Амортизационный анализ алгоритмов

- Амортизационный анализ используется в случаях, когда различные операции происходят с значительно различающейся частотой, что может влиять на точность оценки “традиционными” методами
- В качестве примера здесь и далее будем рассматривать бинарный счетчик, устроенный следующим образом: пусть есть массив A размера k, состоящий только из нулей и единиц. Сопоставим ему число x, являющееся его десятичным представлением(младший бит A[0], старший - A[n-1]). Операцию инкремента определим как увеличение соответствующего двоичного числа на 1. Например так будет выглядеть увеличение счетчика с 0 до 16:

Значение счетчика	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1
10	0	0	0	0	1	0	1	0
11	0	0	0	0	1	0	1	1
12	0	0	0	0	1	1	0	0
13	0	0	0	0	1	1	0	1
14	0	0	0	0	1	1	1	0
15	0	0	0	0	1	1	1	1
16	0	0	0	1	0	0	0	0

- Зададимся теперь целью проанализировать, какая сложность будет у исполнения n последовательных операций над нашим счетчиком. Грубый подход даст оценку O(nk), поскольку для счетчика с максимальным значением стоимость одной операции(соответствующей обнулению) и будет O(k)
- Интуитивно ясно, что оценку можно улучшить, поскольку чем “тяжелее” операция, тем реже она происходит, что “дает надежду”, что в среднем сложность будет меньше, чем O(nk). Далее попробуем формализовать наши предположения

Банковский метод

- В банковском методе(или в методе бухгалтерского учета) каждой операции присваивается так называемая амортизированная стоимость, которая может отличаться от фактической стоимости. В момент исполнения операции ее амортизированная стоимость сравнивается с фактической и если первая оказывается больше, то ее излишки идут в “кредит”, который после можно использовать для “оплаты” ситуаций, когда фактическая стоимость операции, напротив, больше амортизированной.
- Амортизационную стоимость операций стоит выбирать из расчета, что для любой последовательности операций сумма их амортизационных стоимостей должна быть выше суммы их фактических стоимостей, откуда следует требование, что в любой момент времени сумма кредита должна быть неотрицательной
- Теперь можем, используя соображения выше оценить более точно сложность операции инкремента для нашего счетчика: пусть амортизационная стоимость операции, когда бит становится равным единице, равна 2 монетам. Тогда при установке каждого бита в 1 тратится 1 монета на то, чтобы произвести непосредственно установку и откладывается в кредит 1 монета для последующей оплаты установки бита в 0. Поскольку бит не может быть установлен в 0 без предыдущей установки в 1, требование на неотрицательность кредита выполнено
- Поскольку в ходе инкрементирования в итоге в единицу обращается не более одного бита, а все обращения в ноль могут быть оплачены за счет накопленных кредитов(что показано выше), итоговая амортизированная сложность одного инкремента составляет O(1), а значит n операций выполняются за O(N)

Метод потенциалов

- Другим методом проведения амортизационного анализа является метод потенциалов. В нем мы вводим функцию $\Phi(d_i)$, отображающую i-тое состояние системы на множество действительных чисел(или другое множество, изоморфное R по сложению и сравнению). Амортизированная стоимость для i-той операции в данном случае вводится как $c_i^{\text{амортизированная}} = c_i^{\text{фактическая}} + \Phi(d_i) - \Phi(d_{i-1})$, откуда, очевидно, следует, что полная амортизированная стоимость равна полной фактической стоимости плюс разности потенциалов в финальном и начальном состоянии.

- Отсюда видно, что функцию потенциала надо определить таким образом, чтобы $\Phi(d_n) \geq \Phi(d_0)$, или, поскольку обычно неизвестно, какое количество операций будет произведено $\Phi(d_i) \geq \Phi(d_0)$ для любого i от 0 до n,
- Оценим теперь амортизированную сложность инкрементирования бинарного счетчика методом потенциалов: определим потенциал состояния как количество единиц в нем. Пусть i-тая операция обнуляет t_i бит. Тогда фактическая сложность не превышает $t_i + 1$. Если $\Phi(d_i) = 0$, то $t_i = k$, иначе $\Phi(d_i) = \Phi(d_{i-1}) - t_i + 1$. Итого имеем $1-t_i \geq \Phi(d_i) - \Phi(d_{i-1})$. Отсюда $2 \geq c_i^{\text{амортизационная}}$, откуда сразу получаем, что общая сложность равна O(N)