PRÁCTICA Nº6

Conjuntos y Aplicaciones.

1.- INTRODUCCIÓN.

La teoría de conjuntos es muy amplia, sin embargo y debido a lo abstracta que es, resulta bastante complicado trasladarla al ordenador. En esta práctica resolveremos algunos de los pocos problemas que se pueden trasladar al ordenador.

2.- UNIÓN, INTERSECCIÓN Y COMPLEMENTARIO.

Consideremos tres conjuntos, A={1,2,3,4,5,6,7,8,9}, B={a,b,c,d,f,g,h,j,k,} y Z={a,b,h,m,z,3,6,8,0}. Como ya sabemos en Mathemetica se introducen así:

A={1,2,3,4,5,6,7,8,9}
B={a,b,c,d,f,g,h,j,k,}
Z={a,b,h,m,z,3,6,8,0}

Para calcular la unión utilizaremos la función **Union[list1,list2,...]**, entonces podemos calcular A∪B y A∪B∪Z, como sigue:

Union[A,B]
Union[A,B,Z]

Y el programa nos responderá:

{1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,j,k}
y
{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,g,h,i,j,k,m,z}

Para calcular la intersección, utilizaremos la función **Intersection[list1,list2,...]**, entonces podemos calcular A∩V y A∩B∩Z como sigue:

Intersection[A,Z]
Itersection[A,B,Z]

Y el programa responderá:

{3,6,8}
y
{}

es evidente que {} es el conjunto vacío.

Para calcular el complementario de un conjunto en un conjunto X, utilizaremos la función **Complement[Conjunto X, list1, list2,...]** que calcula el conjunto complementario en X de la unión de las lista 1, 2,.... Entonces si X=A∪B∪Z, Podemos calcular el complementario de Z y A∪B como sigue:

X=Union[A,B,Z];
Complement[X,Z]
Complement[X,A,B]

Y el programa responderá:

{1,2,4,5,7,9,c,d,e,f,g,i,j,k}
y
{0,m,z}

Ejercicio 1:

Considerando X, A, B y Z, los conjuntos anteriores, comprobar que el complementario de Z, el complementario de A∪B y el conjunto (A∪B)∩Z forman una partición de X.

Solución:

3.- PRODUCTO CARTESIANO

Recordemos que el producto cartesiano de dos conjunto A y B es AxB={(a,b) | a∈ A y b∈ B}. El producto cartesiano es fácil de calcular utilizando una sencilla rutina que recorra todos los posibles pares ordenados, podemos ver como funciona en el siguiente ejemplo.

Ejemplo 1:

Calcular el producto cartesiano de los conjuntos: A = {a, b, c, d, e, f, g, h} y B = {1, 2, 3, 4, 5, 6, 7, 8, 9}.

SOLUCIÓN:

PROGRAMA	COMENTARIOS	
A = {a, b, c, d, e, f, g, h}; **B = {1, 2, 3, 4, 5, 6, 7, 8, 9};**	Introducimos los conjuntos.	
cadena = "";		
For[j = 1, j <= Length[A], j++,		
For[k = 1, k <= Length[B], k++, **Cadena1 = StringJoin["(", ToString[A[[j]]], ",", ToString[B[[k]]], ")"];** **If[j≠1|| k≠1, cadena = StringJoin[cadena, “,”]];** **Cadena = StringJoin[cadena, cadena1]** **]**	Bucle que recorre la segunda componente del par ordenado.	Bucle que recorre la primera componente de par ordenado.
];		
Print["A = ", A] **Print["B = ", B]** **Cartesiano=ToExpression[StringJoin[“{“,cadena,”}”]]**	Salida de resultados.	

Obsérvese que para calcular el producto cartesiano de tres conjuntos podríamos escribir de nuevo el programa anterior pero tendríamos que incluir 3 bucles anidados o bien utilizar la asociatividad y calcular primero AxB y luego (AxB)xC, aunque en este último caso la presentación de los elementos no es por ternas, el elemento (a,b,c) será representado por ((a,b),c).

Ejercicio 2:

Si A={a,b,c,56} y B={1,2,3,4,5,6,7,8,9}, calcular AxB y AxAxB.

Solución:

4.- PARTES DE UN CONJUNTO.

