Добрый день,

Я не программист и данная подача материала вам может показаться ближе и понятней и тут отражены все основные понятия необходимые нам по программе.

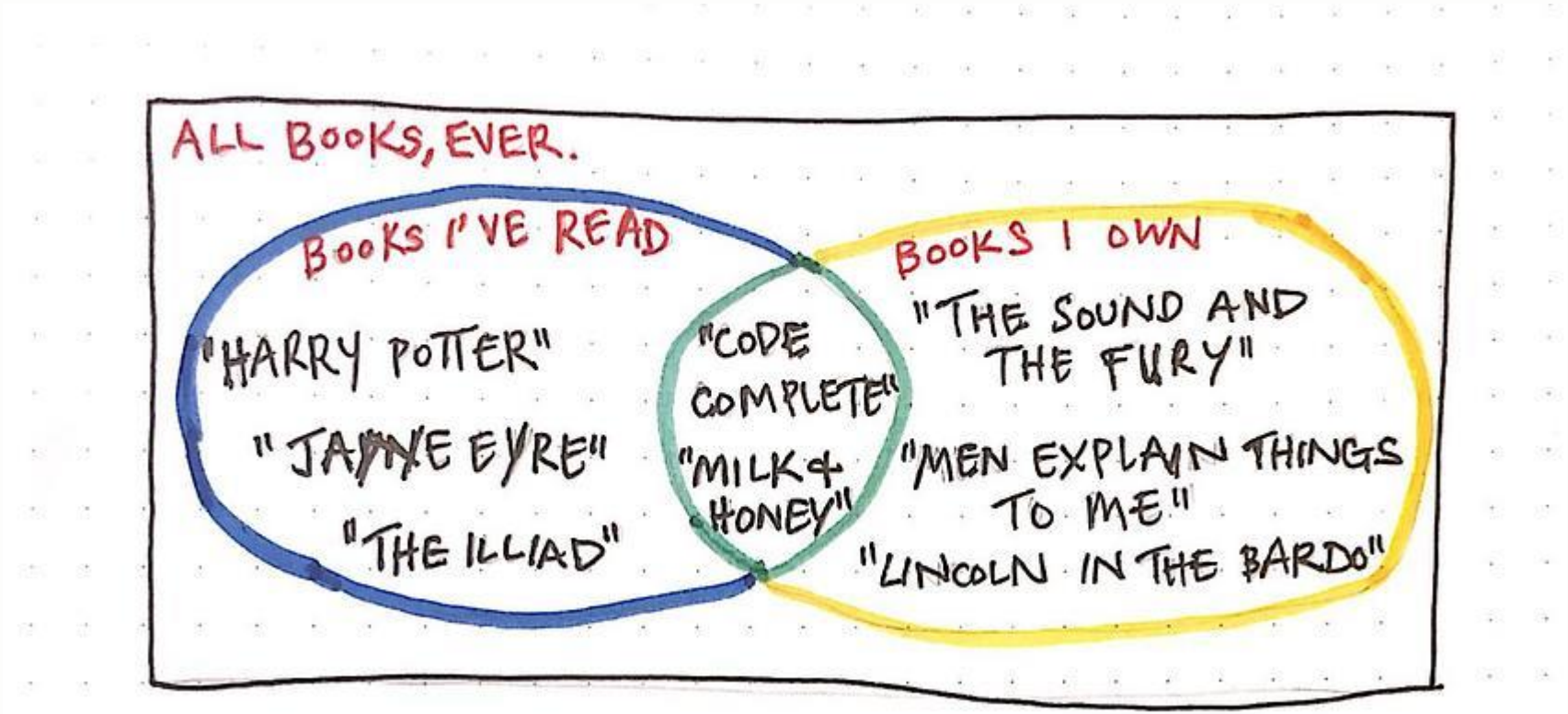
И ответьте письменно на вопросы:

1. Множества, их виды.
2. Способы задания множеств
3. Подмножества
4. Универсальное множество
5. Способы задания множеств
6. Операции над множествами и их свойства
7. Бинарные отношения
8. Области определения и значений бинарного отношения
9. Типы бинарных отношений
10. Операции над бинарными отношениями и их свойства
11. Основные свойства бинарных отношений во множестве А
12. Виды бинарных отношений во множестве
13. Функциональные отношения
14. Отображения и их типы
15. Подстановки как отображения

