

Для начала рассмотрим теоретические ассимптотики заполнения стеков на разных структурах данных.

Очевидно, что массив — самая медленная реализация стека. При каждом добавлении элемента необходимо заново выделять память и перекопировать весь стек. Таким образом, сложность добавления элемента — $O(N)$. Тогда сложность заполнения всего массива:

$$\sum_{i=1}^N i = \frac{N(N+1)}{2} \sim N^2 \text{ при } n \rightarrow \infty, \quad (1)$$

откуда сложность заполнения всего массива — $O(N^2)$.

Для списка все просто: добавление каждого элемента имеет сложность $O(1)$, откуда заполнение всего стека — $O(N)$.

Для амортизированного списка немного сложнее. Каждую степень двойки (в нашем случае) он переписывается. Тогда сложность можно написать:

$$N + \sum_{i=1}^{\log_2(N)} 2^i, \quad (2)$$

что можно оценить как:

$$N + \sum_{i=1}^{\log_2(N)} 2^i < N + N(1 + \frac{1}{2} + \frac{1}{4} + \dots) = N(1 + \sum_{i=0}^{\infty} 2^{-i}) = 3N. \quad (3)$$

Т. е. сложность заполнения так же линейная $O(N)$.

Результаты измерений

Для массива при стандартной линеализации логарифмом получим степень ≈ 1.7 , что не вполне соответствует теории. Возможная причина — полиномиальная сложность с членом N , существенным при меньших N . Тогда логично сделать линеализацию $t(N^2 + N)$.

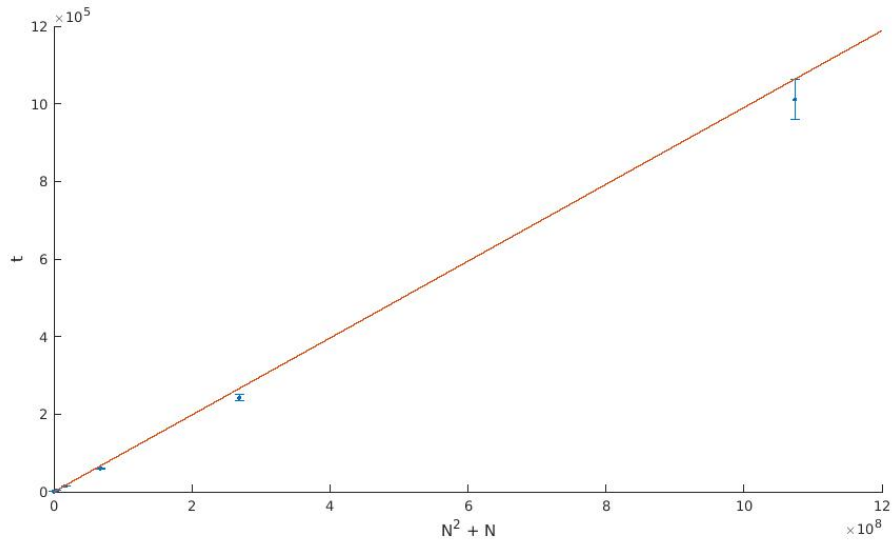


Рис. 1: Линеализация зависимости $t(N)$.

Полученный коэффициент в уравнении $t = k(N^2 + N)$ по методу хи-квадрат $k = (1 \pm 1)10^{-3}$. Значение функции хи-квадрат — $\chi^2 = 55$, что, вообще, говорит о плохой аппроксимации. Полученная зависимость:

