego *Dobble* de orden n un símbolo s puede aparecer en a lo más n cartas.
- A los juegos *Dobble* en los que cada símbolo símbolo aparece en exáctamente n cartas se les llama juegos *Dobble* completos.

Propiedades del juego *Dobble*

- En general tenemos que un juego *Dobble* de orden n un símbolo s puede aparecer en a lo más n cartas.
- A los juegos *Dobble* en los que cada símbolo símbolo aparece en exáctamente n cartas se les llama juegos *Dobble* completos.
- Un juego *Dobble* de orden n se dice que es *óptimo* si este es un juego de orden n con el número máximo de cartas.

Propiedades del juego *Dobble*

- En general tenemos que un juego *Dobble* de orden n un símbolo s puede aparecer en a lo más n cartas.
- A los juegos *Dobble* en los que cada símbolo símbolo aparece en exáctamente n cartas se les llama juegos *Dobble* completos.
- Un juego *Dobble* de orden n se dice que es *óptimo* si este es un juego de orden n con el número máximo de cartas.
- Un juego *Dobble* completo de orden n es un juego óptimo.

Propiedades del juego *Dobble*

- En general tenemos que un juego *Dobble* de orden n un símbolo s puede aparecer en a lo más n cartas.
- A los juegos *Dobble* en los que cada símbolo símbolo aparece en exáctamente n cartas se les llama juegos *Dobble* completos.
- Un juego *Dobble* de orden n se dice que es *óptimo* si este es un juego de orden n con el número máximo de cartas.
- Un juego *Dobble* completo de orden n es un juego óptimo.
- Tambien tenemos que, en general en un juego *Dobble* completo aparece cada par de símbolos $x, y \in \Sigma$ exáctamente una vez.

Propiedades del juego *Dobble*

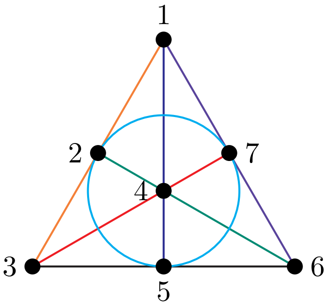
Ahora hacemos una lista de las propiedades del juego *Dobble* de orden n

- $D1$: **Cada par de cartas tiene únicamente un símbolo en común**, es decir, para dos cartas diferentes $C_1 \neq C_2$, tenemos que $C_1 \cap C_2 = s$, con $s \in \Sigma$.
- $D2$: **No existe un símbolo en común para todas las cartas**, es decir $\bigcap_{k \in C} = \emptyset$
- $D3$: **Un par de símbolos a_1 y a_2 está en a lo mas en una carta**
- $D4$: En un juego *Dobble* de orden n completo, **Un par de símbolos a_1 y a_2 está en a lo mas en una carta**

Representación geométrica del juego *Dobble*

Representemos las cartas de un juego *Dobble* como líneas y los símbolos de esas cartas como puntos incidentes a esas líneas.

n=3	1	2	3	4	5	6	7
1	1	2	3				
2	1			4	5		
3	1					6	7
4		2		4		6	
5		2			5		7
6			3	4			7
7			3		5	6	



Representación geométrica del juego *Dobble*

Podemos reescribir las propiedades de un juego *Dobble* de orden n en terminos de puntos y líneas

- $D1$: Cada par de líneas intersectan en exáctamente un punto.
- $D2$: No hay un punto que sea incidente a todas las líneas.
- $D3$: Hay a lo mucho una sola línea que pasa por el par de puntos p_1 y p_2, $p_1 \neq p_2$
- $D4$: En un juego *Dobble* de orden n completo, **hay exáctamente una línea que pasa por el par de puntos p_1 y p_2, $p_1 \neq p_2$**

Representación geométrica del juego *Dobble*

- Existe un objeto matemático que en un principio es muy parecido a esta reprentación geométrica.
- Este objeto es el plano proyectivo finito.

Definición plano proyectivo finito

Un plano proyectivo es un conjunto de puntos P y un conjunto de subconjuntos de P llamados líneas, que cumplen lo siguiene:

- $P1$: Existe exáctamente una línea que pasa por un par de puntos distintos.
- $P2$: Cada par de líneas intersectan en exáctamente un punto.
- $P3$: Existen al menos tres puntos no colineales.
- $P4$: Existen al menos tres puntos en cada línea.

