Stack and Queue

Inst. Nguyễn Minh Huy

Contents

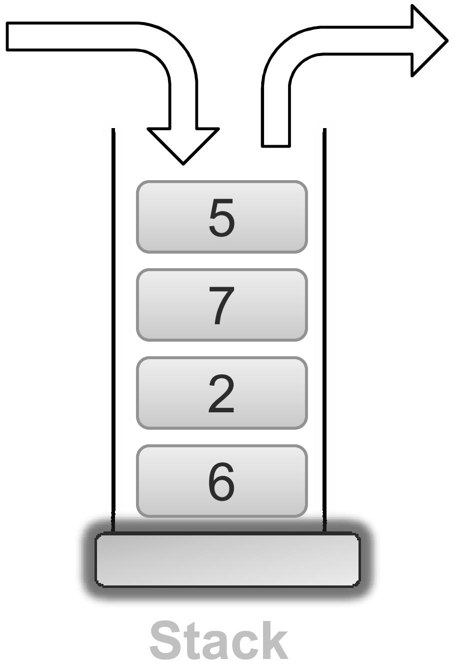
- Stack.
- Queue.

Contents

- **Stack.**
- Queue.

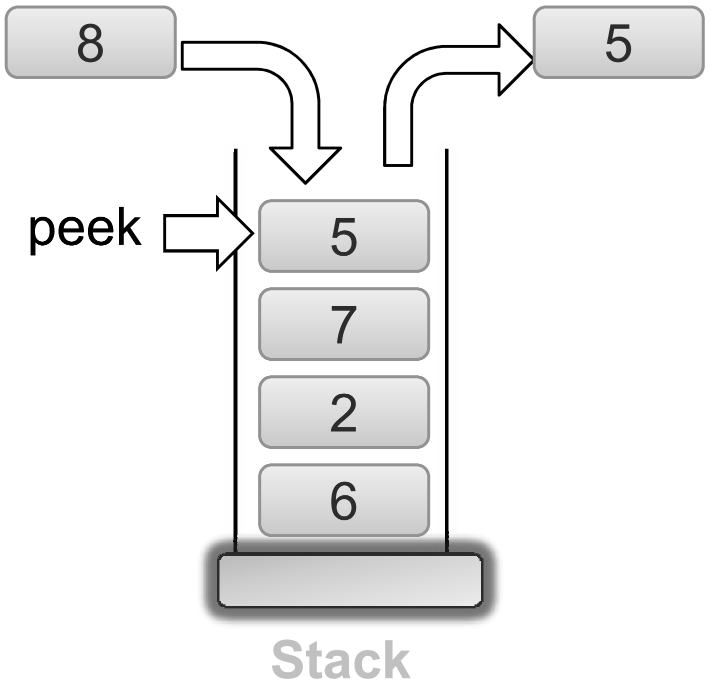
Stack

- Stack concept:
  - Collection of elements accessed by LIFO method.
  - LIFO (**L**ast **I**n **F**irst **O**ut):
    - Last insert, first removed.



Stack

- Operations on stack:
  - init: initialize stack.
  - isEmpty: check empty.
  - isFull: check full.
  - push: insert element.
  - pop: remove element.
  - peek: read element.

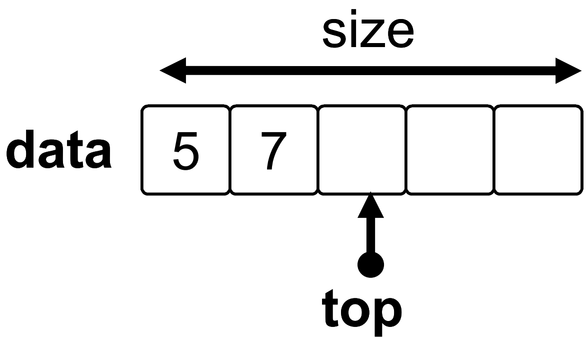


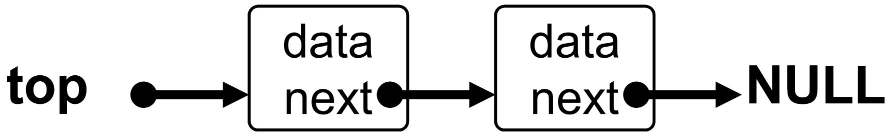
Stack

- Stack implementation:
  - Declaration:

```
// Use dynamic array.
struct Stack
{
        int  *data;
        int   size;
        int   top;
};
```

```
// Use linked list.
struct Stack
{
        Node  *top;
};
```

