«Лабораторная работа "Сравнение скорости работы динамического массива и односвязного списка"»

Алгоритмы и структуры данных

Ахатулы Жанарыс

2022

Содержание

1 Подготовка

Необходимо написать два стека: на списке и на массиве.
Необходимо для динамического массива и списка реализовать следующие функции:

1. **Construct(uint64_t, start_size)** — создает массив заданного размера, который потом будет изменяться в ходе исполнения (только для массива)
2. **Push(int value)** — добавляет элемент в конец контейнера
3. **Top()** — возвращает последний элемент
4. **Pop()** — удаляет последний элемент, не возвращая его
5. **Destroy()** — уничтожает контейнер, освобождая память

2 Оптимальная политика реаллокации массива

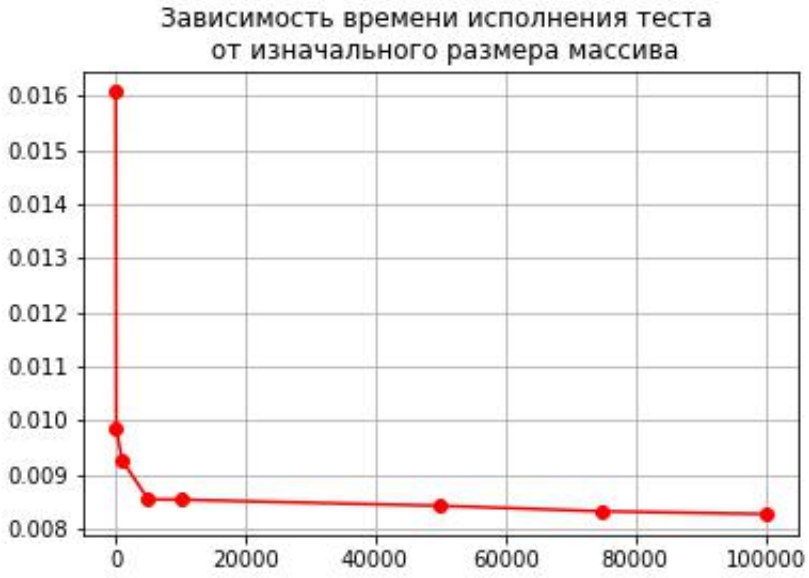
2.1 Поиск изначального оптимального размера

При создании динамического массива у него имеется изначальный размер. Ппредлагается рассмотреть следующие варианты: 10, 100, 1000, 5000, 10000, 50000, 75000, 100000. Далее необходимо провести тест с политикой реаллокации "увеличение размера на изначальную длину". То есть, выбрав параметр 1000, массив при расширении будет увеличиваться на тысячу.

Тест: запушить 10^6 интов в цикле. Далее удалить половину элементов и снова довести размер массива до миллиона. Засечь время исполнения теста.

Далее необходимо построить график зависимости времени теста от изначального размера. Минимум на этом графике и будет изначальным размером массива, с которым будем работать далее.

Результаты выполнения:



На графике по началу виден резкий спад по времени по мере увеличения начального размера, предположительно из-за более редкого обращения к функции realloc, но последние три значения разнятся на очень небольшую величину. Из этого можно предположить, что для последующих начальных размеров, больше 100000 и не сильно отличающихся друг от друга у нас будет равномерное убывание по времени.