Se dice que el plano proyectivo es de orden n cuando las líneas tienen $n+1$ puntos.

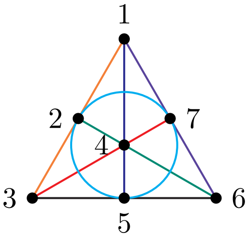


Figura 1: Plano proyectivo de orden 2

Definición plano proyectivo finito

Un plano proyectivo es un conjunto de puntos P y un conjunto de subconjuntos de P llamados líneas, que cumplen lo siguiene:

- $P1$: Existe exáctamente una línea que pasa por un par de puntos distintos.
- $P2$: Cada par de líneas intersectan en exáctamente un punto.
- $P3$: Existen al menos tres puntos no colineales.
- $P4$: Existen al menos tres puntos en cada línea.

Se dice que el plano proyectivo es de orden n cuando las líneas tienen $n+1$ puntos.

Equivalencia plano proyectivo y juego *Dobble* completo

- La propiedad $P1$ es equivalente a la propiedad $D4$.
- La propiedad $P2$ es equivalente a la propiedad $D1$.
- La propiedad $P3$ es equivalente a la propiedad $D2$ cuando el orden del juego *Dobble* es $n \geq 3$
- La propiedad $P4$ es equivalente a la propiedad $D2$ cuando el orden del juego *Dobble* es $n \geq 3$
- Por lo tanto existe un juego *Dobble* completo de orden n si y solo si existe un plano proyectivo finito de orden $n-1$ (por ejemplo, el juego *Dobble* de orden 3 es equivalente al plano proyectivo de orden 2)

Equivalencia plano proyectivo y juego *Dobble* completo

Algunas propiedades de los planos proyectivos finitos que nos serán útiles son las siguientes:

- En general no existen planos proyectivos finitos para cualquier n.
- Pero existen para todo n primo. Eso implica que un juego *Dobble* de orden n completo existe cuando $n-1$ es primo.
- Un plano proyectivo finito de orden n tiene n^2+n+1 puntos y n^2+n+1 líneas. Eso implica que un juego *Dobble* de orden n completo tiene $(n-1)^2+(n-1)+1 = n^2-n+1$ símbolos y cartas.

Algoritmo para construir un juego *Dobble* completo

- Primero consideremos un ejemplo en el caso $n = 4$.
- $n - 1 = 3$ es primo por lo tanto existe el juego *Dobble* completo.
- Los renglones representan las cartas y las columnas los símbolos.

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

- A las primeras 4 cartas le agregamos el símbolo 1, y despues los símbolos $2, 3, 4$; $5, 6, 7$; $8, 9, 10$; y $11, 12, 13$; a cada carta, respectivamente
- Despues agregamos a 3 cartas el símbolo 2, a otras 3 el símbolo 3 y por úlitmo el símbolo 4 a otras 3 cartas.
- Debemos de tener $4^2 - 4 + 1 = 13$ cartas, y tenemos $4 + 3((4 - 2) + 1) = 13$ cartas.
- Lo que nos falta es solo llenar las cartas que están incompletas.

Algoritmo para construir un juego *Dobble* completo

- Esta tabla está dividida en varias secciones.

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

- La parte naranja es llamada renglón.
- Es decir, un renglón es un conjunto de $n-1$ cartas tal que su primer símbolo en la tabla es el mismo.

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

- La parte celeste es llamada una columna.

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

- La intersección de un renglón y una columna es llamada una celda (la parte color verde)

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

- A la parte que llevamos construida del juego le llamaremos la parte fija. (color blanco)
- A la parte que nos falta llenar la llamamos no fija. (color gris)

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5								
6		2				6							
7		2					7						
8			3		5								
9			3			6							
10			3				7						
11				4	5								
12				4		6							
13				4			7						

Algoritmo para construir un juego *Dobble* completo

Desde ahora cuando mencionemos a los renglones y columnas solo nos referiremos a la parte de los renglones y columnas que están en la parte no fija.

Podemos notar lo siguiente en la tabla.

- En general, hay $(n-1)(n-2+1) = n^2 - 2n - 1$ renglones y $(n-1)(n-2) = n^2 - 3n + 2$ columnas en la parte no fija de un juego *Dobble* de orden n.

