Amortized Analysis and Resizing an Array

Alexandra Stefan

Amortized cost

- Usage: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html

"The add operation runs in amortized constant time, that is, adding n elements requires O(n) time."

- Assume a sequence of N operations is performed.
- The ***amortized cost*** of one operation is the total cost of the N operations divided by N

$$amortized\ cost = \frac{cost(op_1) + cost(op_2) + cost(op_3) + \cdots + cost(op_N)}{N}$$

- "Cost" can represent
  - Time complexity
  - Space complexity

(This is the aggregate method for computing the amortized cost. Other methods, not covered: accounting method and potential method.)

Example

Operation	Cost
op1	1
op2	2
op3	3
op4	1
op5	5
op6	1
op7	1

$$totalCost = 1 + 2 + 3 + 1 + 5 + 1 + 1$$

$$amortized\ cost = \frac{totalCost}{7}$$

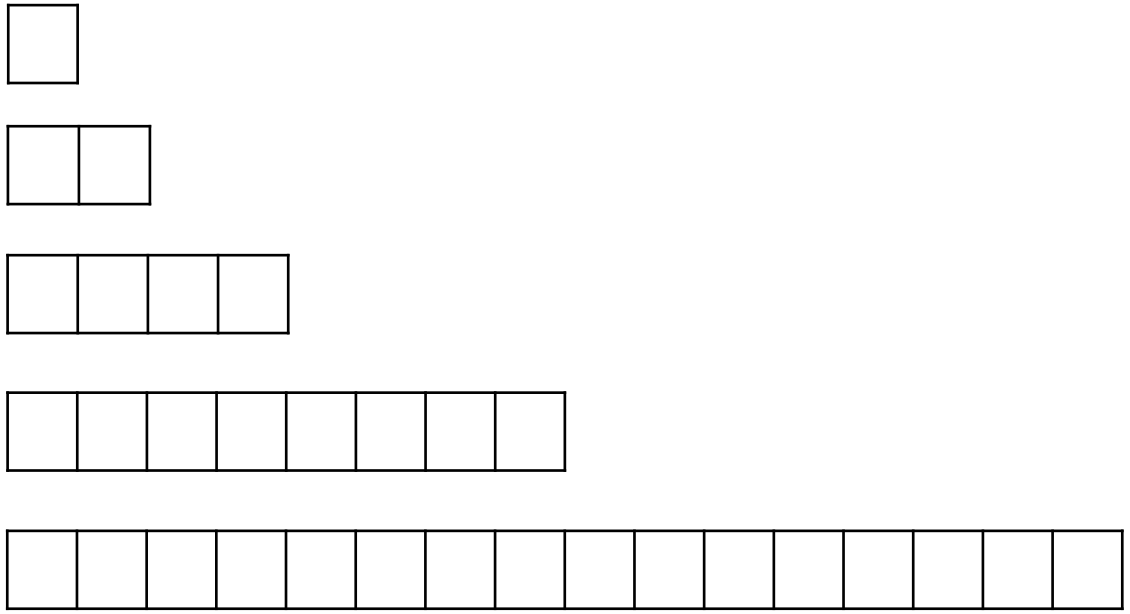
$$= \frac{1 + 2 + 3 + 1 + 5 + 1 + 1}{7} = 1.86$$

Resizing an array (1/3)

- Start with array of capacity 1
- Insert in it numbers: 1,2,3,4,5,6,7,8,9
- If there is space, the cost to write 1 item is 1
- If the array is full => resize
  - allocate double the space,
  - **copy the old items** to the new space
  - write new item (cost 1)

cost	Add
	1
	2
	3
	4
	5
	6
	7
	8
	9

capacity = capacity*2



Calculation for this example :

$$All\ copy\ cost =$$

$$\boldsymbol{Amortized\ cost} = \frac{\qquad}{\qquad} = \frac{\qquad}{\qquad} = \frac{\qquad}{\qquad}$$

Resizing an array (2/3)

- Start with array of capacity 1
- Insert in it numbers: 1,2,3,4,5,6,7,8,9
- If there is space, the cost to write 1 item is 1
- If the array is full :
  - allocate double the space,
  - **copy the old items** to the new space
  - write new item (cost 1)

