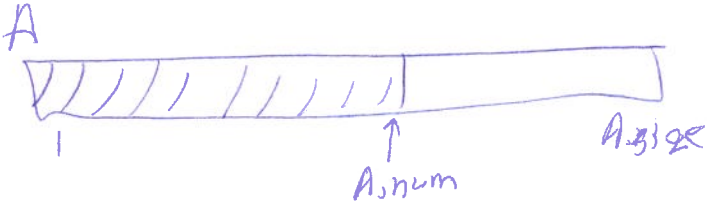
CSCE 750
10/24/2023

Resizable array (cont.)
Mergeable Heaps

Array A A.size = # entries

Supports a collection A.num = # items in the collection. Basic ops:

Lookup (by index)
InsertAtEnd



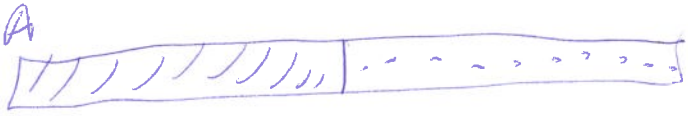
Clearly A.num ≤ A.size

Resizing: when A.num = A.size, the allocate array of size 2*A.size, copy items into new array before actual insertion

Insertion { without resize: time 1
with resize: time A.size + 2

Use potential method to show amortized cost is O(1).

Want Φ to increase a constant amount with each non-resizing insertion, then drop enough when a resize happens to make the amortized cost constant.



$A.num = A.size$: $\Phi \approx A.size$

$A.num = \frac{A.size}{2}$ $\Phi \approx 0$

Just linearly interpolate:

Define $\Phi(A) = 2 * A.num - A.size + 1$

Initially: $A.num = 0$ $A.size = 1$ ⟶ to handle initial case

[$\Phi = 0$ initially, $\Phi \geq 0$ always] ✓

(Claim: Amortized cost of any operation is $O(1)$.

Prove: Lookup: Actual cost is 1

Amortized cost is $1 + \Phi(A_{after}) - \Phi(A_{before})$

$= 1 = O(1)$

InsertAtEnd.

If No resize: Actual cost is 1

Amortized cost is $1 + \Phi(A_{after}) - \Phi(A_{before})$

$A_{after.size} = A_{before.size}$

$A_{after.num} = A_{before.num} + 1$

$= 1 + 2 * A_{after.num} - A_{after.size} + 1$

$- (2 * A_{before.num} - A_{before.size} + 1)$

$= 1 + 2(A_{after.num} - A_{before.num})$

$= 1 + 2 = 3 = O(1)$

If resize:

$A_{after.num} = A_{before.num} + 1$

$A_{after.size} = 2 * A_{before.size}$

Amortized cost = Actual cost + $\Delta\Phi$

$= A_{before.size} + 2 + 2 * A_{after.num} - A_{after.size} + 1$

$- (2 * A_{before.num} - A_{before.size})$

$= A_{before.size} + 2 + 2 + A_{before.size} - 2 * A_{before.size}$

(2 num's)

$= 4 = O(1)$ ☑

Problem: Binary counter: Linked list of binary digits representing an unsigned integer in binary. Want to support:

initially 0

- ~~Read~~ Write out — O(# digits)
- Increment — amortized time O(1)

→ support Decrement in O(1) amortized time O(1)

Mergeable Heaps

Merge two heaps into a single heap

For binary heap — tables, worst-case time $O(n \lg n)$

where n is the total number of items in both heaps. (4)

Use linked structures

- Fibonacci Heaps (not covered)
- Binomial Heaps