Algoritmo para construir un juego *Dobble* completo

Desde ahora cuando mencionemos a los renglones y columnas solo nos referiremos a la parte de los renglones y columnas que están en la parte no fija.

Podemos notar lo siguiente en la tabla.

- En general, hay $(n-1)(n-2+1) = n^2 - 2n - 1$ renglones y $(n-1)(n-2) = n^2 - 3n + 2$ columnas en la parte no fija de un juego *Dobble* de orden n.
- Cada carta puede tener a lo mucho un simbolo de cada columna (y por celda), ya que si tuviera dos o mas tendria dos simbolos en comun con una de las cartas que tiene el símbolo 1 al principio.

Algoritmo para construir un juego *Dobble* completo

Desde ahora cuando mencionemos a los renglones y columnas solo nos referiremos a la parte de los renglones y columnas que están en la parte no fija.

Podemos notar lo siguiente en la tabla.

- En general, hay $(n-1)(n-2+1) = n^2 - 2n - 1$ renglones y $(n-1)(n-2) = n^2 - 3n + 2$ columnas en la parte no fija de un juego *Dobble* de orden n.
- Cada carta puede tener a lo mucho un simbolo de cada columna (y por celda), ya que si tuviera dos o mas tendria dos simbolos en comun con una de las cartas que tiene el símbolo 1 al principio.
- Cada carta tiene que tener un simbolo de cada columna porque a la carta le faltan $n-2$ simbolos, solo hay $n-2$ columnas y por lo de arriba una carta tiene a lo mucho un simbolo por columna

Algoritmo para construir un juego *Dobble* completo

- La $i-$*ésima* carta de un renglón no puede tener un simbolo en comun con otra *i-ésima* carta de otro renglón diferente

Algoritmo para construir un juego *Dobble* completo

- La $i-\acute{e}sima$ carta de un renglón no puede tener un simbolo en comun con otra *i-ésima* carta de otro renglón diferente
- Entonces el simbolo de la carta *i-ésima* de una celda no puede coincidir con el simbolo de la carta *i-ésima* de otra celda si las dos celdas estan en una misma columna

Algoritmo para construir un juego *Dobble* completo

- La $i-ésima$ carta de un renglón no puede tener un simbolo en comun con otra *i-ésima* carta de otro renglón diferente
- Entonces el simbolo de la carta *i-ésima* de una celda no puede coincidir con el simbolo de la carta *i-ésima* de otra celda si las dos celdas estan en una misma columna
- Por lo tanto, como cada carta tiene un simbolo por columna, solo hay $n-2$ columnas y $n-2$ cartas por renglón sin contar la carta *i-ésima*, entonces la carta *i-ésima* de un renglón coincide con otra carta *j-ésima* de otro renglón, $i \neq j$.

Algoritmo para construir un juego *Dobble* completo

- Con estas observaciones podemos representar a una columna (la parte no fija) como una matríz de $n-1 \times n-1$.

n=3	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	4									
2	1				5	6	7						
3	1							8	9	10			
4	1										11	12	13
5		2			5			8			11		
6		2				6			9			12	
7		2					7			10			13
8			3		5				9				13
9			3			6				10	11		
10			3				7	8				12	
11				4	5					10		12	
12				4		6		8					13
13				4			7		9		11		

Algoritmo para construir un juego *Dobble* completo

- Podemos suponer que los símbolos de la columna a la que hace referencia la matriz son $a_1, a_2, \ldots, a_{n-1}$
- Tenemos que asignar exáctamente un símbolo de la columna a cada carta.
- El renglón *i-ésimo* de la matríz corresponde a la celda *i-ésima* de la columna correspondiente a la matriz (naranja y azúl).

n=3	...	...	a_1	a_2	a_3	b_1	b_2	b_3
1	...	...	...			...		
5	...	...	a_1			b_1		
6				a_2			b_2	
7					a_3			b_3
8	...	...		a_2				b_3
9					a_3	b_1		
10			a_1				b_2	
11	...	...			a_3		b_2	
12			a_1					b_3
13				b_2		b_1		