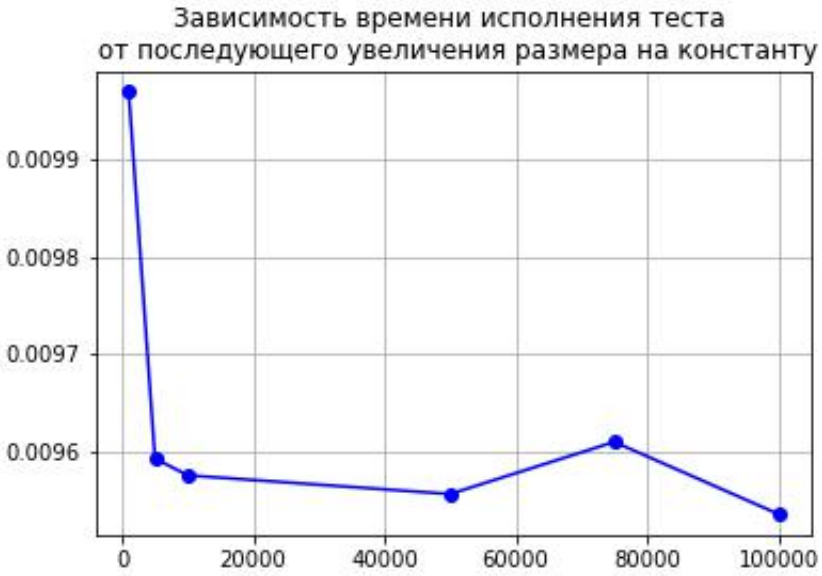
Вывод по пункту 2.1: По графику можно судить, что самым оптимальным начальным размером для стэка на массиве является значение 100000. Судя по графику можно предположить, что при рассмотрений больше числа данных, самым оптимальным будет максимальное среди всех значении, также по причине более редкого обращения к функции realloc.

2.2 Поиск оптимальной политики реаллокации

2.2.1 Увеличение на константу

Необходимо провести тот же тест, что и в пункте 2.1. с изначальным размером, выбранным там же. Увеличивать размер массива надо на одну из следующих констант: 1000, 5000, 10000, 50000, 75000, 100000. Далее построить график времени теста от константы реаллокации.

Результаты выполнения:



На графике также видно, что по началу происходит резкое убывание по времени по мере увеличения значения, также предположительно из-за уменьшения количества обращений к функций realloc, но предпоследнее значение больше по времени двух своих соседей. Из этого невозможно предположить как будут вести себя последующие значения больше 100000, хотя логично также предположить равномерно убывающую прямую.

Вывод по пунтку 2.2.1: По графику можно судить, что самой оптимальной политикой реаллокации на константу, при условии начального размера равного 100000, является значение 100000. Также из графика можно допустить гипотезу о том, что при рассмотрении большего количества данных по политике реаллокации на константу, будет также являться максимальное среди всех значении, аналогично по причиние редкого обращения к функции realloc.

2.2.2 Увеличение в константу

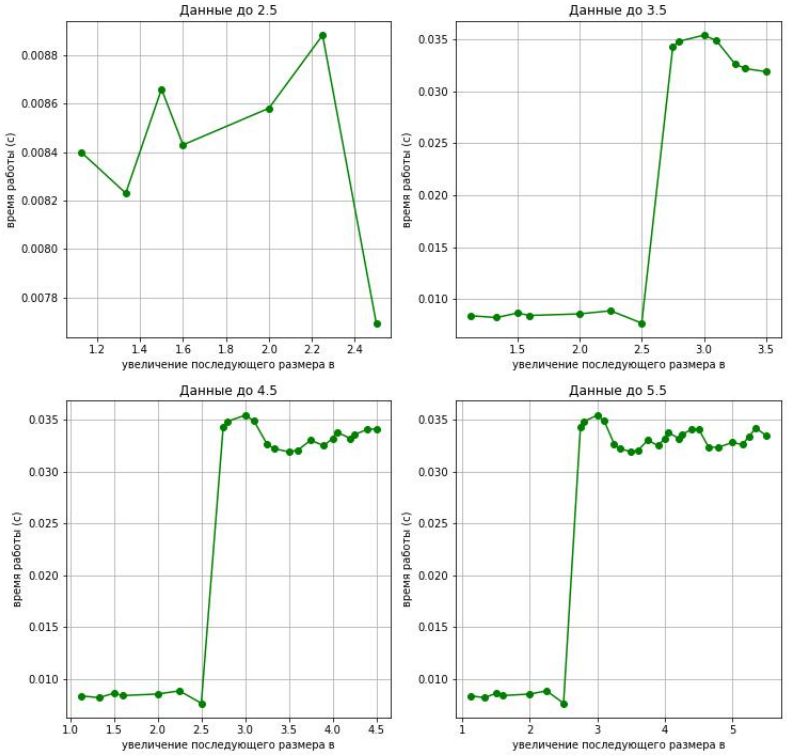
Необходимо провести тот же тест, что и в пункте 2.1. с изначальным размером, выбранным там же. Увеличивать размер массива надо в одну из следующих констант: 9/8, 4/3, 3/2, 8/5, 2, 9/4, 10/4. Далее построить график времени теста от константы реаллокации. Исходя из выбранных параметров, реализовать массив с оптимальной по времени политикой реаллокации.

