Numerical Recipes
(Recetas numéricas)

Ejemplo de un libro donde hay subrutinas ya escritas

Se puede encontrar acá:

https://websites.pmc.ucsc.edu/~fnimmo/eart290c_17/NumericalRecipesinF77.pdf

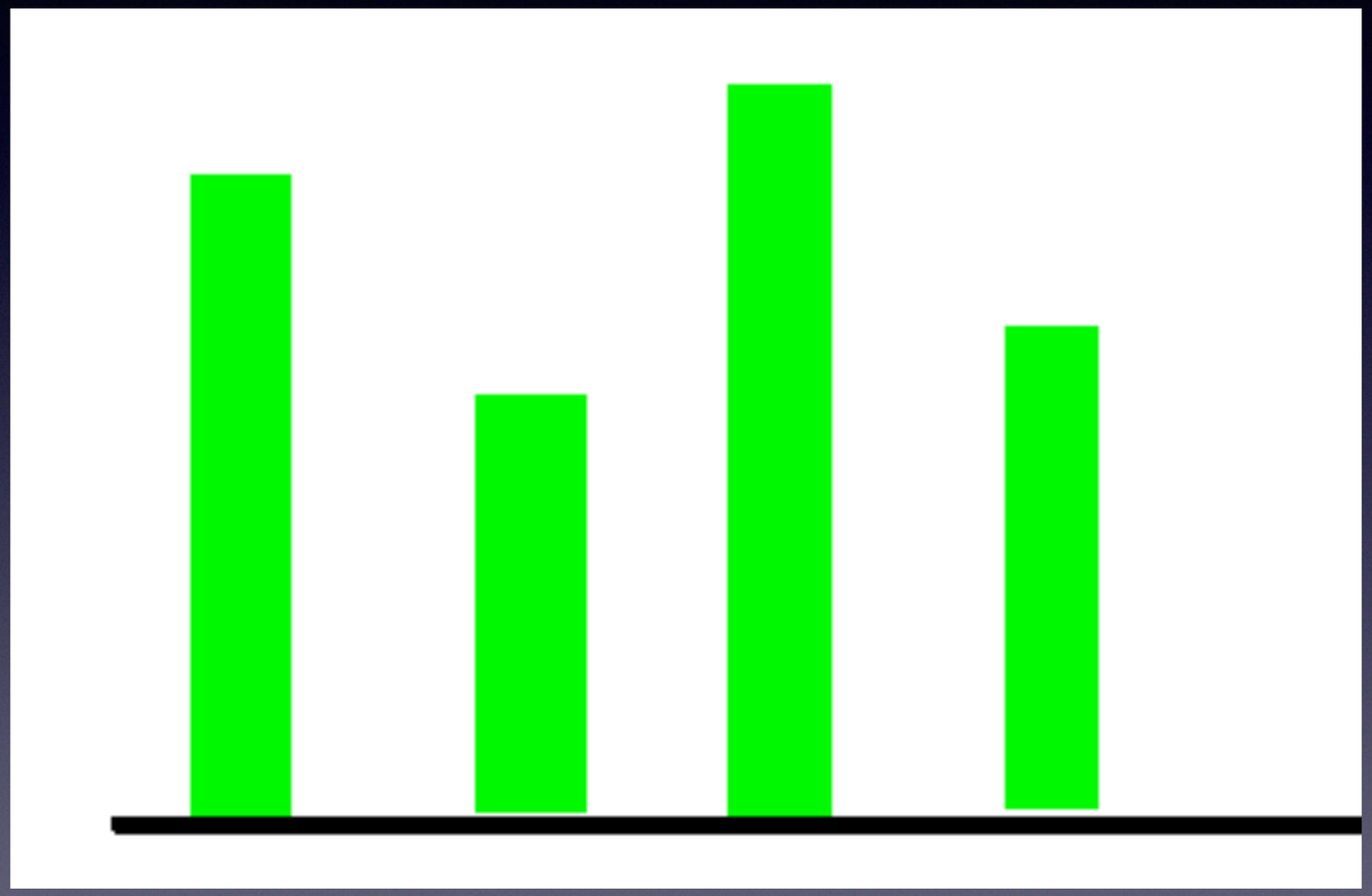
- Sistemas de ecuaciones -> Página 22
- Integración de funciones -> Página 123
- Números al azar -> Página 266

Aplicaciones actuales:

Ordenar (sort)

Ordenar datos es un problema de la informática y nos compete porque en ciencia muchas veces tenemos ordenar datos por alguna característica previo a su análisis

¿Cómo sería el Algoritmo?

