Combinatoria

Oscar Cárdenas Aroca

14 de noviembre de 2018

Combinatoria: La Combinatoria es la parte de las matemáticas que estudia las formas en que los permite contar el número de situaciones que se pueden dar al someter a un conjunto finito a las acciones de ordenar y/o elegir entre sus elemento.

Las formas de combinatoria son:

1. **Permutaciones**: La acción que se hace en la permutación es ORDENAR.

   Hay 2 tipos de permutaciones

   a) **Con repetición**:donde n es el número de cosas que puedes elegir, y eliges r de ellas (Se puede repetir, el orden importa)

   b) **Sin repetición**:donde n es el número de cosas que puedes elegir, y eliges r de ellas (No se puede repetir, el orden importa)

2. **Combinaciones**: Corresponden al número de formas en que se puede extraer subconjuntos a partir de un conjunto dado.

0.1. Fórmulas matemáticas

1. **Permutaciones**:

$$\frac{n!}{(n-r)!}$$

   Donde n es la cantidad de elementos y r es la cantidad de elementos que se pueden tomar.
   NOTA: n tiene que ser mayor o igual que r.

2. **Combinaciones**:

$$\frac{n!}{r!(n-r)!}$$

Donde n es la cantidad de elementos y r es la cantidad de elementos que se van a combinar entre si.
NOTA: n tiene que ser mayor o igual que r.

0.2. Enfoque y planteamiento en lenguajes de programación de las combinatorias

Las combinatorias se enfocan en las teorías de probabilidades y en sus aplciaciones practicas.

Las combinatorias se establece para trabajarse en las teorías de problabilidades en las cuales se requiere para poder llegar a datos como por ejemplo la probalididad de sacar 2 reyes de un mazo de 52 cartas, etc.

Por lo tanto es necesario poder trabajarlas en lenguajes de programción para llegar a resultados más precisos y obtnerlos más rápido.

0.3. Algoritmos

1. PERMUTACIÓN.

   a) PERMUTACIÓN en lenguaje C++:
   Teniendo en cuenta la fórmula de permutación

$$\frac{n!}{(n-r)!}$$

   Se da el siguiente algoritmo.

```
#include<bits/stdc++.h>
using namespace std;

int factorial(int n){
    if (n==0) return 1;
    else return n * factorial (n-1);
}

double Permutacion (int n, int r){
    double perm;
    int aux;
    aux=n-r;
```

```
    perm= factorial(n)/factorial(aux);
    return perm;
}

int main(){
    ios::sync_with_stdio(false);
        cin.tie(NULL);
        int n, r;
        cin>>n>>r
    cout<<Permutacion(n,r)<<"\n";
    return 0;
}
```

b) PERMUTACIÓN (Simplificación) en lenguaje C++:

Aquí una mejora del algoritmo anterior, donde se implementa la simplificación.

```
#include<bits/stdc++.h>
using namespace std;

long permutacion(int start, int finish)
{
    long f = start;
    start--;
    while(start >= finish)
    {
        f *= start;
        start--;
    }
    return f;
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int n, r;
    cin>>n>>r;
    cout<<permutacion(n, ((n-r)+1))<<"\n";
    return 0;
}
```

2. COMBINACIÓN.

a) COMBINACIÓN en C++:

Teniendo en cuenta la fórmula de combinación

$$\frac{n!}{r!(n-r)!}$$

Se da el siguiente algoritmo.

```
#include<bits/stdc++.h>
using namespace std;

int factorial(int n){
    if (n==0) return 1;
    else return n * factorial (n-1);
}

double Combinacion(int n, int r){
    double comb;
    int aux;
    aux=n-r;
    comb= factorial(n)/(factorial(aux) * factorial(r))
        ;
    return comb;
}


int main(){
    ios::sync_with_stdio(false);
        cin.tie(NULL);
        int n, r;
        cin>>n>>r
    cout<< Combinacion(n,r)<<"\n";
    return 0;
}
```

b) COMBINACIÓN (Simplificación) en lenguaje C++:

Aquí una mejora del algoritmo anterior, donde se implementa la simplificación.

```
#include<bits/stdc++.h>
using namespace std;


int factorial(int n){
    if (n==0) return 1;
    else return n * factorial (n-1);
}

long Simplificacion(int start, int finish)
{
    long f = start;
    start--;
    while(start >= finish)
    {
        f *= start;
```

```
        start--;
    }
    return f;
}



int main()
{
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int n, r;
    cin>>n>>r;
    cout<<(Simplificacion(n, ((n-r)+1))/factorial(r))<<
        "\n";
    return 0;
}
```

NOTA: Los algoritmos con simplificación son mejoras de los algoritmos principales.