Результаты выполнения:

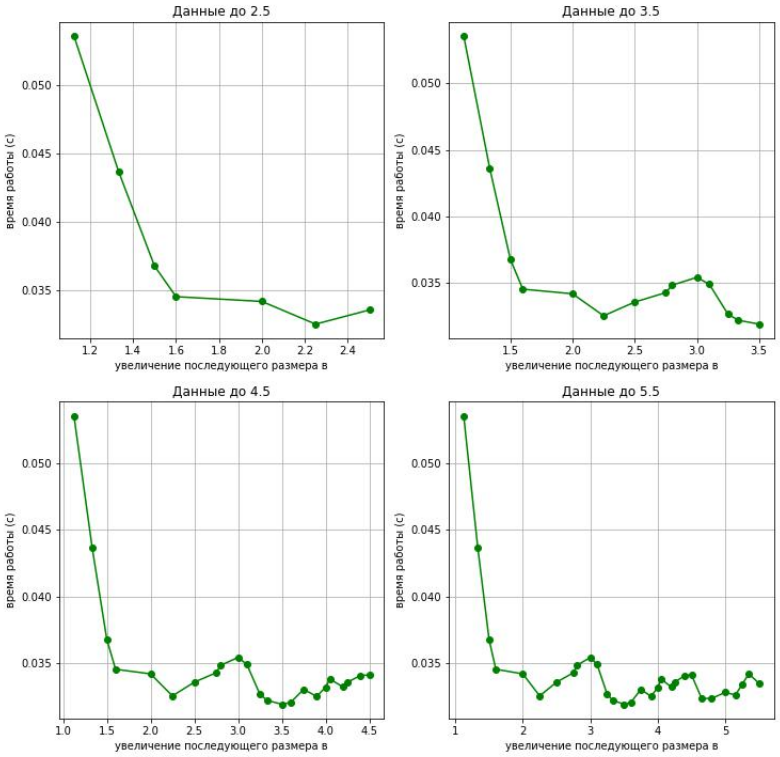
График хаотичен и нельзя только предполагать, что это закономерный итог частого обращения к realloc, хотя однозначным победителем является 10/4. Можно только предположить, что здесь совокупность факторов, таких как начальный размер массива, константы, в которые нужно увеличивать размер, тип компилятора, процессора и т.д.

Поэтому попробуем взять большее количество данных, и посмотрим на результаты:

Первый набор графиков с сохранением результатов по времени из графика выше:



Второй набор графиков уже с полным перетестированием всех параметров:



В первом наборе можем заметить сильную разницу между например первым и последующими графиками, причем было также планомерное увеличение значения 'увеличить в некоторую константу', в отличий от второго графика, который как и других тестах показывает более менее равномерный спад и варьируются данные от 3.5 до 5.5 приблизительно на одном уровне. Поэтому имеется склонность более ко второму графику, т.е. при последующих перетестировках с увеличением данных будет схожость именно со вторым набором.

Вывод по пунтку 2.2.2: По графику можно судить, что самой оптимальной политикой реаллокации в константу от текущего размера, при условии начального размера равного 100000, является значение 10/4. А также если судить по второму набору графиков, самым оптимальным будет значение $14/4 = 3.5$. Здесь, в отличий от других результатов, где самым оптимальным по времени было самое максимальное значение, самое менее времязатратное является серединное значение, и из этого предполагается, что этот оптимум является таковым только для конкретных проведенных теестов, а при более нагруженных тестах результаты будут схожими с предыдущими тестами, т.е. также самым менее времязатратным будет максимальное значение.