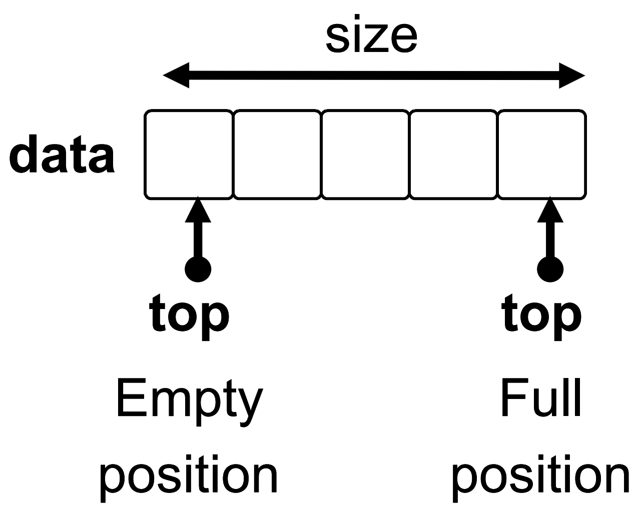




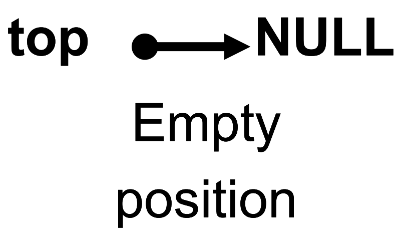
Stack

- Stack implementation:
  - init: initialize empty stack.
  - isEmpty: check top position.
  - isFull: check top position.

Use dynamic array



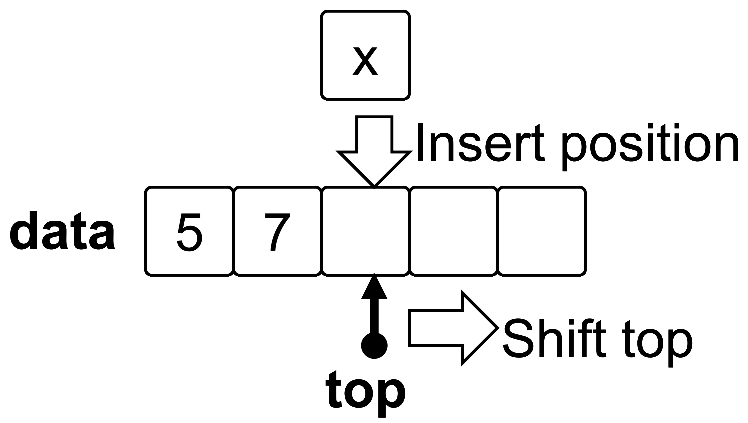
Use linked list



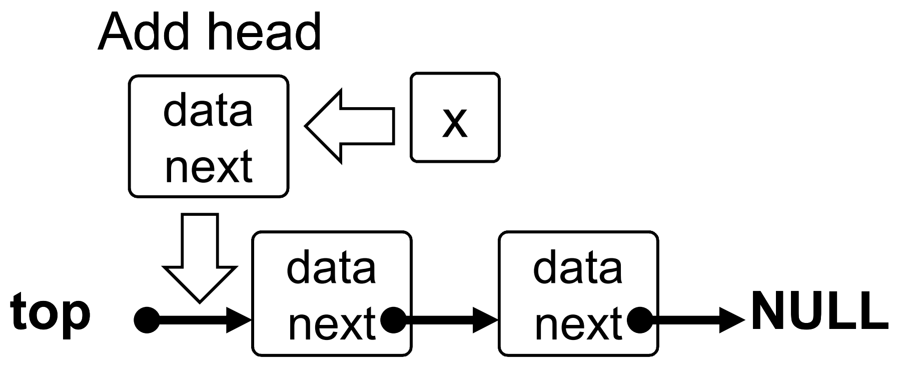
Stack

- Stack implementation:
  - Push: insert element into stack.

