UNIVERSIDAD NACIONAL

SAN AGUSTIN DE AREQUIPA

Facultad de Ingeniería, Producción y Servicios

Escuela Profesional de Ciencia de la Computación

Laboratorio 1 - La Maldición de la Dimensionalidad

Presentado por:

Parizaca Mozo, Paul Antony

CUI:

20210686

Github:

https://github.com/PaulParizacaMozo/Estructuras-De-Datos-Avanzadas

Curso:

Estructuras de Datos Avanzadas

Docente:

Rosa Paccotacya Yanque

Arequipa, Perú

2023

Descripción del laboratorio

● Generar 100 puntos aleatorios entre 0 y 1 de dimensión d
● Calcular la distancia entre todos los pares de puntos (Distancia Euclidiana) (Hint 4950 distancias)
● Generar un histograma (pueden usar Python) de las distancias obtenidas para cada dimensión

Introducción

En el presente informe se va a generar 100 puntos aleatorios entre 0 y 1 en distintas dimensiones, específicamente en las siguientes dimensiones: 10, 50, 100, 500, 1000, 2000, 5000. Luego se va a calcular la distancia entre todos los pares ordenados generados y una vez obtengamos esas distancias generamos un histograma para analizar la tendencia de las distancias obtenidas.

Metodología

Para la generación de los puntos aleatorios se usará el lenguaje de programación c++, en el cual crearemos un programa para la generación de puntos, el programa consta de una clase Espacio la cual tiene como finalidad crear dos archivos de texto: data.txt y distancias.txt.
Una vez tengamos los dos archivos creados se ejecutará automáticamente un programa en python el cual usará como entrada los dos archivos para la generación de los histogramas haciendo uso de la librería Matplotlib.
El codigo de dichos programas se encuentra en el siguiente repositorio de Github:
https://github.com/PaulParizacaMozo/Estructuras-De-Datos-Avanzadas

Ejemplo de ejecución del programa:

-Ejecutamos el siguiente comando en consola donde se encuentren nuestros programas

```
❯ g++ -g -o eject main.cpp -pedantic && ./eject
```

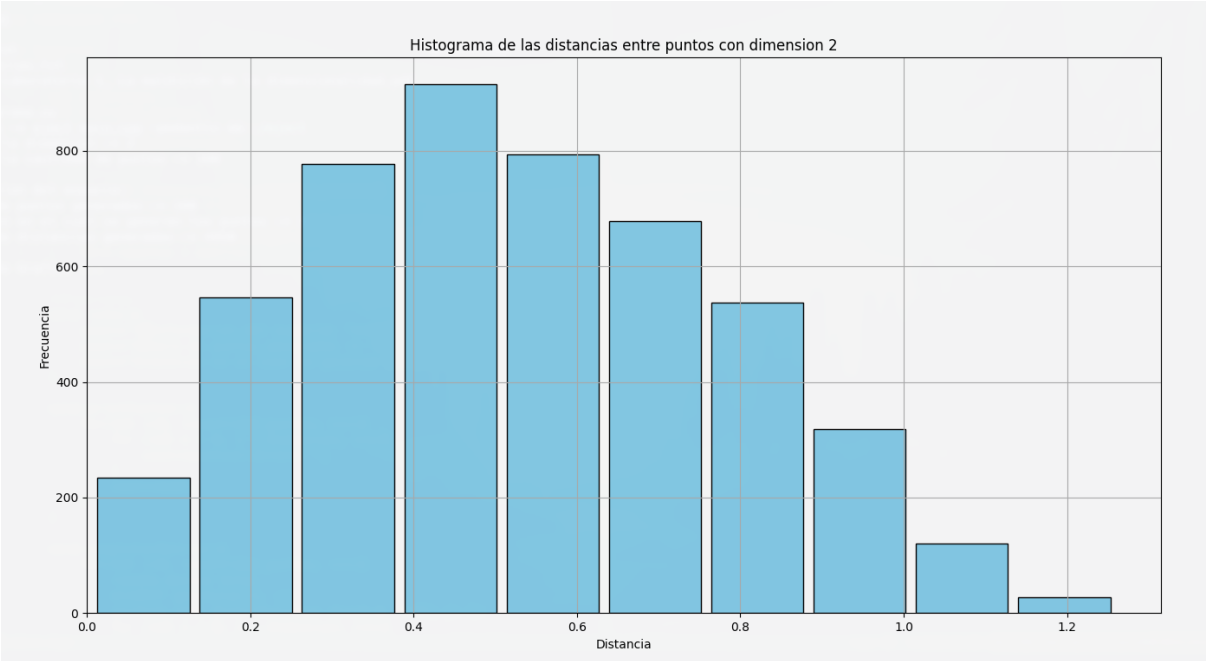
-Nos pedirá ingresar la dimensión en la cual se quiere trabajar y la cantidad de puntos a generar

```
❯ g++ -g -o eject main.cpp -pedantic && ./eject
Ingrese la dimension -> 2
Ingrese la cantidad de puntos -> 100

Informacion del espacio:
Numero de puntos generados -> 100
Dimension en el cual se generan los puntos -> 2
Numero de distancias generadas -> 4950

Iniciando Grafica ...
```

