Universidad Rey Juan Carlos

Matemática Discreta y Algebra
Curso 2008-09

PRÁCTICA 3

COMBINATORIA

1. Introducción y Objetivos

En esta práctica se aborda el uso de Maple para resolver problemas de **combinatoria**.

El objetivo es que tras esta práctica el alumno conozca las instrucciones básicas de Maple útiles en **combinatoria** y sea capaz de calcular el numero de variaciones, permutaciones y combinaciones con esta herramienta informática.

Instrucciones

1. Lee con atención el texto. Cuando aparezcan comandos o procedimientos de Maple (líneas en rojo), intenta averiguar para qué sirven (puedes emplear la ayuda de Maple) y después ejecútalos. Puedes ejecutarlos varias veces y cambiar en ellos lo que te parezca para entenderlos mejor o mejorarlos.
2. Las secciones 2,3,4,5 contienen la información básica de la práctica, mientras que en la sección 6 se aborda, de manera opcional, el estudio de otras funciones combinatorias incluidas en los paquetes de instrucciones de Maple.
3. Al final de la práctica se incuye una sección con varios ejercicios tipo test sobre conceptos explicados a lo largo de este documento yq ue deberían ser resueltos por el alumno/a sin problema, sise ha comprendido el contenido de la práctica.

Maple dispone de un conjunto de funciones específicas para combinatoria. En Maple, hay varios conjuntos de funciones específicas (denominados *paquetes*) que deben cargarse aparte usando el comando **`with()`**. En este caso las funciones propias de teoría de combinatoria estan guardadas en el paquete combinat de Maple. Para cargar el paquete procedemos como siempre con la instrucción:

```
> with(combinat);
Warning, the protected name Chi has been redefined and unprotected

[Chi, bell, binomial, cartprod, character, choose, composition, conjpart, decodepart,
```

encodepart, *fibonacci*, *firstpart*, *graycode*, *inttovec*, *lastpart*, *multinomial*, *nextpart*, *numbcomb*, *numbcomp*, *numbpart*, *numbperm*, *partition*, *permute*, *powerset*, *prevpart*, *randcomb*, *randpart*, *randperm*, *setpartition*, *stirling1*, *stirling2*, *subsets*, *vectoint*]

2. Cardinales de conjuntos

Comenzamos con las funciones que presenta Maple para trabajar con conjuntos finitos y calcular sus cardinales.
Recordamos que los conjuntos se definían con llaves.

```
> A:={1,2,3,4};
```

$$A := \{1, 2, 3, 4\}$$

Y que la operación **nops()** calculaba el número de elementos, es decir, el cardinal del conjunto A:

```
> nops(A);
```

$$4$$

```
> C:={seq(n^2,n=-10..10)}; nops(C);
```

$$C := \{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$$

$$11$$

Las operaciones habituales de conjuntos, como son la unión, la intersección y la diferencia de conjuntos se pueden efectuar con Maple, con lar ordenes **union, intersection** y **minus** .respectivamente.

```
> B:={1,a,b,c};
```

$$B := \{1, a, b, c\}$$

```
> A union B;
```

$$\{1, 2, 3, 4, a, b, c\}$$

```
> A intersect B;
```

$$\{1\}$$

```
> A minus B;
```

$$\{2, 3, 4\}$$

Podemos hacer un procedimiento simple para construir el conjunto producto cartesiano:

```
> cartesiano:=proc(A::set,B::set)
  local i,j,C:
  C:={}:
  for i from 1 to nops(A) do
     for j from 1 to nops(B) do
     C:=C union {[[op(A)][i],[op(B)][j]]}
    od:
  od: C; end:
```

Veamos unos ejemplos:

```
> cartesiano({1,2},{1,m});
```

$$\{[1, l], [1, m], [2, l], [2, m]\}$$

```
> cartesiano({maria,juana,luisa},{pedro,pepe,juan});
```

$$\{[luisa, pepe], [luisa, juan], [maria, pedro], [maria, pepe], [maria, juan], [juana, pedro], [juana, pepe], [juana, juan], [luisa, pedro]\}$$

```
> 
```

Aplicamos la teoría de conjuntos para responder la siguiente cuestión:

Ejemplo: ¿cuántos números enteros positivos menores o iguales que 1000 son a la vez el cuadrado de un número entero y múltiplos de 4?.