Use dynamic array



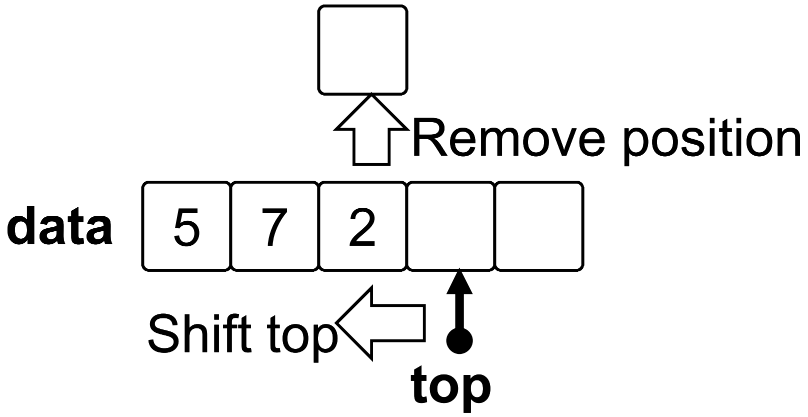
Use linked list



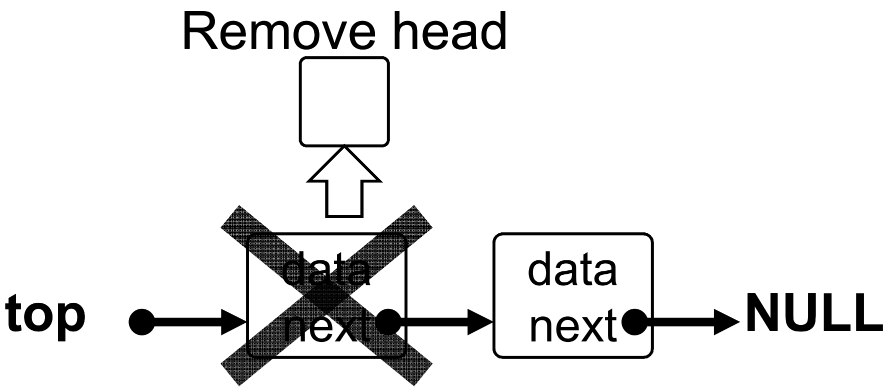
Stack

- Stack implementation:
  - Pop: remove element from stack.

Use dynamic array



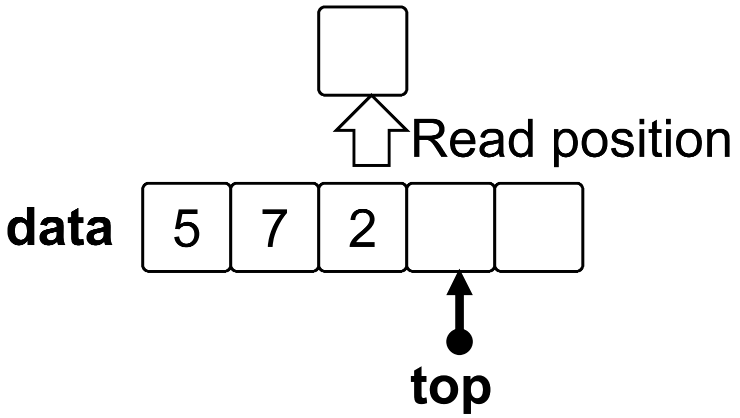
Use linked list



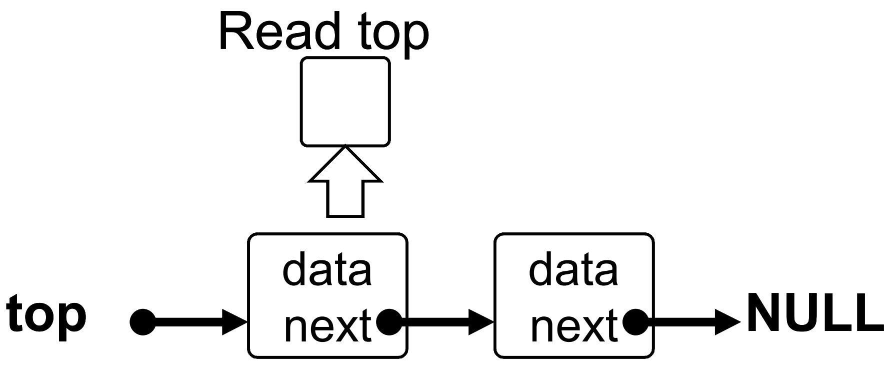
Stack

- Stack implementation:
  - Peek: read element from stack, do not remove.