-Nos mostrará la gráfica

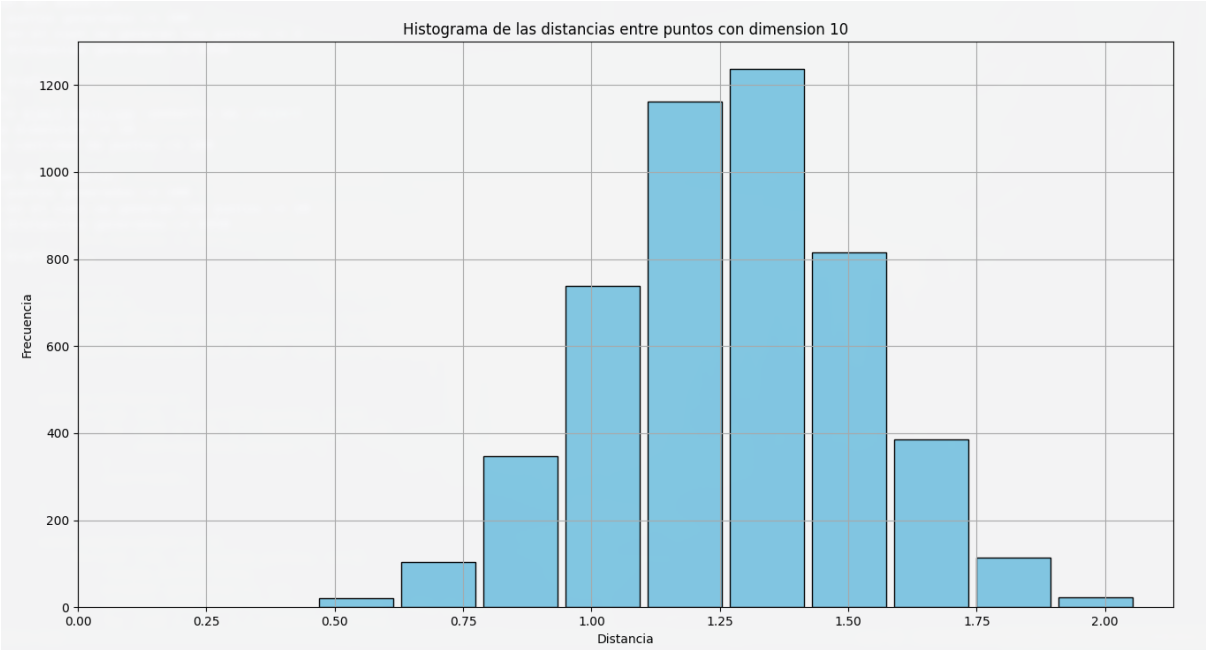


-Y con eso culminará la ejecución del programa

```
Finalizado.
 ~/U/6semestre/E/Lab1 on  main !4 took 2m 1s
```