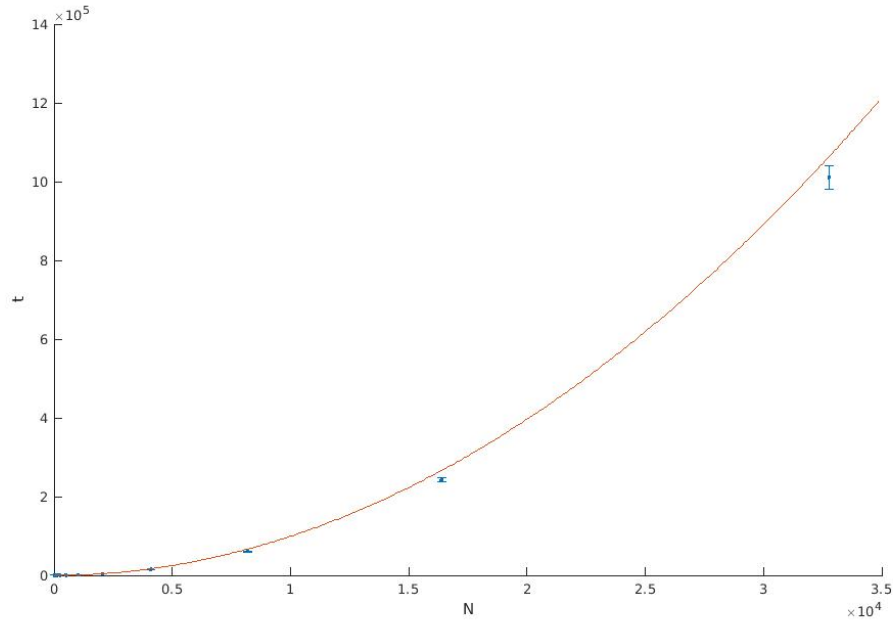


Рис. 2: Полученная зависимость для массива.

Теперь рассмотрим список и амортизированный массив. Их будем аппроксимировать логарифмом по основанию 2.

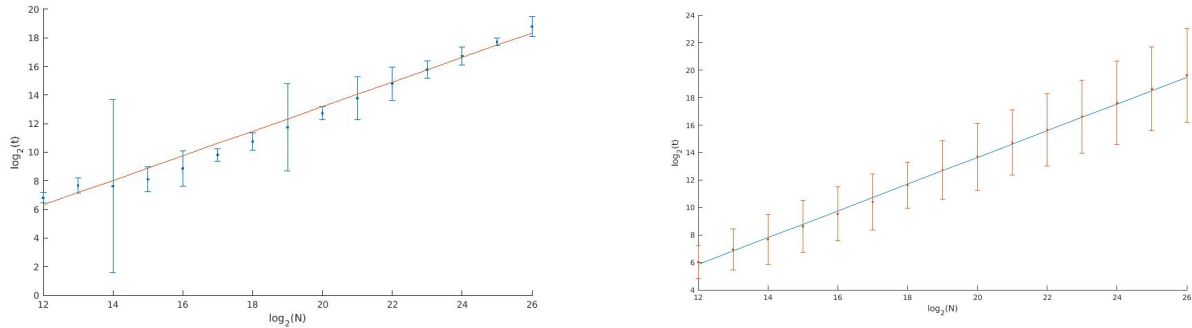


Рис. 3: Справа — линеализация логарифмом времени заполнения стека на списке, слева — на амортизированном массиве.

Получим линейные уравнения для логарифмических осей вида $\log_2(t) = k \log_2(N) + b \Rightarrow t = 2^b N^k$.

Для массива $k = (86 \pm 3)10^{-2}$, $b = (-4.0 \pm 0.6)$. Для списка — $k = (0.97 \pm 0.13)$, $b = (-5.8 \pm 2.3)$

Откуда полученные зависимости:

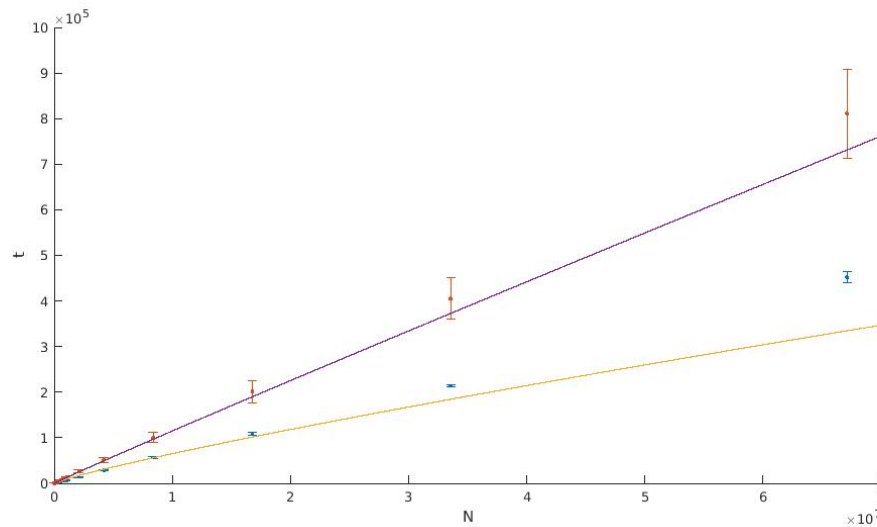


Рис. 4: Полученные зависимости для односвязного списка (отмечена фиолетовым) и амортизированного массива (отмечен желтым).

Заметим, что для списка показатель хи-квадрат $\chi^2 \approx 0.01$, что говорит о больших погрешностях. Причина — постоянное выделение новой памяти — операции, время которой может произвольно меняться. Заметим, что использование массива при этом намного более эффективное по времени, несмотря на одинаковую теоретическую асимптотику. Это говорит о том, что выделение памяти (реже используемое при заполнении массива) — ‘дорогая’ по времени операция. Для массива же аппроксимация удачна — $\chi^2/\text{approx} 1.2$, несмотря на показатель степени, отличающийся от теоретического на более, чем 2σ .

