1) You are working on an algorithm, which is adding rows and columns to a matrix. Each call to an add function costs $i + c$ where i is the i-th call of the function and c is some constant. Every call where $i = k^2$ for some k costs i^2. Meaning the cost function is :

$$c_i = \begin{cases} i + c & ; i \neq k^2, k \in \mathbb{N} \\ i^2 & ; i = k^2, k \in \mathbb{N} \end{cases}$$

what is the amortized cost of this function?

Sum:

Cost of n operations:

$$\text{Cost} = \sum_{i=1}^{n} i+c \; + \sum_{i=1}^{\sqrt{n}} \underbrace{-i^2 - c}_{\text{subtract squares}} + \underbrace{i^4}_{\text{add squares}} = (n-\sqrt{n})c + \frac{n^2+n}{2} - \frac{2\sqrt{n}^3 + 3\sqrt{n}^2 + \sqrt{n}}{6} + \frac{6\sqrt{n}^5 + 15\sqrt{n}^4 + 10\sqrt{n}^3 - \sqrt{n}}{30}$$

$$= \boxed{O(n^{2,5})} \text{ for n operations}$$

Per operation:

$$\frac{\text{Cost}}{n} = \boxed{O(n^{1,5})} \text{ per operation}$$

http://www.math.com/tables/expansion/power.htm

2) **Exercise 2: Amortization**

You are working on a dynamic table which will only support inserts. Instead of doubling table size when it is full you decide to increase it for 10% only. Is amortized cost of insert still constant? Prove using potential method.

Let:

l = "current number of elements in the array"

S = "siz of the array"

$\Phi(T_i) = c\,l_i + d\,s_i$; for some constants c & d

Inserting in non-full array: $l_{i-1} = l_i - 1$, $s_i = s_{i-1}$, $c_i = 1$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$
$$= 1 + c\,l_i + d\,s_i - c(l_i - 1) - d\,s_i$$
$$= 1 + c \quad \Rightarrow \quad c \geq -1$$

Inserting in full array: $l_{i-1} = l_i - 1$, $s_i = 1.1 \cdot s_{i-1} = 1.1 \cdot (l_i - 1)$, $c_i = l_i$

$s_{i-1} = l_{i-1}$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1})$$
$$= l_i + c\,l_i + 1.1\,d(l_i - 1) - c(l_i - 1) - d(l_i - 1)$$
$$= l_i + \cancel{c\,l_i} + 1.1\,d\,l_i - 1.1\,d - \cancel{c\,l_i} + c - d\,l_i + d$$
$$= \underbrace{l_i(1 + 1.1\,d - d)}_{=0} \underbrace{- 0.1\,d + c}_{} \geq 0$$

$\Rightarrow 1 + 0.1\,d = 0$ $\qquad 1 + c \geq 0$

$d = -10$ $\qquad c \geq -1$

$\Phi(D_i) \geq 0$: $\quad c\,l_i - 10 \cdot s_i \geq 0$ $\quad s_i \leq 1.1 \cdot l_i$

$c\,l_i - 11\,l_i \geq 0$ $\quad l_i \geq 0$

$l_i(c - 11) \geq 0$

$c - 11 \geq 0$

$c \geq 11$

choosing $c = 11$ $\Rightarrow$ $\Phi(D_i) = 11\,l_i - 10\,s_i$!

$\hat{c}_i = 12 \Rightarrow$ $\boxed{\text{inserting is done in constant time}}$

3) # Exercise 3: Approximation

Suppose you are given a symmetric 4-SAT formula, which is described with 4-CNF formula F with n clauses, each clause consisting of 4 literals. For example: $F = (x_1 \vee x_2 \vee \neg x_4 \vee x_5) \wedge (x_4 \vee \neg x_2 \vee \neg x_1 \vee x_3) \wedge (\neg x_3 \vee x_2 \vee \neg x_5 \vee x_1)$.

In 4-SAT we accept each clause if it evaluates to 1. In symmetric 4-SAT we accept a clause if it evaluates to 1 and a clause in which we negate each literal also evaluates to 1. Or in other words symmetric 4-SAT accepts each clause if it has one literal that assigns to 0 and one that assigns to 1.

We know that symmetric MAX 4-SAT this is NP-complete problem so we make a simple approximation algorithm. In symmetric MAX 4-SAT we try to satisfy as many clauses as possible. Lets set each variable to 0 with probability 0.5 and to 1 with probability 0.5.

Find the approximation factor for this algorithm.

Note: In 4-CNF each clause can not have the same literal twice or have a variable x_i and its negation $\neg x_i$.

n Variables $x_1 \dots x_n$

m clauses

A_i = clause i is satisfied (initial & symetric value)

$$Y_i = I\{A_i\} = \begin{cases} 1 & ; A_i \\ 0 & ; \text{otherwise} \end{cases}$$

<u>$P(A_i)$:</u>

we need to satisfy 1, 2 or 3 variables to satisfy initial or symetric clause.

$$\left.\begin{aligned} 1: &\quad \binom{1}{4} \cdot \frac{1}{2} \cdot \left(\frac{1}{2}\right)^3 = \frac{4}{16} \\ 2: &\quad \binom{2}{4} \cdot \left(\frac{1}{2}\right)^2 \left(\frac{1}{2}\right)^2 = \frac{6}{16} \\ 3: &\quad \binom{3}{4} \left(\frac{1}{2}\right)^3 \left(\frac{1}{2}\right) = \frac{4}{16} \end{aligned}\right\} + \Rightarrow \frac{7}{8} = P(A_i)$$

$$Y := Y_1 + \dots + Y_m$$

$$E[Y] = \sum_{i=1}^{m} E[Y_i] = m \cdot P(A_i) = \frac{7m}{8} \quad \Rightarrow$$

$Y^* \leq m$ (we can satisfy at most m clauses)

$\frac{Y^*}{Y} \leq \frac{m}{\frac{7}{8} \cdot m} = \frac{8}{7} \Rightarrow$ $\boxed{\frac{8}{7}\text{-approx algorithm}}$