Your Ultimate Guide To Landing Top AI roles

2.3.8

Amortized Analysis of Dynamic Array

DECODE AiML

→ In Amortized analysis, we average the time required to perform a sequence of operations over all operations performed.

→ Using an Amortized analysis, we can show that the average cost of an operation is small, even though a single operation within the sequence might be expensive.

→ Example:

Suppose you commute by bus and have 2 options

① Single ticket – Rs 50 per ride

② Monthly travel pass – Rs 1000, unlimited ride

↳ If you ride bus once a month → Rs 1000

If you ride bus 50 times a month → [Rs 1000 (1st), 0 (2nd), 0, 0, 0 ---- 0 (50th)]

Cost per ride = Rs 1000/50 = Rs 20 → Cheaper per ride.

→ Amortized analysis is implicitly used in cost analysis of many Algorithms.

DECODE AiML

Dynamic Array

→ A dynamic array is an array that:
① Automatically resizes itself when it becomes full.
② Allows O(1) random access like a normal array.
③ Grows in Amortized O(1) time per insertion.

* How it works?
→ we start with empty dynamic array of size 0.

→ Let's define
① when the array is full, we double its size.
② Insert n element one by one

→ Each insert works like this
① If space is available → insert in O(1) time
② If full - create a new array of double size and

↳ Copy all old elements
↳ Insert the new element.

* Amortized Time Complexity Analysis

Resize no.	New Capacity	Copy Cost
1	1	0
2	2	1
3	4	2
4	8	4
5	16	8
....	....	....
K	$2^{(K-1)}$	$2^{(K-2)}$

Total Cost of n insertion is
① Each insertion takes 1 unit
② Plus Cost of Copying during resizes

→ Let's say $2^{(K-1)} = n$

$K - 1 = \log n$

$K = 1 + \log n$

Total Cost of n Insertion $= \underbrace{n}_{\text{Simple Insert}} + \underbrace{1+2+4+\cdots+n/2}_{\text{Resize}}$

↳ Geometric Series $\Rightarrow 2^0 + 2^1 + 2^2 + \cdots\cdots\ n/2$

$$S_n = \frac{a(r^{n-1}-1)}{r-1} = \frac{1\cdot(2^{\log n}-1)}{2-1} = (n-1)$$

$$T(n) = n + (n-1) = 2n-1$$

$$T(n) = O(n)$$

↳ Amortized Cost per Insertion $= \frac{T(n)}{n} = \frac{2n-1}{n} \approx 2 = O(1)$

→ Dynamic Array functionality in Python can be achieved using **List** data structures.

List in Python

→ Python List is a dynamic array

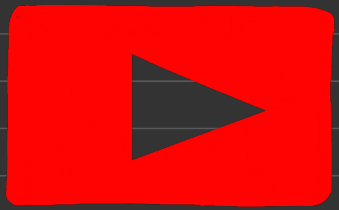
→ Not a linked list — uses contiguous memory.

→ Elements are stored as references to Python objects

→ Performance Summary.

Operation	Time Complexity	Notes
append()	Amortized - O(1)	Occasional O(n) when resized
insert(i, x)	O(n)	Element must be shifted
pop()	O(1)	Last elements
pop(i)	O(n)	shift elements
list[i]	O(1)	direct index access
len(list)	O(1)	Stored in metadata

DECODE AiML

Like  Subscribe