Definimos el conjunto de los cuadrados menores que 1000.

```
> evalf(sqrt(1000));
```

$$31.62277660$$

```
> 32**2;
```

$$1024$$

```
> F:={seq(n^2,n=1..31)};
```

$$F := \{ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676, 729, 784, 841, 900, 961 \}$$

Definimos ahora el conjunto de los múltiplos de 4 menores o iguales que 1000.

```
> G:={seq(4*n,n=1..250)}:
```

Por tanto debemos calcular el cardinal de la intersección.

```
> nops(F intersect G);
```

$$15$$

Los números enteros que cumplen ambas condiciones son exactamente:

```
> F intersect G;
```

$$\{ 4, 16, 36, 64, 100, 144, 196, 256, 324, 400, 484, 576, 676, 784, 900 \}$$

En efecto todos ellos son múltiplos de 4 y el cuadrado de un número entero.

```
> 
```

Ejercicios.

Después de haber leído detenidamente los contenidos de esta sección deberías ser capaz de responder a las siguientes preguntas:

(1) De los números naturales menores o iguales que 10.000 cuántos múltiplos de 5 no son múltiplos de 7.

(a) 1725
(b) 2175
(c) 1715

```
> 
> 
> 
```

(2) Determinar cuántos números naturales menores que 10000 son múltiplos de 7 pero no son pares ni múltiplos de 3:

(a) 477
(b) 476
(c) 478

```
> 
> 
```

```
>
```

(3) Calcular el número de enteros positivos (n>0) menores que 10.000 que son a la vez el cuadrado de un número entero y el cubo de un número entero.

(a) 4
(b) 7
(c) 17

```
>
>
>
```

3. Permutaciones

Comenzamos ahora a estudiar los comandos que tiene Maple para el cálculo de variaciones, permutaciones y combinaciones.
Aunque siguiendo el orden de los apuntes deberíamos empezar con las variaciones de *m* elementos tomadas de *n* en *n* vamos a comenzar sin embargo con las permutaciones. Maple no tiene implementada una función para calcular las variaciones, pero sí que la tiene para permutaciones y combinaciones, de modo que con ellas podremos definir las variaciones.
El número de permutaciones de m elementos se calcula con numbperm:

```
> numbperm(6);
```

$$720$$

También se puede usar el símbolo `!` o la expresión `factorial()` para calcular el factorial de un número:

```
> 6!; factorial(6);
```

$$720$$

$$720$$

La función se puede aplicar a un conjunto en lugar de a su cardinal:

```
> numbperm({a,b,c,d,e,f});
```

$$720$$

Para generar todas las permutaciones de un conjunto, esto es, las distintas maneras de ordenarlo, empleamos permute:

```
> permute({a,b,c});
```

$$[[a, b, c], [a, c, b], [b, a, c], [b, c, a], [c, a, b], [c, b, a]]$$

Observad que el resultado es una secuencia.

Y para generar aleatoriamente una permutación de los elementos de un conjunto usamos la función randperm:

```
> randperm({a,b,c,d,e,f});
```

$$[e, a, f, d, c, b]$$

Si queremos introducir **repeticiones**, entonces debemos dar los datos con una **secuencia**:

```
> permute([c,o,c,o,d,r,i,l,o]):
> numbperm([c,o,c,o,d,r,i,l,o]);
```

$$30240$$

La línea en que enumeran las permutaciones de la palabra cocodrilo la hemos terminado con dos puntos para no llenar pantallas con las 30240 permutaciones distintas de la palabra.

Y podemos verificar que es correcto según las fórmulas de clase:

```
> 9!/(2!*3!);
```

$$30240$$

También la función **multinomial()** permite calcular las permutaciones con repetición. De esta manera las permutaciones con repetición se pueden calcular por estos dos caminos:

```
> is(numbperm([c,o,c,o,d,r,i,l,o])=multinomial(9,2,3,1,1,1,1));
```

$$true$$

```
>
```

4. Combinaciones

Para calcular el **número de combinaciones** de *n* elementos (por ejemplo 2 elementos) escogidos entre los elementos de un conjunto (por ejemplo el conjunto *{María , Juan , Eva , Luis , Teresa, Pablo}*), esto es, sus subconjuntos de *n* elementos, empleamos la función numcomb,

```
> numbcomb({Maria, Juan, Eva, Luis, Teresa, Pablo},2);
```

$$15$$

Aunque también se puede hacer empleando los números combinatorios *m sobre n* que son los coeficientes en el binomio de Newton.

```
> binomial(6,2);
```

$$15$$

Para ver la fórmula con la que trabaja **binomial()** podemos usar de nuevo **convert()**.

```
> convert(binomial(m,n),factorial);
```

