UNER Facultad de Ciencias de la Administración

Algoritmos y Programación

Métodos de Ordenamiento -Shell-

2018

Prof. Miguel A. Fernández

1

Métodos de Ordenamiento - Shell

Requerimiento:

Es necesario conocer previamente la cantidad de elementos que tiene el array.

Variantes:

- Método Básico
- Método Mejorados

Donald Shell
Marzo 1924 -
Universidad de Michigan

UNER

2

Métodos de Ordenamiento - Shell

Método Básico: Procedimiento:

1. Se calcula la distancia de los elementos a comparar.
Para la primera recorrida dicho valor se obtiene dividiendo por dos la cantidad de elementos del vector.

2. Se recorre el array comparando los dos elementos que están separados por una distancia igual al valor calculado en el punto 1, utilizando la metodología burbuja.

3. Se recalcula el valor correspondiente a la distancia de los elementos a comparar, dividiendo al valor utilizado en la recorrida anterior por 2. Se obtiene así un nuevo valor que resulta igual a la mitad del utilizado en la recorrida anterior. (se toma la parte entera cuando el cociente no sea exacto).

Se inicia una nueva recorrida repitiendo desde el paso 2.

UNER

3

Ordenamiento Shell

D = 9/2 = 4

Primera Recorrida

1	2	3	4	5	6	7	8	9
5	8	0	4	7	6	3	9	3

Cambia

5	6	0	4	7	8	3	9	3

Cambia

Fin Primera Recorrida

5	6	0	4	3	8	3	9	7

Segunda Recorrida:

Cambia

3	6	0	4	5	8	3	9	7

Fin Segunda Recorrida

4

Ordenamiento Shell

D = 9/2 = 4

Tercera Recorrida

1	2	3	4	5	6	7	8	9
3	6	0	4	5	8	3	9	7

Fin Tercera Recorrida

No se realizaron intercambios. Se recalcula distancia

D = D/2 = 2

Primera Recorrida:

3	6	0	4	5	8	3	9	7

Cambia

0	6	3	4	5	8	3	9	7

Cambia

0	4	3	6	5	8	3	9	7

5

Ordenamiento Shell

D = D/2 = 2

1	2	3	4	5	6	7	8	9
0	4	3	6	5	8	3	9	7

Cambia

0	4	3	6	3	8	5	9	7

Fin Primera Recorrida

Segunda Recorrida:

0	4	3	6	3	8	5	9	7

Fin Segunda Recorrida

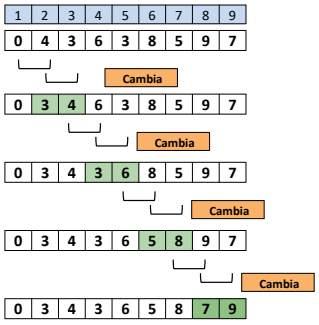
No se realizaron intercambios. Se recalcula distancia

6

Ordenamiento Shell

D = D/2 = 1

Primera Recorrida:

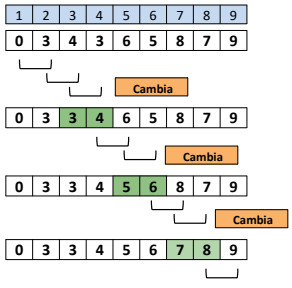


Fin Primera Recorrida

Ordenamiento Shell

D = D/2 = 1

Segunda Recorrida:



Fin Segunda Recorrida

En una tercera recorrida no se realizarían intercambios, Por lo que siendo D=1 finaliza el procedimiento.

Métodos de Ordenamiento - Shell

Método Básico:

Ordenar los datos de un array "Total" que contiene 100 elementos enteros.

```
Program Shell-Uno
Type
   Totales = array [1..100] of integer;
Var
   Total : Totales;
   i, Aux : integer;
   B : Boolean;
```

```
Begin
  D := 100/2
  Mientras D > 0
     B := False
     Mientras B = False
        B := True
        Parai = 1, ( 100 - D ),1
           Si Total(i) > Total( i + D )
                Aux := Total(i)
                Total(i) := Total( i + D )
                Total(i + D ) := Aux
                B := False
           FinSi
        FinPara
     FinMientras
     D := D/2
  FinMientras
FIN
```

UNER

Métodos de Ordenamiento - Shell

Método con retroceso: Procedimiento:

1. Se calcula la distancia de los elementos a comparar.
Para la primera recorrida dicho valor se obtiene dividiendo por dos la cantidad de elementos del vector.