Срок сдачи – 13.01.2021г.

https://ru.hexlet.io/blog/posts/teoriya-mnozhestv-osnovy-i-bazovye-operatsii-nad-mnozhestvami

Теория множеств: основы и базовые операции над множествами



Мы знаем довольно много о структурах данных, понимаем их устройство, разбираемся, какие структуры работают быстро и помогают решать конкретные задачи. Но эти знания бесполезны, если мы не понимаем, как это использовать в реальной жизни. Это похоже на изучение геометрии в школе. Вы долго считаете предмет бесполезным, пока однажды не появляется необходимость рассчитать площадь пола, чтобы заказать новое ковровое покрытие. Впрочем, пользу геометрии можно почувствовать, даже если вы никогда не считали площадь пола в комнате самостоятельно.

Сегодня поговорим о структуре данных, которая в теории очень догматична, а на практике очень популярна. На самом деле вы так или иначе уже сталкивались с этой структурой, а также слышали о ней на уроках математики в школе. Вы уже догадались, что речь идёт о множествах.

Теория множеств без страха

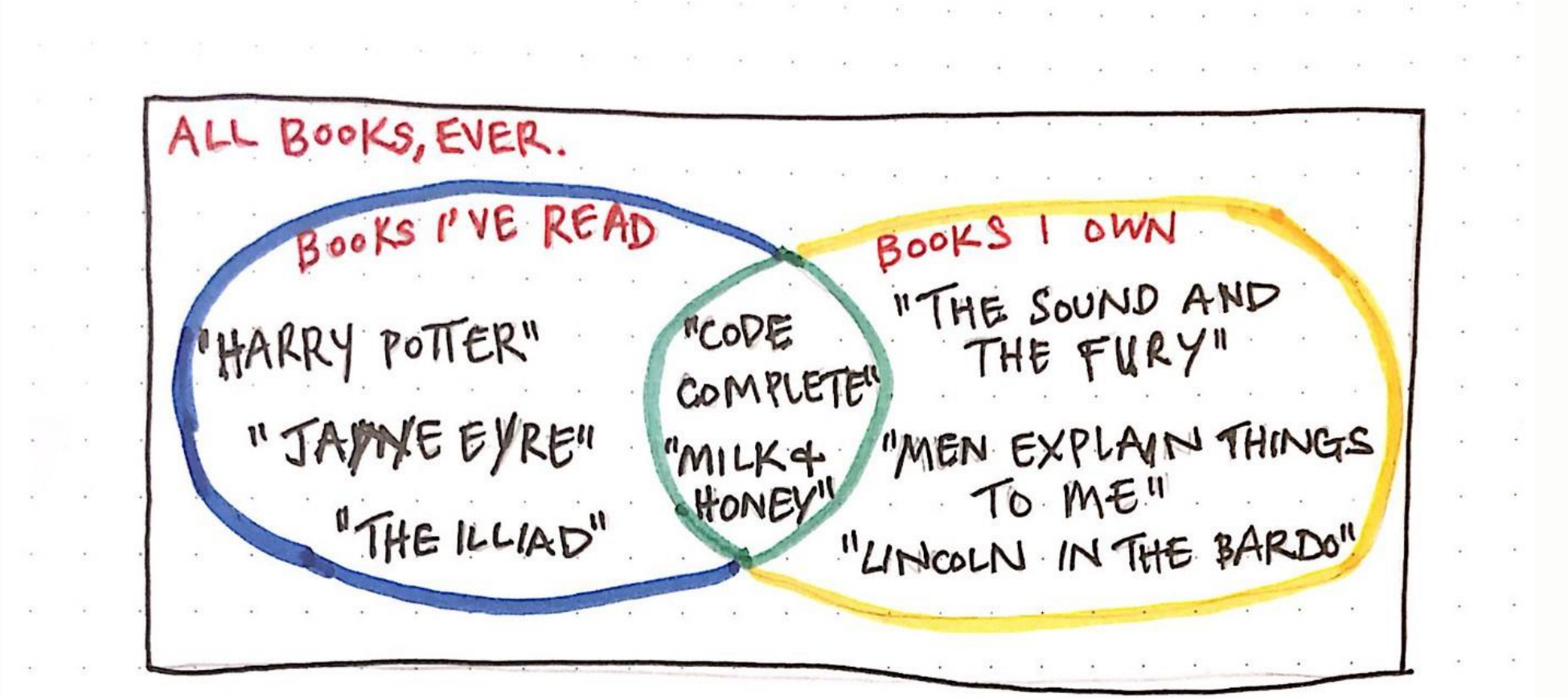
Прежде чем разбирать устройство множеств, давайте поймём, откуда они появляются. То есть давайте сразу погрузимся в теорию — да-да, в теорию множеств! Не бойтесь сложностей — высока вероятность того, что вы уже так или иначе использовали эту теорию. Возможно, вы сталкивались с теорией множеств, когда проходили в школе диаграмму Венна. Диаграмму Венна включили в программу изучения множеств, так как она хорошо иллюстрирует отношения подмножеств.

