Closest Pair Lab

Daniel Gutierrez

Abstract

El proposito de este workshop es aplicar lo trabajado en clase en referencia a los algoritmos, la complejidad de dichos algoritmos y los distintos casos aprendidos. Ademas, al obtener los tiempos de ejecucion y numero de comparaciones, obtener conclusiones en base a ello.

Index Terms

IEEE, IEEEtran, journal, LaTeX, paper, template.

I. Introduction

EL siguiente documento proporciona el análisis de los resultados al experimentar con un algoritmo. En este caso se buscaba generar n coordenadas aleatorias y calcular la distancia minima entre los n puntos. Para realizar esto, se realizo un algoritmo denominadao ”fuerza bruta” y otro recursivo denominado ”divide y venceras”. El algoritmo ”fuerza bruta” consiste en comparar todos los elementos con el resto de elementos, mientras que ”divide y venceras” consiste en subdividir las coordenadas y comparar dentro de esas subdivisiones. Luego de tener esto, se buscaba encontrar las diferencias entre ambos algoritmos para asi sacar conclusiones. Para realizar esto, se utilizo Netbeans IDE 8.2 para el desarrollo de los algoritmos necesarios y además se uso Spyder para graficar los resultados obtenidos en Python con las librerías Numpy y Matplotlib.

Octubre 25, 2022

II. Pseudocodigo de Solucion

Calculo de la distancia entre dos puntos

$d \leftarrow Raiz(x1 - x2)^2 + (y1 - y2)^2)$

Algoritmo para realizar fuerza bruta

for $i = inicio$ hasta $fin - 1$ **do**
  for $j = i + 1$ hasta fin **do**
    if $dist < dmin$ **then**
      $dmin = dist$
    end if
  end for
end for

Algoritmo para separacion de arraylist y calculo de distancia minima

$pos \leftarrow longitud$ div 2
if n mod $2 = 0$ **then**
  $pos \leftarrow pos + 1$
end if
if $n > 3$ **then**
  Se divide la lista desde 0 hasta la mitad-1 y se asigna a $p1$
  Se divide la lista desde la mitad hasta n-1 y se asigna a $p2$
  $min \leftarrow (DAC(p1, r, d), DAC(p2, r, d))$
  $min \leftarrow (min, distancias(p1[n], p2[0]))$
else
  $min \leftarrow min(min, bruteforce(coordenadas, r, 1))$
end if

III. Solucion del Problema

Para la solucion del problema propuesto, se deben utilizar algunos metodos para realizar las operaciones pertinentes. En primera instancia, se va a utilizar el manejo de archivos para guardar informacion en relacion a los tiempos de ejecucion y numero de comparaciones de cada iteracion para asi graficar en python. Para el manejo de archivos se intenta crear un archivo con nombre y Si no es posible crear el archivo retorna el error. Para la creacion de coordenadas se creo una clase llamada punto la cual tiene como parametros la coordenada x y la coordenada y que tengra el punto en el plano cartesiano, y de esa manera se facilita el manejo de las coordenadas al utilizar arraylist de tipo punto. Ademas se utiliza una subrutina

llamada ”bruteforce” la cual realiza el proceso de comparacion entre todos los puntos para obtener distancia minima (este algoritmo se encuentra en la seccion anterior).En sintesis, la subrutina ”bruteforce” realiza el calculo de la distancia de todas las coordenadas entre los puntos inicio y fin, finalmente retorna la distancia minima).Para realizar esto, la subrutina ”bruteforce” recorre todo el arraylist de puntos en ciclos anidados para asi comparar cada elemento con todos los elementos que le siguen al punto a comparar. Para el proceso de ”divide y venceras” se llama a una subrutina llamada ”DAC” la cual esta encargada de dividir el proceso de comparacion en dos partes iguales, se realiza una subdivision de las coordenadas cuando el tamano de coordenadas sea mayor a 3 y asi se realizan subdivisiones recursivas para unicamente comparar puntos que se encuentren cerca y no realizar comparaciones innecesarias, si el numero de coordenadas es menor o igual a 3 ahi se realiza el proceso de ”bruteforce” entre esas 3 coordenadas para asi reducir el numero de comparaciones. Esta subrutina se encuentra en la seccion anterior nombrada ”Algoritmo para separacion de arraylist y calculo de distancia minima”. En sintesis, la subrutina ”DAC” hace uso de la subrutina ”bruteforce” (anteriormente explicada) con la diferencia que se realiza en dos instrucciones distintas, en caso de que hayan mas de 3 coordenadas subdivide el arraylist en una sublista que contiene la mitad de las coordenadas y luego crea otra sublista con la segunda mitad y asi realiza divisiones recursivas, a medida que realiza las subdivisiones calcula la distancia minima entre las dos sublistas y luego la compara con la distancia entre los dos puntos medios para asi encontrar una distancia minima. Luego de eso, cuando la sublista tiene 3 o menos elementos, realiza fuerza bruta y encuentra la distancia minima real entre todos los puntos. Para la division del arraylist, se utiliza un metodo propio de Netbeans el cual crea sublistas.

