ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Lecturer: Prof. B. Gärtner

Informatik für Mathematiker und Physiker HS13

Exercise Sheet 11

Submission deadline: 15:15 - Tuesday 3rd December, 2013
Course URL: http://www.ti.inf.ethz.ch/ew/courses/Info1_13/

Assignment 1 – Skript-Aufgabe 148 (4 points)

a) The file `choosing_numbers.cpp` contains an implementation of the game of choosing numbers. Choose any strategy (or loaded dice) you like and simulate 1 million rounds of the game. Do you beat the second player (denoted as `your_friend` in the code)? Can you find a strategy that beats the fair dice? (Empirically.)

b) Find a loaded dice that beats the fair dice in the game of choosing numbers. (This is a theory exercise.)

Assignment 2 – Skript-Aufgabe 147 (2 points)

Consider the generator `ansic` used in `choosing_numbers.cpp`. Since the modulus is $m = 2^{31}$, the internal computations of the generator will certainly overflow if 32 bits are used to represent `unsigned int` values. Despite this, the sequence of pseudorandom numbers computed by the generator is correct and coincides with its mathematical definition. Explain this!

Assignment 3 (5 points)

Write a class `Vector_2` that implements objects of a 2-dimensional vector space over $\mathbb{R}$ (which we will model by the type `double` for the purpose of this exercise).

a) Implement a (non-trivial) constructor, member functions `x()` and `y()` that let you retrieve the x-and y-coordinates of a vector, and `operator+=`, `operator+`, `operator-=`, `operator-` that should work as one could expect. Make it const-correct!

b) Implement scalar multiplication and division. Make sure, that for `Vector_2 p` and `double d`, not only `p * d` compiles, but also `d * p`, and both versions are working as expected.

c) Provide `operator>>` and `operator<<` that let you read and write points to and form the input stream. (For a point (x, y), the coordinates are simply separated by a space: `x y`).

d) Write a function `squared_distance` that computes the squared distance between two points.

Test your code with the program `Vector_2_test.cpp` that you find on the webpage.

Assignment 4 (5 points)

In this exercise, we implement the k-*means clustering algorithm*. Given a set of n points, and a number k, we want to cluster the given points into k sets in some meaningful way. We call the mean of all the points in one cluster a *cluster center*. Ideally, one could ask to find the *best* clustering, i.e. an assignment of the points into k sets (a clustering), such that the sum of all the distances between the points and their receptive cluster center is as small as possible. This is a very hard problem in general. The k-means clustering algorithm computes a clustering that in many cases looks very good, but is not necessarily the best possible clustering.

The k-means algorithm works as follows:

1. Read two numbers n and k, the number of points and the number of clusters, and then the coordinates of n 2-dimensional points.

2. Choose k cluster centers. For the purpose of this exercise, you can for instance pick any k points form the input, or simply the k first ones. These cluster centers define the k clusters.

3. Calculate for each point the closest cluster center and assign the point to this cluster.

4. Update the means: Compute for each cluster the mean of the points belonging to this cluster set this as the new cluster center.

5. Repeat the steps 3.-4. until the process stabilizes, i.e. the clusters do not change anymore between the iterations. For the purpose of this exercise, simply repeat 3.-4. 100 times, say.

6. Output the k cluster centers. You can also depict the result graphically by using the `ifm::Window` library. The output could for instance look like in Figure 1 below.

Use the sample inputs provided on the course webpage to test your algorithm. What happens when you change k, the number of desired cluster centers? What happens when you initialize the algorithm with k *different* cluster centers, say the last k points from the input?

Hint: Use the class `Vector_2` from the previous exercise to read the points from the input, calculate the distances, and compute the means. There are also two demo programs on the website that demonstrate how you can produce colorful plots with the `ifm::Window` library.

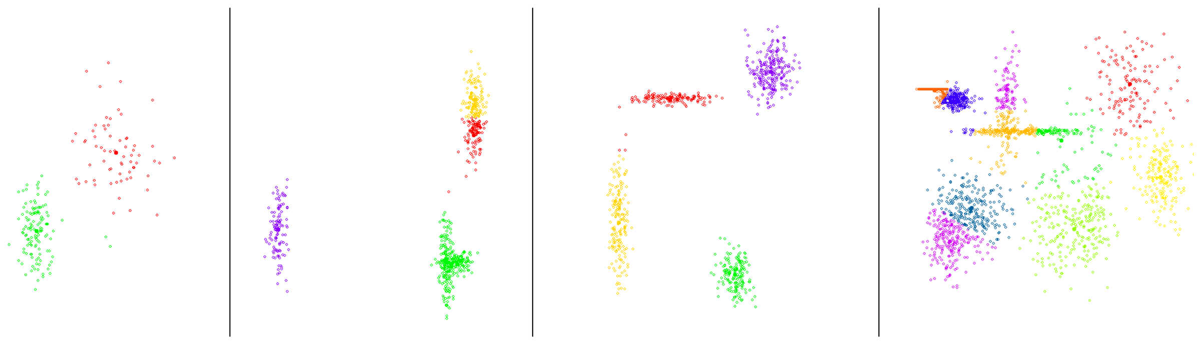

Figure 1: Sample output of the k-means clustering algorithm on the datasets from the website. The clusters are depicted by color, the clusters centers by solid circles. which you can hardly see...