Мы выяснили, что теория множеств не должна никого пугать. Теперь пришло время разобраться, что это за теория на самом деле. Множество — математическая концепция. Теорией множеств описывает отношения множеств.

Множество — ни что иное, как неупорядоченная коллекция, в которой нет дублирующихся элементов.

В этом определении есть три важных слова: «неупорядоченная», «дублирующихся» и «элементов». Эти слова точно передают суть и устройство множества. Если мы это запомним, то будем знать основную информацию о том, как работает эта структура данных.

Нужно понять, почему это важно. Для начала давайте посмотрим на множества в действии. Как сказано выше, отношения множеств удачно иллюстрирует диаграмма Венна. Давайте взглянем на два множества: книги, которые есть у человека дома, и книги, которые этот человек прочитал.



Если вы знакомы с диаграммой Венна, то понимаете, что в центре в зелёном круге находятся книги, которыми человек владеет, и которые он прочитал. Здесь множества пересекаются. Также вы понимаете, что два множества — прочитанные человеком книги и книги, которые есть у человека — существуют внутри другого множества. Это все существующие в мире книги.

Диаграмма Венна — хорошая база для понимания теории множеств, так как с её помощью легче понять более сложные вещи. Допустим, вы хотите представить два множества книг в какой-то структуре данных. Вы уже знаете, что книги надо разделить на два множества: которые человек прочитал и которые есть у него дома. Для удобства назовём первое множество Set X, а второе Set Y. Эти множества после реконфигурации в структуры данных можно представить с помощью диаграммы Венна.

BOOKS I'VE READ: X

X = {
"HARRY POTTER",
"JANE EYRE",
"CODE COMPLETE",
"THE ILLIAD",
"MILK & HONEY"
}

BOOKS I OWN: Y

Y = {
"THE SOUND & THE FURY",
"LINCOLN IN THE BARDO",
"MILK & HONEY",
"MEN EXPLAIN THINGS TO ME",
"CODE COMPLETE"
}

Both of these sets are a collection of distinct, unique objects. There will never be duplicate values in a set. There will also never be a predetermined order to the objects.

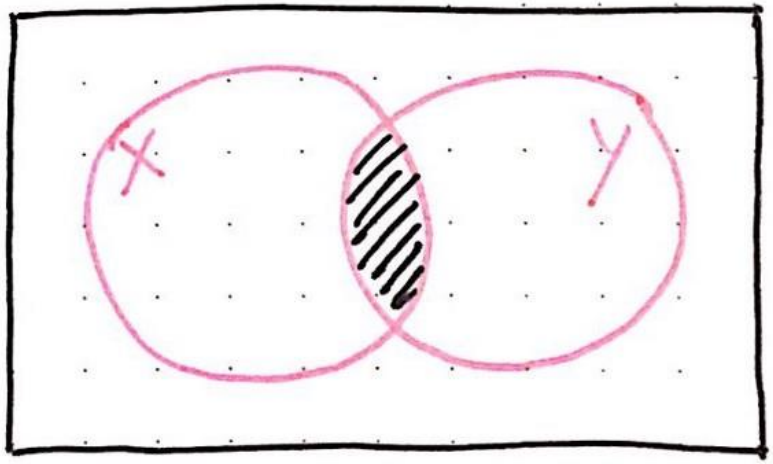
Можно заметить, что множества Set X и Set Y стали похожи на объекты или хэши: элементы внутри них не имеют индексов или других элементов, позволяющих их упорядочить. В них также нет повторяющихся элементов, что делает эти структуры данных множествами. Как вы уже знаете, множество — это коллекция неупорядоченных элементов, которые не повторяются.