Para calcular el conjunto de las partes de un conjunto A vamos a utilizar dos bucles anidados, uno recorrerá los elementos de A y otro recorrerá los elementos que el conjunto partes de A tiene en cada momento, añadiendo en cada ciclo el elemento de A al que nos estemos refiriendo por el bucle 1 al elemento de P(A) al que nos estemos refiriendo por el segundo bucle. Veamos con un poco más de detalle como funciona:

Supongamos que A = {a, b, c}, en un principio cogemos P(A) = {∅} y se le añaden el resto de elementos que le faltan como sigue; supongamos que el contador del bucle 1 es j, entonces tendremos 3 ciclos:

- j = 1, cogemos el elemento a∈ A, se añade ∅∪{a}={a} a P(A), por tanto:
  $$P(A) = \{\emptyset, \{a\}\}.$$
- j = 2, cogemos el elemento b∈ A, se añade ∅∪{b}={b} y {a}∪{b}={a,b} a P(A), por tanto:
  $$P(A) = \{\emptyset, \{a\}, \{b\}, \{a,b\}\}$$
- j = 3, cogemos el elemento c∈ A, se añade ∅∪{c}={c}, {a}∪{c}={a,c}, {b}∪{c}={b,c}, {a,b}∪{c}={a,b,c} a P(A), por tanto:
  $$P(A) = \{\emptyset, \{a\}, \{b\}, \{a,b\}, \{c\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$$

Ejemplo 2:

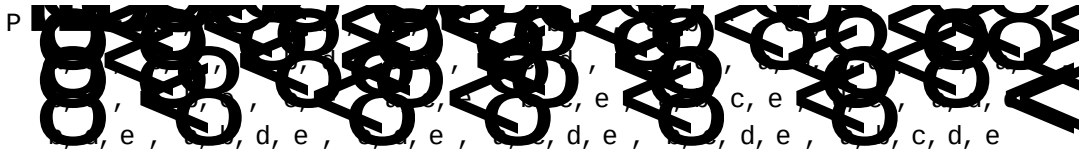
Calcula las partes del conjunto: A = {a, b, c, d, e}

SOLUCIÓN:

PROGRAMA	COMENTARIOS	
A = {a, b, c, d, e,};	Introducimos el conjunto.	
Partes = {{}};	Definimos el conjunto partes de A en principio sólo con el conjunto vacío.	
For[j = 1, j <= Length[A], j++,		
temp = Length[Partes];		
For[i = 1, i <= temp, i++,	Bucle que recorre todos los elementos de Parte de A.	Bucle que recorre todos los elementos de A.
Partes = Append[Partes, Append[Partes[[i]], A[[j]]]];		
];		
];		
Print["A=", A]	Salida de resultados.	
Print["P(A)=", Partes]		

Y el programa devolverá:

A= {[illegible], c, d, e



También podemos utilizar las funciones de Mathemática que se encuentran en un paquete de Matemática discreta, para usarlo lo primero que debemos de hacer es cargar dicho paquete de funciones, para ello tendremos que escribir:

```
<< DiscreteMath`Combinatorica`
```

Las funciones nuevas que vamos a estudiar son. KSubsets[], NthSubsets[], GrayCode[] y Subsets[].

La función **KSubsets[conjunto, n]** calcula todos los subconjuntos con n elementos. Por ejemplo los subconjuntos de A={1,2,3,4} con 2 elementos se calcularía así:

KSubsets[{1,2,3,4},2]

Y podemos calcular las partes del conjunto X={1,2,3,4,5,6,7} con el programa:

PROGRAMA	COMENTARIOS
X = {1, 2, 3, 4, 5, 6, 7};	Introducimos el conjunto.
Partes = {};	
For[j = 0, j <= Length[X], j++, Partes = Union[Partes, KSubsets[X, j]]];	Bucle que va desde 0 hasta el número de elementos de X, y calcula en cada caso todos los subconjuntos de X.
Partes	Salida de resultados.

