Queues

CS 240

Alex Vondrak

ajvondrak@csupomona.edu

Winter 2012

Queues

Definition (Queue)

A queue is a linear data structure of items arranged from rear to front. It's defined by three operations:

- enqueue: To insert an item, you place it at the rear (before any other items)
- dequeue: To remove an item, you remove the front element
- peek: You may look at the front item of the queue without removing it; to look at anything underneath, you must dequeue the front

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Let's enqueue the elements 1, 2, and 3.

What should this look like in the array?

(A)

							3	2	1

(B)

							1	2	3

(C)

1	2	3							

(D)

3	2	1							

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Let's dequeue an element (after enqueueing 1, 2, and 3), returning 1. What should the array look like after this?

(A)								3	2	
(B)									3	2
(C)		2	3							
(D)	2	3								

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

How many indices should we keep track of?

(A) One: `front`

(B) One: `rear`

(C) Two: `front` and `rear`

(D) None of the above

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
							3	2	1

`rear = 7, front = 9`

Let's take a closer look at dequeueing an element.
What value do we return?

(A) `this.data[this.rear]`

(B) `this.data[this.front]`

(C) `this.data[0]`

(D) `this.data[this.data.length - 1]`

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
							3	2	1

`rear = 7, front = 9`

Let's take a closer look at dequeueing an element.
How do we change **`this`**`.front`?

(A) Decrement it by 1

(B) Increment it by 1

(C) Don't; keep it the same

(D) None of the above

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
							3	2	

`rear = 7, front = 8`

Let's take a closer look at enqueueing an element, 4.
How should we change **`this`**`.rear`?

(A) Decrement it by 1

(B) Increment it by 1

(C) Don't; keep it the same

(D) None of the above

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
							3	2	

`rear = 7, front = 8`

Let's take a closer look at enqueueing an element, 4.
How should we change **`this`**`.rear`?

(A) Decrement it by 1

(B) Increment it by 1

(C) Don't; keep it the same

(D) None of the above

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
						4	3	2	

`rear = 6, front = 8`

Let's take a closer look at enqueueing an element, 4.
How did we change the array to the one pictured above?

(A) `this.data[this.rear] = item`

(B) `this.data[this.front] = item`

(C) `this.data[this.rear - 1] = item`

(D) None of the above

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
10	9	8	7	6	5	4	3	2	

`rear = 0, front = 8`

Let's take a closer look at enqueueing even more elements.
If we enqueue 11 at this point, how can we fit it in the array?

(A) Grow the array

(B) Loop **`this`**`.rear` around to 9

(C) Let **`this`**`.rear` be -1

(D) We can't

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
10	9	8	7	6	5	4	3	2	11

`rear = 9, front = 8`

Let's take a closer look at enqueueing even more elements.
If we enqueue 12 at this point, how can we fit it in the array?

(A) Grow the array

(B) Overwrite the 2

(C) Let **`this`**`.rear` be 8

(D) We can't

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
10	9	8	7	6	5	4	3	2	11

`rear = 9, front = 8`

Let's take a closer look at enqueueing even more elements.
If we grow the array, can we just copy the above elements as-is into a bigger array?

(A) Yes

(B) No

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
10	9	8	7	6	5	4	3	2	11

`rear = 9, front = 8`

Let's consider the remaining few queue methods before we implement them in full.

Will the `size()` method involve a simple calculation as in `ArrayStack`?

(A) Yes

(B) No

Multiple Choice Question

Imagine storing queue elements in an array, similar to `ArrayStack`.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
10	9	8	7	6	5	4	3	2	11

`rear = 9, front = 8`

Let's consider the remaining few queue methods before we implement them in full.
Will the `isEmpty()` method be any more complicated than the one in `ArrayStack`?

(A) Yes

(B) No