

Таблица 1. Время работы операций анализируемых структур данных.

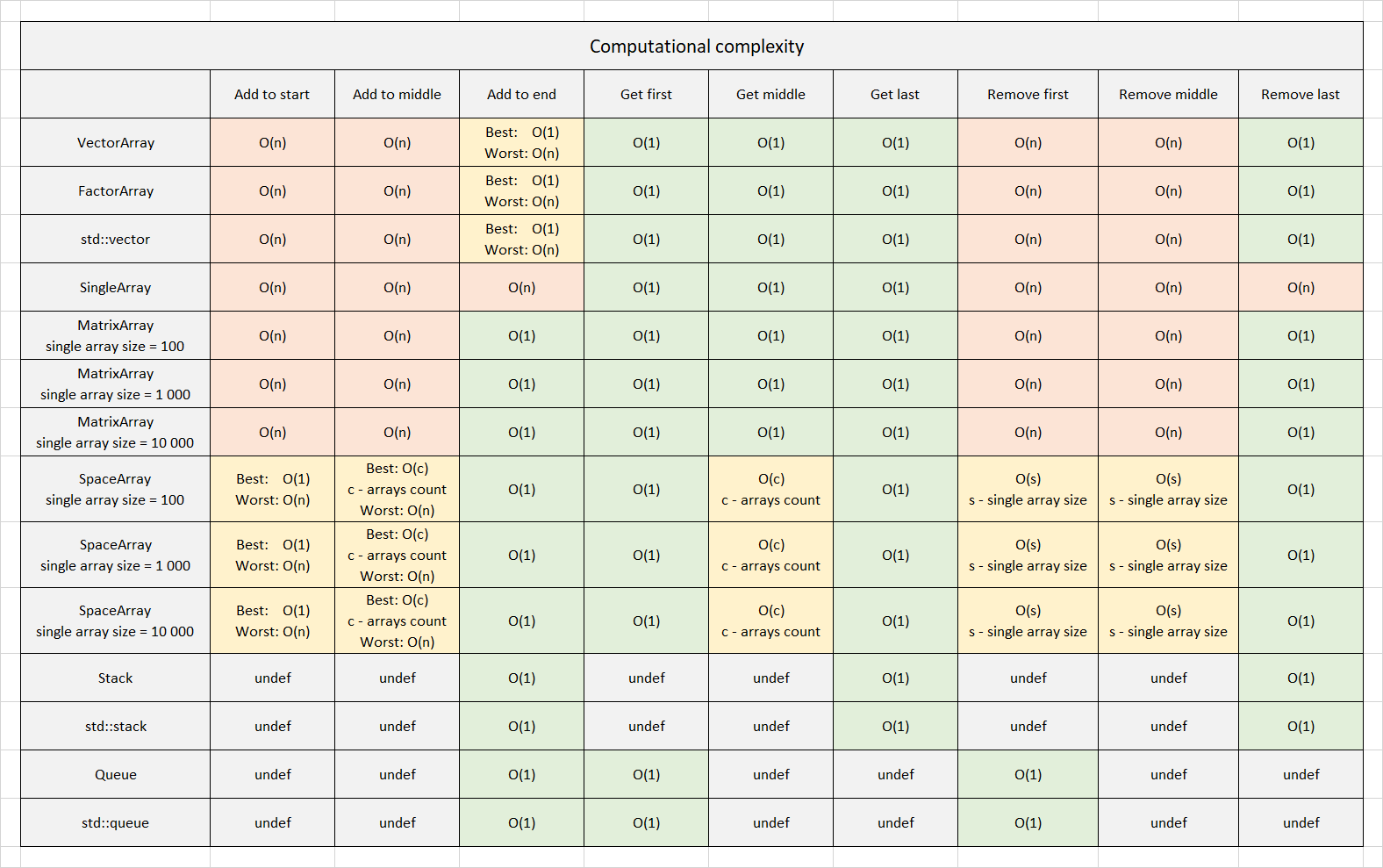


Таблица 2. Вычислительная сложность операций анализируемых структур данных

**VectorArray:**

*Плюсы:*

1. Динамическое изменение размера.
2. Быстрое добавление в конец за O(1) (за исключением добавлений при которых приходится выделять новый участок памяти).
3. Быстрый доступ к любому индексу в массиве для чтения – O(1).