La función **NthSubset[n,Conjunto]** calcula l subconjunto n-ésimo de nuestro conjunto, empezando por el vacío que es n=0, y terminando por el propio conjunto. Por ejemplo podemos calcular el conjunto nº 12 del ejemplo anterior:

NthSubset[12,{1,2,3,4,5,6,7}]

Utilizando esta función, también podemos calcular las partes de un conjunto, lo único que necesitamos saber cuantos elementos tiene las partes de un conjunto. Como para un conjunto A con k elementos, el número de elementos de partes de A es 2^k, entonces podemos calcular las partes como sigue:

A = {1, 2, 3, 4}
Table[NthSubset[n, A], {n, 0, (2^Length[A]) - 1}]

La función **GrayCode[Conjunto]** calcula directamente las partes de un conjunto.

GrayCode[A]

Nótese que los tres métodos que hemos visto calculan las partes de un conjunto de forma distinta y además el orden en el que aparecen los conjuntos es distinto para cada caso, utilizando Ksubset, aparecen ordenados por el número de elementos; utilizando NthSubset, nos aparece el mismo orden que con la rutina que se diseñó en el ejemplo, y el último lo hace de forma que cada conjunto difiere de su posterior en un elemento.

En el paquete de Matemática Discreta, disponemos de otra función que resulta interesante: **Subsets[n]**. Esta función calcula las partes del conjunto {1,2,3,...,n}.

Ejercicio 3:

Calcular por todos los métodos que se han visto las partes de A={a,b,c,d}, B={1,2,3} y $A \cup B$.

Solución:

5.- APLICACIONES.

Teniendo en cuenta que los únicos conjuntos que estamos tratando son finitos, entonces la definición de aplicaciones es bastante sencilla.

Ejemplo 3:

si queremos definir una aplicación entre los conjunto A={a,b,c,d} y B={1,2,3,4} definida por:

$$f: A \to B$$
$$f(a) = 1,\ f(b) = 3,$$
$$f(c) = 3,\ f(d) = 2.$$

Lo haremos así:

```
A = {a, b, c, d};
B = {1, 2, 3, 4};
f[a] = 1;
f[d] = 2;
f[b] = 3;
f[c] = 3;
```

También será bastante sencillo programar unas rutinas que nos determinen cuando una aplicación es inyectiva, sobreyectiva o biyectiva:

PROGRAMA	COMENTARIOS
A = {a, b, c, d}; **B = {1, 2, 3, 4};**	Introducimos los conjuntos dominio y codominio de la aplicación.
f[a] = 1; **f[d] = 2;** **f[b] = 3;** **f[c] = 3;**	Definimos la aplicación.
Imagen = {};	Suponemos que la imagen de la aplicación en un principio es el conjunto vacío.
Do[Imagen = Union[Imagen, Append[{}, f[A[[i]]]], {i, 1, Length[A]}];	Calculamos el conjunto imagen.
Print["El conjunto imagen es: ", Imagen];	Mostramos el conjunto imagen.
If[Intersection[B, Imagen] == B, Print["Es sobreyectiva"], **Print["No es sobreyectiva"]];**	Comprobamos si es sobreyectiva.
If[Length[A] == Length[Imagen], Print["Es inyectiva"], **Print["No es inyectiva"]];**	Comprobamos si es inyectiva.
If [Length[A] == Length[B] && Length[B] == Length[Imagen], **Print["Es biyectiva"], Print["No es biyectiva"]];**	Comprobamos si es biyectiva.

Ejercicio 4:

Dados los conjuntos A={a,b,c,d} y B={1,2,3}. Define una aplicación entre A y P(B) que sea inyectiva y otra entre P(B) y A que sea sobreyectiva.

Solución:

Obsérvese que se podría usar el grafo de la aplicación para definirla, por ejemplo la aplicación f: A $\rightarrow$ B del ejemplo 3 tendría por grafo, G_f = {(a,1), (b,3), (c,3), (d,2)} que introduciríamos en el ordenador de la siguiente manera:

G = {{a,1},{b,3},{c,3},{d,3}}

Y ahora podríamos realizar un programa de forma análoga al anterior que determinase si f es inyectiva, sobreyectiva o biyectiva. Obsérvese que dado un grafo cualquiera, éste puede estar asociado a una correspondencia que no sea necesariamente una aplicación, resultaría también en consecuencia interesante realizar un programa que determinase si el grafo G está asociado a una aplicación.

Ejercicio 5:

Dados dos conjuntos A = {1,2,3,4,5} y B = {a,b,c,d,e}, sea G = {(1,a), (3,c), (5,e)} $\subseteq$ AxB un grafo, realizar un programa que determine si (A,B;G) es una aplicación.