IV. Metodo ”main”

El metodo ”main” es donde se llevan a cabo las operaciones tales como realizar el numero de repeticiones, la creacion de los archivos y el guardado de datos en archivos.Se realizan 10 repeticiones donde se varia el tamaño desde 4 hasta 1048576 y se maneja un arreglo para los tiempos de ejecucion y numero de comparaciones del ”divide y venceras”, ademas se manejan las mismas coordenadas. De esta manera se garantiza igualdad de condiciones y asi se calculan los tiempos de ejecucion de cada iteracion al igual que el numero de comparaciones realizadas en cada caso. Para la creacion de coordenadas aleatorios, se utiliza una subrutina que se encarga de generar ”n” numeros aleatorios y asi genera uno para la coordenada x y otro para la coordenada y, de igual forma el limite para el numero generado aumenta proporcional al numero de coordenadas, es decir, si se van a generar 100 elementos el numero maximo para una coordenada sera 100-1. Se utiliza el metodo random de java para asi garantizar que el experimento sea aleatorio en su totalidad.Por ultimo, se escribe en cada archivo los datos pertinentes de cada iteracion para cada caso para luego graficar en python con estos datos.

V. Script usado para graficar

Como se mencionó en la introducción, se utilizo Spyder el cual es un entorno de desarrollo de Python para graficar los resultados y así sacar conclusiones. El script consiste en extraer los datos del archivo de texto y guardar cada columna de datos en un arreglo para cada variable, luego grafica el modelo matematico esperado para asi comparar los resultados al graficar el numero de coordenadas contra el tiempo y el numero de comparaciones. Al momento de tener los resultados, se observo una gran diferencia en ambos casos con lo cual varia el modelo matematico, de esa manera se logra ver el comportamiento esperado de una mejor manera y asi se explica en la siguiente seccion.

VI. Conclusion

En base a los gráficos obtenidos (Apendices A y B), se puede ver que el mejor caso es el que se realiza con ”divide and conquer”, en donde el numero de comparaciones aumenta a menor razón y los tiempos de ejecución son menores, aun cuando el numero de coordenadas es muy grande como lo son 1048576 coordenadas se observa que el numero de comparaciones y tiempo de ejecución es mucho menor a los números que se observan en el otro caso, esto hace ver que en efetcto el ”divide and conquer” es mucho mas eficiente. El algoritmo ”brute force” se comporta aproximadamente de manera cuadratica n^2 mientras que el ”divide and conquer” se comporta de manera lineal n con lo cual se concluye que en efecto el rendimiento del ”divide and conquer” es mucho mas eficiente. La diferencia en complejidades se debe en que al recorrer dos ciclos anidados para comparar cada punto con el resto de puntos se hacen aproximadamente n^2 comparaciones por punto y n va decreciendo a medida que se comparan elementos mientras que en el ”divide and conquer” unicamente se realiza divide and conquer unicamente cuando el tamano de coordenadas de la sublista es menor o igual a 3 con el fin de reducir en gran cantidad el numero de comparaciones, asi se logro obtener que el numero de comparaciones fue alrededor de $\frac{n}{2}$.Aunque en las graficas no se logre apreciar, por medio de otro software para graficar se logra ver la gran diferencia a medida en que el tamaño ”n” aumenta. Por ultimo, se puede concluir que se podria mejorar la eficiencia del algoritmo ”divide and conquer” si se realizan mas divisiones recursivas cada vez que se divide el arreglo y de esa manera se realizan menos comparaciones, sin embargo se logra apreciar la gran diferencia con el metodo utilizado con lo que no se justifica optimizar para efectos de demostracion.

