Analisis e Implementacion del Problema del par de puntos más cercanos utilizando los metodos de Fuerza bruta y Recursividad

Esteban Camilo Ramirez Pereira
Ingenieria de Sistemas
Universidad del Norte Barranquilla,Colombia
ecpereira@uninorte.edu.co

***Resumen*: Este documento tiene como objetivo realizar un análisis acerca de la eficiencia de dos algoritmos, uno que hace uso de iteraciones y otro que usa recursividad, cuya finalidad es encontrar el par de puntos más cercanos dentro de una lista de coordenadas. Para implementar estos algoritmos se creó un ArrayList de tamaño N=6 el cual tiene coordenadas creadas a partir de valores generados aleatoriamente. Al realizar estos algoritmos, que fueron ejecutados en el lenguaje de programación Java, y calcular su complejidad, se llegó a la conclusión que el algoritmo más eficiente es el implementado con fuerza bruta, con una complejidad $O(3+n^2)$**

Keywords—Algoritmo, Complejidad, Coordenadas, ArrayList

I. Definicion de problema

Dados N puntos en un espacio métrico bidimensional, se debe encontrar el par de puntos con la distancia más pequeña entre ellos[1].

Ejemplo:

N=3

Input: {[-7, 4], [2, 4], [2, 3]}

Output: La distancia mas corta es entre los puntos [2,3] y [2,4] con un valor de 1

II. Metodologia

Para realizar el siguiente trabajo, se realizó un programa en el lenguaje de programación Java, haciendo uso del IDE Netbeans. En este, se creó un ArrayList de tamaño N=6, cuyo contenido es vectores de tipo entero (int[]), los cuales representan cada uno un punto dentro de un espacio métrico bidimensional. El contenido de estos puntos es generado de manera aleatoria, en un rango de -6 a 6.

```
ArrayList<int[]> coordenadas = new ArrayList<int[]>();

for (int i = 0; i < 6; i++) {
    int x = -6 + (int)(Math.random() * ((6 - (-6)) + 1));
    int y = -6 + (int)(Math.random() * ((6 - (-6)) + 1));
    int[] arr = {x, y};
    coordenadas.add(arr);
}
```

Una vez se generó este ArrayList, se ordeno de forma ascendente, con el objetivo de facilitar la comparación entre las distintas coordenadas que este almacena.

```
Collections.sort(coordenadas, new Comparator<int[]>() {
 public int compare(int[] a, int[] b) {
      if(a[0] - b[0]==0) {
          return a[1]-b[1];
      }else
          return a[0]-b[0];
      }
});
```

Una vez ordenado el ArrayList, se ejecutarán dos algoritmos: uno con iteraciones y otro recursivo, que comparten el objetivo de encontrar la distancia más corta entre los puntos. Para hallar esta distancia se hace uso de la **Fórmula de la distancia entre dos puntos**[2], también conocida como **Teorema de Pitágoras.**

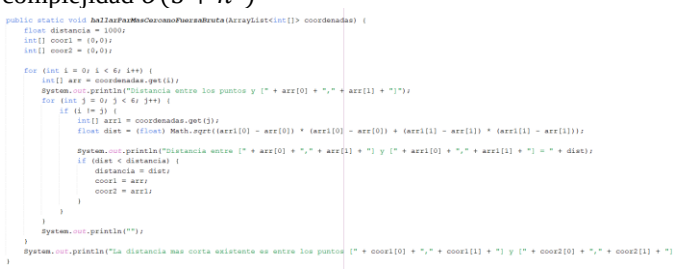
$$A(x_1, y_1)$$

$$B(x_2, y_2)$$

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

III. Resultados

El algoritmo realizado con fuerza bruta tiene una complejidad $O(3+n^2)$



El algoritmo realizado con recursividad posee una complejidad $O(nlogn)$

Independientemente de su implementación, ambos algoritmos arrojan los mismos resultados

```
------------------------Coordenadas generadas------------------------
[3,5]
[5,-5]
[6,-4]
[-2,0]
[-5,-1]
[-6,-1]

---------Coordenadas mas cercanas aplicando fuerza bruta-------------

La distancia mas corta existente es entre los puntos [-6,-1] y [-5,-1] con un valor de 1.0
---------Coordenadas mas cercanas aplicando recursividad------------

La distancia mas corta existente es entre los puntos [-6,-1] y [-5,-1] con un valor de 1.0
```

IV. Conclusiones

A partir de las complejidades calculadas para ambos algoritmos, se concluye que en el algoritmo en que se usan más iteraciones hay mayor eficacia, debido a su menor nivel de complejidad. También hay que decir que independiente de esto, los dos algoritmos son óptimos y rápidos para ser considerados eficientes, debido a la baja cantidad de datos.

Referencias

[1] Geometría analítica. (2021, nov 10) “Fórmula de la distancia entre dos puntos (geometría)”. [Online]. Disponible en: https://www.geometriaanalitica.info/formula-de-la-distancia-entre-dos-puntos-geometria-ejemplos-y-ejercicios-resueltos/

[2] Wikipedia. (2022, sept 8) “”. [Online]. Disponible en: https://es.wikipedia.org/wiki/Problema_del_par_de_puntos_más_cercanos