CH-231-A

Algorithms and Data Structures

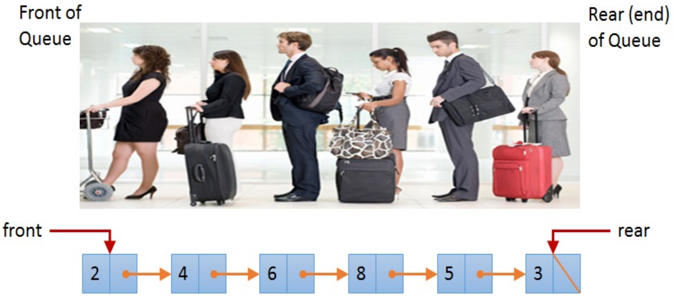
ADS

Lecture 20

Dr. Kinga Lipskoch

Spring 2022

Queue (1)



Front pointer

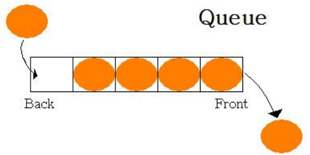
*Pointing to **first** element of Queue*

Rear pointer

*Pointing to **Last** element of Queue*

Queue (2)

- Elementary dynamic data structure.
- Implements idea of dynamic set.
- Delete operation is called `dequeue`.
- Insert operation is called `enqueue`.
- FIFO principle (First In First Out):
  The element that is removed from the queue is the oldest one in the queue.



Queue Operations

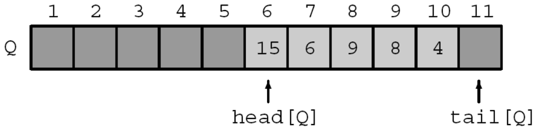
Modify operations:

- *Enqueue*(Q, x):
  Add element x at the tail of queue Q.
- *Dequeue*(Q):
  If queue is non-empty, remove head element and return it.

Queue Example (Array Implementation) (1)

- *head*[Q] and *tail*[Q] mark the index of the first entry and the one following the last entry of the queue.
- Example:

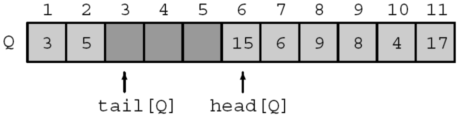
  Queue with 5 elements between indices 6 (head) and 11 (tail).



- We can also have under- and overflow.

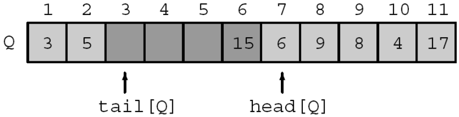
Queue Example (Array Implementation) (2)

- Apply operations *Enqueue*$(Q, 17)$, *Enqueue*$(Q, 3)$, and *Enqueue*$(Q, 5)$:



	1	2	3	4	5	6	7	8	9	10	11
Q	3	5				15	6	9	8	4	17
			tail[Q]			head[Q]					

- Apply operation *Dequeue*(Q) returning entry 15:

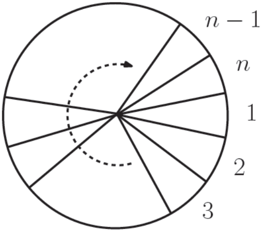


	1	2	3	4	5	6	7	8	9	10	11
Q	3	5				15	6	9	8	4	17
			tail[Q]				head[Q]				

Queue: Modulo Operations

Circular structure of filling the array with queue entries:

- $head[Q] = 1$ and $tail[Q] = 5$:
  4 entries
- $head[Q] = n - 1$ and $tail[Q] = 1$:
  2 entries
- $head[Q] = n$ and $tail[Q] = n - 1$:
  $n - 1$ entries (full queue)



Queue Operations (Array Implementation) (3)

```
Enqueue(Q,x)
if tail[Q] = head[Q]-1 then
  error ‘overflow’
Q[tail[Q]] ← x
if tail[Q] = length[Q]
  then tail[Q] ← 1
  else tail[Q] ← tail[Q]+1
```

```
Dequeue(Q)
if tail[Q] = head[Q] then
  error ‘underflow’
x ← Q[head[Q]]
if head[Q] = length[Q]
  then head[Q] ← 1
  else head[Q] ← head[Q]+1
return x
```

Queue Operations: Complexity

```
Enqueue(Q,x)
if tail[Q] = head[Q]-1 then
  error 'overflow'
Q[tail[Q]] ← x
if tail[Q] = length[Q]
  then tail[Q] ← 1
  else tail[Q] ← tail[Q]+1
```

```
Dequeue(Q)
if tail[Q] = head[Q] then
  error 'underflow'
x ← Q[head[Q]]
if head[Q] = length[Q]
  then head[Q] ← 1
  else head[Q] ← head[Q]+1
return x
```

Complexity:
when implemented as an array all operations are O(1).