Universidad Rey Juan Carlos

Grado en Matemáticas

Algoritmos de ordenación I

Geometría computacional

Autores: Guillermo Grande Santi y

Alejandro López Adrados

Marzo, 2022

Índice

1 Objetivos

El objetivo de esta práctica era la implementación de tres algoritmos de ordenación distintos y compararlos con una gráfica entre ellos. Para esta comparación, hemos calculado el tiempo que tardaban en ejecutarse con una cantidad distinta de elementos al algoritmo.

2 Metodología

2.1 Algoritmo de la burbuja

El algoritmo de la burbuja consiste en recorrer con un bucle for los elementos de un vector de dos en dos comparándolos entre ellos. Si el primer elemento es mayor que el segundo se invierte su orden y en caso contrario, se mantiene. Esto se realiza a lo largo de todo el vector, devolviendo así el vector ordenado.

2.2 Algoritmo de inserción

El algoritmo de inserción consiste en recorrer mediante un bucle for el vector, fijándose una posición k en cada iteración. Después de esto, se toma el elemento k+1 y se compara con todos los anteriores hasta que se encuentra un elemento menor o el principio del vector. En esta posición se insertará el elemento k+1, desplazando hacia la derecha el resto. Tras recorrer el vector por completo, éste quedará ordenado.

2.3 Algoritmo de selección

El algoritmo de selección consiste en recorrer el vector y encontrar el elemento mínimo. Una vez encontrado, se pondrá en la primera posición. Repitiendo esto con los siguientes mínimos y colocándolos en posiciones sucesivas al primero se obtendrá finalmente el vector ordenado.

2.4 Cálculo de tiempos y gráfico

Para calcular los tiempos hemos usado la función "proc.time()", que determina el tiempo real y de la CPU en segundos desde que se inició el proceso en R. Por tanto, por cada vez que se llama al algoritmo se coge el tiempo justo antes de llamarlo, para así restar al tiempo total de ejecución el tiempo que tardó en llegar hasta justo antes de la llamada al algoritmo.

En cuánto al gráfico, se cogen como variable independiente un vector formado por los diferentes números de elementos que se usarán en el algoritmo. Por otro lado, como variable dependiente se crean tres vectores (uno por cada algoritmo) en los que se guardarán los tiempos de ejecución de cada uno de los algoritmos con el número de elementos que indica la variable independiente. Una vez tenemos estos datos, se usan las funciones "plot()" y "lines()", que se encargarán de dibujar la gráfica.

3 Conclusiones

Como conclusión, observando la gráfica podemos intuir que el algoritmo de la burbuja en un principio parece el menos eficiente con un número de elementos pequeño, mientras que el algoritmo de selección parece el mejor. Sin embargo, cuando este número va aumentando, se puede observar que el algoritmo de la burbuja acaba siendo el más eficiente (podemos ver además que crece de manera lineal) mientras que los otros, pese a empezar con mejores tiempos, terminan creciendo en mayor medida y siendo menos eficientes.

