Data Structures and Algorithms

Mohammad GANJTABESH

Department of Computer Science,
School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

mgtabesh@ut.ac.ir

Dept. of Computer Science
University of Tehran

$11 \times 2_m + 1 \times 5_m = 27_m / 12$

Amortized Analysis

Data Structures and Algorithms
Undergraduate course

Mohammad GANJTABESH
mgtabesh@ut.ac.ir

Dept. of Computer Science
University of Tehran

Amortized Analysis

Amortized analysis refers to determining the time-averaged running time for a sequence of operations.

Example

Consider selecting the k^{th} smallest element in an array by sorting the array...

How to perform the amortized analysis?

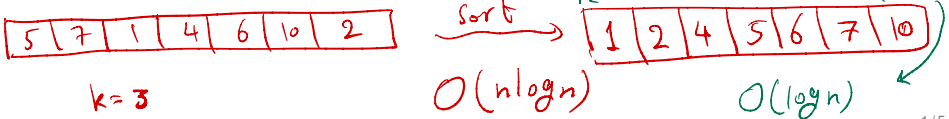
- Aggregate method → average cost per operation.
- Accounting method
- Potential method

assign a cost per operation

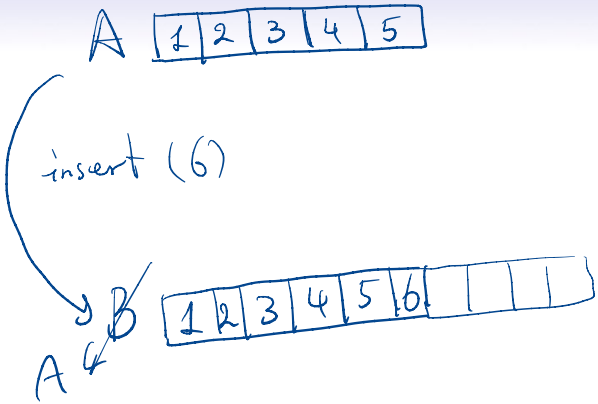
k-Select $O(n)$ ✓

Time Complexity for single execution $= O(n \log n)$

n execs $= O(n \log n) + O(n)$



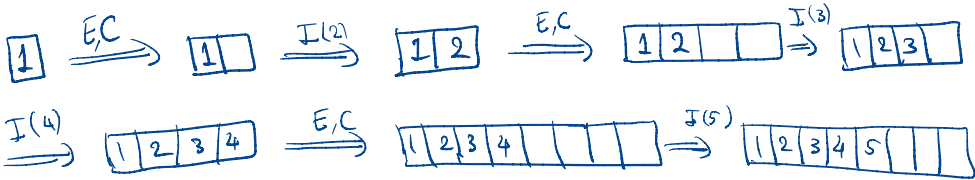
Array :



Example: Dynamic Array

Idea: Double the array size when it becomes full.

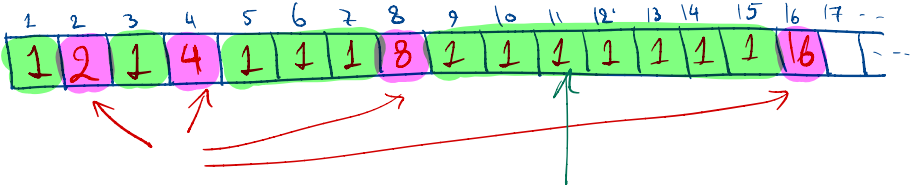
- Insertion Time Complexity: $O(n)$ (which is not tight!)
- Cheap operation: Insertion with $O(1)$
- Expensive operation: array expansion and copy with $O(n)$



Dynamic Array: Aggregate method

$\frac{O(\text{total cost})}{n} = \frac{O(n)}{n} = O(1)$

* simplest way.
* $\frac{O(\text{total cost})}{\#\ \text{operation}}$ should be calculated.