1	2	3
2	3	1
3	1	2

1	2	3
3	1	2
2	3	1

Algoritmo para construir un juego *Dobble* completo

- La columna *i-ésima* de la matríz corresponde a las cartas *i-ésimas* de las celdas correspondientes a la columna (verde)
- Una entrada de la matríz (i, j) corresponde al símbolo de la celda *i-ésima* y la carta *j-ésima* de esa celda, esa celda estando en la columna correspondiente a la matríz.

n=3	…	…	a_1	a_2	a_3	b_1	b_2	b_3
1	…	…		…			…	
5	…	…	a_1			b_1		
6				a_2			b_2	
7					a_3			b_3
8	…	…		a_2				b_3
9					a_3	b_1		
10			a_1				b_2	
11	…	…			a_3		b_2	
12			a_1					b_3
13				b_2		b_1		

1	2	3
2	3	1
3	1	2

1	2	3
3	1	2
2	3	1

Algoritmo para construir un juego *Dobble* completo

- Los renglones de la matríz no pueden tener símbolos repetidos ya que las cartas no pueden tener símbolos repetidos de la misma celda.
- Las columnas de la matríz no pueden tener símbolos repetidos, ya dos *i-ésimas* cartas de dos celdas diferentes no pueden compartir otro símbolo en común (ya comparten uno de la parte fija de la tabla).

n=3	...	...	a_1	a_2	a_3	b_1	b_2	b_3
1	...	...		...			...	
5	...	...	a_1			b_1		
6				a_2			b_2	
7					a_3			b_3
8	...	...		a_2				b_3
9					a_3	b_1		
10			a_1				b_2	
11	...	...			a_3		b_2	
12			a_1					b_3
13				b_2		b_1		

1	2	3
2	3	1
3	1	2

1	2	3
3	1	2
2	3	1

Algoritmo para construir un juego *Dobble* completo

- Ahora veremos que las $n-2$ matrices que se forman con las $n-2$ columnas tienen que cumplir con que, al concatenar cada par de matrices entrada por entrada, se tienen que todos los elementos sean diferentes (todas las matrices tienen que usar el mismo conjunto de valores).
- A un conjunto de matrices que cumplen las propiedades antes mencionadas se les conoce como cuadrados latinos mutualmente ortogonales.

n=3	$\dots$	$\dots$	a_1	a_2	a_3	b_1	b_2	b_3
1	$\dots$	$\dots$		$\dots$			$\dots$	
5	$\dots$	$\dots$	a_1			b_1		
6				a_2			b_2	
7					a_3			b_3
8	$\dots$	$\dots$		a_2				b_3
9					a_3	b_1		
10			a_1				b_2	
11	$\dots$	$\dots$			a_3		b_2	
12			a_1					b_3
13				b_2		b_1		

1	2	3
2	3	1
3	1	2

1	2	3
3	1	2
2	3	1

11	22	33
23	31	12
32	13	21

Algoritmo para construir un juego *Dobble* completo

- Queremos probar que con un conjunto de $n-2$ cuadrados latinos ortogonales podemos agregar los símbolos faltantes a las cartas. (cuadrados ortogonales $\implies$ juego válido de *Dobble*)
- Probaremos la contrapositiva, tener entradas no válidas en las columnas no fijas (las columnas a las que corresponden las matrices), implica que esas matrices no forman un conjunto de cuadrados latinos ortogonales ($\neg$ juego válido $\implies$ $\neg$ cuadrados ortogonales).

Algoritmo para construir un juego *Dobble* completo

- Para que un conjunto de cartas sea un juego *Dobble* tiene que cumplir las propiedades $(D1)$ y $(D2)$.
- La propiedad $D2$ (no hay un símbolo en común para todas las cartas) se cumple por construcción de la tabla.
- Así que solo tendremos que tener en cuenta que pasa cuando no se cumple con la propiedad $(D1)$ (Solo un símbolo en común por par de cartas.)

Algoritmo para construir un juego *Dobble* completo

- Supongamos que hay dos cartas con más de un símbolo en común. entonces existen los siguientes casos.
- Las dos cartas están en el mismo renglón de la tabla.
- En este caso ya comparten un símbolo un símbolo en la parte fija de la tabla.
- Así que solo comparten al menos un símbolo en la parte no fija.
- Entonces tendríamos un cuadrado representando a la columna en que coincide un símbolo a de esta forma.
- Su matriz asociada no forma un cuadrado latino.