*Минусы:*

1. При добавлении элементов в начало или середину приходится перемещать все элементы массива, что значительно увеличивает время работы алгоритма добавления. Сложность добавления в начало – O(n).
2. Не смотря на быстрое добавление в конец, при заполнении массива эта операция может отработать за O(n) из-за необходимости выделить больший кусок памяти в heap и скопировать в него ранее добавленные елементы.
3. Медленное удаление из начала и середины массива – O(n).

**FactorArray:**

*Плюсы:*

1. Те же плюсы что и у VectorArray
2. При переполнении выделяет больший объем памяти чем VectorArray, в связи с этим массив реже переполняется и нам не приходиться слишком часто выделять новые участки heap, благодаря этому FactorArray выигрывает у VectorArray в скорости добавления в конец.

*Минусы:*

1. Занимает много не используемой памяти.
2. Остальные минусы те же что и для VectorArray

**SingleArray**

*Плюсы:*

1. Одним из немногих плюсов SingleArray является его экономичность. Может использоваться в ситуациях когда на счету каждый байт памяти, при этом элементы добавляются и удаляются очень редко.
2. Доступ к любому элементу массива за O(1).

*Минусы:*

1. При каждом добавлении или удалении элемента необходимо выделять новый участок памяти, в связи с этим добавление и удаление работают в десятки раз медленней чем у VectorArray и FactorArray.

**std::vector**

*Плюсы:*

1. Те же плюсы что и у VectorArray.

*Минусы:*

1. Те же что и у VectorArray.

**MatrixArray**

*Плюсы:*

1. При переполнении массива не нужно каждый раз выделять больший объём памяти и копировать ранее добавленные элементы.

*Минусы:*

1. Даже при достаточно большом размере подмассива (10 000 элементов) MatrixArray показал более скромные результаты чем VectorArray и FactorArray при добавлении 1 миллиона элементов. Полученный результат связан с большим количеством обращений к памяти для выделения новых подмассивов.
2. Также к минусам можно отнести немного усложнённую операцию поиска индекса в MatrixArray.

**SpaceArray**

*Плюсы:*

1. Удаление элемента из любой части массива затрагивает только подмассив из которого производиться удаление.
2. Благодаря сохранению подмассивов в полупустом состоянии добавление элементов также затрагивает только подмассив.

*Минусы:*

1. Одним из минусов является усложнённая реализация, из-за разной заполненности каждого из подмассивов возникает необходимость в хранении дополнительной информации.
2. Усложнённый поиск подмассива и индекса для добавления или удаления элемента

**Stack**

*Плюсы:*

1. Быстрые добавление и удаление не затрагивают остальные элементы стека.

*Минусы:*

1. Специфическая структура данных (LIFO) не имеющая возможности доступа к элементам по индексу. Тем не менее эту особенность сложно назвать минусом, так как принцип LIFO используется во многих алгоритмах.
2. Невозможно удалить или прочитать элемент без удаления всех предшествующих элементов.

**std::stack**

*Плюсы:*

1. Стоит отметить, что по сравнению с самостоятельно реализованным стэком std::stack показал лучшие временные результаты при добавлении 1 миллиона элементов. Этот результат связан с использованием массива внутри std::stack, тогда как самодельный стэк реализован на основе односвязного списка, что является причиной слишком частого выделения новой памяти
2. Остальные плюсы те же что и у Stack

*Минусы:*

1. Те же что и у Stack

**Queue**

*Плюсы:*

1. Как и в stack - добавление и удаление не затрагивают остальные элементы и работают за O(1).

*Минусы:*

1. Структура данных работающая по принципу (FIFO), предоставляет доступ только к первому элементу очереди. Как и в случае со стеком, данную особенность сложно отнести к минусам.

**std::queue**

*Плюсы:*

1. Ситуация с быстродействием та же что и у stack (std::queue показала себя лучше чем самодельная очередь в связи с внутренней реализацией).
2. Остальные плюсы те же что и у Queue

*Минусы:*

1. Те же что и у Queue.