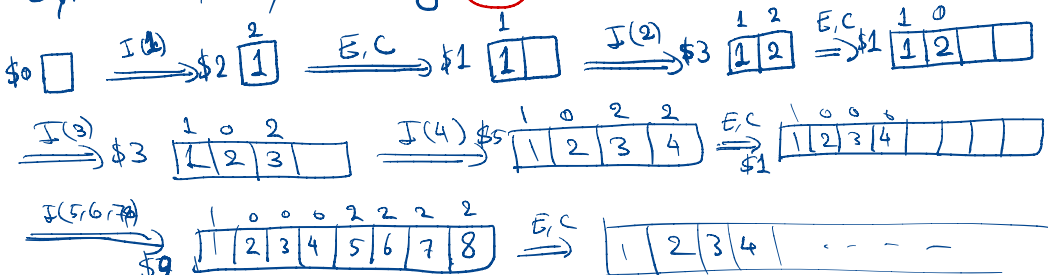
$O(\text{total cost}) = O(\sum \text{cost of } n \text{ ops}) = O(\sum \text{cost of cheap ops} + \sum \text{cost of expansive ops})$

$\Longrightarrow \begin{cases} O(\sum \text{cost of } n \text{ insertions}) = O(n) \\ O(\sum \text{cost of expansions}) = 1+2+4+8+\cdots < 2n = O(n) \end{cases}$

Dynamic Array: Accounting method (Banker's method)

- Assign a cost to each operation (relative to i)
- any surpass goes into a bank.
- Overcharge the cheap ops to save enough money for more expensive ops.

Dynamic Array : charge 3 units per simple op.



Time Complexity for each operation = 3 = O(1)

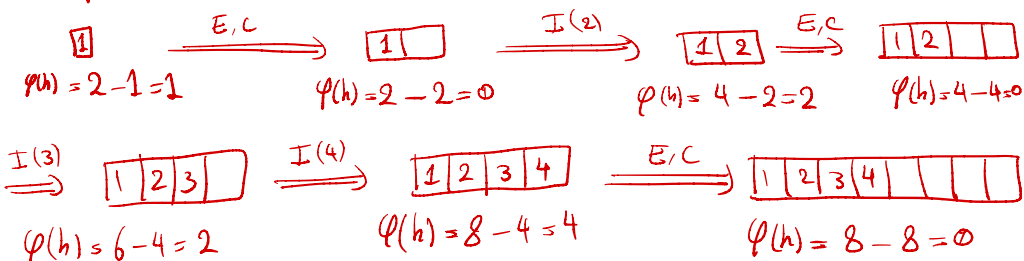
Dynamic Array: Potential method

* Similar to the accounting method.
* The potential (energy) is analogous to the bank.
* Potential must be ≥ 0.
* Potential method involves potential function $\varphi(h)$: (h: state)
  - if the cost of an operation is low, the potential increases, otherwise it decreases.
  - can be used to independently derive potential at any state.
  - " " " " calculate the potential difference (change in cost between two operations).

* Finding the appropriate potential function is the challenging part!

Dynamic Array: Potential method

Example : Dynamic Array. $\varphi(h) = 2n - \text{size of the array}$



In general : $\varphi(h) = 2k - 2^{\lceil \log_2 k \rceil}$

Amortized time of the i^{th} operation h_i : $\underbrace{c_i}_{\text{cost of op.}} + \underbrace{\varphi(h_i) - \varphi(h_{i-1})}_{\text{potential difference}}$

Dynamic Array: Potential method

Cont.

$\Longrightarrow$ {

- cheap / simple operation:

$$c_i + (2i - size) - (2(i-1) - size) = 1 + 2 = 3$$

(after the operation: $2i - size$; before the operation: $2(i-1) - size$; -2)

similar to accounting method

- expensive / complex operation:

$$(i+1) + \varphi(h_i) - \varphi(h_{i-1}) = (i+1) + (2i - 2i) - (2(i-1) - i)$$

$(i+1)$ $\Downarrow$ i copy + 1 insert

$2i$: the size after expansion

i: the size before expansion

$$= i + 1 - 2i + 2 + i$$

$$= 3$$

}

$\Longrightarrow$ Amortized Time = $3 = O(1)$ (in all cases)