Resultados

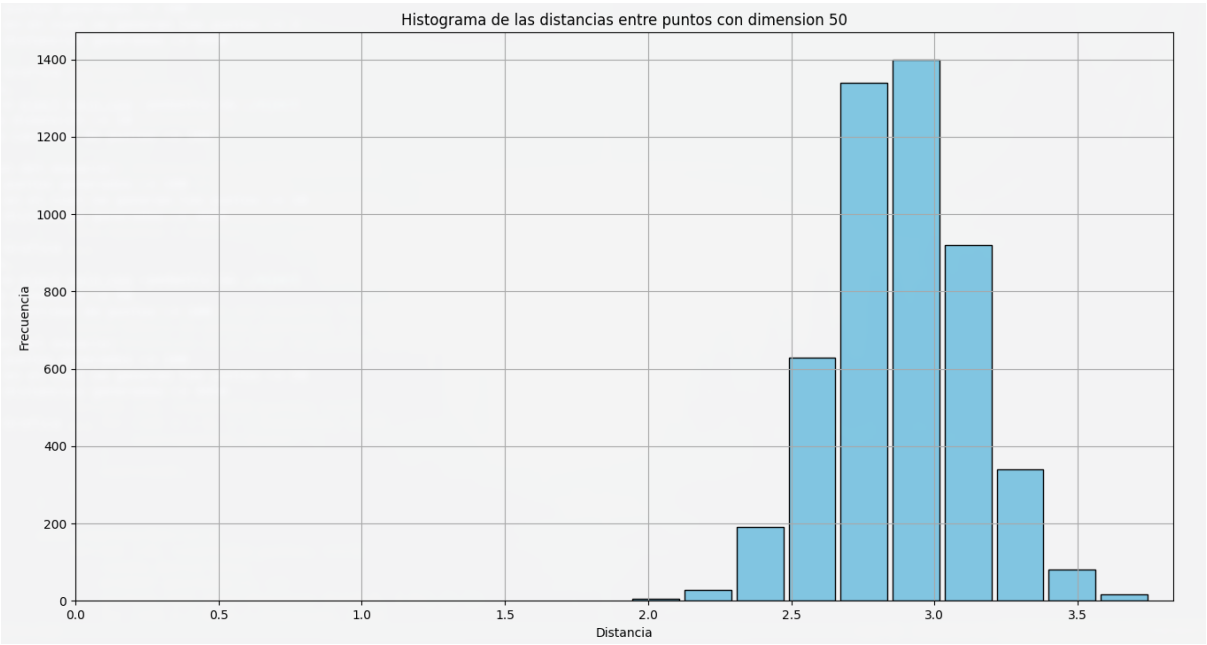
-Dimensión 10



Eje x: Los valores de las distancias se encuentran entre 0.25 y 2.25.

Eje y: Teniendo como frecuencia máxima un valor menor a 1300 y superior a 1200.

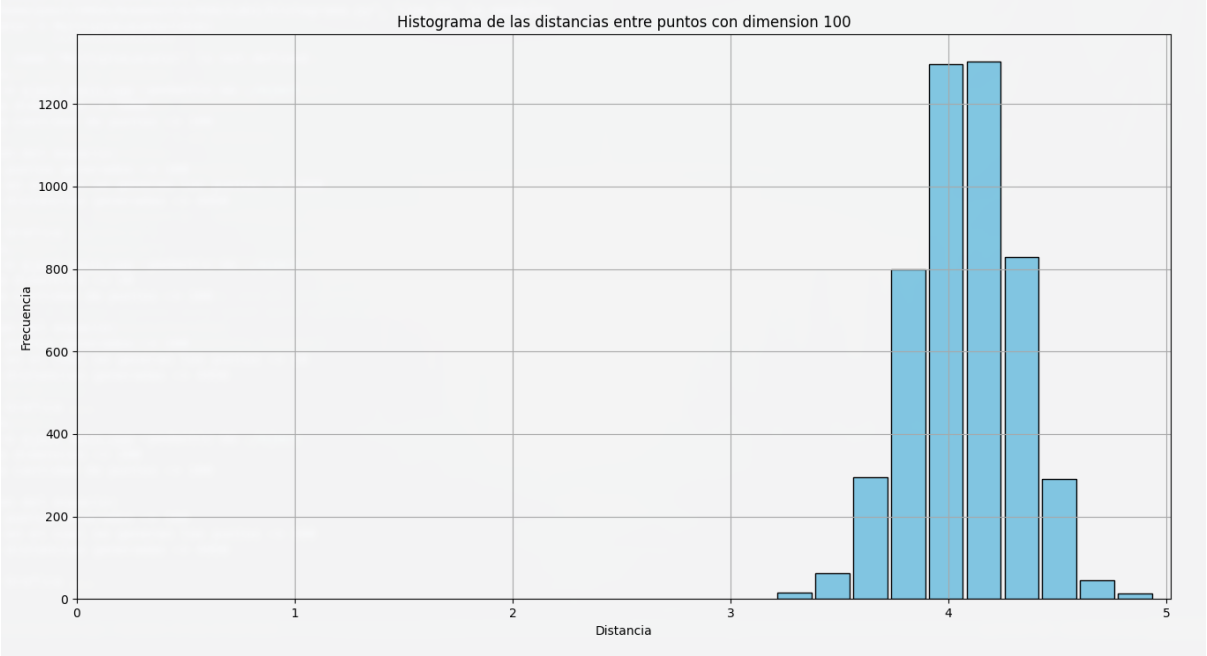
-Dimensión 50



Eje x: Los valores de las distancias se encuentran entre 1.5 y 4.0.

Eje y: Teniendo como frecuencia máxima un valor de 1400.