Сравним полученные зависимости:

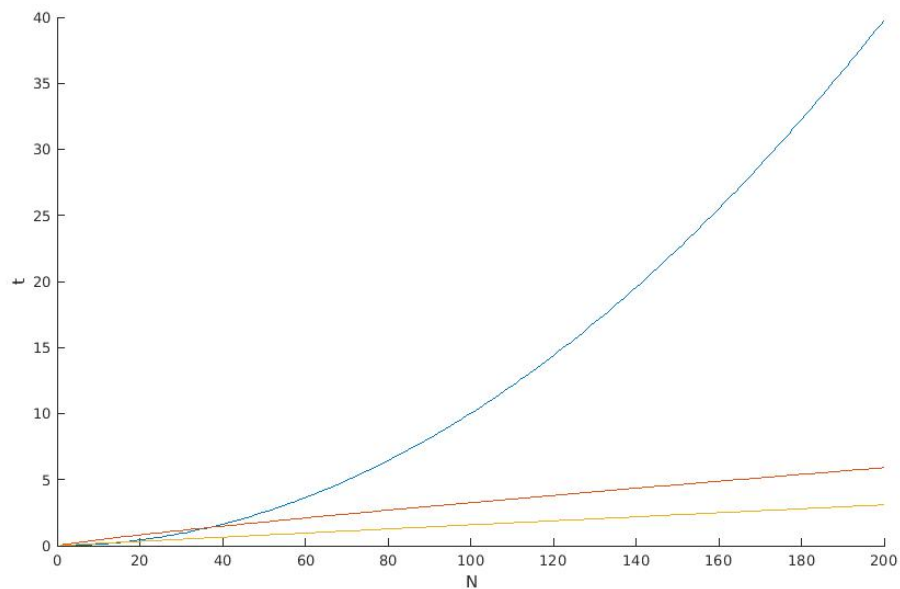


Рис. 5: Сравнение времени заполнения стеков разными структурами данных. Синий — массив. Оранжевый — список. Желтый — амортизированный массив.

Заметим, что полученный результат (показатель степени) немного меньше единицы. Возможно, это связано с факторами, вызванными выделением новой памяти при перекопировании и т. д. Интересно тогда сравнить амортизированный массив со статическим, в котором всю память мы выделим сразу. При этом не будет так же и многократного копирования и мы будем по-сути лишь заполнять массив.

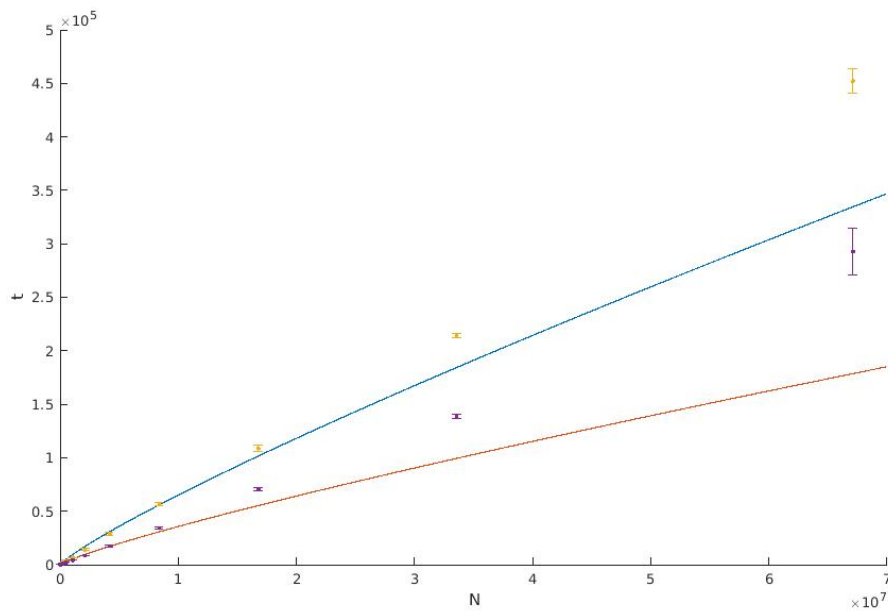


Рис. 6: Сравнение амортизированного массива (отмечен оранжевым) и статического (голубым).

Получим, что и в этом случае показатель степени отклоняется от единицы. Полученные результаты отклоняются от теоретических. Все это говорит о факторах, влияющих на время выполнения программы и не зависящих от неё.