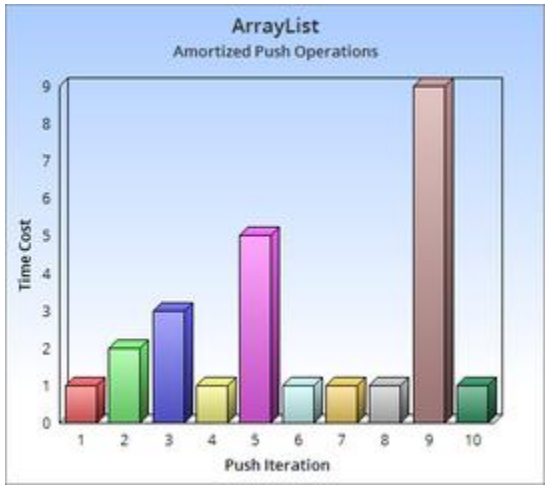
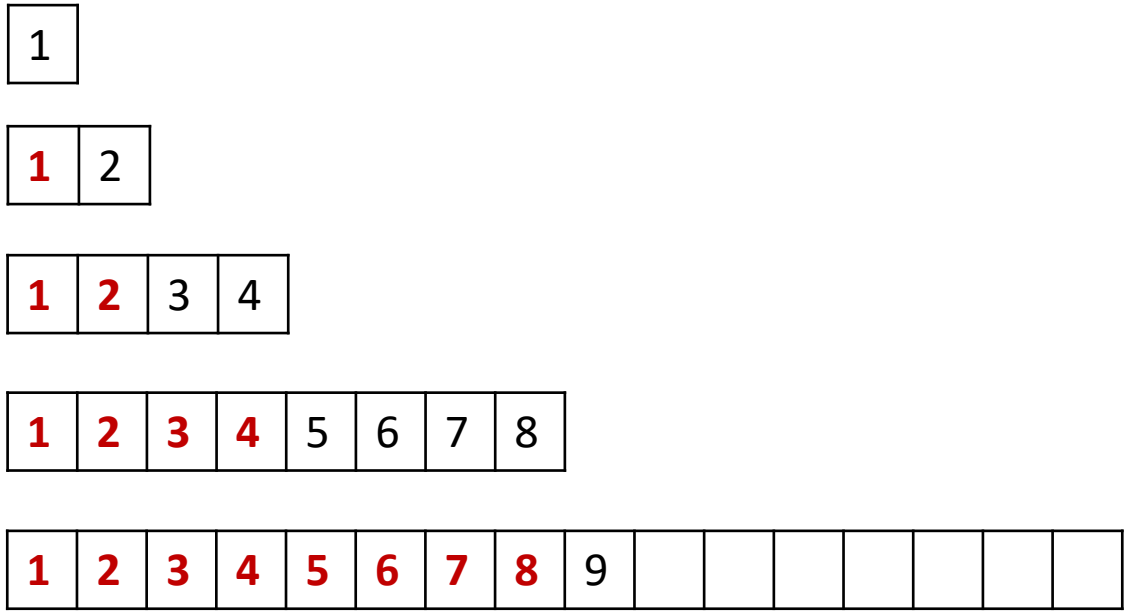


Image from Wikipedia
https://en.wikipedia.org/wiki/Amortized_analysis

cost	Add
1	1
1+1=2	2
2+1=3	3
1	4
4+1=5	5
1	6
1	7
1	8
8+1=9	9

capacity = capacity*2



Calculation for this example :

$$All\ copy\ cost = \mathbf{1+2+4+8} = 15 = 2*8-1$$

$$\textbf{Amortized cost} = \frac{Total\ cost}{9} = \frac{All\ copy\ cost + 9*write1}{9} = \frac{1+2+4+8+9*1}{9}$$

Resizing an array (3/3)

- Start with array of capacity 1
- Insert in it numbers: 1,2,3,4,5,6,7,8,9
- If there is space, the cost to write 1 item is 1
- If the array is full :
  - allocate double the space,
  - **copy the old items** to the new space
  - write new item (cost 1)