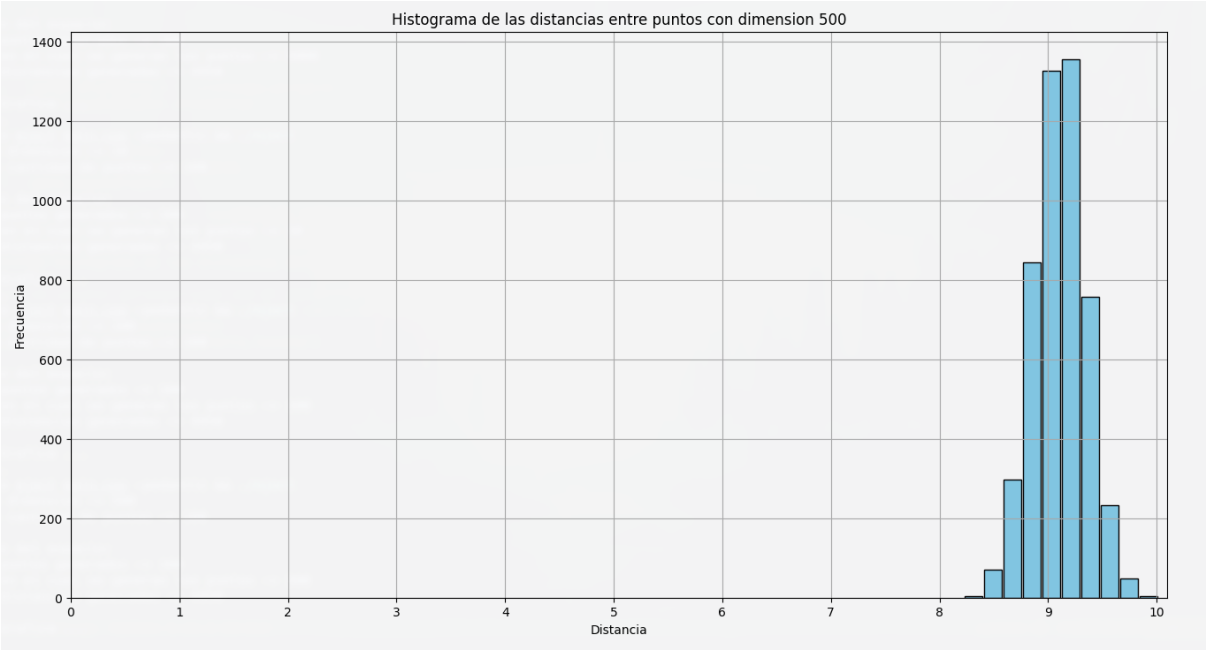
-Dimensión 100



Eje x: Los valores de las distancias se encuentran entre 3.0 y 5.0.

Eje y: Teniendo como frecuencia máxima un valor menor a 1300 y superior a 1200.

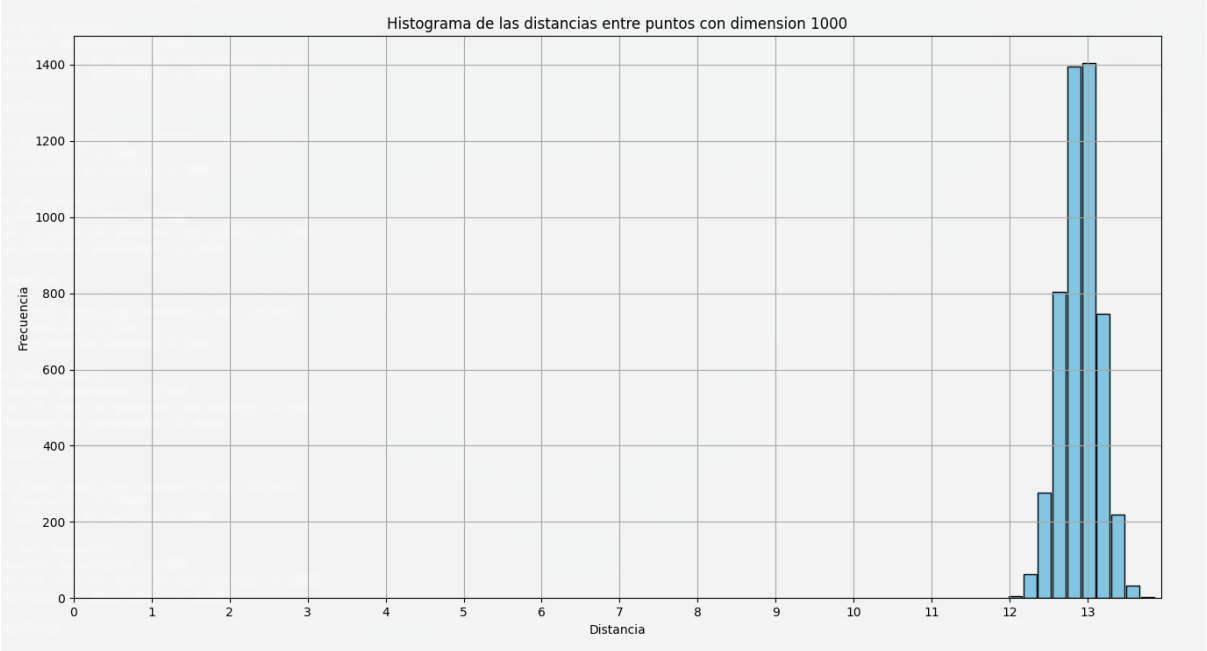
-Dimensión 500



Eje x: Los valores de las distancias se encuentran entre 8.0 y 10.0.

Eje y: Teniendo como frecuencia máxima un valor menor a 1400 y superior a 1300.

