**Задание 1.** *Разреженные числовые матрицы* – это матрицы, в которых только относительно малая часть элементов не равна нулю (<https://en.wikipedia.org/wiki/Sparse_matrix>). Например, если в матрице размером 100x100 только угловые элементы ненулевые (их 4 штуки), то мы определённо можем назвать такую матрицу разреженной. Очевидно, что в целях экономии памяти нулевые элементы разреженной матрицы желательно не хранить. Вам необходимо создать и протестировать класс для представления разреженной матрицы.

1. Создать класс SparseMatrix для представления разреженной целочисленной матрицы (элементы матрицы имеют тип int). Обязательное условие – класс должен иметь конструктор, позволяющий задать размеры матрицы (количество строк и количество столбцов, строго больше нуля), и индексатор с двумя индексами для доступа к элементам матрицы. В качестве внутреннего хранилища вы можете использовать любую коллекцию на ваш выбор. Желательно обеспечить приемлемое быстродействие и разумный расход памяти.
2. Для удобства отладки переопределить в классе матрицы метод ToString().
3. Реализовать в классе SparseMatrix интерфейс IEnumerable<int>. Должны возвращаться **все** элементы матрицы, включая нулевые (делается построчный обход).
4. Описать в классе SparseMatrix метод GetNonzeroElements(). Тип возвращаемого значения – IEnumerable<(int, int, int)>. Метод возвращает набор кортежей вида (*индекс\_i*, *индекс\_j*, *значение*) для всех ненулевых элементов. Индексы должны быть упорядочены по столбцам, затем по строкам – то есть сначала ненулевые элементы из первого столбца, затем – ненулевые элементы из второго столбца, и т. д.
5. Снабдить класс SparseMatrix методом GetCount(x). Он должен возвращать, сколько раз элемент x встречается в матрице. При реализации метода GetCount(x) учтите, что x может быть равен 0.