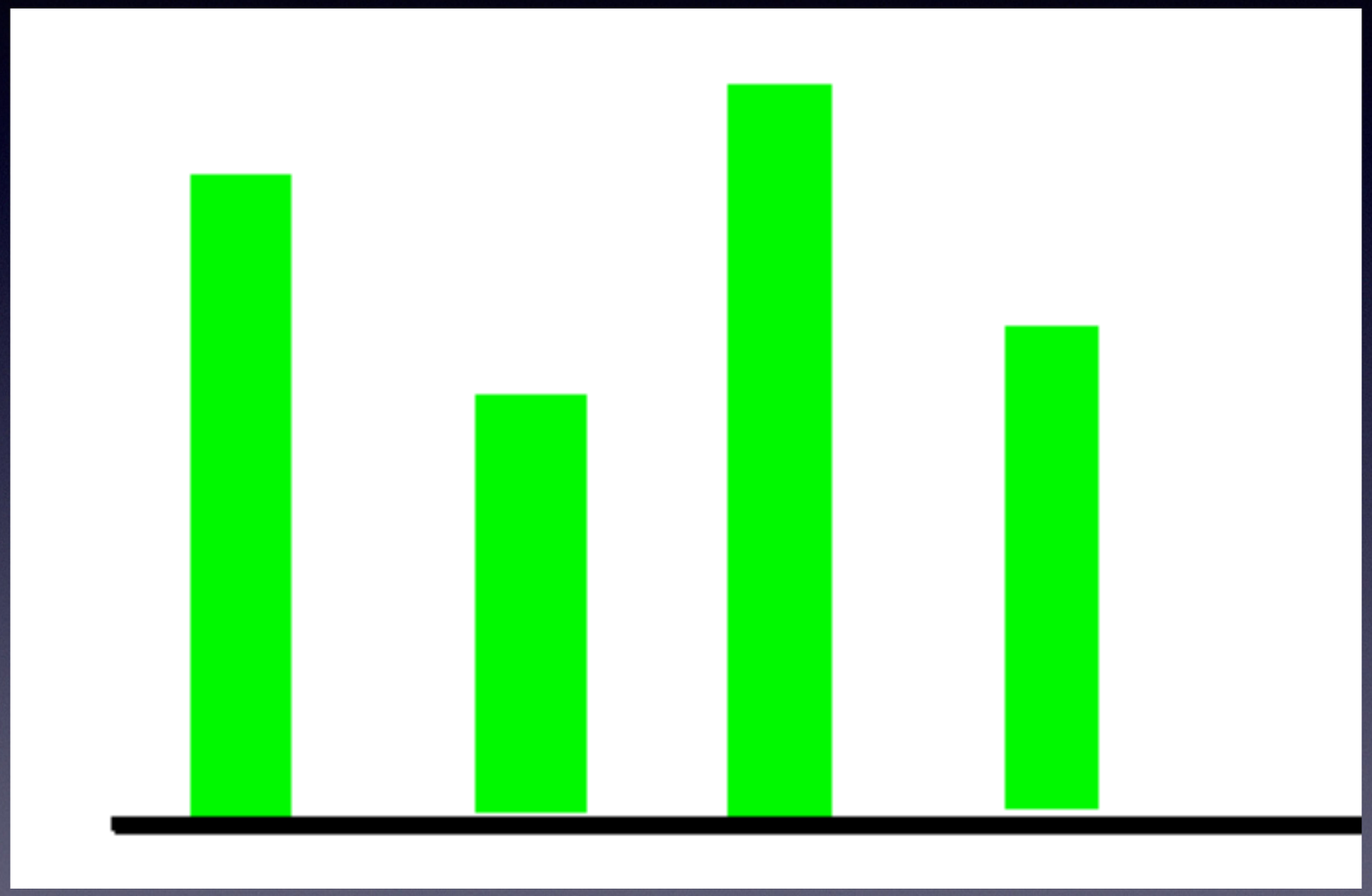


Método de Selección

Programas - Método de selección

```
subroutine selec(n,a)
real*4 a(10000000)

do i=1,n-1
  do j=i+1,n

    if(a(i).gt.a(j)) then
      temp=a(i)
      a(i)=a(j)
      a(j)=temp
    endif

  enddo
enddo

return
end
```

¿Cómo sería el Algoritmo?