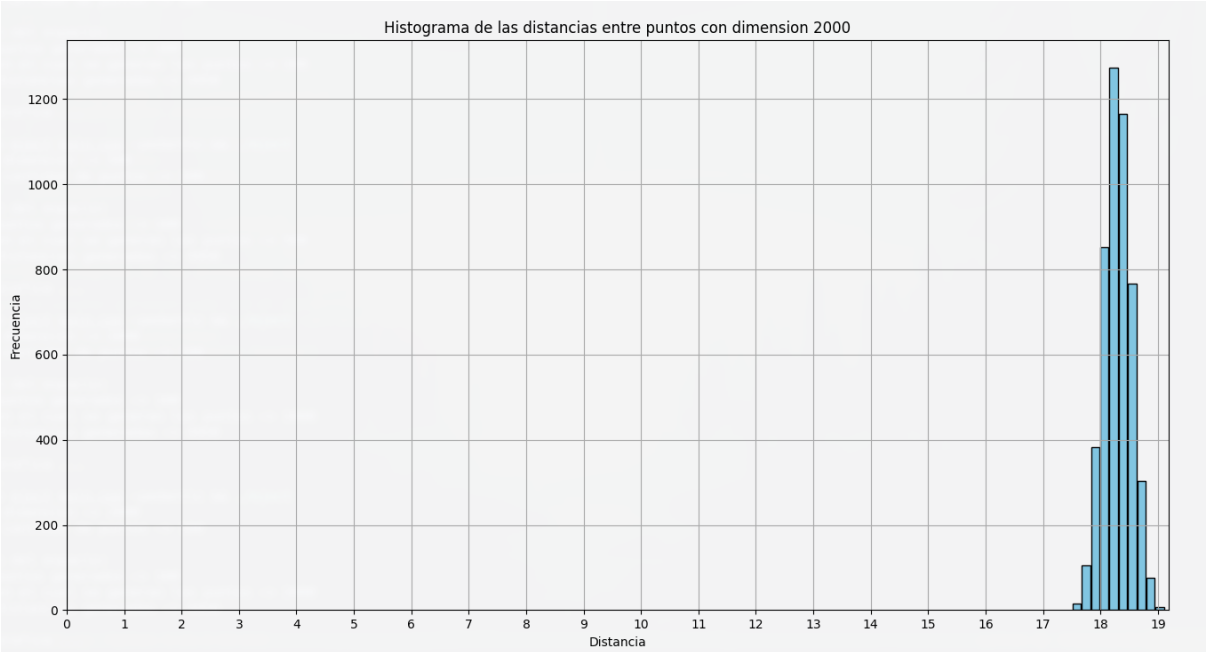
-Dimensión 1000



Eje x: Los valores de las distancias se encuentran entre 12.0 y 14.0.

Eje y: Teniendo como frecuencia máxima un valor que ronda los 1400.

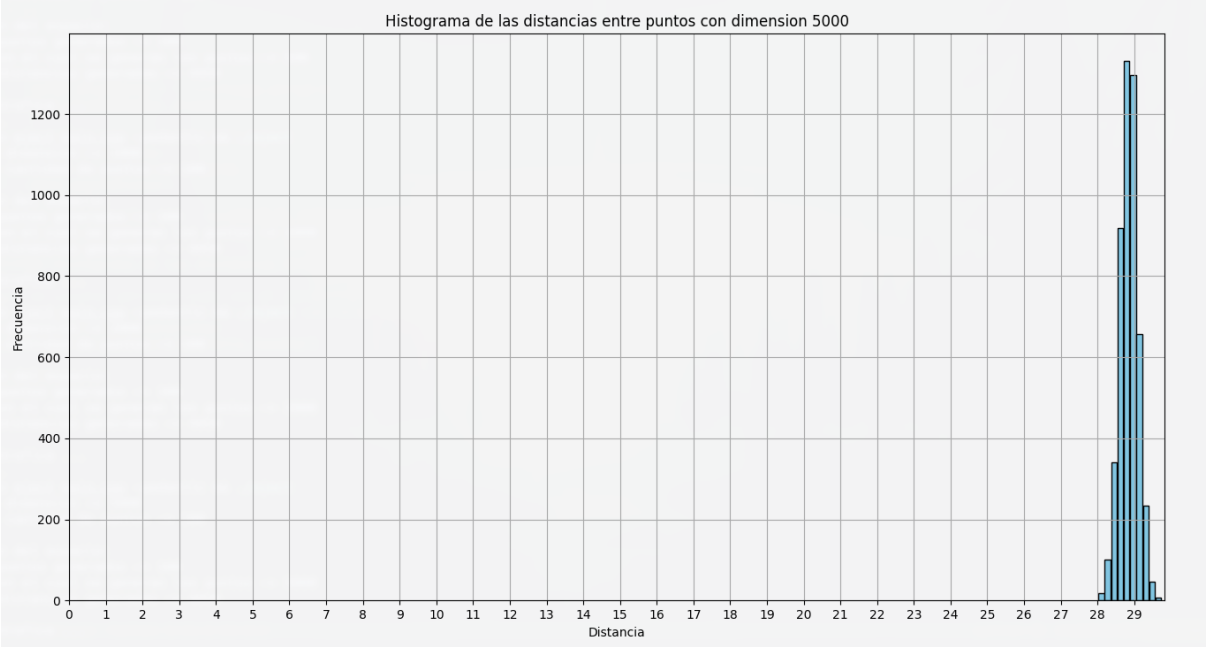
-Dimensión 2000



Eje x: Los valores de las distancias se encuentran entre 17.5 y 19.5.