Use dynamic array

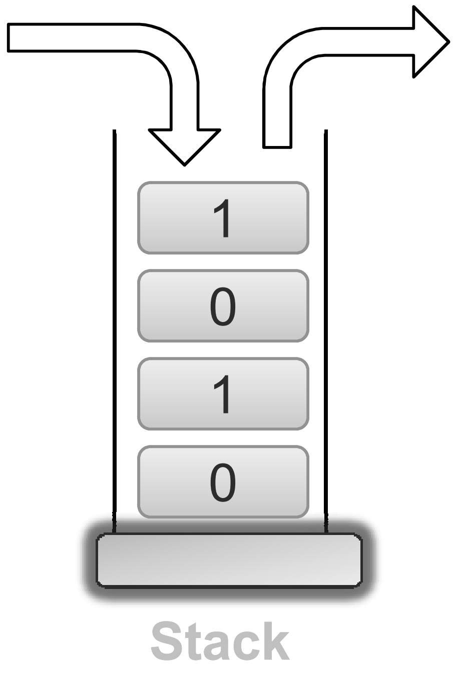


Use linked list



Stack

- Stack applications:
  - Perform reversed operations:
    - Convert decimal to binary.
  - Process expression:
    - Reversed Polish Notation.
  - Simulate recursion.

