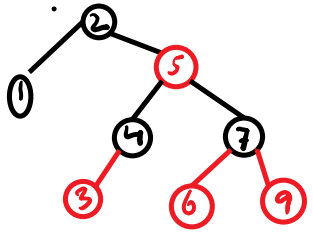


Habib University
shaping futures

CS 201 Data Structure II (L2 / L5)

Muhammad Qasim Pasta

qasim.pasta@sse.habib.edu.pk

ArrayList: List ADT – using Array

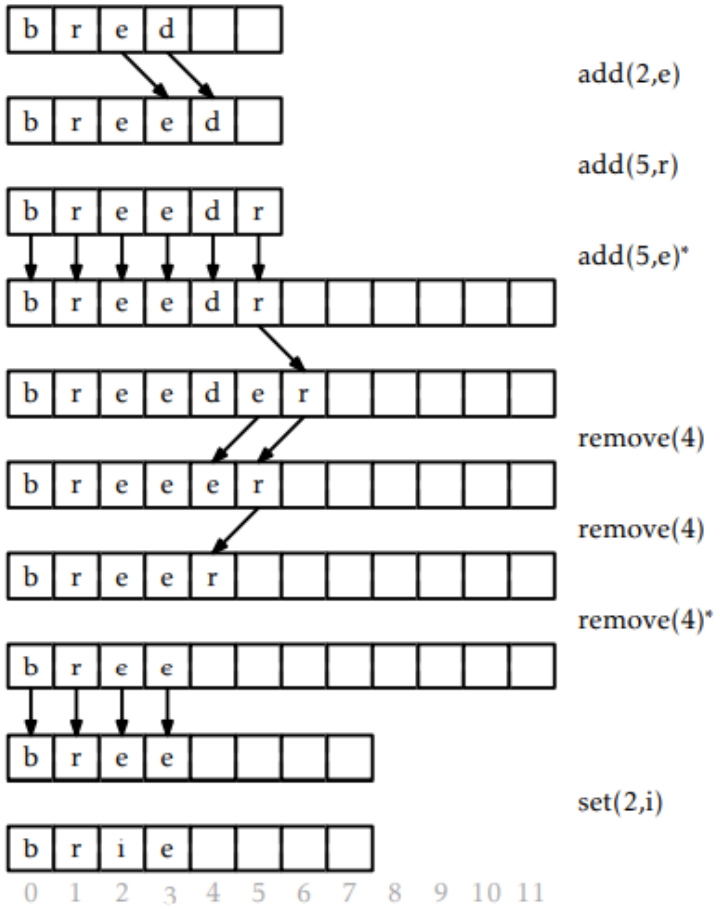
- Implementing List operations using array
- Array will be used to store the data i.e. elements
- Array has its limitations: homogenous and size (why?)
- Operations:
  - set(i,x): returns the value at i^{th} position and set value of x to i^{th} position
  - get(i): returns the value at i^{th} position
  - add(i,x): set the value of x at i^{th} position by pushing existing elements from i to i+ 1
  - remove(i): remove the value at i^{th} position by pushing existing elements from i to i-1
- What if we do not want to limit the size – using array?

Growing and Shrinking

- Because of fixed size array
  - Scenario when the number of elements can't be determined before hand
  - Either create a enough large array – issues?
  - Recreate array and copy elements – issues?

- Resize the array as per usage
- Grow – if the array is full
  - Increase the size by 2n
- Shrink – if array size is greater than 3 * n
  - Decrease the size by n/2

List: Add / Remove operations for with resize



Lemma 2.1. If an empty List is created and any sequence of $m \geq 1$ calls to *add(i,x)* and *remove(i)* are performed, then the total time spent during all calls to *resize()* is $O(m)$.

- Any time resize() is called, the number of calls to add and remove since the last call to resize() is at least $\frac{n}{2} - 1$
- If we prove this, then we need to prove that the number of add (i,x) or remove(i) calls between two resizes is at least $\frac{n_i}{2}$
- Two cases:
  - For add
  - For remove

$$\sum_{i}^{r} (\frac{n_i}{2} - 1) \leq m$$

$$\sum_{i}^{r} n_i \leq 2m + 2r$$

$$\sum_{i}^{r} O(n_{i)} \leq O(m + r)$$

i	n_i	$\frac{n_i}{2} - 1$
1	2	0
2	4	1
3	8	3
4	16	7
5	32	15
6	64	31

Case 1: resize is being called by add(i,x)

- Let's consider that array a is full where $length(a) = n = \ \ n_i$

1	2	3	4	5	6	7	8
X	X	X	X	Y	Y	Y	Y

- When did the last time resize() call?

Case 1: resize is being called by add(i,x)

- Let's consider that array a is full where $length(a) = n = \ \ n_i$

1	2	3	4	5	6	7	8
X	X	X	X	Y	Y	Y	Y

- When did the last time resize() call?
- When array has $\frac{n_i}{2}$ elements
- How may add(i,x) calls made since the last resize()?
- $\frac{n_i}{2}$ calls

1	2	3	4	5	6	7	8
X	X	X	X	Y	Y	Y	Y

Case 2: resize is being called by remove(i)

- Let's consider that size array a is more than thrice of the number of elements

  $\text{length(a)} \geq 3\text{n} = 3\text{n}_\text{i}$ (or $n_i = \frac{length(a)}{3}$)

1	2	3	4	5	6	7	8
X	X						

Case 2: resize is being called by remove(i)

- Let's consider that size array a is more than thrice of the number of elements

  $\text{length(a)} \geq 3\text{n} = 3\text{n}_\text{i}$ (or $n_i = \frac{length(a)}{3}$)

- The number of elements in the array when last time remove(i,x) called? = $\frac{length(a)}{2}$

1	2	3	4	5	6	7	8
X	X						

1	2	3	4	5	6	7	8
X	X	Y	Y				

Total elements removed = no of elements when last time remove() called – number of elements present in the array

$$\boldsymbol{R \geq \frac{length(a)}{2} - \frac{length(a)}{3}}$$

$$\boldsymbol{R \geq \frac{length(a)}{6}}$$

$$\boldsymbol{R \geq \frac{1}{2}(\frac{length(a)}{3})}$$

$$\boldsymbol{R \geq \frac{1}{2}(n_i)}$$

ADT in the textbook	Purpose	
ArrayStack	List ADT using array	
ArrayQueue	Queue using Array	