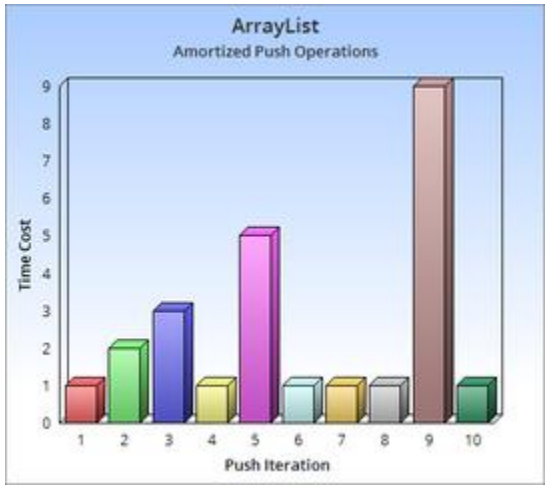
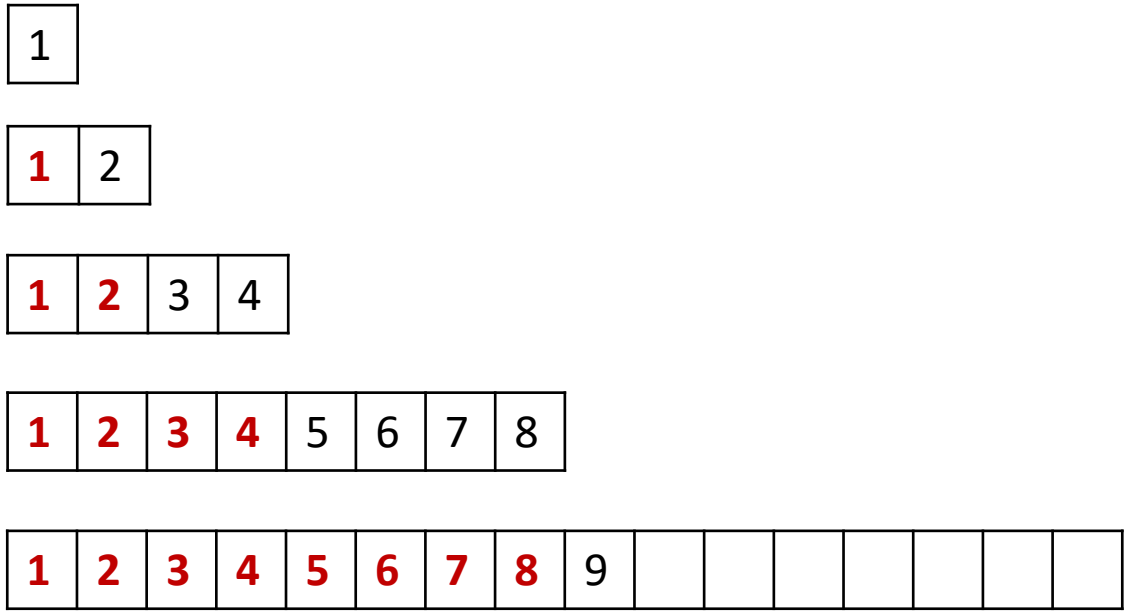


Image from Wikipedia
https://en.wikipedia.org/wiki/Amortized_analysis

cost	Add
1	1
1+1=2	2
2+1=3	3
1	4
4+1=5	5
1	6
1	7
1	8
8+1=9	9

capacity = capacity*2



Calculation for this example :

$$All\ copy\ cost = \mathbf{1+2+4+8} = 15 = 2*8-1$$

$$\boldsymbol{Amortized\ cost} = \frac{Total\ cost}{9} = \frac{All\ copy\ cost + 9*write1}{9} = \frac{1+2+4+8+9*1}{9}$$

General formula:

