**LABORATORIO DE PROGRAMACIÓN**

**Prof.: Hugo Araya Carrasco**

Para este laboratorio considere los códigos siguientes que implementan 3 algoritmos de ordenamiento.

**Bubble Sort**

int array[N];

int i,j,aux;

for(i=0;i<N-1;i++){

for(j=0;j<N-i-1;j++){

if(array[j+1]<array[j]){

aux=array[j+1];

array[j+1]=array[j];

array[j]=aux;

}

}

}

**Shell Sort**

int array[N];

for(salto=N/2;salto!=0;salto = salto/2){

cambios = 1;

while (cambios != 0){

cambios=0;

for(i=salto;i<N;i++){

if(array[i-salto]>array[i]){

aux=array[i];

array[i]=array[i-salto];

array[i-salto]=aux;

cambios++;

}

}

}

}

**Quick Sort**

#include<stdio.h>

#define N \_\_\_

void swap(int\* a, int\* b){

int t = \*a;

\*a = \*b;

\*b = t;

}

int particion (int arr[], int bajo, int alto){

int pivote, i, j;

pivote = arr[alto];

i = (bajo - 1);

for (j = bajo; j <= alto- 1; j++){

if (arr[j] <= pivote){

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[alto]);

return (i + 1);

}

void quickSort(int arr[], int bajo, int alto){

int pi;

if (bajo < alto){

pi = particion(arr, bajo, alto);

quickSort(arr, bajo, pi - 1);

quickSort(arr, pi + 1, alto);

}

}

int main(){

int arr[N], i;

int n = N;

quickSort(arr, 0, n-1);

return 0;

}

Usted debe generar 3 archivos de datos con 1.000.000 de elementos (conteniendo sólo números de a lo más 6 dígitos), el primer archivo debe estar ordenado en forma ascendente, el segundo debe estar ordenado en forma descendente y el tercer archivo debe contener los datos en forma aleatoria. En cada uno de los casos de prueba usted debe utilizar los mismo elementos y escribir el tiempo empleado para ordenar dicho caso.

Los resultados deben ser presentados en forma de gráfico (considere construir tablas como las mostradas).

BUBBLE SORT

|  |  |  |  |
| --- | --- | --- | --- |
| S.O. |  |  |  |
| Algoritmo | Bubble Sort |  |  |
|  |  |  |  |
| n | Ordenado | Inverso | Aleatorio |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 101000 |  |  |  |
| 150000 |  |  |  |
| 400000 |  |  |  |
| 500000 |  |  |  |
| 600000 |  |  |  |
| 700000 |  |  |  |
| 800000 |  |  |  |
| 900000 |  |  |  |
| 1000000 |  |  |  |

Shell Sort

|  |  |  |  |
| --- | --- | --- | --- |
| S.O. |  |  |  |
| Algoritmo | Shell Sort |  |  |
|  |  |  |  |
| n | Ordenado | Inverso | Aleatorio |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 200000 |  |  |  |
| 300000 |  |  |  |
| 400000 |  |  |  |
| 500000 |  |  |  |
| 600000 |  |  |  |
| 700000 |  |  |  |
| 800000 |  |  |  |
| 900000 |  |  |  |
| 1000000 |  |  |  |

Quick Sort

|  |  |  |  |
| --- | --- | --- | --- |
| S.O. |  |  |  |
| Algoritmo | Quick Sort |  |  |
|  |  |  |  |
| n | Ordenado | Inverso | Aleatorio |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 200000 |  |  |  |
| 300000 |  |  |  |
| 400000 |  |  |  |
| 500000 |  |  |  |
| 600000 |  |  |  |
| 700000 |  |  |  |
| 800000 |  |  |  |
| 900000 |  |  |  |
| 1000000 |  |  |  |

Para medir el tiempo de ejecución del programa puede utilizar el siguiente código como ejemplo:

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

int main(){

time\_t t\_1,t\_2;

t\_1=time(NULL);

**// Bloque a medir el tiempo**

t\_2=time(NULL);

printf("\n\nEstadistica del programa: Ordenamiento .... ");

printf("\n\nTiempo 1 = %d",t\_1);

printf("\n\nTiempo 2 = %d",t\_2);

printf("\n\nTiempo Total = %lf\n",difftime(t\_2,t\_1));

return 0;

}

Para generar un conjunto de 1.000.000 de datos aleatorios utilice el siguiente programa llamado genera\_aleatorio.c, después de compilado debe ejecutarlo desde la línea de comando de la forma siguiente genera\_aleatorio > datos\_aleatorios.

#include <stdio.h>

#include <time.h>

int main(){

int numero,i;

srand(time(0));

numero = 1000000

printf("%d\n",numero);

for (i=0; i< numero; i++)

printf("%d\n",rand()%100000);

return 0;

}

Para generar un conjunto de 1.000.000 de datos ordenados de menor a mayor, utilice el siguiente programa llamado genera\_ordenado.c, después de compilado debe ejecutarlo desde la línea de comando de la forma siguiente genera\_ordenado > datos\_ordenados.

#include <stdio.h>

#include <time.h>

int main(){

int numero,i;

srand(time(0));

numero=1000000;

printf("%d\n",numero);

for (i=0; i< numero; i++)

printf("%d\n",i);

return 0;

}

Para generar un conjunto de 1.000.000 de datos ordenados de mayor a menor, utilice el siguiente programa llamado genera\_invertido.c, después de compilado debe ejecutarlo desde la línea de comando de la forma siguiente genera\_invertido > datos\_invertidos.

#include <stdio.h>

#include <time.h>

int main(){

int numero,i;

srand(time(0));

numero=1000000;

printf("%d\n",numero);

for (i=numero; i>0; i--)

printf("%d\n",i);

return 0;

}