# ΕΡΓΑΣΙΑ 2

## ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΑΛΓΟΡΙΘΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

ΜΑΝΤΖΟΥΡΑΝΗΣ ΓΕΩΡΓΙΟΣ **sdi1700076**

ΜΙΧΟΠΟΥΛΟΣ ΜΙΧΑΛΗΣ **sdi1700091**

## NN and Clustering for time series

**20 / 12 / 2021**

**In this project we used Lsh and Clustering structures as well as in the previous project**

**https://github.com/mixo091/Emiris-Project1. In order to find the Approximate Nearest Neighbor we used the metrics below :**

- **Euclidean Distance**
- **Discrete Frechet**
- **Continuous Frechet**

# Download Repository

git clone https://github.com/mixo091/Emiris-Project1.git

# Compilation and Execution

- **A part**:
  - **make search**
    - **./ search -i TestSets/<input_file>** -q **TestSets/<query_file>** -k **<int>** -L **<int>** -M **<int>** -probes **<int>** -o **<output_file>** -algorithm **<LSH or Hypercube or Frechet>** -metric **<discrete or continuous>** -delta **<double>**
      - **k :** number of Lsh functions
      - **L:** number of hash tables
      - **M:** max num of possible items to check (hypercube)
      - **probes:** max neighbors to check (hypercube)
      - **algorithm:** Lsh Hypercube Frechet
      - **metric:** discrete / continuous (**only if Frechet algorithm is given)**
      - **delta:** sample double number
  - **make clean** : remove object files and targets

- **B part:**
    - **make cluster**
        - /cluster -i ./TestSets/FinalSets/nasd_input.csv -c config.txt -o ./Results/result.txt -update MeanVector -assignment Classic -complete 1
        - ./cluster -i ./TestSets/FinalSets/nasd_input.csv -c config.txt -o ./Results/result.txt -update MeanVector -assignment Classic -complete 0
        - /cluster -i ./TestSets/FinalSets/nasd_input.csv -c config.txt -o ./Results/result.txt -update MeanVector -assignment Classic -complete 0
        - /cluster -i ./TestSets/FinalSets/nasd_input.csv -c config.txt -o ./Results/result.txt -update MeanVector -assignment Hypercube -complete 0

    - **make clean**

# Header Files

- **Clustering**
  - Clustering.hpp → Class used for clustering and B part of this project
  - ClusteringUtilities.hpp → Helping functions for clustering
- **Data**
  - Data.hpp → This is our main class for representing our data
- **frechet → Fred Library**
  - config.hpp
  - curve.hpp
  - frechet.hpp
  - point.hpp
  - simplification.hpp
  - types.hpp
- **hashTable**
  - HashTable.hpp →This is our hash table
- **hypercube**
  - hypercube.hpp → Hypercube class
- **LSH**
  - lsh_Cfrechet.hpp → lsh class used for Continuous Frechet , chlid class of lsh
  - lsh.hpp → main lsh class

# Cpp files

- **The below are used for fred library only**
  - config.cpp
  - curve.cpp
  - frechet.cpp
  - interval.cpp
  - point.cpp
  - simplification.cpp

- Functionality.cpp → helping functions for arguments parsing, calculation, brute force implementantion and more
- a3_main.cpp → this is our main function of A part

## Some points to mention

For the implementation of A3 subtask of A part, we at first store the input time series as well as the query series, in a vector , without taking into consideration the variable **t of time.** So we have a vector in **R** space of some points. We start by doing the **filtering** and then by initialising **lsh_CFrechet** object, we continue with snapping and padding, in order to create the key of our hash table. The insertion of that key is done with the use of **lsh** of previous project. For that purpose **lsh_CFrechet , inherits from Lsh class.** Finally, we call **perform_continuous_frechet(…)** in order to calculate the distances between the filtered query curves and the filtered input curves.