Нелинейные структуры данных

Эти структуры данных отличаются от массивов и списков. Массивы — это последовательные контейнеры. Элементы в них расположены по порядку. При добавлении нескольких элементов в определённом порядке, они остаются в этом порядке.

Нелинейные структуры данных необязательно остаются в том порядке, в котором их добавляют. При добавлении или удалении элементов может измениться порядок других элементов. Внутри они состоят из деревьев и куч, рассмотренных в предыдущей части.

Существует много вариаций таких структур данных. Самыми базовыми являются словарь данных, а также упорядоченное и неупорядоченное множества.

Словарь данных

Обычный словарь состоит из набора слов (ключа) и определения (значения). Поскольку ключи находятся в алфавитном порядке, любой элемент можно найти очень быстро.

Если словари не были бы отсортированы, то поиск слов в них был невероятно сложным.

Существует два основных способа сортировки элементов в словаре: сравнение или хэш. Традиционное упорядочивание сравнением обычно более интуитивно. Оно похоже на порядок бумажном словаре, где всё отсортировано по алфавиту или по числам.

При сортировке элементов таким образом может потребоваться функция сравнения. Обычно эта функция по умолчанию является оператором «меньше чем», например $a < b$.

Второй способ сортировки элементов — использование хэша. Хэш — это просто способ преобразования блока данных в одно число. Например, строка «blue» может иметь хэш 0xa66b370d, строка «red» — хэш 0x3a72d292. Когда словарь данных использует хэш, он обычно считается неотсортированным. В действительности он всё равно отсортирован по хэшу, а не по удобному человеку критерию. Словарь данных работает тем же образом. Есть небольшая разница в скорости между использованием словарей с традиционной сортировкой и сортировкой по хэшу. Различия так малы, что их можно не учитывать.

В C++ есть семейство контейнеров *map*/*mutimap* или *unordered_map*/*unordered_multimap*. В Java семейство называется *HashMap*, *TreeMap* или *LinkedHashMap*. В C# это *Dictionary* или *SortedDictionary*. Каждое из них имеет собственные особенности реализации, например сортировка по хэшу или сравнением, допущение дубликатов, но в целом концепция одинакова. Заметьте, что в каждой из стандартных библиотек имеется их упорядоченная версия (в которой задаётся сравнение) и неупорядоченная версия (где используется хэш-функция). После добавления элементов в словарь данных вы сможете изменять значения, но не ключ.

Вернёмся к аналогии с бумажным словарём: можно менять определение слова, не перемещая слово в книге; если изменить написание слова, то придётся удалять первое написание и повторно вставлять слово с новым написанием. Подробности работы вы можете узнать в учебниках. Достаточно знать, что словари очень быстры при поиске данных, и могут быть очень медленными при добавлении или удалении значений.

Упорядоченное и неупорядоченное множество

Упорядоченное множество — это почти то же самое, что и словарь. Вместо ключа и значения в нём есть только ключ. Вместо традиционного словаря со словами и определениями там только слова. Множества полезны, когда вам нужно хранить только слова без дополнительных данных. В C++ семейство структур называется *set*/*multiset* или *unordered_set*/*unordered_multiset*. В Java это *HashSet*, *TreeSet* или *LinkedHashSet*. В C# они называются *HashSet* и *SortedSet*.

Как и в случае со словарями, существуют упорядоченные версии (где задаётся сравнение) и неупорядоченные версии (где используется хэш-функция). После добавления ключа его тоже нельзя изменять. Вместо этого нужно удалить старый объект и вставить новый. Часто они реализуются точно так же, как словарь данных, просто хранят только значение. Поскольку они реализуются так же, то они имеют те же характеристики. В множествах очень быстро ищутся и находятся значения, но они медленно работают при добавлении и удалении элементов.

Заключение

Классы контейнеров словарей данных, упорядоченных и неупорядоченных множеств очень полезны для быстрого поиска данных. Часто они реализуются как деревья или хэш-таблицы, которые очень эффективны в этом отношении. Используйте их тогда, когда требуется один раз создать данные и часто ссылаться на них. Они не так эффективны при добавлении и удалении элементов. Внесение изменений в контейнер может вызвать смещение или изменение порядка внутри него. Если вам необходимо следовать этому паттерну использования, то лучше выбрать упорядоченный связанный список.