2. Se inicia una recorrida del vector comparando los dos elementos que están separados por una distancia igual al valor calculado en el punto 1. Si no están en el orden requerido, se los cambia de lugar.

3. Se recalcula el valor correspondiente a la distancia de los elementos a comparar, dividiendo al valor utilizado en la recorrida anterior por 2. Se obtiene así un nuevo valor que resulta igual a la mitad del utilizado en la recorrida anterior. (se toma la parte entera cuando el cociente no sea exacto).

Se inicia una nueva recorrida repitiendo desde el paso 2.

UNER

Métodos de Ordenamiento - Shell

Se inicia una nueva recorrida repitiendo desde el paso 2.

Cuando en cualquier recorrida, es necesario realizar un intercambio, el método exige que se 'retroceda', restituyendo las posiciones de la comparación inmediata anterior para volver a comparar los elementos e intercambiarlos si corresponde.

Este retroceso debe continuar hasta que:

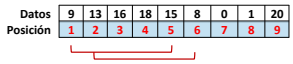
a) la comparación efectuada indique que los elementos están en el orden correspondiente.
b) cuando se llegue al inicio del vector.

Una vez finalizado el proceso de retroceso, se debe continuar la recorrida desde el punto en que se inició dicho retroceso.

El procedimiento termina cuando finaliza una recorrida en la que se compararon e intercambiaron elementos ubicados en posiciones contiguas.

Métodos de Ordenamiento - Shell

Dado el arreglo:

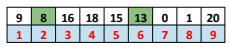


PRIMERA RECORRIDA

Valor de D = 9/2 luego D= 4

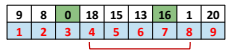
Compara: i=1 (1-5) No cambia

Suma 1 a i y compara: i=2 (2-6): Cambia



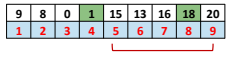
Retroceso: i=i-D (-2) No es posible.

Retoma, suma 1 a i y Compara: i=3 (3-7): Cambia



Retroceso: i=i-D (-1) No es posible.

Retoma, suma 1 a i y compara: i=4 (4-8): Cambia



Retroceso: i=i-D (0) No es posible.

Retoma, suma 1 a i y Compara: i=5 (5-9): No Cambia

FIN PRIMERA RECORRIDA

Métodos de Ordenamiento

Situación al final de la primer recorrida

9	8	0	1	15	13	16	18	20
1	2	3	4	5	6	7	8	9

SEGUNDA RECORRIDA
Valor de D = 4/2 luego D=2
Compara: i=1 (1-3) cambia

0	8	9	1	15	13	16	18	20
1	2	3	4	5	6	7	8	9

Retroceso: i=i-D (-1) no es posible
Retoma, suma 1 a i y compara: i=2
(2,4) cambia

0	1	9	8	15	13	16	18	20
1	2	3	4	5	6	7	8	9

Retroceso: i=i-D (0) no es posible
Retoma, suma 1 a i y compara: i=3
(3,5) No cambia

Suma 1 a i y compara: i=4 (4-6) No cambia
Suma 1 a i y compara: i=5 (5-7) No cambia
Suma 1 a i y compara: i=6 (6-8) No cambia
Suma 1 a i y compara: i=7 (7-9) No cambia

FIN SEGUNDA RECORRIDA

13

Métodos de Ordenamiento

TERCERA RECORRIDA
Valor de D = 2/2 luego D=1
Compara: i=1 (1-2) No cambia

0	1	9	8	15	13	16	18	20
1	2	3	4	5	6	7	8	9

Suma 1 a i y compara: i=2 (2-3) No cambia
Suma 1 a i y compara: i=3 (3-4) Cambia

0	1	8	9	15	13	16	18	20
1	2	3	4	5	6	7	8	9

Retrocede: [j=3; i=i-D; i=2]
Compara: i=2 (2-3) No Cambia

Retoma, suma1 a i y Compara: i=4 (4-5) No Cambia
Suma 1 a i y compara: i=5 (5-6) Cambia

0	1	8	9	13	15	16	18	20
1	2	3	4	5	6	7	8	9

Retrocede: [j=5; i=i-D; i=4]
Compara: i=4 (4-5) No Cambia

Retoma, suma1 a i y Compara: i=6 (6-7) No Cambia
Suma 1 a i y compara: i=7 (7-8) No Cambia
Suma 1 a i y compara: i=8 (8-9) No Cambia

FIN TERCERA RECORRIDA (ULTIMA)

¡ ARRAY ORDENADO !