Eje y: Teniendo como frecuencia máxima un valor menor a 1300 y superior a 1200.

-Dimensión 5000



Eje x: Los valores de las distancias se encuentran entre 28.0 y 30.0.
Eje y: Teniendo como frecuencia máxima un valor menor a 1300 y superior a 1200.

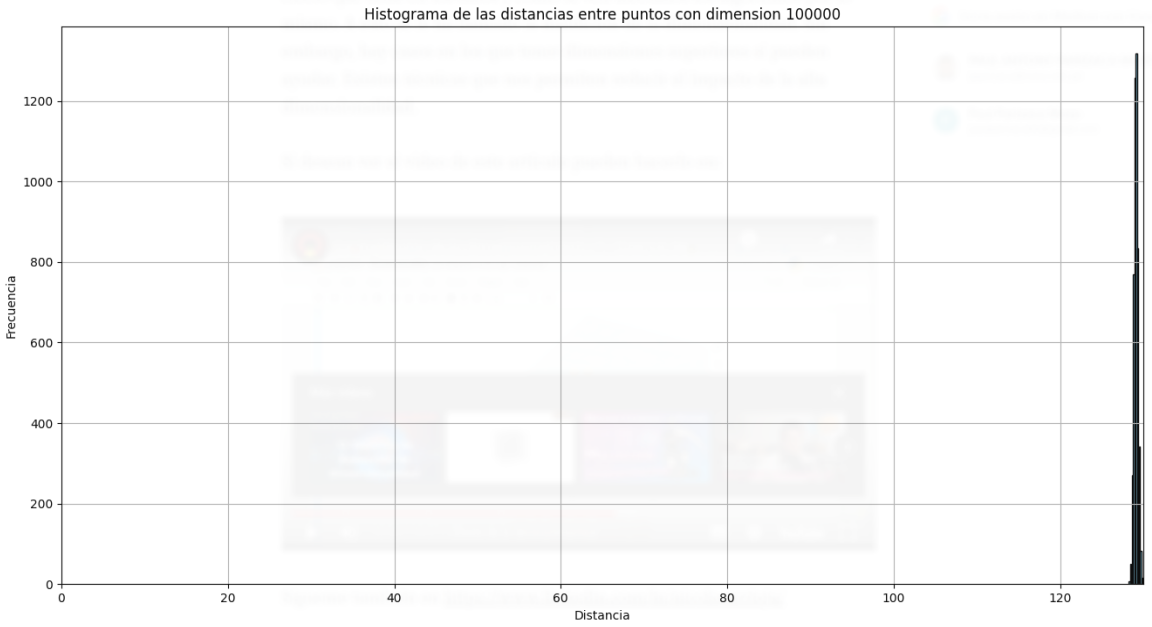
Análisis

Teniendo en cuenta los histogramas de dimensiones 10,50,100,500,1000,2000,5000 respectivamente podemos notar que hay un cambio significativo en las distancias de un punto con otro punto del mismo espacio que varía según se incrementa el número de dimensiones para el espacio.

Haciendo una comparación entre las distancias de dimensión 10 y las distancias de dimensión 1000 podemos ver que las distancias de dimensión 10 oscilan entre 0.25 y 2.25 mientras que las distancias de la dimensión 1000 oscilan entre 12.0 y 14.0. Con esto podemos ver que las distancias entre puntos van en incremento. Y comparando más aún las distancia de la dimensión 10 con las distancias de la dimensión 5000 en la cual las distancias entre un punto y otro oscilan entre 28.0 y 30.0 se nota el incremento significativo entre una distancia de una dimensión baja comparado con una dimensión alta.

Por ello también podemos decir que las distancias en las dimensiones bajas son representativas, que queremos decir con esto, que las distancias entre puntos pueden proporcionar información útil sobre cómo se relacionan los datos entre sí. Y a medida que se va creciendo el espacio en dimensiones las distancias entre los puntos se empieza a expandir, a crecer y por ello se van volviendo menos representativas y con un incremento

aún mayor de dimensiones se va perdiendo en su totalidad esta característica en la siguiente imagen podemos ver las distancias con una dimensión de 100000.



Viendo el histograma las distancias son muy similares entre ellas, se puede decir que entre un punto y otro las distancias son equidistantes.

Conclusión

-La alta dimensionalidad puede ser un problema y es necesario entender el efecto que tiene en los datos y como se ven afectados los algoritmos por lo mismo.[3]

-Las distancias entre puntos en dimensiones bajas son más representativas,significativas.

-A medida que las dimensiones incrementan en el espacio dejan de ser representativas, significativas.