$$\frac{m!}{n!\,(m-n)!}$$

Si lo que queremos es obtener la **lista de todas las combinaciones** empleamos **choose()**:

```
> choose({Maria, Juan, Eva, Luis, Teresa, Pablo},2);
```

$$\{\{Pablo, Maria\}, \{Pablo, Luis\}, \{Pablo, Teresa\}, \{Teresa, Eva\}, \{Juan, Pablo\},$$
$$\{Juan, Maria\}, \{Juan, Luis\}, \{Juan, Teresa\}, \{Juan, Eva\}, \{Pablo, Eva\},$$
$$\{Maria, Luis\}, \{Maria, Teresa\}, \{Maria, Eva\}, \{Luis, Teresa\}, \{Luis, Eva\}\}$$

Obsérvese que el resultado es un conjunto.

Si a las funciones **numbcomb()** y **choose()** no se les da el segundo argumento, hacen el cálculo correspondiente a todos los valores desde cero hasta el número de elementos del conjunto.

```
> numbcomb({Maria, Juan, Eva, Luis, Teresa, Pablo});
```

$$64$$

Que es el cardinal del conjunto ***partes de*** *{María , Juan , Eva , Luis , Teresa, Pablo}* formado por todos sus subconjuntos.

```
> 2^6;
```

$$64$$

Este conjunto es:

```
> choose({Maria, Juan, Eva, Luis, Teresa, Pablo});
```

$$\{\{\}, \{Pablo, Maria\}, \{Pablo, Luis\}, \{Pablo, Teresa\}, \{Teresa, Eva\},$$
$$\{Pablo, Maria, Luis, Teresa, Eva\}, \{Juan, Pablo, Maria, Luis, Teresa, Eva\},$$
$$\{Juan, Pablo\}, \{Juan, Maria\}, \{Juan, Luis\}, \{Juan, Teresa\}, \{Juan, Eva\},$$

{ *Pablo*, *Eva* }, { *Maria*, *Luis* }, { *Maria*, *Teresa* }, { *Maria*, *Eva* }, { *Luis*, *Teresa* },
{ *Luis*, *Eva* }, { *Maria* }, { *Juan*, *Pablo*, *Maria* }, { *Luis* }, { *Juan*, *Pablo*, *Luis* },
{ *Juan*, *Maria*, *Luis* }, { *Pablo*, *Maria*, *Luis* }, { *Juan*, *Pablo*, *Maria*, *Luis* }, { *Teresa* },
{ *Juan*, *Maria*, *Eva* }, { *Pablo*, *Maria*, *Eva* }, { *Juan*, *Pablo*, *Maria*, *Eva* },
{ *Juan*, *Luis*, *Eva* }, { *Pablo*, *Luis*, *Eva* }, { *Juan*, *Pablo*, *Luis*, *Eva* }, { *Maria*, *Luis*, *Eva* },
{ *Juan*, *Maria*, *Luis*, *Eva* }, { *Pablo*, *Maria*, *Luis*, *Eva* }, { *Juan*, *Pablo*, *Maria*, *Luis*, *Eva* },
{ *Juan* }, { *Pablo* }, { *Maria*, *Luis*, *Teresa*, *Eva* }, { *Juan*, *Maria*, *Luis*, *Teresa*, *Eva* },
{ *Luis*, *Teresa*, *Eva* }, { *Juan*, *Luis*, *Teresa*, *Eva* }, { *Pablo*, *Luis*, *Teresa*, *Eva* },
{ *Juan*, *Pablo*, *Luis*, *Teresa*, *Eva* }, { *Juan*, *Teresa*, *Eva* }, { *Pablo*, *Teresa*, *Eva* },
{ *Juan*, *Pablo*, *Teresa*, *Eva* }, { *Maria*, *Teresa*, *Eva* }, { *Juan*, *Maria*, *Teresa*, *Eva* },
{ *Pablo*, *Maria*, *Teresa*, *Eva* }, { *Juan*, *Pablo*, *Maria*, *Teresa*, *Eva* }, { *Eva* },
{ *Juan*, *Pablo*, *Eva* }, { *Juan*, *Pablo*, *Teresa* }, { *Juan*, *Maria*, *Teresa* },
{ *Pablo*, *Maria*, *Teresa* }, { *Juan*, *Pablo*, *Maria*, *Teresa* }, { *Juan*, *Luis*, *Teresa* },
{ *Pablo*, *Luis*, *Teresa* }, { *Juan*, *Pablo*, *Luis*, *Teresa* }, { *Maria*, *Luis*, *Teresa* },
{ *Juan*, *Maria*, *Luis*, *Teresa* }, { *Pablo*, *Maria*, *Luis*, *Teresa* },
{ *Juan*, *Pablo*, *Maria*, *Luis*, *Teresa* } }

```
>
```

Podemos generar una combinación de forma aleatoria con randcomb

```
> randcomb({chocolate,vainilla,fresa,limon},2);
```

{ *vainilla*, *chocolate* }

```
> randcomb(5,3);
```

{ 1, 3, 4 }

```
>
```

En el último caso se eligen al azar tres números del conjunto *{1,2,3,4,5}*.

