# $k$–Nearest Neigbors

## 1 $k$-NN classifier

1. Load the training set and choose a positive integer $k$.
2. Recieve an example to classify.
3. Pick $k$ vectors from the training set with the shortest (euclidean) distance from the new example.
4. Classify the new example according as the class which occurs most frequently among the nearest neighbors picked in 3.

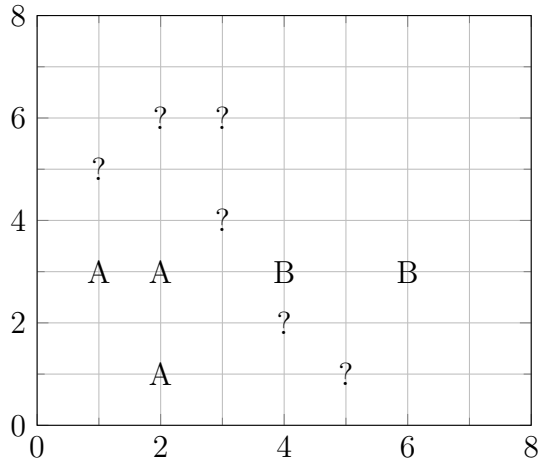
## Questions

**Question 1.**

Classify the following examples as A or B using the $k$-NN method with $k = 3$.
**Training set:** A(1, 3), A(2, 1), A(2, 3), B(4, 3), B(6, 3).

**Examples to classify:**

- $(1, 5)$
- $(3, 6)$
- $(2, 6)$
- $(4, 2)$
- $(3, 4)$
- $(5, 1)$



**Question 2.**

Similarly to Question 1, classify the following examples based on the training set ($k = 3$):

**Training set:** A(5, 4, 1), A(4, 3, 0), B(1, 2, 3), B(2, 0, 4), C(6, 1, 1), C(5, 0, 1).

**Examples to classify:** (4, 4, 0), (1, 1, 5), (6, 0, 0).

# Mini-project: $k$-NN

The goal is to implement the $k$-NN classifier. The program should take 3 arguments:
`k`: positive natural number being the k-NN hiperparameter.
`train-set`: name of the file containing csv train set.
`test-set`: name of the file containing csv test set.

**Requirements:**

- The program should apply $k$-NN classifier based on the train set to each vector from the test set and produce the accuracy (proportion of correctly classified examples from the test set).
- The program should additionally provide a simple interface (not necessarily graphical) to enable the user to input single vectors to be classified.
- Test the program using training data in `iris.data` and test data in `iris.test.data`.
- **Important:** the program should be able to load any dataset (in a format similar to `iris.data`), with an arbitrary number of dimensions/classes.
- **Optional extension:** prepare a graph (excel, python, etc.) showing the accuracy vs the value of $k$.
- **Optional extension:** also classify examples in the *WDBC* dataset provided in the files `wdbc.data` and `wdbc.test.data` [Source].