Вывод по пункту 2.2: Сравнивая время работа самого оптимального увеличения 'на' и 'в' константу от текущего размера, можно сделать вывод, что самое оптимальное увеличение будет **увеличение в 10/4 от текущего размера**.

Вывод по пункту 2: В этом пунтке были найдены самые оптимальные параметры для стэка на массиве ввиде: начальный размер, последующее реаллоцирование на константу и последующее реаллоцирование в константу от текущего размера. А также было произведено сравнение последующего реаллоцирования 'на' и 'в' константу от текущего размера и выбрано самое оптимальное из них. По результатам тестирования оказалось, что самым оптимальным параметром ввиде начальный размер является значение в 100000, для последующее реаллоцирования на константу является аналогично значение в 100000 и для последующего реаллоцирования в константу от текущего размера является значение 10/4. По результатам сравнения двух увеличений 'на' и 'в', прорезюмировалось, что самое оптимальное является увеличение в 10/4 от текущего размера. Таким образом выяснилось, что для стэка на массиве лучшими показателями являются: начальный размер - 100000, увеличение 'в' константу от текущего размера - 10/4.

3 Сравнение для стэка

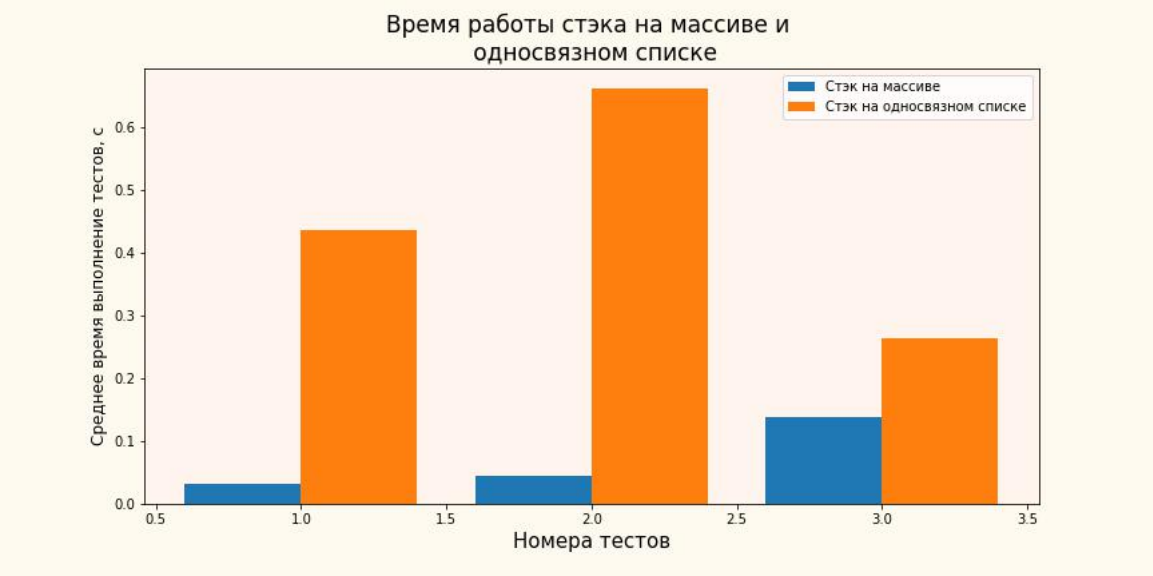
Необходимо реализвать два стека: **ArrayStack** и **ListStack**. Далее провести для них следующие тесты (каждый провести хотя бы по три раза и усреднить время тестов):

1. Моделирование стека вызова функций. Надо запушить 10^6 интов, потом удалить половину элементов и запушить четверть - останется 750000 интов. Повторять удаление половины и вставку четверти, пока не останется в стеке меньше 100000 элементов (получится 9 итераций).