Об операциях с множествами без боли

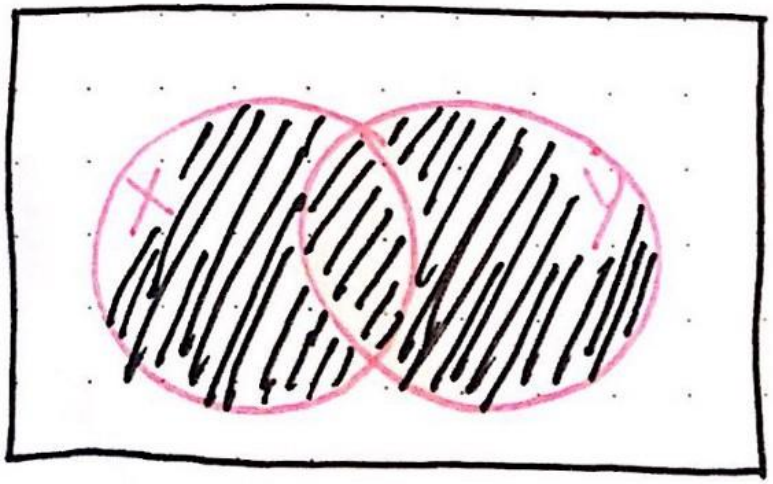
Какие возможности открывает представление множеств в формате структур данных? С ними теперь можно выполнять разные операции. Две самые важные операции, которые выполняются над множествами — это пересечение и объединение.

Intersections vs. Unions

$X \cap Y$, or the intersection of X & Y



$X \cup Y$, or the ~~intersect~~ union of X & Y



Пересечение множеств часто записывается с помощью такой нотации: X ∩ Y. Пересечение определяет, где два множества пересекаются. Другими словами, эта операция возвращает все элементы, которые входят в два множества. В нашем примере пересечение Set X и Set Y возвращает все книги, которые человек читал и которые есть у него дома. Хороший ключ к пониманию пересечения — ключевое слово «и». Мы получаем книги, которые человек читал **и** которые есть у него дома. Несмотря на то, что полученные с помощью пересечения книги существуют в двух множествах, мы не повторяем их, так как в множестве могут быть только уникальные элементы.

Объединение двух множеств обозначается так: X ∪ Y. Объединение возвращает общность двух множеств или объединённое множество. Иными словами, с помощью объединения множеств можно получить новое множество элементов, которые существуют хотя бы в одном исходном множестве. В нашем случае объединение вернёт все книги, которые человек читал, а также все книги, которые есть у него дома. Обратите внимание, если книга входит одновременно в Set X и Set Y, она не может дублироваться в новом множестве после объединения, так как в множества входят только уникальные элементы.

С помощью диаграммы Венна пересечение и объединение можно представить так:

Operations on Sets

X = {
"HARRY POTTER",
"JANE EYRE",
"MILK & HONEY",
"CODE COMPLETE",
"THE ILLIAD"
}

Y = {
"MILK & HONEY",
"MEN EXPLAIN THINGS TO ME",
"LINCOLN IN THE BARDO",
"CODE COMPLETE",
"THE SOUND & THE FURY"
}

INTERSECTIONS

$X \cap Y$: yields another set of all the elements that are both in X and Y.