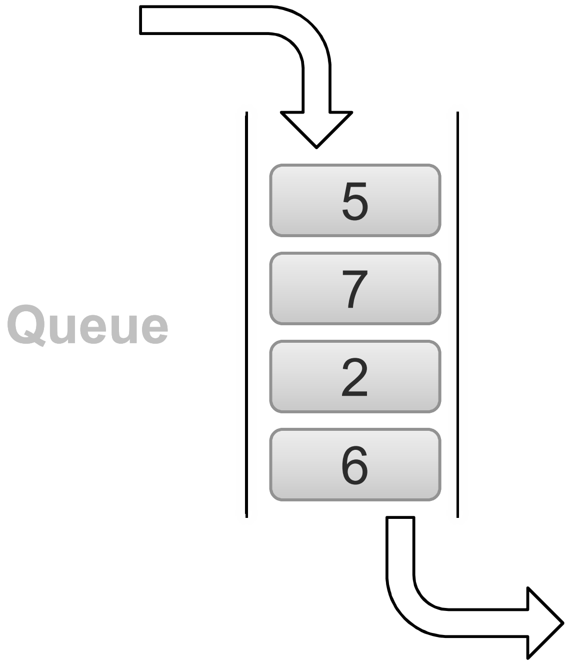


Contents

- Stack.
- **Queue.**

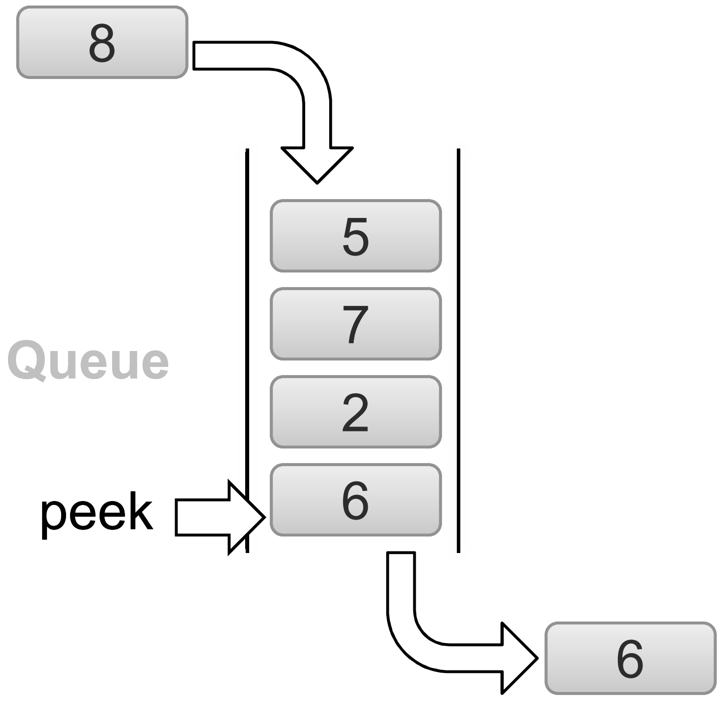
Queue

- Queue concept:
  - Collection of elements accessed by FIFO method.
  - FIFO (**F**irst **I**n **F**irst **O**ut):
    - First come first serve.
    - First insert first remove.



Queue

- Operations on queue:
  - init: initialize queue.
  - isEmpty: check empty.
  - isFull: check full.
  - push: insert element.
  - pop: pop element.
  - peek: read element.

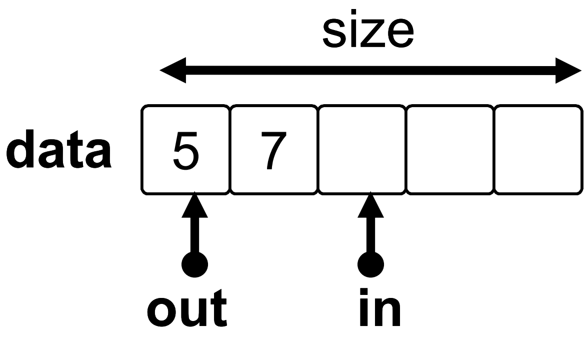


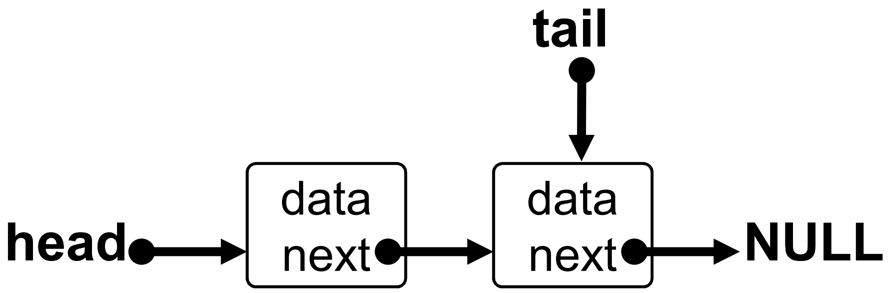
Queue

- Queue implementation:
  - Declaration:

```
// Use dynamic array
struct Queue
{
        int  *data;
        int   size;
        int   in;
        int   out;
};
```

```
// Use linked list
struct Queue
{
        Node  *head;
        Node  *tail;
};
```

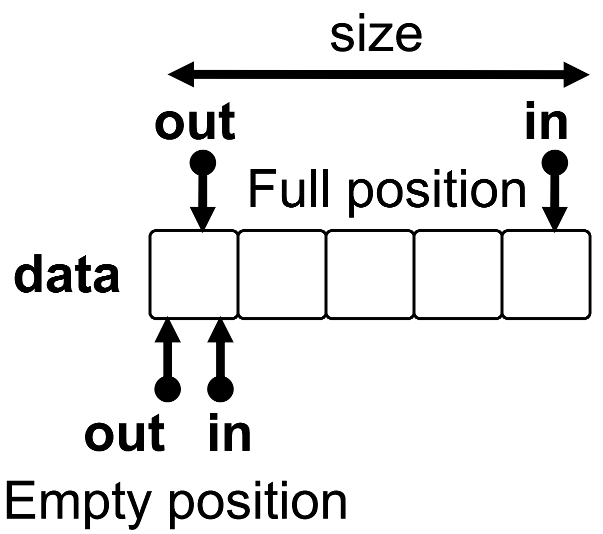




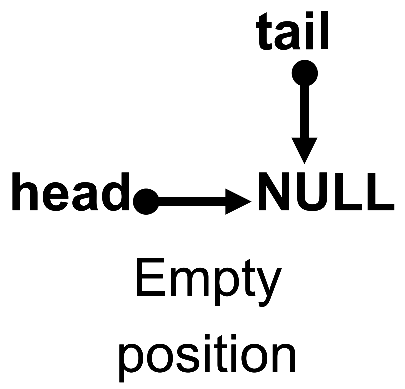
Queue

- Queue implementation:
  - init: initialize empty queue.
  - isEmpty: check in and out position.
  - isFull: check in and out position.

