Lecture 3

Amortized Analysis (Take 2)

Dynamic Arrays (Lists)

initialize(): Creates empty list
append(x): Add x to end of list
get(i): Return i^{th} element

Naive: Allocate array of size $n+1$, move everything over
$O(n)$ time per append!!

Better: Doubling. Start with array of capacity 1
When run out of space → double + move old elements over.

Efficiency

Best case: $O(1)$ time (available space)

Worst case: $O(n)$ time (array was full)

Key idea: Amortized analysis

Analyse a sequence of operations on the data structure.

Amortized cost (aggregate method)

ac = total cost of a sequence of n operations / # operations.

Comparison model? ✗

Model (costs)

- Writing to a value in an array costs 1
- Moving an item from one to another costs 1
- Everything else is free

Theorem Cost of n appends is at most $3n$

Most recent costs $\leq n$

2^{nd}-last resize costs $\leq n/2$ ...

Cost of inserts $= n$

Cost of resizes $\leq$
$n + n/2 + n/4 + \ldots + 1 \leq 2n$

total $\leq n + 2n \leq 3n$

Theorem Amortized cost of append is ≤ 3

Proof: $3m / m = 3$ □

Bankers Method

Review 122

The Potential Method

kind of a generalization of bankers method

Potential function

Φ : data structure states $\rightarrow \mathbb{R}$

Amortized cost (Potential method)

operation takes data structure from S_{i-1} to S_i

$$aC = C + \Phi(S_i) - \Phi(S_{i-1})$$

amortized cost = actual cost + (difference in potential)

$$\sum ac_i = \sum \left(c_i + \left(\Phi(S_i) - \Phi(S_{i-1}) \right) \right)$$

$$= \underbrace{\sum c_i}_{\text{total cost}} + \underline{\Phi(S_m) - \Phi(S_0)}$$

If $\Phi(S_m) \geq \Phi(S_0)$

$$\sum c_i = \sum ac_i - \Phi(S_m) + \Phi(S_0)$$

$$\sum c_i \leq \sum ac_i$$

Lists via Potentials

Educated guessing + trial and error

Key idea

* Cheap operations should increase potential
* Expensive operations should decrease it

List State = { n : # list elements, c : capacity of array }

Cheap operation : Insert item Φ ↗

Expensive operation : Resize Φ ↘

$\Phi(n, c) = n - c$ (first guess)

$n \leq c$, so $\Phi(n, c) \leq 0$

$\Phi(n, c) = n - \frac{c}{2}$ (second guess)

$a_c = c + \Phi(S_i) - \Phi(S_{i-1})$

	a_c	c	Old Φ	New Φ	$\Delta\Phi$
Insert	2	1	$n - \frac{c}{2}$	$n+1 - \frac{c}{2}$	1
Resize	$\frac{n}{2}$	n	$\frac{n}{2}$	0	$-n/2$

$\Phi(n, c) = 2\left(n - \frac{c}{2}\right)$ (third guess)

	ac	c	old Φ	New Φ	$\Delta\Phi$
Insert	3	1	$2\left(n - \frac{c}{2}\right)$	$2\left(n+1 - \frac{c}{2}\right)$	2
Resize	0	n	$2 \cdot \frac{n}{2} = n$	0	$-n$

Amortized cost of append = 3 □

An even-more dynamic array

Add a "pop" operation. Removes last element.

Shrink array when $n \leq c/4$ and a pop happens. Halve the size of the capacity and move everything.

Property: After a grow or shrink, $n = c/2$

Cost model

- Costs 1 to erase an element from array.

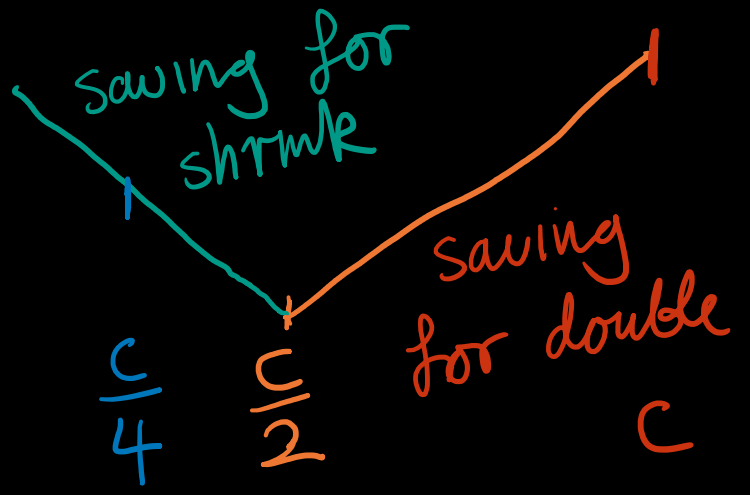
Analyze growing-shrinking list

$$\Phi(n,c) = \begin{cases} 2(n - \frac{c}{2}) & n \geq \frac{c}{2} \\ 2(\frac{c}{2} - n) & n < \frac{c}{2} \end{cases}$$

appending above 1/2 cap. potential ↗

popping below 1/2 cap. potential ↗

	ac	c	old Φ	new Φ	$\Delta\Phi$
Insert	3	1	"	"	$\leq +2$
Grow	0	n	n	0	$-n$
pop	2	1	$\frac{c}{2} - n$	$\frac{c}{2} - (n-1)$	1
Shrink	0	$c/4$	$\frac{c}{4}$	0	$-c/4$



Theorem cost of pop (amortized) = 2

$\Phi(S_0) = \frac{c}{2} - n = 1$ $\quad \Phi(S_m) \geq 0$

$n = 0$
$c = 2$

Charge 1 to initialization