"and"

$X \cap Y$ = {"CODE COMPLETE", "MILK & HONEY"}

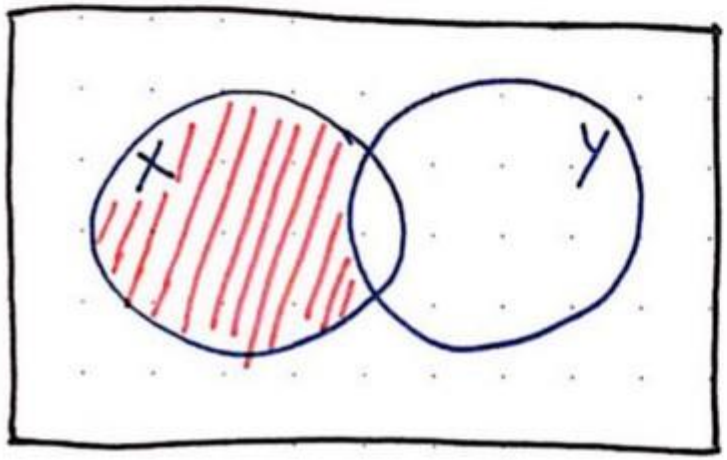
Теперь давайте рассмотрим более сложные вещи. Объединение и пересечение — важные операции над множествами, но это только азы теории. Нам надо познакомиться с другими операциями, чтобы решать более серьёзные задачи. Важно понимать разность множеств и относительные дополнения множеств. Ниже мы разберём, почему это важные операции, но сначала нужно понять, как они работают.

Как понятно из названия, разность множеств определяет разницу между множествами. Иными словами, мы определяем, какие элементы останутся в множестве X, если удалить из него все элементы, которые содержатся в множестве Y. Это действие можно обозначить так: X — Y. В примере на иллюстрации ниже разница между множеством X и множеством Y — это элементы, которые существуют в Set X, но не существуют в Set Y. Они обозначены буквами C, Z и W.

Slightly More Complex Set Operations

SET DIFFERENCE

X − Y: yields the difference between two sets, or all the elements in set **X** that are not in set **Y**.



$X = \{m, L, A, C, Z, W\}$

$Y = \{X, N, O, L, A, m\}$

$X - Y = \{C, Z, W\}$

Относительное дополнение — противоположность разности множеств. Например, относительное дополнение Y по сравнению с X возвращает все элементы множества Y, которые не входят в множество X. Относительно дополнение можно обозначить так: X \ Y. Относительное дополнение X \ Y фактически возвращает такой же набор элементов, как разность Y — X. В нашем примере множество Y меньше множества X. Единственный элемент, который входит в Set Y, но не входит в Set X — число 2.

По сути, мы просто вычитаем множество X из множества Y и отвечаем на вопрос: что существует в Y, чего нет в X?

Вы могли заметить, что в части примеров мы имеем дело со строками, в другой части в качестве элементов выступают буквы и числа. Здесь надо подчеркнуть важный момент: множество может включать любой тип элементов или объектов. Вы можете рассматривать множества как хэши: они включают любые сущности, если те встречаются в множестве только один раз.

Теперь давайте рассмотрим ещё одну операцию, она самая сложная из всех. Но не пугайтесь, с ней тоже можно разобраться.

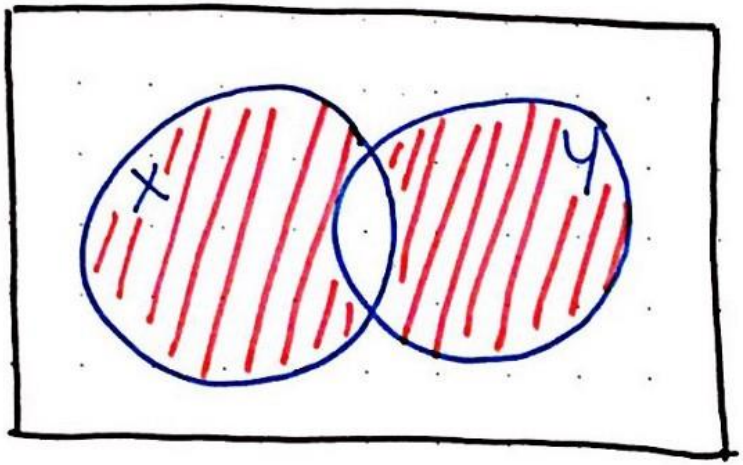
В некоторых случаях требуется найти противоположность пересечению множеств. Иными словами, речь идёт о книгах, которые есть у человека, и книгах, которые он прочитал, но которые не входят одновременно в оба множества. Как назвать это подмножество? И как найти его?