De nuevo, como en el caso de las permutaciones, si queremos que haya elementos repetidos los datos deben introducirse como una secuencia.

```
> choose([a,b,a],2);
```

[[*a*, *a*], [*a*, *b*]]

```
> numbcomb([a,b,a],2);
```

2

Observa que éstas no son las combinaciones con repetición de 2 elementos tomadas de dos en dos, que son por la fórmula vista en clase:

```
> binomial(2+2-1,2);
```

3

Ya que en las combinaciones elegidas arriba, falta la pareja *(b,b)*.

Se puede introducir una función en dos variables que permita calcular el número de **combinaciones con repetición** de *m* elementos tomadas de *n* en *n*.

```
> CR:=(m,n)->binomial(m+n-1,n);
```

$CR := (m, n) \rightarrow \text{binomial}(m + n - 1, n)$

```
> CR(2,2);
```

3

```
>
```

5. Variaciones

Vamos a construir una fórmula para las variaciones y para las variaciones con repetición.
Como las combinaciones de m elementos tomadas de n en n son el cociente de las variaciones de m elementos tomadas de n en n por las permutaciones de n, la fórmula oportuna será:

```
> V:=(m,n)->binomial(m,n)*n!;
```

$$V := (m, n) \to \text{binomial}(m, n)\, n!$$

```
> V(3,5);
```

$$0$$

```
> V(3,3);
```

$$6$$

O como un procedimiento:

```
> V2:=proc(m,n)
  local s,i:
  if n>m then s:=0 else
  s:=1:
  for i from 0 to n-1 do
  s:=s*(m-i):
  od:
  fi:
  s; end:
```

```
> V2(5,3);
  V2(5,10);
```

$$60$$

$$0$$

Y también podemos construir una fórmula para las variaciones con repetición:

```
> VR:=(m,n)->m^n;
```

$$VR := (m, n) \to m^n$$

```
> VR(2,3);
```

$$8$$

Y dado un conjunto de m elementos podríamos diseñar un procedimiento que construyera primero todas las combinaciones de n elementos con el comando *choose*, y luego hiciera todas las permutaciones para cada una de estas combinaciones. El resultado es el conjunto de todas las variaciones de m elementos tomados de n en n, esto es, las listas ordenadas de n elementos en un conjunto de m elementos.

```
> variaciones:=proc(A,n)
  local C,D,i:
  C:=[op(choose(A,n))]:
  D:={}:
  for i from 1 to nops(C) do
  D:= D union {op(permute([op(C[i])]))}: od:
  D; end:
```

Veamos algunos ejemplos:

```
```

```
> variaciones({a,b,c,d},3);
```

$$\{[a,d,b],[c,a,d],[c,d,a],[d,a,c],[d,c,a],[a,b,c],[a,c,b],[b,a,c],[b,c,a],$$
$$[c,a,b],[c,b,a],[b,d,c],[b,a,d],[b,d,a],[d,a,b],[d,b,a],[a,d,c],[a,b,d],$$
$$[a,c,d],[b,c,d],[c,b,d],[c,d,b],[d,b,c],[d,c,b]\}$$

En efecto, se han obtenido todas:

```
> is(nops(variaciones({a,b,c,d,e},3))=V(nops({a,b,c,d,e}),3));
```

true

Otros ejemplos:

```
> variaciones({1,2,3},2);
```

$$\{[1,2],[2,3],[1,3],[2,1],[3,1],[3,2]\}$$

```
> variaciones({p,q,r,s,t,u,v},7):
> is(nops(%)=7!);
```

true

```
>
```

6. Otras funciones combinatorias (OPCIONAL)

Las funciones **partition()**, **numbpart()** y **randpart()** sirven para calcular el número de formas distintas en que se puede descomponer un número natural en suma de otros, para hallar todas las posibles descomposiciones y para hallar una de ellas al azar respectivamente. Veamos un ejemplo:

```
> partition(10);
```

