Universidad Autónoma de Baja California

Facultad de Ciencias Químicas e Ingeniería

Algoritmos y Estructura de Datos

Análisis de la eficiencia del Algoritmo de ordenamiento Shell

Integrantes: Saul Ivan Flores – 01217102

Fecha: Abril 27, 2018

Código utilizado

```c
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "./universales/menu.h" //  se incluye libreria de generacion eh
impresion de menus
#include "./universales/titles.h"// se incluye para impresion de titulos
#include "./universales/array.h"// se incluye para impresion de titulos

/*
  COMPILACION DEL PROGRAMA:
  1. Acceder al root del projecto
  2. ejecutar siguente comando en terminal
      "gcc -o main main.c ./universales/menu.c ./universales/titles.c
./universales/array.c"
  3. Ejecutrar comando de ejecucion
      "./main"
*/

void shellShort();
void ansImpresion();
int main() {
  printTitle("Análisis de la Shell Sort");
  //int arr[] = {9, 7, 5, 2, 1, 4, 3, 6};   //base case
  //int arr[] = {1, 2, 3, 4, 5, 6, 7, 9};   //best case
  //int arr[] = {9, 7, 6, 5, 4, 3, 2, 1};   //worst case
  //int arr[] = {1, 2, 3, 4, 5, 6, 7, 9, 10};   //best case odd
  int arr[] = {10, 9, 7, 6, 5, 4, 3, 2, 1};   //worst case odd

  int size = sizeof(arr)/sizeof(arr[0]);    //tamano del arreglo
  shellShort(arr, size);                    //se manda a llamar metodo de
ordenamieto shellSort
  printf("\tTamano de arreglo:  %d\n", size); //impresion del tamano del arreglo
  return 0; //salida de funcion
}
//Funcion que ejecuta el metodo de ordenamieto shell sort
void shellShort(int arr[], int size) {
  printArray(arr, size);  //funcion impresion de arreglo declarada en lib
array.c
  int i, j, middle, iterationCounter; //declaracion de funcion
  clock_t start = clock(); //inicio la variable tipo clock
  middle = size / 2 ;    //Se obtiene el punto medio de arreglo; si arreglo es
impar ej. n = 7; n/2 = 3.5 se considera solo el 3 del resultado obtenido
  iterationCounter = 0; //contador de iteraciones
```

```
for(middle; middle > 0; middle /= 2) { //ciclo for mientras middle sea mayor a cero al final de cada iteracion dividelo por 2
    for(i = middle; i < size; i++) { //mientras i sea menor al tamano total del arreglo incrementa i
      int tmp = arr[i]; //variable temporal almacena el valor en al posicion indicda, esto se utiliza para itercambiar valores en el arreglo
      for(j = i; j >= middle && arr[j - middle] > tmp; j -= middle) {
        arr[j] = arr[j - middle];
        printArray(arr, size); //impresion de arreglo en su estado actual
        iterationCounter++; //cantidad total de iteracion en el 3r for anidado
      }
      arr[j]  = tmp; // se le asigna el valor de tmp a la posicion j;
    }
  }
  clock_t end = clock(); //inicio variable tipo clock
  double timeSpent = (double)(end - start) / CLOCKS_PER_SEC;
  ansImpresion(timeSpent, iterationCounter);  //mando valores obtenidos a funcion impresora de rsultado
}

void ansImpresion(double timeSpent, int i) {
  printf("\n\n");
  printf("\tTiempo de ejecucion: %f\n", timeSpent);
  printf("\tCantidad de iteraciones: %d\n", i);
}
```

Par

Peor de los casos (n = 8)	2	5	8	10
Tiempo de ejecución	106µS	224µS	287µS	2µS
# de iteraciones	12	28	28	0
Mejor de los casos (n = 8)				
Tiempo de ejecución	4µS	3µS	3µS	3µS
# de iteraciones	0	0	0	0

Impar

Peor de los casos (n = 9)	2	5	8	10
Tiempo de ejecución	94μS	367μS	307μS	3μS
# de iteraciones	8	36	36	0
Mejor de los casos (n = 9)				
Tiempo de ejecución	4μS	4μS	3μS	2μS
# de iteraciones	0	0	0	0

Conclusión

Utilizando el análisis empírico generamos un estimado de tiempos de ejecución, utilizando tres ciclos for anidados implementamos el algoritmo. Se logra observar que en el caso donde saltó (en el código se representa como middle) se divide entre 2 ambos arreglos(par e impar) en el peor de los casos representan un mejor tiempo de ejecución. Mientras se incremente el valor en el cual el salto se divide el tiempo de ejecución tiende aumentar, el divisor no debe exceder el tamaño del arreglo si no el algoritmo no itera ya que se genera un valor de 0 que es asignado a salto. En tiempos de ejecución promedio en el peor de los casos, los arreglos par obtuvieron mejor tiempo.