$\dots$	a	$\dots$
$\dots$	$\dots$	$\dots$
$\dots$	a	$\dots$

a	$\dots$	a
$\dots$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$

Algoritmo para construir un juego *Dobble* completo

- Caso 2: Las dos cartas están en renglones diferentes.
- Tenemos dos subcasos, el primero es cuando las dos cartas $C1$ y $C2$ están en los renglones *i-ésimo* y *j-ésimo*, respectivamente, con $i \neq j$. El segundo es cuando $i = j$.
- Cuando $i \neq j$, se necesita que coincidan al menos dos símbolos en la parte no fija, entonces la matríz que los representa es de la siguiente forma.
- Estos dos matrices no son mutualmente ortogonales.

$$
\begin{array}{c|c|c|c}
 & & & \\
\hline
 & \begin{array}{c}\cdots\\ \mathrm{a}\\ \cdots\end{array} & \begin{array}{c}\cdots\\ \mathrm{b}\\ \cdots\end{array} & \\
\hline
 & \begin{array}{c}\cdots\\ \cdots\\ \mathrm{a}\end{array} & \begin{array}{c}\cdots\\ \cdots\\ \mathrm{b}\end{array} & \\
\hline
 & & &
\end{array}
$$

$$
\begin{array}{c}
\begin{array}{|c|c|c|}
\hline
\cdots & \cdots & \cdots \\ \hline
\cdots & \mathrm{a} & \cdots \\ \hline
\cdots & \cdots & \mathrm{a} \\ \hline
\end{array}
\\[1em]
\begin{array}{|c|c|c|}
\hline
\cdots & \cdots & \cdots \\ \hline
\cdots & \mathrm{b} & \cdots \\ \hline
\cdots & \cdots & \mathrm{b} \\ \hline
\end{array}
\end{array}
=
\begin{array}{|c|c|c|}
\hline
\cdots & \cdots & \cdots \\ \hline
\cdots & \mathrm{ab} & \cdots \\ \hline
\cdots & \cdots & \mathrm{ab} \\ \hline
\end{array}
$$

Algoritmo para construir un juego *Dobble* completo

- En el caso que $i = j$, entonces esas dos cartas ya tienen un símbolo en común en la parte fija, así que solo tienen que coincidir una ves mas, quedando de esta forma.
- Su matriz asociada no forma un cuadrado latino.

...	a	...
...	...	...
...	...	...
	a	
...	...	...
..	..	..

...	a	...
...	a	...
...	...	...

Algoritmo para construir un juego *Dobble* completo

- Por lo tanto un conjunto de cuadrados latinos ortogonales llena la parte faltante de la tabla, usando la representación de matríz de las columnas.
- Tambien el juego generado es un juego completo, ya que tiene $n^2 - n + 1$ cartas y símbolos.

Algoritmo para construir un juego *Dobble* completo

- Solo nos falta saber una manera de generar los cuadrados latinos ortogonales.
- Existe una formula para generarlos, es la siguiente.
- Si n es un número primo, entonces para $i, j \in \{1, 2, \ldots, n\}$, $k \in \{1, 2, \ldots, n-1\}$

$$A_k(i, j) = [k \cdot (i-1) + (j-1)] \bmod n$$

Siendo A_k una matríz de $n \times n$

Algoritmo para construir un juego *Dobble* completo

- Recapitulando, el algoritmo para generar el juego *Dobble* n completo funciona de la siguiente manera.
- Comprobamos que $n-1$ sea un número primo, si no lo es no es posible generar el juego con el algoritmo.
- Generamos la parte fija de la tabla.
- Generamos los $n-2$ cuadrados latinos ortogonales de $n-1 \times n-1$.
- Convertimos cada cuadrado latino en una columna de la tabla.

Costo del algoritmo

- La salida del algoritmo es de tamaño $(n^2 - n + 1)n$ símbolos en total $(n^2 - n + 1)$ cartas con n símbolos cada una.
- Así que el costo tiene que ser de al menos $\Omega(n^3)$
- Nótese que en la parte fija hacemos un número constante de operación por cada símbolo que agregamos (la estructura es fija)
- También podemos usar la fórmula de los cuadrados latinos para asignar directamente a cada carta el símbolo correspondiente por renglón, así que tambien hacemos una cantidad constante de operaciones por símbolo.
- Así que el costo del algoritmo es $\Theta(n^3)$