$$[[1,1,1,1,1,1,1,1,1,1],[1,1,1,1,1,1,1,1,2],[1,1,1,1,1,1,2,2],$$
$$[1,1,1,1,2,2,2],[1,1,2,2,2,2],[2,2,2,2,2],[1,1,1,1,1,1,1,3],$$
$$[1,1,1,1,1,2,3],[1,1,1,2,2,3],[1,2,2,2,3],[1,1,1,1,3,3],[1,1,2,3,3],$$
$$[2,2,3,3],[1,3,3,3],[1,1,1,1,1,1,4],[1,1,1,1,2,4],[1,1,2,2,4],[2,2,2,4],$$
$$[1,1,1,3,4],[1,2,3,4],[3,3,4],[1,1,4,4],[2,4,4],[1,1,1,1,1,5],[1,1,1,2,5],$$
$$[1,2,2,5],[1,1,3,5],[2,3,5],[1,4,5],[5,5],[1,1,1,1,6],[1,1,2,6],[2,2,6],$$
$$[1,3,6],[4,6],[1,1,1,7],[1,2,7],[3,7],[1,1,8],[2,8],[1,9],[10]]$$

Es decir, ha encontrado todas las maneras en que 10 se puede poner como suma de números naturales menores o iguales que 10.

Para contarlas se usa:

```
> numbpart(10);
```

42

Y en efecto calcula el cardinal del conjunto anterior:

```
> nops(partition(10));
```

42

De igual manera si se quiere conseguir una partición aleatoriamente:

```
> randpart(10);
```

$$[1,1,2,6]$$

Observa que si se repite el comando anterior:

```
> randpart(10);
```

$$[1,1,1,1,2,4]$$

[>
Como la elección es aleatoria, es demasiada casualidad que haya salido igual.

7. Ejercícios

Después de haber leído detenidamente los contenidos de las secciones anteriores deberías ser capaz de responder a las siguientes preguntas:

(1) Determinar el número de palabras que se pueden construir permutando las letras de la palabra "semaforos".
(a) 90721
(b) 91720
(c) 90720
[>
[>
[>
(2) ¿De cuantas maneras se pueden sentar 11 personas en dos mesas, una de 6 comensales y otra de 5 comensales, teniendo en cuenta tanto la distribución en las dos mesas, como el orden en cada mesa?:
(a) 36916800
(b) 43916800
(c) 39916800
[>
[>
[>
(3) Determinar cuántos subconjuntos de menos de 5 elementos (de 0 a 4 elementos) tiene el conjunto
[> `E:={a,b,c,d,f,e,r,t,y,n,l}:`
(a) 432
(b) 554
(c) 562
[>
[>
[>
(4) Dado un conjunto de 54 personas hay que elegir un delegado, un subdelegado y un vocal. De entre los restantes hay que elegir a 3 personas para que les acompañen. Determinar cuántas posibles delegaciones hay.
(a) 3099889800
(b) 3099259800
(c) 3099889800
[>
[>
[>
(5) Sean los conjuntos A y B descritos abajo.
```
> A:={1,2,3,4,5,e,r,t,y,u,i,o,p}:
  B:={1,2,3,4,5,e,r,t,g,y,u,h,j,l}:
```
Determinar cuántos subconjuntos de 3 elementos comparten A y B.

(a) 120
(b) 130
(c) 134
[>
[>
[>

(6) Sea un alfabeto con 27 letras. Determinar cuántos códigos alfanuméricos (formados por números y letras) de 7 símbolos se pueden establecer, sabiendo que los dos primeros símbolos han de ser letras y los dos últimos han de ser números.

(a) 13662633690
(b) 4061864070
(c) 3692603700
[>
[>
[>

(7) En un taller hay 10 coches que necesitan reparaciones. Calcular de cuantas formas distintas se pueden
elegir y ordenar los primeros 4 coches que se van a arreglar.

(a) 210
(b) 420
(c) 5040
[>
[>
[>

(8) Determinar cuantos posibles resultados pueden acontecer al lanzar 25 monedas simultáneamente.

(a) 16777216
(b) 300
(c) 26
[>
[>
[>

(9) Sea un alfabeto con 27 letras. Determinar cuantas contraseñas alfanuméricas (formadas por números y letras, distinguiendo mayúsculas y minúsculas) de 8 símbolos se pueden establecer, sabiendo que los tres primeros símbolos han de ser letras y los
dos últimos han de ser números.

(a) 515978035200
(b) 4127824281600
(c) 417942208512
[>
[>
[>
[>