Método de la burbuja (Bubble sort)

Programas - Método de la Burbuja

```
subroutine bubble(n,a)
real*4 a(10000000)
logical bandera

bandera=.true.


do while(bandera)
  bandera=.false.

  do i=1,n-1

    if(a(i).gt.a(i+1)) then
      temp=a(i)
      a(i)=a(i+1)
      a(i+1)=temp
      bandera=.true.
    endif

  enddo

enddo
return
end
```

Otros algoritmos (mucho mejores)

Inserción Directa (Straight Insertion)

Es el método natural que se usa para ordenar un mazo de cartas en los juegos de naipes.
En el peor de los casos estaríamos en un algoritmo cuyo tiempo crece con N^2 y por lo tanto se aconseja su uso para muestras pequeñas $N<20$
Es un método se utiliza de apoyo en otros métodos más sofisticados

Otros algoritmos (mucho mejores)

Método de las cáscaras (Shell´s method)

Es un método que se basa en el anterior. Divido la muestra en grupos de a dos y uso inserción directa.
Luego junto los grupos de a 4 elementos y repito así hasta que cubro toda la muestra.
$t \sim N^{1.25}$

Ordena rápido pocos elementos y pierde poco tiempo con la ya ordenado
En promedio es mucho más rápido que N^2

Compararemos métodos con un programa

```
c------- Burbuja ----------------------


      call leer(n,a)
      open(23,file='/proc/uptime')
      read(23,*)ti,t2
      close(23)


      call bubble(n,a)


      open(23,file='/proc/uptime')
      read(23,*) tf,t2
      close(23)


      time=tf-ti
      write(*,*) 'Tiempo de la Burbuja=',time
```

Y repitamos este procedimiento para varios de los algoritmos y veamos cuanto tardan en ordenar 100,000 números.

Resultados

Algoritmo	10^5 números segundos	10^6 números segundos
Burbuja	36.07	3678.71
Selección	25.95	2547.27
Inserción Directa	6.56	646.6
Cáscaras	0.02	0.3
Heapsort	0.02	0.19
Quicksort	0.01	0.12

Y si no quiero perder el “orden” original

Tengo que construir un vector de índices (tipo "función identidad”), tal que el primer elemento tiene un 1 y el segundo un 2 y así sucesivamente.

```
⋮
IF(algo) then
temp=I(j)
I(j)=I(j+1)
I(j+1)=temp
⋮
```

Entonces ordenamos I(j) tal que A(I(j)) lo vea como el vector ordenado y A(j) es el original

Por ejemplo si tengo un par ordenado y quiero conservar ese orden

```
⋮
IF(algo) then
temp=X(i)
X(i)=X(i+1)
X(i+1)=temp
temp=Y(i)
Y(i)=Y(i+1)
Y(i+1)=temp
⋮
```

Librerías

Son colecciones de subroutines y funciones ya pre-compiladas.

Conociendo sus argumentos y orden por la documentación son muy fáciles de usar.

Las hay para cualquier tema y en particular son muy útiles en ciencia las que que trabaja con álgebra vectorial y con GPUs.

También son muy útiles las que sirven para realizar figuras.

Subroutines del Sistema Operativo

Vienen con el SO, y hay que tener documentación para usarlas. Veamos dos ejemplos muy útiles:

Call getenv()
Call getarg()
Call system()

Call system()

Sirve para acceder a comandos del sistema operativo desde mi programa

```
.
.
.
CALL SYSTEM('ls *.dat > mis_archivos.txt')
OPEN(45, file='mis_archivos.txt')
.
.
.
```