Use dynamic array



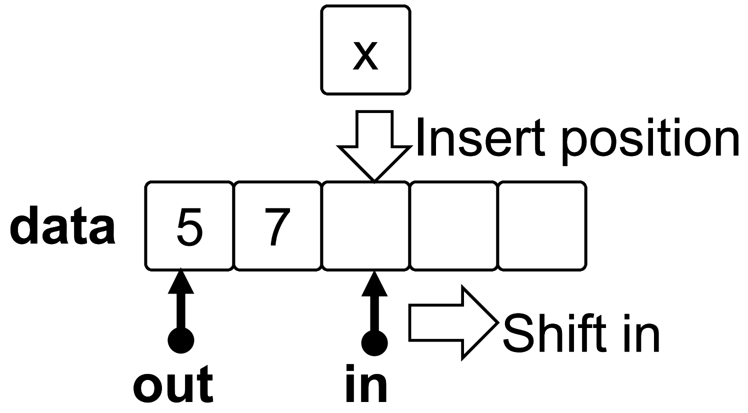
Use linked list



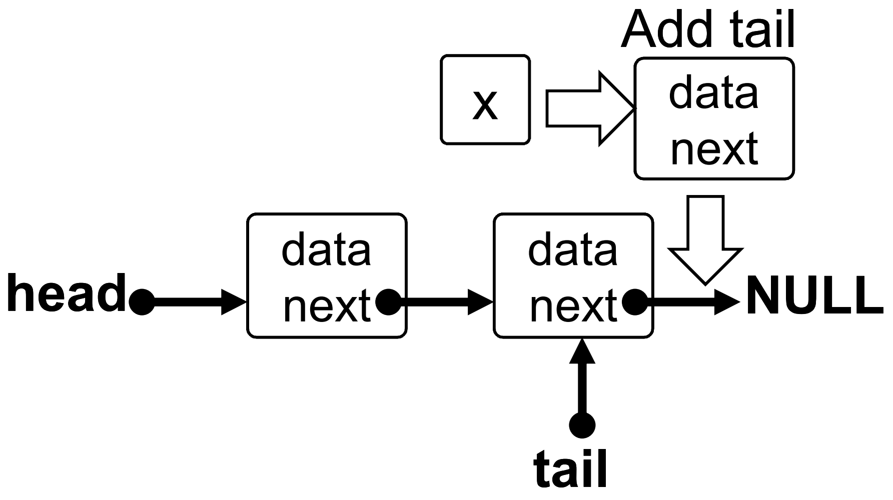
Queue

- Queue implementation:
  - Push: insert element into queue.