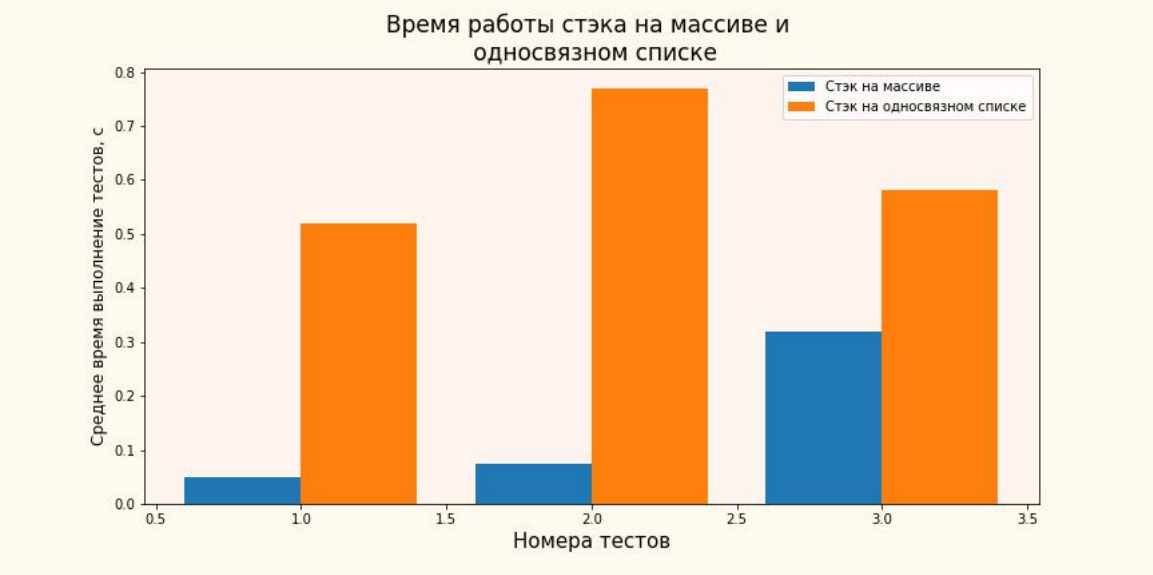
2. Моделирование нагруженного стека вызова функций. Надо запушить 10^6 интов, потом 100 раз удалить 10000 элементов и добавить столько же. Далее как в первом тесте провести 9 итераций удаления-вставки и снова 100 раз удалить 10000 элементов и добавить столько же.

3. Моделирование случайной последовательности команд. Надо научиться средствами языка генерировать случайные числа из множества $\{1, 2\}$. Далее сначала довести размер стека до миллиона, а потом выполнить миллион инструкций следующего вида: каждое выпадение единицы добавлять элемент, а на двойку - попать из стека. Засекать время после доведения размера стека до миллиона.

Результаты сравнения:



Из пункта 1.2.2 выяснилось, что по результатам второго набора графика самым оптимальным является значение 3.5, поэтому будет логичным протестировать два стэка, но со вторым параметром 'увеличение в' 3.5 раза от текущего размера стэка на массиве и показать **результаты сравнения:**



По двум гистограммам видна сильная разница работы по времени двух стэков. Однако стоит заметить, что по времени стэк на массиве, с последующим увеличением размера в 3.5 раза, работает чуть дольше стэка, с последующим увеличением размера в 2.5 раза.

Итоговый вывод: В лабораторной работе 'Сравнение скорости работы динамического массива и односвязного списка' были выполнены две части. В первой части 'Нахождение оптимальной для платформы политики реаллокации массива' были по итогу найдены оптимальный изначальный размер массива для стэка равного 100 000 и оптимальный размер для последующего реаллоцирования в 10/4 от изначального размера.

Во второй части 'Сравнение для стека' было сравнение времени работы стэка реализованного на динамически выделяющемся массиве и односвязном списке. Можно резюмировать, что стэк, реа-

лизованный на массиве, будет намного оптимальнее и намного менее времязатратнее для разных операции, чем стэк на односвязном списке. Здесь имеется гипотеза о том, что такой результат получился из-за того, что процессор при работе со стэком на массиве подгрузил в кэш линию некоторый (возможно весь) участок массива и т.к. массив - последовательный контейнер, соотственно обращение к массиву быстрее нежели к односвязному списку, где происходит хаотичное обращение к данным.