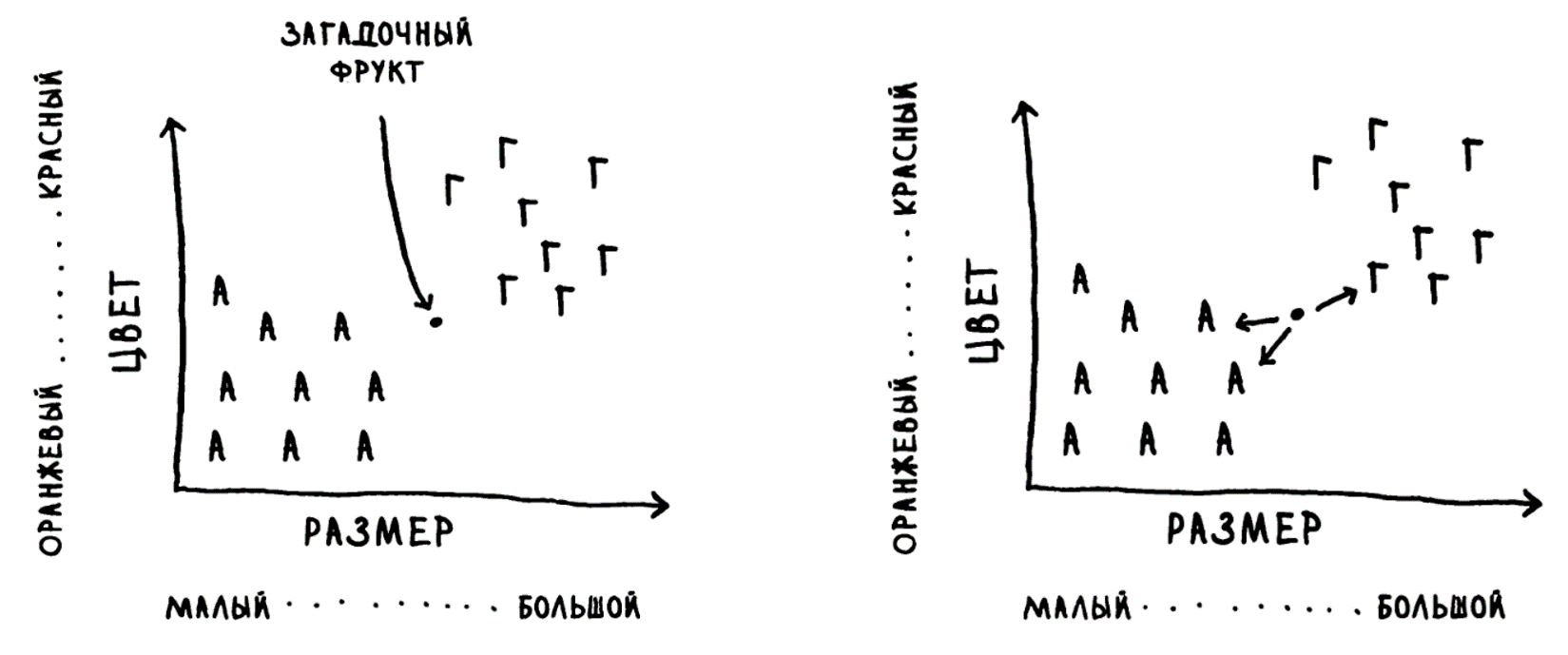
**Задание 2.**

1. Создайте простой класс для представления *книги*. Сохраните в классе книги *название книги* (строка, не пустая, не null), *дату публикации* (возможно равная null) и *набор авторов книги* (коллекция неповторяющихся строк, возможно пустая).
2. *13-значный ISBN книги* – это строка, имеющая формат XXX-X-XX-XXXXXX-X или XXXXXXXXXXXXX, где X – это цифра 0..9. Создайте класс *Каталог* – это коллекция, **подобная словарю**, в которой хранятся книги. Предусмотрите доступ к книге по ключу-строке – по ISBN книги. Учтите важный нюанс – если книгу поместили в каталог, применив ключ 123-4-56-789012-3, то её можно извлечь и по ключу 123-4-56-789012-3, и по ключу 1234567890123. Корректность самого ISBN в этой задаче можете не проверять (не надо проверять контрольные цифры, корректность самого формата проверить желательно – например, при помощи регулярных выражений).

**Задание 3.** *Метод k-ближайших соседей* – один из простейших алгоритмов машинного обучения (<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>).

Суть алгоритма: имеется набор объектов с известными **названиями** и числовыми **характеристиками**. Эти характеристики дают возможность описать объект как точку в -мерном пространстве. Пусть есть объект, у которого заданы характеристики, но нет названия. Чтобы найти название этого объекта, найдём ближайших к нему известных объектов. То название, которое будет у ближайших соседей повторяться чаще, сделаем названием нашего неизвестного объекта (алгоритм классификации).

Пример: апельсины и грейпфруты, . Неизвестный объект оказался апельсином.



1. Необходимо реализовать алгоритм метода k-ближайших соседей в двумерном пространстве характеристик. Вход алгоритма: набор известных объектов, две характеристики неизвестного объекта и значение . Каждый объект состоит из строкового названия и двух (для простоты) числовых характеристик (можно использовать кортежи). Выход алгоритма: строка с названием неизвестного объекта. Постарайтесь при реализации алгоритма использовать возможности класса Parallel или технологии PLINQ.
2. Создайте консольное приложение для тестирования алгоритма. Консольное приложение читает входные данные из текстового файла. Пример файла приведён ниже. Полный файл – в приложении к заданию.

**Пример файла с данными**

orange,1.0,1.1

orange,1.0,2.0

orange,4.0,2.0

orange,4.0,3.2

orange,2.0,4.3

orange,3.0,4.1

orange,3.0,3.1

grapefruit,6.2,8.5

grapefruit,7.0,7.0

grapefruit,7.2,8.0

grapefruit,8.0,7.0

grapefruit,8.2,8.0

grapefruit,8.0,9.0

grapefruit,8.5,8.6