-En un espacio con dimensiones altas tienden a ser similares, indistinguibles en algunos casos y con esto perder el cómo se relacionan las distancias entre puntos la una con la otra.

Código

Para el siguiente programa se trabajó con c++ y python para la elaboración del programa

–main.cpp

En el cual tenemos una clase Espacio

```cpp
#include <random>
#include <vector>
#include <math.h>
#include <fstream>
using namespace std;

class Espacio{
    public:
        //Atributos
        int dimension;
        int cantidad_puntos;
        vector<vector<double>>puntos;
        vector<double>distancias;

        //Constructor
        Espacio(int dimension,int cantidad_puntos){
            this->dimension = dimension;
            this->cantidad_puntos = cantidad_puntos;
            vector<std::vector<double>> matriz(cantidad_puntos,vector<double>(dimension)); n: value: n:
            this->puntos = matriz;
        }
```

La cual tiene atributos como dimensión, cantidad_puntos, un vector de vectores de tipo double puntos y un vector de tipo double para almacenar las distancias generadas.

La clase tiene funciones

info(): Muestra información del espacio(dimensión y cantidad de puntos)

```cpp
        //Funciones
        void info(){
            cout<<"\nInformacion del espacio: "<<endl;
            cout<<"Numero de puntos generados -> "<<puntos.size()<<endl;
            cout<<"Dimension en el cual se generan los puntos -> "<<puntos[0].size()<<endl;
        }
```

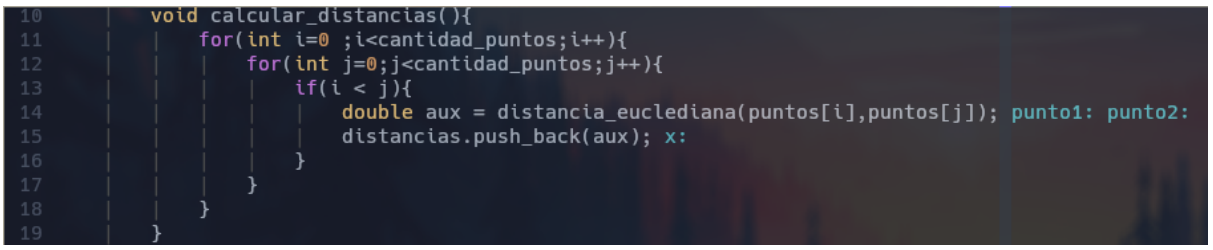
completarMatriz(): Completa la matriz con números generados aleatoriamente entre 0 y 1

```cpp
        void completarMatriz(){
            for(int i=0; i<cantidad_puntos; i++){
                random_device hola;
                mt19937 gen(hola()); sd:
                uniform_real_distribution<> dis(0.0, 1.0); a: b:
                for (int n = 0; n < dimension; ++n){
                    puntos[i][n] = dis(gen);
                }
            }
        }
```

distancia euclidiana(): Algoritmo para calcular la distancia euclidiana

```cpp
        double distancia_euclediana(vector<double>punto1,vector<double>punto2){
            double distancia = 0;
            for (int i=0 ;i<dimension;i++) {
                distancia += pow(punto2[i]-punto1[i],2); x: y:
            }
            distancia = sqrt(distancia); x:
            return distancia;
        }
```

calcular_distancias(): Calcula todas las distancias de todos los puntos con los demas puntos



```
void calcular_distancias(){
    for(int i=0 ;i<cantidad_puntos;i++){
        for(int j=0;j<cantidad_puntos;j++){
            if(i < j){
                double aux = distancia_eucledianа(puntos[i],puntos[j]); punto1: punto2:
                distancias.push_back(aux); x:
            }
        }
    }
}
```

