Analysis and Design of Algorithms

National University of Singapore

Algorithms

CS3230

Tutorial

Week 8

Question 1

Which of the following statements is **false**?

- The amortized cost for insert in dynamic tables is $\Theta(1)$.

- In the accounting method, the amortized cost $\hat{c}_i$ is always greater than the actual cost c_i of an operation.

- $\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i \geq 0$ where $\hat{c}_i$ and c_i are the amortized and actual costs of the i-th operation respectively.

Solution

Answer: B is false

- **A**: For insert in dynamic tables allocate \$3.
  - \$1 used immediately for insert
  - \$2 used later to transfer to new array when array doubled
- **C**: $$\sum_{i=1}^{n} \hat{c}_i - \sum_{i=1}^{n} c_i \geq 0.$$
  - Sum of amortized cost must be larger than sum of total cost (bank balance ≥ 0)
- **B is false**: $\hat{c}_i > c_i$ does not always hold, e.g. insert when table doubled

Question 2

- Consider a data structure that is based on a queue with four operations:
  - ENQUEUE(a): Add the element a into the queue
  - DEQUEUE(): Dequeue a single element from the queue
  - DELETE(k): Dequeue k elements from the queue
  - ADD(A): Enqueue all elements in A
- **Claim**: ENQUEUE, DEQUEUE and DELETE run in amortized O(1) time while ADD runs in amortized O(|A|) time.
- Using accounting method, can you show that these time complexities are correct?
- (Please state the charge for each operation.)

Solution

- ENQUEUE(a) is charged \$2,
  - \$1 is for immediate insert
  - \$1 is store in the bank for the future dequeue operation of a

- DEQUEUE() is charged \$0
  - The element is deleted using \$1 from the bank

- DELETE(k) is charged \$0
  - The k elements are deleted using \$k from the bank

- ADD(A) is charged \$(2|A|)
  - There are |A| enqueue. Each enqueue is charged \$2
    - \$1 is for immediate insert
    - \$1 is stored in the bank for the future dequeue operation

Solution

- After the insertion of element *x* (in ENQUEUE and ADD operations), $1 is assocateed to *x* in the bank.

- When we dequeue the element *x* (in DEQUEUE and DELETE operations), we can use $1 from the bank for dequeue of *x*.

- Hence, the bank never goes negative.

Question 3

- Consider a data structure that is based on a queue with four operations:
  - ENQUEUE(a): Add the element a into the queue
  - DEQUEUE(): Dequeue a single element from the queue
  - DELETE(k): Dequeue k elements from the queue
  - ADD(A): Enqueue all elements in A
- **Claim**: ENQUEUE, DEQUEUE and DELETE run in amortized O(1) time while ADD runs in amortized O(|A|) time.
- Using Potential method, can you show that these time complexities are correct?
- (Please state your potential function.)

Solution

- Let D_i be the data structure after i^{th} operation.
- Let $\Phi(D_i)$=the number of elements in the queue.

- For ENQUEUE(a),
  - Actual cost $c_i=1$
  - The queue has 1 more element after ENQUEUE, hence, $\Phi(D_i) - \Phi(D_{i-1}) = 1$.
  - Amortized cost $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 2$.

Solution

- Let D_i be the data structure after i^{th} operation.
- Let $\Phi(D_i)$=the number of elements in the queue.

- For DEQUEUE(),
  - Actual cost $c_i=1$
  - The queue has *1* less element after ENQUEUE, hence, $\Phi(D_i) - \Phi(D_{i-1}) = -1$.
  - Amortized cost $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 0$.

Solution

- Let D_i be the data structure after i^{th} operation.
- Let $\Phi(D_i)$=the number of elements in the queue.

- For DELETE(k),
  - Actual cost $c_i = k$
  - The queue has k less element after ENQUEUE, hence, $\Phi(D_i) - \Phi(D_{i-1}) = -k$.
  - Amortized cost $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 0$.

Solution

- Let D_i be the data structure after i^{th} operation.
- Let $\Phi(D_i)$=the number of elements in the queue.

- For ADD(A),
  - Actual cost $c_i = |A|$
  - The queue has $|A|$ more element after ENQUEUE, hence, $\Phi(D_i) - \Phi(D_{i-1}) = |A|$.
  - Amortized cost $\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) = 2|A|$.

- In conclusion, ENQUEUE, DEQUEUE and DELETE run in amortized O(1) time while ADD runs in amortized O(|A|) time.

Question 4

Delete x from T;

$n \leftarrow n-1$;

If ($n = 0$)

  free(T);

Else

  If(n = size(T)/2)

  { $T' \leftarrow$ **createTable**($n/2$);

  copy(T,T');

  free(T);

  $T \leftarrow T'$

  }

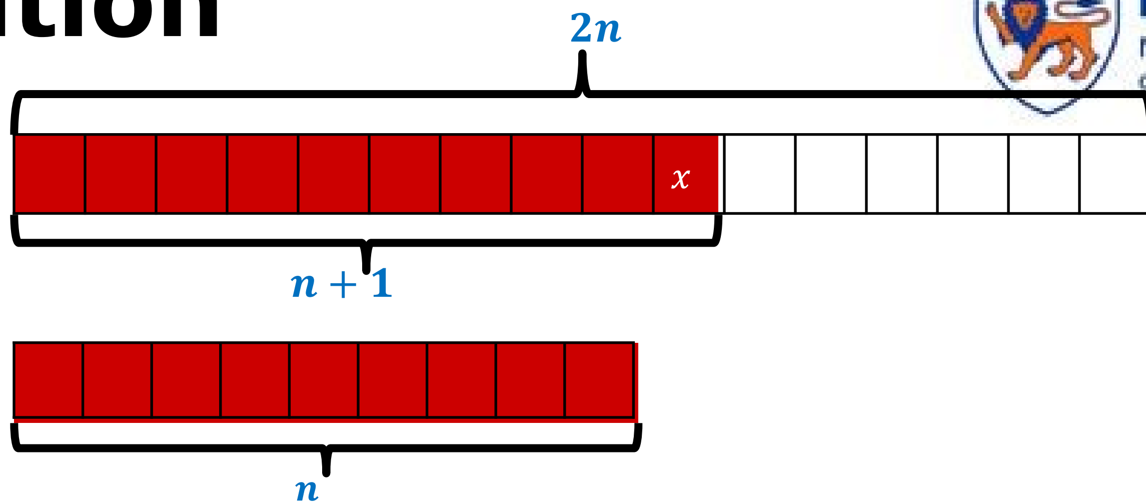
Note, T is the dynamic table that supports only deletions.

Using Potential method show that the amortized cost of each Deletion operation is O(1).

(State your potential function.)

Solution

$\phi = c\,(n-1)$



Before **Delete**(x)

$\phi = 0$

After **Delete**(x)

Operation Delete(x)	Actual Cost	$\Delta(\phi)$	Amortized Cost
Case 1: when table does not shrink	c	c	$2c$
Case 2: when table shrinks to half	$cn + c$	$c(1-n)$	$2c$

ϕ at any stage = c (size(T)$-n$)

How can you express **number of empty slots** in terms of **size** and n ?

A quantity that is decreasing during expensive step