Solución:

6.- RELACIONES BINARIAS.

Una relación binaria R sobre un conjunto A, es un subconjunto de AxA, $R \subseteq AxA$. Los elementos de AxA son de la forma (x,y) donde x, $y \in A$. Nosotros representaremos los pares ordenados en el programa de la forma {x,y}. Por tanto si A={a,b,c} y consideramos la relación binaria R={(a,a),(b,b),(a,b),(b,c)}, los introduciremos en el ordenador así:

```
A={a,b,c};
R={{a,a},{b,b},{a,b},{c,b}};
```

Recordemos que una relación binaria podía verificar 4 propiedades: reflexiva, simétrica, transitiva y antisimétrica, en el siguiente ejemplo se puede ver como se comprueba si una relación binaria verifica alguna de estas propiedades:

Ejemplo 3:

Sean A={a,b,c,d,e,f,g,h} y R = {(a,a), (b,b), (c,c), (d,d), (e,e), (f,f), (g,g), (h,h), (a,b) ,(b,a)}, determinar si R es reflexiva, simétrica, transitiva o antisimétrica, y comprobar si R es relación de orden o relación de equivalencia.

SOLUCIÓN:

PROGRAMA	COMENTARIOS
A = {a, b, c, d, e, f, g, h};	Conjunto
R = {{a, a}, {b, b}, {c, c}, {d, d}, {e, e}, {f, f}, {g, g}, {h, h}, {a, b}, {b, a}};	Relación binaria.
Reflexiva = True; **For[n = 1, n <= Length[A], n++,** **If[Intersection[{{A[[n]], A[[n]]}}, R] == {{A[[n]], A[[n]]}}, Null,** **Reflexiva = False]];**	Comprobamos si es reflexiva.
Simetrica = True; **For[m = 1, m <= Length[R], m++,** **If[Intersection[{{R[[m, 2]], R[[m, 1]]}}, R] == {{R[[m, 2]], R[[m, 1]]}},** **Null, Simetrica = False]** **];**	Comprobamos si es simétrica
Transitiva = True; **For[p = 1, p <= Length[R], p++,** **For[q = 1, q <= Length[R], q++,** **If[R[[p, 1]] == R[[q, 2]],** **If[Intersection[{{R[[q, 1]], R[[p, 2]]}}, R] == {{R[[q, 1]], R[[p, 2]]}},** **Null, Transitiva = False]** **];** **];** **];**	Comprobamos si es transitiva
Antisimetrica = True; **For[r = 1, r <= Length[R], r++,** **If[Intersection[{{R[[r, 2]], R[[r, 1]]}}, R] == {{R[[r, 2]], R[[r, 1]]}} && !** **(ToString[R[[r, 1]]] == ToString[R[[r, 2]]]), Antisimetrica = False]** **];**	Comprobamos si es antisimétrica

If[Reflexiva, Print["R es reflexiva"], Print["R no es reflexiva"]] **If[Simetrica, Print["R es simétrica"], Print["R no es simétrica"]]** **If[Transitiva, Print["R es Transitiva"], Print["R no es Transitiva"]]** **If[Antisimetrica, Print["R es antisimétrica"], Print["R no es antisimétrica"]];** **If[Reflexiva && Simetrica && Transitiva, Print["R es una relación de equivalencia"], Print["R no es relación de equivalencia"]];** **If[Reflexiva && Antisimetrica && Transitiva, Print["R es una relación de orden"], Print["R no es relación de orden"]]**	Salida de resultados.

Obsérvese que utilizando el ejemplo anterior podemos comprobar si una relación binaria es de orden o de equivalencia.

Ejercicio 5:

Dada la relación binaria R={(a,a),(b,b),(c,c),(a,b),(c,b)}, comprobar que propiedades verifica.

Solución:

7.- NOTACIÓN EN CONJUNTOS.

Estos son algunos de los símbolos que incluye el Mathematica y que se usan habitualmente en conjuntos.

SIMBOLO	
\[EmptySet]	∅
\[Subset]	⊂
\[SubsetEqual]	⊆
\[Superset]	⊃
\[SupersetEqual]	⊇
\[NotSubset]	⊄
\[Intersection]	∩
\[Union]	∪
\[Element]	∈
\[NotElement]	∉