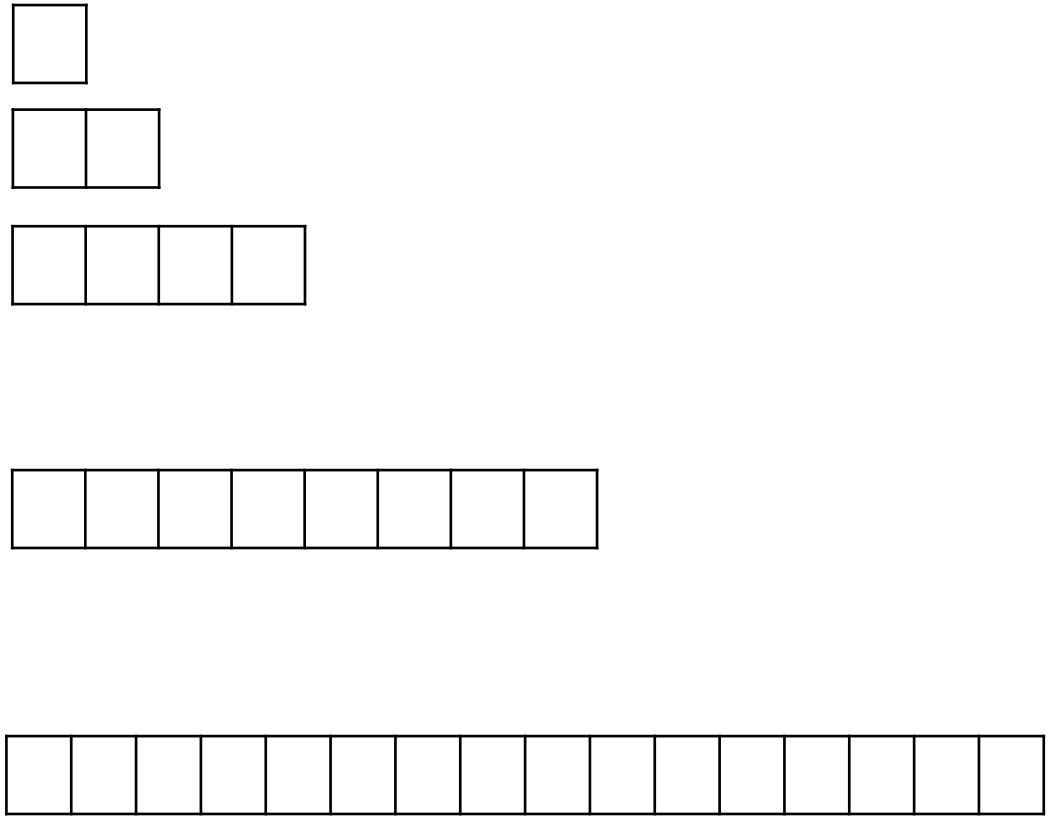
$$All\ copy\ cost = \mathbf{1+2+4+8+\cdots}\frac{\boldsymbol{N}}{\mathbf{4}}+\frac{\boldsymbol{N}}{\mathbf{2}}+\boldsymbol{N} = 2N-1 = \Theta(N)$$

$$\boldsymbol{Amortized\ cost} = \frac{Total\ cost}{N} = \frac{All\ copy\ cost+N*write1}{N} = \frac{\Theta(N)+N*\Theta(1)}{N} = \frac{\Theta(N)+\Theta(N)}{N} = \frac{\Theta(N)}{N} = \boldsymbol{\Theta(1)}$$

Meets the ArrayList requirement for add() !

Resize array: *2 vs + constant

capacity = capacity*2



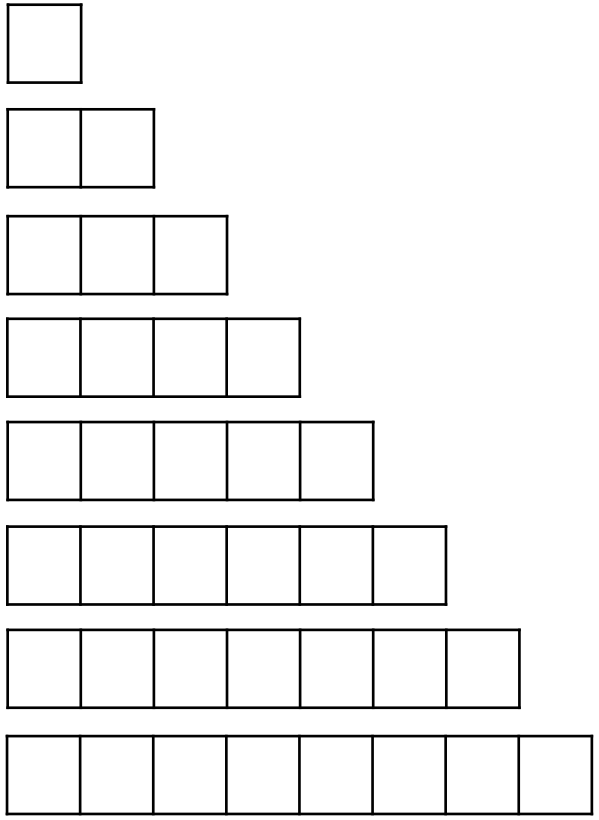
1+2+4+8+16+…+N/4+N/2+N = 2N-1

All copy cost : Θ(N) =>

Amortized cost =

$$= \frac{\boldsymbol{\Theta(N)} + \Theta(N)}{N} = \frac{\Theta(N)}{N} = \boldsymbol{\Theta(1)}$$