Правильное название для этого кейса — симметрическая разность множеств. Также употребляют термины «дизъюнктивное объединение» и «несвязное объединение». Симметрическая разность возвращает все элементы, которые входят в одно из множеств, но не входят в пересечение этих множеств. Пример на иллюстрации поможет разобраться с дизъюнктивным объединением.

SYMMETRIC DIFFERENCE

* Also referred to as DISJUNCTIVE UNION

$X \triangle Y$: yields all the elements that exist in either of the sets, but do not exist in the intersection $(X \cap Y)$ of the two sets.



$X = \{A, B, C, 1, 2, 3\}$

$Y = \{X, Y, Z, 1, 2, 3\}$

$X \triangle Y = \{A, B, C, X, Y, Z\}$

В примере выше симметрическая разность похожа на поиск относительного дополнения множества X и множества Y. Если подходить к этому с позиции математики, поиск симметричной разницы — то же самое, что и объединение относительных дополнений множества X и множества Y. Эту операцию можно записать так: $X \triangle Y = (X \setminus Y) \cup (Y \setminus X)$.

Но не дайте сбить себя с толку!

Всё, что нужно для поиска симметрической разности — найти элементы, которые есть в множестве X, но отсутствуют в множестве Y, и какие элементы есть в множестве Y, но отсутствуют в множестве X. Иными словами, надо найти уникальные элементы в каждом множестве.

В примере выше числа 1, 2 и 3 входят в множества X и Y одновременно. А буквы A, B, C, X, Y, Z входят только в множества X или Y. Поэтому они представляют симметрическую разность множеств X и Y.

Мы рассмотрели теоретические вопросы. Теперь можно посмотреть, как теория множеств работает на практике.

Множества вокруг нас

К этому моменту вы наверняка задумались, зачем надо изучать теорию множеств. Это хороший вопрос, и пришло время ответить на него.

Уже догадались? Множества повсюду. Это структуры данных, которые мы можем использовать при работе с разными языками программирования, например, Python, Java, Ruby, JavaScript и так далее. Если вы знакомы с этими или другими языками программирования, то уже вспомнили методы, которые позволяют работать с множествами.

Вот пример на JavaScript.

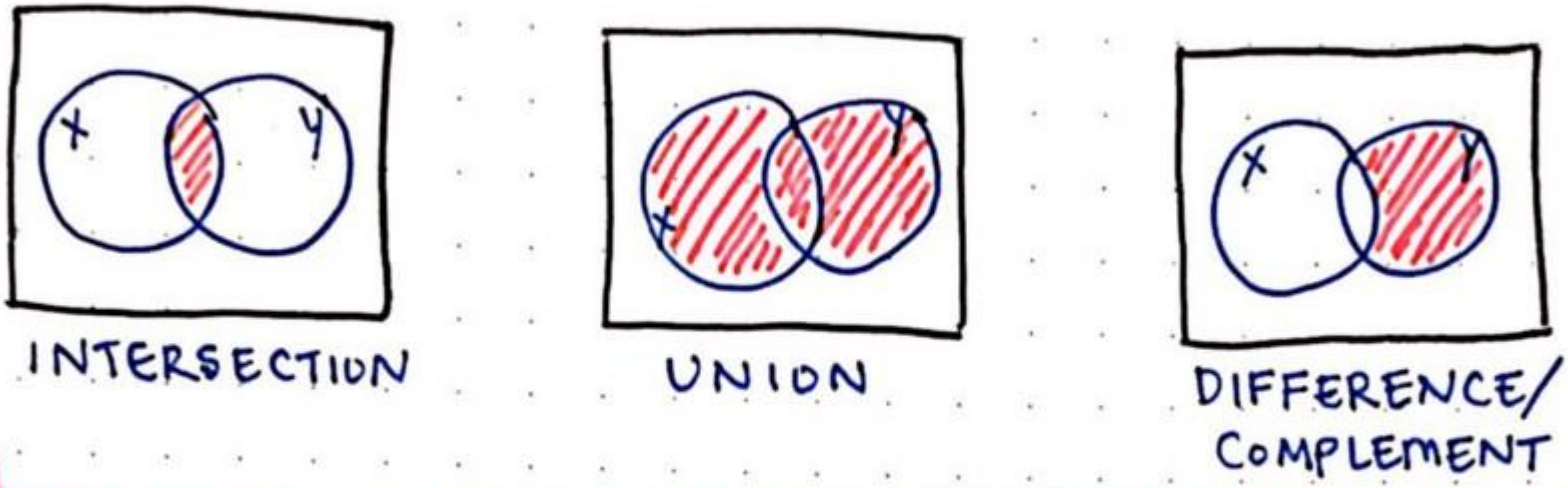
```
var s = new Set();

s.add(2);
// Set { 2}
s.add(45);
// Set { 2, 45}
s.add(45);
// Set { 2, 45}
```

```
s.add('hello world!');
// Set { 2, 45, 'hello world!' }

s.has(2);
// true
s.has('cats');
// false

s.size;
// 3

s.delete(45);
// Set { 2, 'hello world!' }

s.has(45);
// false
```

