**PRACTICA Nro. 1: COMPLEJIDAD DE ALGORITMOS**

**Utilizando el software Concorde con un timer de 60 segundos, llene el cuadro siguiente**

<http://www.math.uwaterloo.ca/tsp/concorde/downloads/downloads.htm>

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instancias (Nro. De ciudades)** | **Método Exacto (Busca el óptimo)**  **cutting-plane method** | | | **Método: Heurísticas**  **(Buscan una buena solución en un tiempo razonable)** | | |
| **Greedy** | **Nearest neighbor** | **Random** |
| **Optimo**  **(Si/no)** | **Valor objetivo** | **Tiempo** | **Valor objetivo** | **Valor objetivo** | **Valor objetivo** |
| **50** | **SI** | **560** | **0.03** | **633** | **757** | **2603** |
| **100** | **SI** | **806** | **0.08** | **1019** | **1099** | **5394** |
| **300** | **SI** | **1302** | **1.92** | **1561** | **1574** | **15457** |
| **400** | **SI** | **1534** | **0.8** | **1811** | **1832** | **21102** |
| **500** | **SI** | **1674** | **0.9** | **1961** | **2140** | **25606** |
| **600** | **SI** | **1784** | **29** | **2119** | **2190** | **31177** |
| **700** | **SI** | **1921** | **25** | **2342** | **2479** | **37300** |
| **800** | **NO** | **2454** | **60** | **2361** | **2618** | **41631** |
| **900** | **NO** | **2174** | **60** | **2629** | **2752** | **46942** |
| **1000** | **SI** | **2258** | **36** | **2576** | **2909** | **49178** |
| **1200** | **NO** | **2503** | **60** | **2936** | **3263** | **63657** |
| **1400** | **NO** | **2671** | **60** | **3012** | **3494** | **74820** |
| **1600** | **NO** | **2886** | **60** | **3323** | **3721** | **82758** |
| **1800** | **NO** | **2994** | **60** | **3552** | **3872** | **95964** |
| **2000** | **NO** | **3179** | **60** | **3685** | **4070** | **102856** |
| **2500** | **NO** | **4001** | **60** | **4204** | **4467** | **128652** |
| **3000** | **NO** | **4132** | **60** | **4565** | **4939** | **158171** |
| **4000** | **NO** | **5001** | **60** | **5234** | **5819** | **207872** |
| **5000** | **NO** | **5654** | **60** | **5875** | **6388** | **261288** |
| **6000** | **NO** | **5346** | **60** | **6380** | **6991** | **312456** |

1. Para el método exacto haga una gráfica Instancias x Tiempo. Discutir la gráfica.
2. Haga un solo gráfico Instancias x Valor objetivo los cuatro métodos. Discutir la gráfica

**PRACTICA Nro. 2: COMPLEJIDAD DE ALGORITMOS**

**Programe los métodos de ordenamiento:**

**a) Método de la burbuja**

**b) Método de ordenamiento por** **inserción**

**Con un timer máximo de 120 segundos, genere listas de números en forma aleatoria y llene el cuadro siguiente**

|  |  |  |
| --- | --- | --- |
| **Instancias (Lista de números)** | **Tiempo de ordenamiento** | |
| **Método de la burbuja** | **Método de ordenamiento por** **inserción** |
| **100** |  |  |
| **200** |  |  |
| **600** |  |  |
| **1000** |  |  |
| **2000** |  |  |
| **4000** |  |  |
| **6000** |  |  |
| **8000** |  |  |
| **10000** |  |  |
| **20000** |  |  |
| **30000** |  |  |
| **40000** |  |  |
| **50000** |  |  |
| **10000** |  |  |

Haga un solo gráfico Instancias x Tiempo de ordenamiento para los dos métodos. Discutir la gráfica

Codigo en Python

Programa1.py

def ordenar\_burbuja(L):

for i in range(1,len(L)):

for j in range(0, len(L)-i):

if L[j] > L[j+1]:

temp=L[j]

L[j]=L[j+1]

L[j+1]=temp

return L

programa2.py

*def ordenar\_insertando(L):*

*for j in range(1,len(L)):*

*i=j-1*

*temp=L[j]*

*while i > -1 and L[i] > temp:*

*L[i+1]=L[i]*

*i=i-1*

*L[i+1]=temp*

*return L*

Ejemplo de como generar la lista de números aleatorios

>>> G1=random.sample(range(10\*\*3),10\*\*3)

>>> G2=random.sample(range(10\*\*4),10\*\*4)

>>> G3=random.sample(range(10\*\*5),10\*\*5)