4 Anexos

4.1 Código de algoritmos

```
#Algoritmo Burbuja
burbuja = function(vec)
{
  vecAux = vec
  for(i in seq(1, length(vecAux)-1))
  {
    #Si un numero es mayor que el de la siguiente posición,
    #se intercambian sus posiciones.
    if(vecAux[i] > vecAux[i+1]) {
      aux = vecAux[i+1]
      vecAux[i+1]=vecAux[i]
      vecAux[i]=aux
    }
  }

  if(isTRUE(all.equal(vec, vecAux)))
  {
    return(vecAux)
  }
  else {
    return(burbuja(vecAux))
  }
}

#Algoritmo Inserción
insercion = function(vec)
{
```

```
  for (i in seq(1, length(vec)))
  {
    actual = vec[i];
    j = i-1;
    #Desplazamiento de los elementos de la matriz
    while ((j>0) && (vec[j]>actual))
    {
      vec[j+1]=vec[j];
      j = j-1;
    }
    #Insertar el elemento en su lugar
    vec[j+1]=actual;
  }
  return(vec);
}

#Algoritmo Selección
seleccion = function(vec)
{
  for (i in 1:(length(vec)-1))
  {
    minimo = i
    temp = vec[i]
    for(j in (i:length(vec)))
    {
      if(vec[j]<temp) {
        minimo = j
        temp = vec[j]
      }
    }
```

```
    vec[minimo] = vec[i]
    vec[i] = temp
  }
  return(vec)
}

#Vector de números aleatorios y llamada a los algoritmos
#Variable auxiliar
aux<-0
#Para cambiar el número de elementos se debe modificar
#la siguiente variable
elementos<-40
vec<-runif(elementos,0,1000)

#Cálculo de los tiempos que tarda cada uno de los algoritmos
tiempoBurbuja<-proc.time()
burbuja(vec)

##  [1]  54.12011  97.33363 108.74554 110.39779 172.48501 182.45229 197.44292
##  [8] 241.22138 277.13178 289.40766 309.39760 320.95801 334.45671 412.69840
## [15] 420.95599 423.26966 453.49283 509.38263 528.50608 561.41935 564.85082
## [22] 571.48404 644.96248 686.32807 709.54960 716.85019 731.20050 745.79297
## [29] 750.37892 755.81309 771.17311 775.02778 801.11160 801.48238 880.93294
## [36] 882.08305 884.27710 884.94873 924.15652 976.14157

proc.time()-tiempoBurbuja

##    user  system elapsed
##   0.062   0.006   0.068

tiempoInser<-proc.time()
insercion(vec)
```

```
##  [1]  54.12011  97.33363 108.74554 110.39779 172.48501 182.45229 197.44292
##  [8] 241.22138 277.13178 289.40766 309.39760 320.95801 334.45671 412.69840
## [15] 420.95599 423.26966 453.49283 509.38263 528.50608 561.41935 564.85082
## [22] 571.48404 644.96248 686.32807 709.54960 716.85019 731.20050 745.79297
## [29] 750.37892 755.81309 771.17311 775.02778 801.11160 801.48238 880.93294
## [36] 882.08305 884.27710 884.94873 924.15652 976.14157
```

```
proc.time()-tiempoInser
```

```
##    user  system elapsed
##   0.018   0.000   0.018
```

```
tiempoSelec<-proc.time()
seleccion(vec)
```

```
##  [1]  54.12011  97.33363 108.74554 110.39779 172.48501 182.45229 197.44292
##  [8] 241.22138 277.13178 289.40766 309.39760 320.95801 334.45671 412.69840
## [15] 420.95599 423.26966 453.49283 509.38263 528.50608 561.41935 564.85082
## [22] 571.48404 644.96248 686.32807 709.54960 716.85019 731.20050 745.79297
## [29] 750.37892 755.81309 771.17311 775.02778 801.11160 801.48238 880.93294
## [36] 882.08305 884.27710 884.94873 924.15652 976.14157
```

```
proc.time()-tiempoSelec
```

```
##    user  system elapsed
##   0.021   0.000   0.021
```

4.2 Representación de la gráfica

```
# Datos
Elementos<-c(500,1000,2500,5000,7500,10000,25000,50000)
Tiempo<-c(0.06,0.19,0.94,2.61,4.19,5.84,16.84,35.3)
y2<-c(0.03,0.04,0.2,0.75,1.68,2.95,18.68,75.32)
```

```
y3<-c(0.02,0.04,0.14,0.5,1.11,1.96,12.11,48.2)
# Gráfico
plot(Elementos, Tiempo, type = "l", main="ALGORITMOS")
legend(x = "topleft",          # Posición
       legend = c("Burbuja", "Inserción","Selección"), # Textos de la leyenda
       lty = c(1),           # Tipo de líneas
       col = c(1, 2, 3),           # Colores de las líneas
       lwd = 2)                 # Ancho de las líneas
lines(Elementos,y2, type ="l", col=2)
lines(Elementos,y3, type ="l", col=3)
```