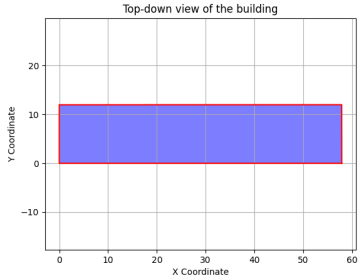
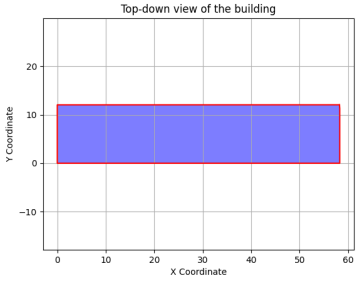
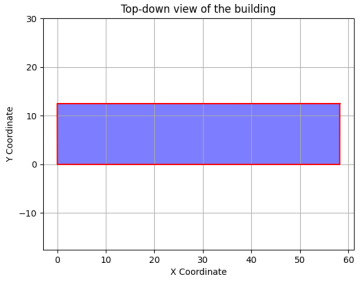
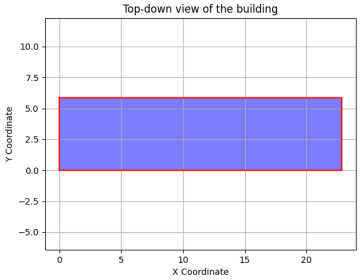
Andmekaeve EX4

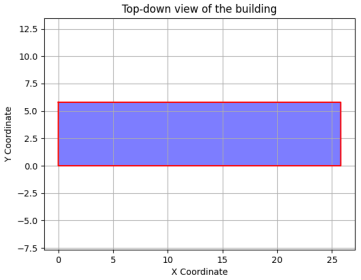


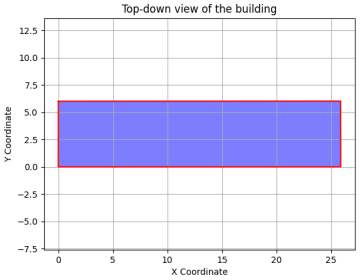




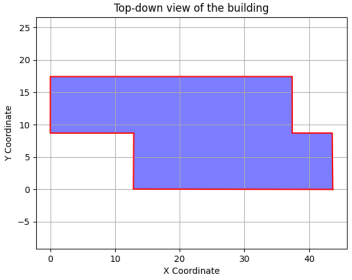
112017313, 112016810, 112016804
peaaegu identsed

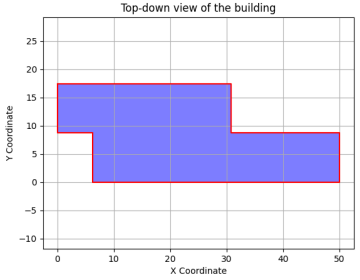


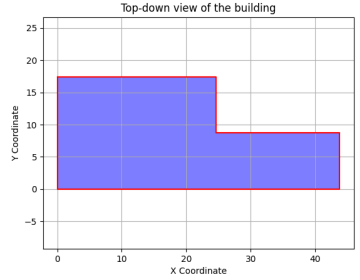




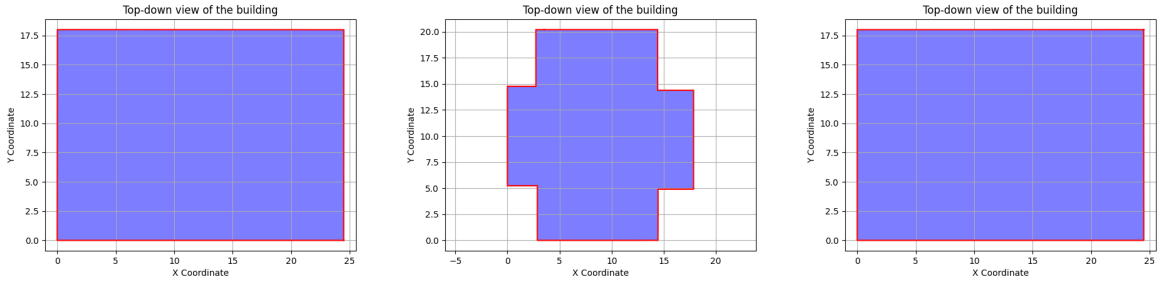
120230405, 120218032, 120217240
peaaegu identsed







121396267, 121281541, 121396267
sarnased, kuid mitte identsed, eriti viimane



121366297, 101019414, 120863258
Viimased pildid (peaksid olema) Rocca Towerid, kuid üks valikutest läks vist valesti :), seda on näha ka nii EX3, kui ka EX4 sarnasusmaatriksis, et üks hoone ei sarnane mitte ühegi teisega.

Minu EX4 lahendamisel otsisin kujunditevahelisi sarnasusi, hoone koordinaatide vahelistest absoluutsetest kaugustest hoolimata. Kõigepealt kasutasin kõigi koordinaatide peal kohandatud normaliseerimist, mis normaliseeris iga kujundi puhul eraldi X ja Y teljed. Seejärel otsisin endale algoritmi mille abil sarnasust leida, kõige paljulubavam kujundite sarnasusalgoritm mille ma leidsin oli Hausdorffi distants, leidsin kõikide normaliseeritud kujundite omavahelise Hausdorffi distantsi ja lisasin selle sarnasusmaatriksisse. Ma ütleksin, et jäin tulemusega rahule. Maatriksis on selgelt näha ühte rocca towerit, mis teiste sekka ei sobi ja seda, et teised ehitistekolmikud on üsna sarnased.