Очевидно, что имена методов могут меняться в зависимости от языка. Например, метод *has* из примера выше в Ruby называется *include?*, но эти методы работают практически одинаково. А в Python при работе с множествами можно использовать методы *intersection*, *union* и *symmetric_difference*.

Но в чём именно польза множеств? Понятно, что с ними можно работать в разных языках программирования, но зачем это нужно на практике?

Time-Complexity of Sets



The time-complexity of all of these set operations is $O(\text{length}(X) + \text{length}(Y))$.

Imagine two sets, X + Y:

$X = \{A, B, C, 1, 2, 3\}$ $Y = \{X, Y, Z, 1, 2, 3\}$

Один из моментов — множества могут сэкономить вам много времени. Помните все эти сложные операции — *intersection*, *union*, *difference*? Уже догадались? Продолжительность выполнения этих операций зависит от размера множеств. Это связано с тем, что для выполнения операций нам надо обойти все элементы множества. Обычно даже гигантские множества можно обойти достаточно быстро.

Но как насчёт основных операций? Как насчёт добавления элементов в одно из множеств, удаления элементов, поиска конкретного элемента в множестве? Все эти операции выполняются за константное время или 0(1). Это очень мощный инструмент, и это значит, что множества могут быть даже более удобной структурой данных, чем словарь или хэш.

Но подождите, почему все операции с множествами выполняются так быстро? Как это возможно? Как оказалось, под капотом множества представляют собой хэши. Теперь вся информация собирается воедино. С хэш-таблицами знакомо большинство программистов, но почему с их помощью так удобно реализовывать множества?

Это возможно благодаря нескольким факторам. Первый: в хэш-таблицах каждый элемент всегда имеет уникальный индекс. Это очень хорошо с точки зрения реализации множеств, так как множества могут включать только уникальные элементы. Второй фактор: в хэш-таблицах порядок элементов не имеет значения. В множествах порядок элементов тоже не имеет значения. Наконец, хэш-таблицы обеспечивют константное время доступа 0(1). Это идеально для выполнения базовых операций с множествами.

Читайте также

Haskell — язык, позволяющий глубже понять программирование. Как он устроен и почему его выбирают разработчики?

Заключение

Теория множеств используется в разных областях computer science. Это важная для программистов концепция, понимание которой помогает разработчикам эффективно работать с данными.

Адаптированный перевод статьи Set Theory: the Method To Database Madness by Vaidehi Joshi.

Множества и бинарные отношения

http://amstarm.ru/warehouse/Akimova/%D0%9C%D0%95%D0%A2%D0%9E%D0%94%D0%98%D0%A7%D0%9A%D0%98/%D0%91%D0%B5%D0%BB%D1%8F%D0%B5%D0%B2%D0%B0%20%D0%A2.%D0%AE.%20%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B0%20%D0%B8%20%D0%B1%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D1%8B%D0%B5%20%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D1%8F.pdf

Учебное пособие – очень доступно изложен материал!
Множества и бинарные отношения
для обучающихся специальности «Программирование в компьютерных системах»
Автор: Беляева Татьяна Юрьевна