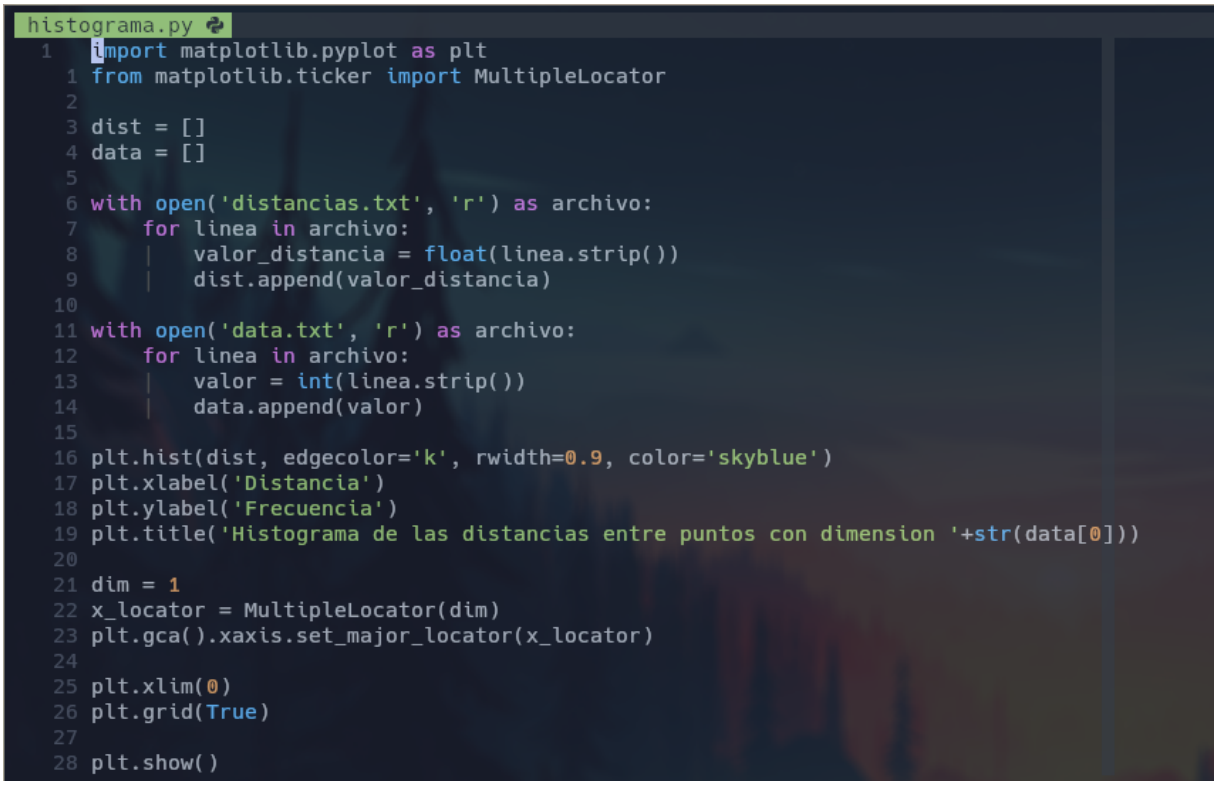
escribir_archivos(): Escribe dos archivos: data.txt y distancias.txt que serán usados por el programa escrito en python para la generación de histogramas.

```
void escribir_archivos(){
    ofstream file;
    file.open("distancias.txt"); s:
    for(auto vec : distancias){ : double
        file<<vec<<'\n';
    }
    file.close();
    file.open("data.txt"); s:
    file<<dimension<<'\n';
    file<<cantidad_puntos<<'\n';
    file.close();
}
};
```

main(): Solicita al usuario ingresar la dimensión y la cantidad de puntos para el espacio y con esa información crear un objeto de la clase espacio y llamar a las funciones necesarias para mostrar el histograma.

```
int main()
{
    int dim,cant;
    cout<<"Ingrese la dimension -> ";
    cin>>dim;
    cout<<"Ingrese la cantidad de puntos -> ";
    cin>>cant;
    Espacio a(dim,cant); dimension: cantidad_puntos:
    a.info();
    a.completarMatriz();
    a.calcular_distancias();
    cout<<"Numero de distancias generadas -> "<<a.distancias.size()<<endl<<endl;
    cout<<"Iniciando Grafica ..."<<endl;
    a.escribir_archivos();
    cin.get();
    system("python histograma.py"); command:
    cout<<"Finalizado."<<endl;
    return 0;
}
```

-Histograma.py



```python
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator

dist = []
data = []

with open('distancias.txt', 'r') as archivo:
    for linea in archivo:
        valor_distancia = float(linea.strip())
        dist.append(valor_distancia)

with open('data.txt', 'r') as archivo:
    for linea in archivo:
        valor = int(linea.strip())
        data.append(valor)

plt.hist(dist, edgecolor='k', rwidth=0.9, color='skyblue')
plt.xlabel('Distancia')
plt.ylabel('Frecuencia')
plt.title('Histograma de las distancias entre puntos con dimension '+str(data[0]))

dim = 1
x_locator = MultipleLocator(dim)
plt.gca().xaxis.set_major_locator(x_locator)

plt.xlim(0)
plt.grid(True)

plt.show()
```

El programa en python leerá la información de los archivos data.txt y distancias.txt y los almacenará en vectores para el manejo de la información que luego será requerida para la creación de los histogramas. Se usó la librería matplotlib para la generación de los histogramas. Código Completo en el repositorio de Github: **https://github.com/PaulParizacaMozo/Estructuras-De-Datos-Avanzadas**

Referencias

1.- uniform_real_distribution
https://en.cppreference.com/w/cpp/numeric/random/uniform_real_distribution

2.- Distancia Euclidiana.
https://www.lifeder.com/distancia-euclidiana/

3.- La Maldición de la Dimensionalidad
https://medium.com/@nicolasarrioja/la-maldici%C3%B3n-de-la-dimensionalidad-f7a6248cf9a

4.- La maldición de la dimensionalidad - 3 - Ciencia de Datos y Machine Learning
https://youtu.be/WlavTxJSDUs