	12128	**12086**	**12021**	**12136**	**11201**	**11201**	**12139**	**10101**	**12021**	**12023**	**12128**	**11201**
121281541	0.00	0.41	0.57	0.41	0.38	0.42	0.09	0.43	0.55	0.56	0.50	0.48
120863258	0.41	0.00	0.49	0.00	0.48	0.48	0.35	0.25	0.49	0.49	0.49	0.48
120217240	0.57	0.49	0.00	0.49	0.40	0.39	0.53	0.49	0.03	0.02	0.59	0.24
121366297	0.41	0.00	0.49	0.00	0.48	0.48	0.35	0.25	0.49	0.49	0.49	0.48
112016804	0.38	0.48	0.40	0.48	0.00	0.40	0.39	0.36	0.40	0.40	0.43	0.40
112017313	0.42	0.48	0.39	0.48	0.40	0.00	0.42	0.48	0.39	0.39	0.43	0.39
121396267	0.09	0.35	0.53	0.35	0.39	0.42	0.00	0.42	0.52	0.53	0.50	0.41
101019414	0.43	0.25	0.49	0.25	0.36	0.48	0.42	0.00	0.49	0.49	0.44	0.49
120218032	0.55	0.49	0.03	0.49	0.40	0.39	0.52	0.49	0.00	0.01	0.58	0.26
120230405	0.56	0.49	0.02	0.49	0.40	0.39	0.53	0.49	0.01	0.00	0.59	0.25
121281521	0.50	0.49	0.59	0.49	0.43	0.43	0.50	0.44	0.58	0.59	0.00	0.53
112016810	0.48	0.48	0.24	0.48	0.40	0.39	0.41	0.49	0.26	0.25	0.53	0.00

Kuigi ma nägin EX4 peal palju rohkem vaeva ja kasutasin keerukamaid meetodeid (Hausdorffi distantsi), siis on EX3 maatriksist gruppe suurema väärtuste vahemiku tõttu kergem gruppe välja lugeda. EX3 arvutuskäik on väga palju lihtsam, kuid sarnasusmaatriks oleneb tugevalt sellest, milliseid andmeid kasutada. EX4 arvutuskäik on palju keerulisem, kuid annab ilmselt hoonete sarnasuse kohta kindlama ülevaate, kasutasin EX3 lihtsalt andmeid, mis gruppide siseselt väga sarnasesid ja muidu erinesid.

import os
import math
import json
import csv
import pandas as pd

```
def normalize_coordinates(coords: list):
    if not coords:
        return []

    x_values, y_values = zip(*coords)  # Separate x and y components

    x_min, x_max = min(x_values), max(x_values)
    y_min, y_max = min(y_values), max(y_values)

    normalized_coords = [
        (
            (x - x_min) / (x_max - x_min) if x_max != x_min else 0,  # Normalize x
            (y - y_min) / (y_max - y_min) if y_max != y_min else 0   # Normalize y
        )
        for x, y in coords
    ]

    return normalized_coords

def calculate_point_distance(point1, point2):
    print("calculation")
```

```
    return ((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)**0.5

def directed_hausdorff_distance(coords1, coords2):
    max_min_dist = 0
    for point1 in coords1:
        min_dist = float('inf')
        for point2 in coords2:
            print(point1)
            print(point2)
            dist = calculate_point_distance(point1, point2)
            min_dist = min(min_dist, dist)
        max_min_dist = max(max_min_dist, min_dist)
    return max_min_dist


def create_comparison_matrix(coordinate_sets, labels=None):
    coordinates_dict = {}
    for coordinates in coordinate_sets:
        coordinates_dict[coordinates[0]] = coordinates[1]

    df = pd.DataFrame.from_dict(coordinates_dict, orient="index")
    df.to_csv("results/coordinates.csv", index=False)
    # Print as a formatted table
    print("\nCoordinates:")
    print(df.to_string())

    n = len(coordinate_sets)
    hausdorff_matrix = [[0.0 for _ in range(n)] for _ in range(n)]

    n = len(coordinate_sets)

    for i in range(n):
        for j in range(i + 1, n):
```

```
            forward = directed_hausdorff_distance(coordinate_sets[i][1], coordinate_sets[j][1])
            backward = directed_hausdorff_distance(coordinate_sets[j][1], coordinate_sets[i][1])
            distance = max(forward, backward)
            hausdorff_matrix[i][j] = distance
            hausdorff_matrix[j][i] = distance

    hausdorff_df = pd.DataFrame(hausdorff_matrix, index=df.index, columns=df.index)
    hausdorff_df.to_csv("results/similarity_matrix.csv")

    print("ran calculate_similarities")
    return hausdorff_df


if __name__ == "__main__":
    pathname = "../EX3/data"
    building_coordinates = []
    for filename in os.listdir(pathname):
        print(filename)
        ehr_code = int(filename.replace(".ehr.json", ""))
        with open(f"{pathname}/{filename}", "r") as f:
            data = json.load(f)

        print(data[0]["ehitis"]["ehitiseKujud"]["ruumikuju"][0]["geometry"]["coordinates"][0])
        coordinates = (ehr_code, normalize_coordinates(data[0]["ehitis"]["ehitiseKujud"]["ruumikuju"][0]["geometry"]
["coordinates"][0]))
        building_coordinates.append(coordinates)

    similarity_matrix = create_comparison_matrix(building_coordinates)

    similarity_matrix.to_csv("results/similarity_matrix.csv")
```