Lecture 17 Amortized Analysis

Thursday, November 2, 2017 2:35 PM

- Dynamic array

	array	length	capacity
initial	[ ]	length = 0	capacity = 1
add(1)	[1]	1	1
add(2)	[1 | 2]	2	2 (2×1)
add(3)	[1 | 2 | 3 |]	3	4 (2×2)
add(4)	[1 | 2 | 3 | 4]	4	4
add(5)	[1 | 2 | 3 | 4 | 5 | | |]	5	8 (2×4)

- analyzing running time
  - time of "add" operation
    - $\underline{(2^k+1)}$-th add operation "heavy" / "expensive"

      running time 2^{k+1}

      (allocating an array of size 2^{k+1}
      copy first 2^k elements
      add the (2^k+1)th element)

    - all other add operation "light"

      running time 1 (change length, put the element into an empty slot)

  - aggregate method

    let t_i be running time of the i-th add operation

$$\text{total running time} = \sum_{i=1}^{n} t_i$$

$$= \underbrace{\sum_{\substack{i=1 \\ i \neq 2^k+1}}^{n} t_i}_{\text{light operation}} + \underbrace{\sum_{\substack{k \\ 2^k+1 \leq n}} t_{2^k+1}}_{\text{heavy operation}}$$

$$= \sum_{\substack{i=1 \\ i \neq 2^k+1}}^{n} 1 + \sum_{\substack{k \\ 2^k+1 \leq n}} 2^{k+1}$$

$$= \sum_{\substack{i=1 \\ i \neq 2^k+1}}^{n} 1 + \sum_{\substack{k \\ 2^k+1 \leq n}} 2^{k+1}$$

$$\leq n + (2+4+8+\cdots+2^{l+1})$$

l: largest number s.t. $2^l+1 \leq n$
$\Rightarrow 2^{l+1} \leq 2n$

$$= n + 2^{l+2} - 2$$

$$\leq n + 4n = 5n$$

amortized time $= \dfrac{\text{total time}}{n} = 5 = O(1)$ $\square$

– charging argument

between two heavy operations

2^k+1 to $2^{k+1}+1$

we have $2^{k+1}+1-(2^k+1)-1$ light operations

$= 2^k - 1$

Cost for the $2^{k+1}+1$ operation is 2^{k+2}

$$\frac{2^{k+2}}{2^k-1} \approx 4$$

save 4 units of time for each light operation

when $(2^{k+1}+1)$th operation (heavy) happens

I have saved $(2^k-1)\cdot 4 = 2^{k+2}-4$ units of time

I need to pay 2^{k+2}

so the additional money (time) to pay is just 4.

in summary: $\begin{cases} \text{for light operation: pay 1, save 4} & (1+4=5) \\ \text{heavy : use all saving, pay 4} & (4) \end{cases}$

Amortized cost $\leq 5 = O(1)$