Use dynamic array



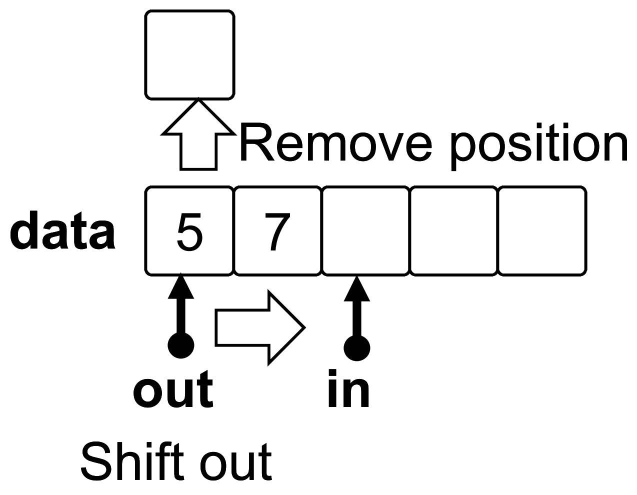
Use linked list



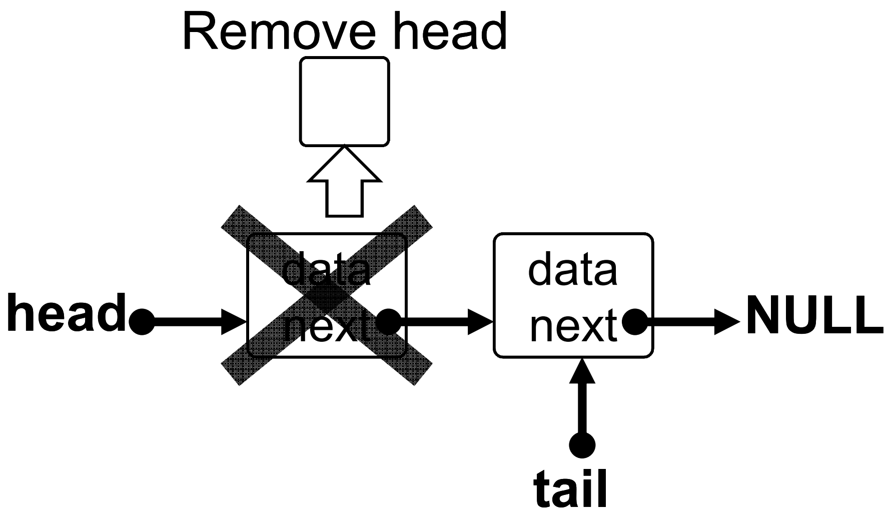
Queue

- Queue implementation:
  - Pop: remove element from queue.

Use dynamic array



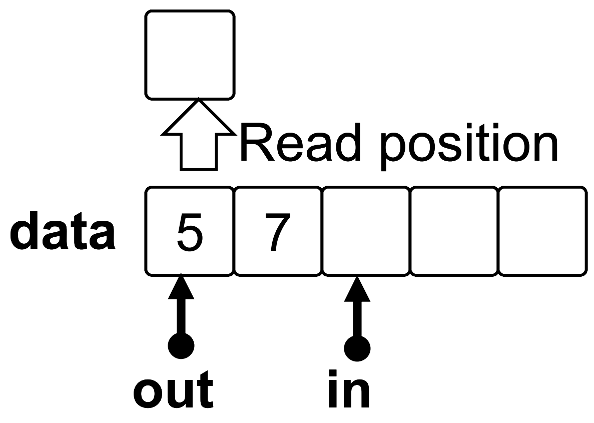
Use linked list



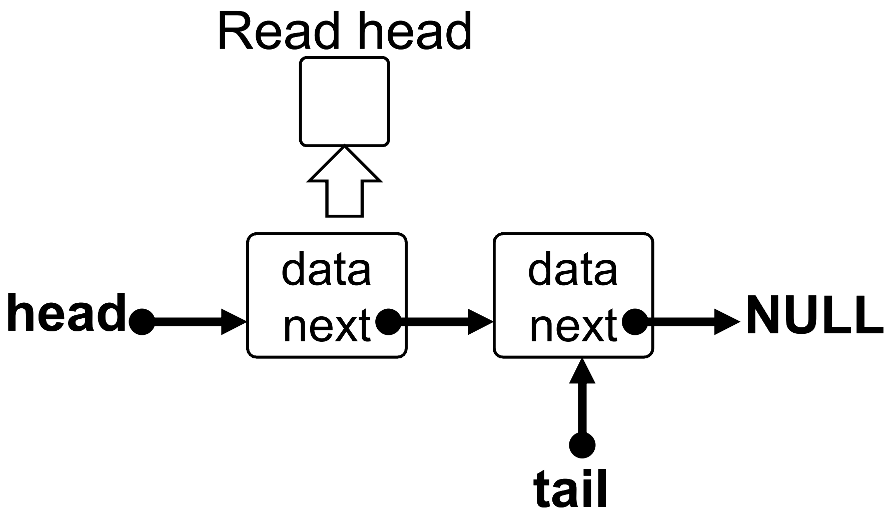
Queue

- Queue implementation:
  - Peek: read element from queue.

Use dynamic array



Use linked list



Queue

- Queue applications:
  - Breadth-first search in tree.
  - System queue.

- Concept:
  - Stack: LIFO – Last In First Out.
  - Queue: FIFO – First In First Out.
- Operations:
  - init, isEmpty, isFull.
  - push, pop, peek.
- Implementations:
  - Dynamic array.
  - Singly linked list.