IC3002 Análisis de Algoritmos

Proyecto corto 1

Objetivo General

Familiarizar al estudiante con la tarea de analizar algoritmos, tanto en su aspecto teórico, como en su forma empírica.

Objetivo específico

Juzgar el desempeño del algoritmo de ordenamiento por inserción, haciendo observaciones empíricas mediante programas, para contrastarlas posteriormente con lo esperado según la teoría.

Descripción del proyecto

Este proyecto consta de 4 etapas

Etapa 1

Escribir programas en C y Python que generen un vector aleatorio de tamaño n, de valores enteros entre 1 y n, mediante muestreo con reposición y distribución discreta uniforme de los valores. Luego los programas ordenarán el vector utilizando el algoritmo de ordenamiento por inserción visto en clase. Se deberá registrar el tiempo consumido por el programa para llevar a cabo el ordenamiento (véase el apéndice). Se deberá correr el programa al menos seis veces, anotando el tiempo que tardó cada vez.

La tarea anterior se deberá efectuar para los siguientes valores de n: 1000, 5000, 10000, 20000, 100000.

Etapa 2

Lo mismo que la etapa 1, excepto que los programas generan el vector ya ordenado.

Etapa 3

Lo mismo que la etapa 1, excepto que el vector inicial se genera ordenado de mayor a menor.

Etapa 4 (Sumarizar los datos)

Hay que tomar todos los datos recogidos por los experimentos y sumarizarlos conjuntamente. Se admite el uso de programas sencillos de manejo de datos, como Excel.

Los datos se analizarán tomando en cuenta cada etapa por separado.

Para cada grupo de datos se tomará el promedio simple de los datos de cada experimento, se calculará el polinomio interpolador de Lagrange correspondiente y se graficará junto con los datos.

Los datos de los experimentos figurarán como gráficos de barras, el polinomio interpolante será un gráfico de línea.

Interface con el usuario.

Python: usar tkinter.

C: usar GTK+.

Detalles de elaboración

- El proyecto es estrictamente individual. La copia dejará una nota de cero en los trabajos que se detecten como tales.
- Fecha de entrega: Jueves 4 de Agosto de 2016, hasta las 11:00pm, enviar todo el material al correo avargas@itcr.ac.cr

Apéndice: La medición del tiempo en Python y C.

Python. Se puede usar la función now() del módulo datetime, como lo muestra el siguiente ejemplo:

```
import datetime

now = datetime.datetime.now()

print
print "Current date and time using str method of datetime object:"
print str(now)

print
print "Current date and time using instance attributes:"
print "Current year: %d" % now.year
print "Current month: %d" % now.month
print "Current day: %d" % now.day
print "Current hour: %d" % now.hour
print "Current minute: %d" % now.minute
print "Current second: %d" % now.second
print "Current microsecond: %d" % now.microsecond

print
print "Current date and time using strftime:"
print now.strftime("%Y-%m-%d %H:%M")
```

C. Se puede usar al efecto la función clock(), que mide el tiempo en ticks de reloj transcurridos desde que inició la ejecución del programa, como lo muestra el siguiente ejemplo:

```
#include <time.h>
#include <stdio.h>
```

```
int main()
{
   clock_t start_t, end_t, total_t;
   int i;

   start_t = clock();
   printf("Starting of the program, start_t = %ld\n", start_t);

   printf("Going to scan a big loop, start_t = %ld\n", start_t);
   for(i=0; i< 10000000; i++)
   {
   }
   end_t = clock();
   printf("End of the big loop, end_t = %ld\n", end_t);

   total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
   printf("Total time taken by CPU: %f\n", total_t  );
   printf("Exiting of the program...\n");

   return(0);
}
```