APPENDIX A
GRAFICA "BRUTEFORCE"

A continuacion se presenta la grafica de los resultados del caso "BruteForce"

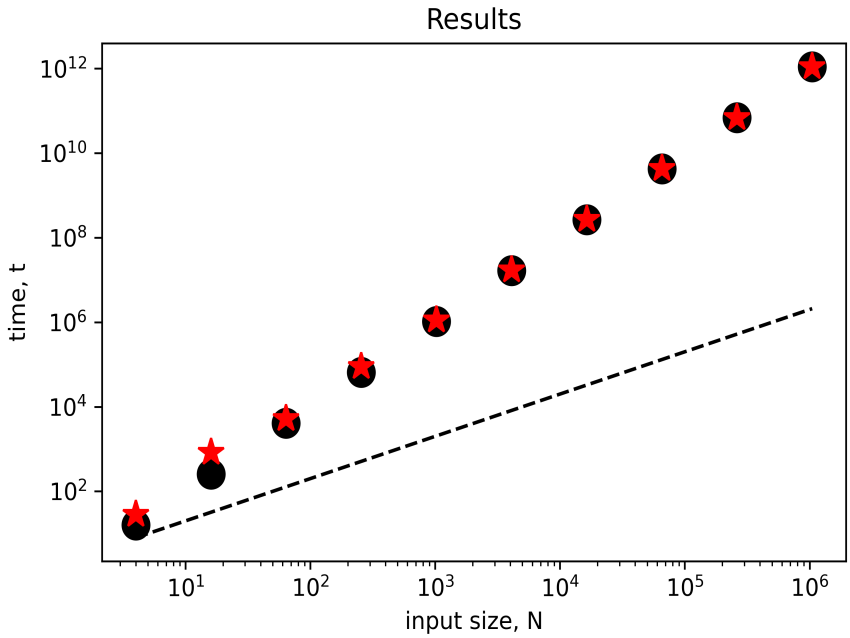

Fig. 1. Grafica Brute Force

APPENDIX B
GRAFICA "DIVIDE AND CONQUER"

A continuacion se presenta la grafica de los resultados del caso "Divide and Conquer".

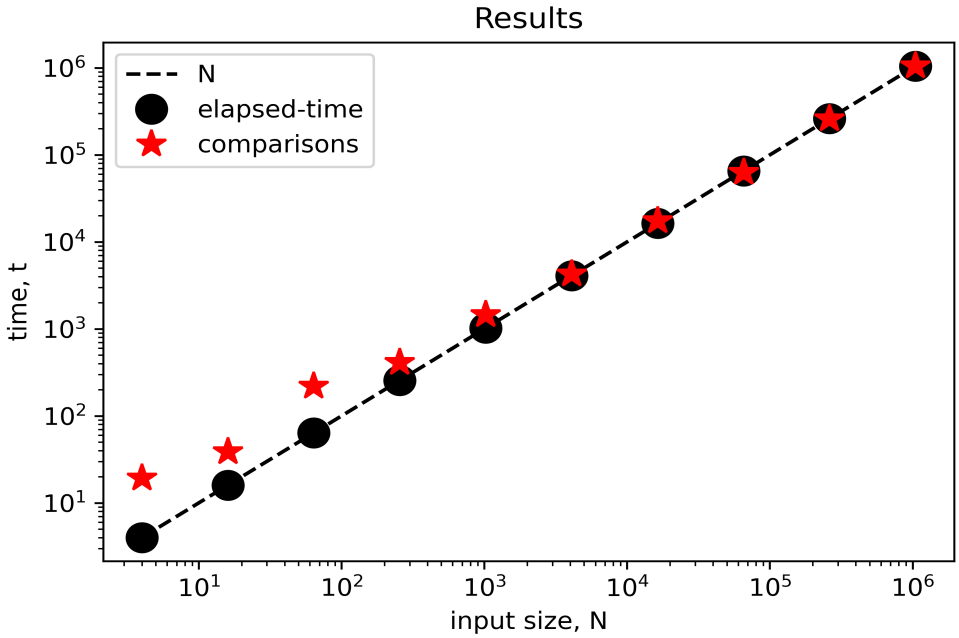

Fig. 2. Grafica Divide and Conquer