capacity = capacity + 1



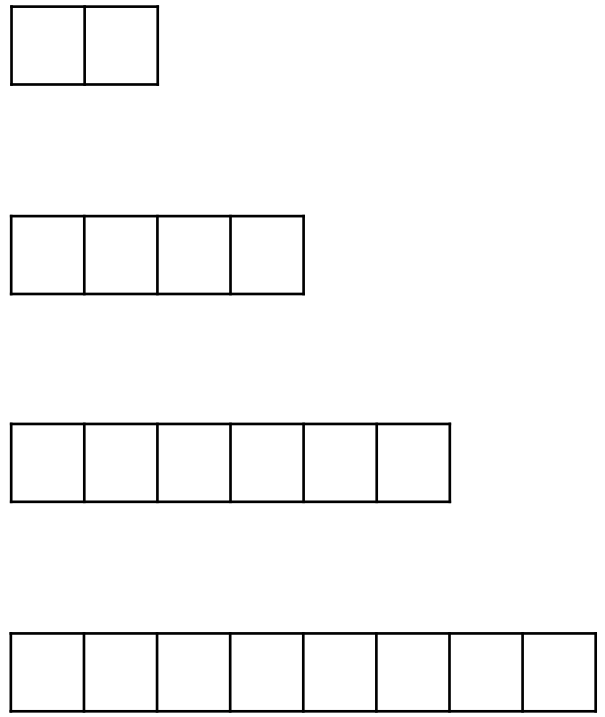
1+2+3+…+(N-2)+(N-1)+N = N(N+1)/2

All copy cost : $\Theta(N^2)$

Amortized cost =

$$\frac{\boldsymbol{\Theta(N^2)} + \Theta(N)}{\mathrm{N}} = \frac{\Theta(N^2)}{\mathrm{N}} = \boldsymbol{\Theta(N)}$$

capacity = capacity + 2



2+4+6+…+(N-2)+N=(N/2)(N/2+1)

All copy cost : $\Theta(N^2)$

Amortized cost =

$$\frac{\boldsymbol{\Theta(N^2)} + \Theta(N)}{\mathrm{N}} = \frac{\Theta(N^2)}{\mathrm{N}} = \boldsymbol{\Theta(N)}$$

Math derivation

- Proof for +2:

$$2 + 4 + 6 + 8 + 10 + 12 + \cdots . + (N-4) + (N-2) + N =$$
$$= 2 * 1 + 2 * 2 + 2 * 3 + 2 * 4 + \cdots + 2 * \frac{\mathrm{N}}{2} =$$
$$= 2\left(1 + 2 + 3 + 4 + \cdots + \frac{N}{2}\right) =$$
$$= 2 * \frac{\frac{N}{2}\left(\frac{N}{2} + 1\right)}{2} = \frac{N^2}{4} + \frac{N}{2} = \Theta(N^2)$$

- Proof for +100:

$$100 + 200 + 300 + 400 + \cdots + (N-100) + N =$$
$$100 * 1 + 100 * 2 + 100 * 3 + 100 * 4 + \cdots + 100 * \frac{\mathrm{N}}{100} =$$
$$100\left(1 + 2 + 3 + 4 + \cdots + \frac{N}{100}\right) =$$
$$100 * \frac{\frac{N}{100}\left(\frac{N}{100} + 1\right)}{2} = 50 * \frac{N^2}{10000} + 50 * \frac{N}{100} = \Theta(N^2)$$

Code view

```
for(capacity = 1; capacity<N; capacity=capacity * 2){
        ….
        copy_array(capacity,….);  // Θ(capacity)
        ….
}
```

```
for(capacity = 0; capacity<N; capacity=capacity+100){
        ….
        copy_array(capacity,….);  // Θ(capacity)
        ….
}
```

Main points

- Amortized time complexity
  - Definition
  - Problems:
    - Given operations and their TC => compute amortized TC of one operation
    - Given total TC for N operations (e.g. NlgN) => compute amortized TC of one operation

- Resizing for *add* operation:
  - Double the space: capacity = capacity * 2 => O(1) amortized time
  - Constant increment: capacity = capacity + 2 => O(N) amortized time
  - => mindful of cost of resizing

- ArrayList in Java
  - *add()* has O(1) amortized time
  - Resizing adds time complexity => avoid resizing if possible
    - => call constructor with expected size as capacity => `new ArrayList<>(N);`
    - => or use `ensureCapacity()` before adding a large number of elements