14

Métodos de Ordenamiento

Recordando:

Método Burbuja – Variante con retroceso

```
Para i = 1, n-1, 1
        z := i
        Mientras  i > 0 and then V(i) > V(i + 1)
                Aux  := V(i)
                V(i) :=  V(i + 1)
                V(i + 1) := Aux
                i := i -1
        Finmientras
        i := z
FinPara
```

UNER

15

Métodos de Ordenamiento - Shell

Solución Algorítmica:

Ordenar los datos de un arreglo "V" de "n" elementos:

```
Begin
D := n / 2
Mientras D > 0
    Para i = 1, (n – D), 1
        j := i
        Mientras i > 0 and then  V(i) > V( i + D)
                AUX := V(i)
                V(i)  := V(i + D)
                V(i + D) := AUX
                i := i - D
        Finmientras
        i := j
    Finpara
    D := D / 2
Finmientras
FIN
```

UNER

16

Métodos de Ordenamiento - Shell

Caso:

Dado un arreglo que contiene los datos de los alumnos de la FCAd se desea ordenarlos en forma creciente por número de documento. El arreglo contiene el número de documento y el nombre de cada alumno.

Se parte de considerar que la cantidad de elementos del arreglo se encuentra en una variable LON.

```
Program Ordena
Type
        Registro  =  Record
            NUD : Integer;
            NOM   : string
        end;
        Alumnos = array [ 1..100] of Registro;
 Var
        Alumno  :  Alumnos;
        Aux : Registro;
        i, j, LON, D : Integer;
```

UNER

17

Métodos de Ordenamiento - Shell

Solución Algorítmica:

```
Begin
  LON := 100
  D := LON / 2
  Mientras D > 0
    Para i = 1, (LON – D), 1
        j := i
        Mientras i > 0 and then ALUMNO(i).NUD > ALUMNO(i+D).NUD
            AUX := ALUMNO(i)
            ALUMNO(i)  := ALUMNO(i+D)
            ALUMNO (i+D) := AUX
            i := i – D
        Finmientras
        i := j
    FinPara
    D := D / 2
  Finmientras
FIN
```

UNER

18

UN ALTO EN EL CAMINO

¿ Cuál es el mejor lenguaje de programación?

La estrategia de solución y el dominio de aplicación

PARADIGMAS DE PROGRAMACIÓN

ESTRUCTURADO	ORIENTADO A OBJETOS	FUNCIONAL	LÓGICO
C Pascal Cobol PHP Python	Ada 95 Smaltalk Delphi Visual Object Java Javascript C++ Python C#	Lisp Haskell	Prolog

UNER

Método Shell

Ejemplo Pascal

```
program ShellSort;
type
   rango = 1..500;
   Tlista = array [rango] of integer;
var
   lista : Tlista;

begin {programa principal}
 clrscr;
 writeln();
 write(' Ordenacion por ShellSort: ');
 ShellSort;
 write(' FIN Ordenacion ShellSort: ');
end.
```

```
procedure ShellSort;
var
    Distancia, i, j, k , aux: integer;
begin
   Distancia := 500 div 2;
   while (Distancia > 0) do
   begin
      for i := (Distancia + 1) to 500 do
      begin
         j := i - Distancia;
         while (j > 0) do
         begin
            k := j + Distancia;
            if (lista[j] <= lista[k]) then
               j := 0
            else
               aux := lista[j]
               lista[j] := lista[k]
               lista[k] := aux
               j := j - Distancia;
         end; {while j > 0}
      end; {for}
      Distancia := Distancia div 2;
      end; {while (Distancia > 0)}
end;
```

Métodos Shell

Solución en JAVA

```
Public static void ShellSort(int[] array){
   int gap = array.length / 2;
   while (gap > 0) {
      for (int i = 0; i < array.length - gap; i++) {
         int j = i + gap;
         int tmp = array[j];
         while (j >= gap && tmp > array[j - gap]) {
            array[j] = array[j - gap];
            j -= gap;
         }
         array[j] = tmp;
      }
      if (gap == 2) {
         gap = 1;
      }else{
         gap /= 2;
      }
   }
   return array;
}
```

UNER

21

Métodos de Ordenamiento - Shell

...Y AHORA....
¡ A TRABAJAR !

UNER

22