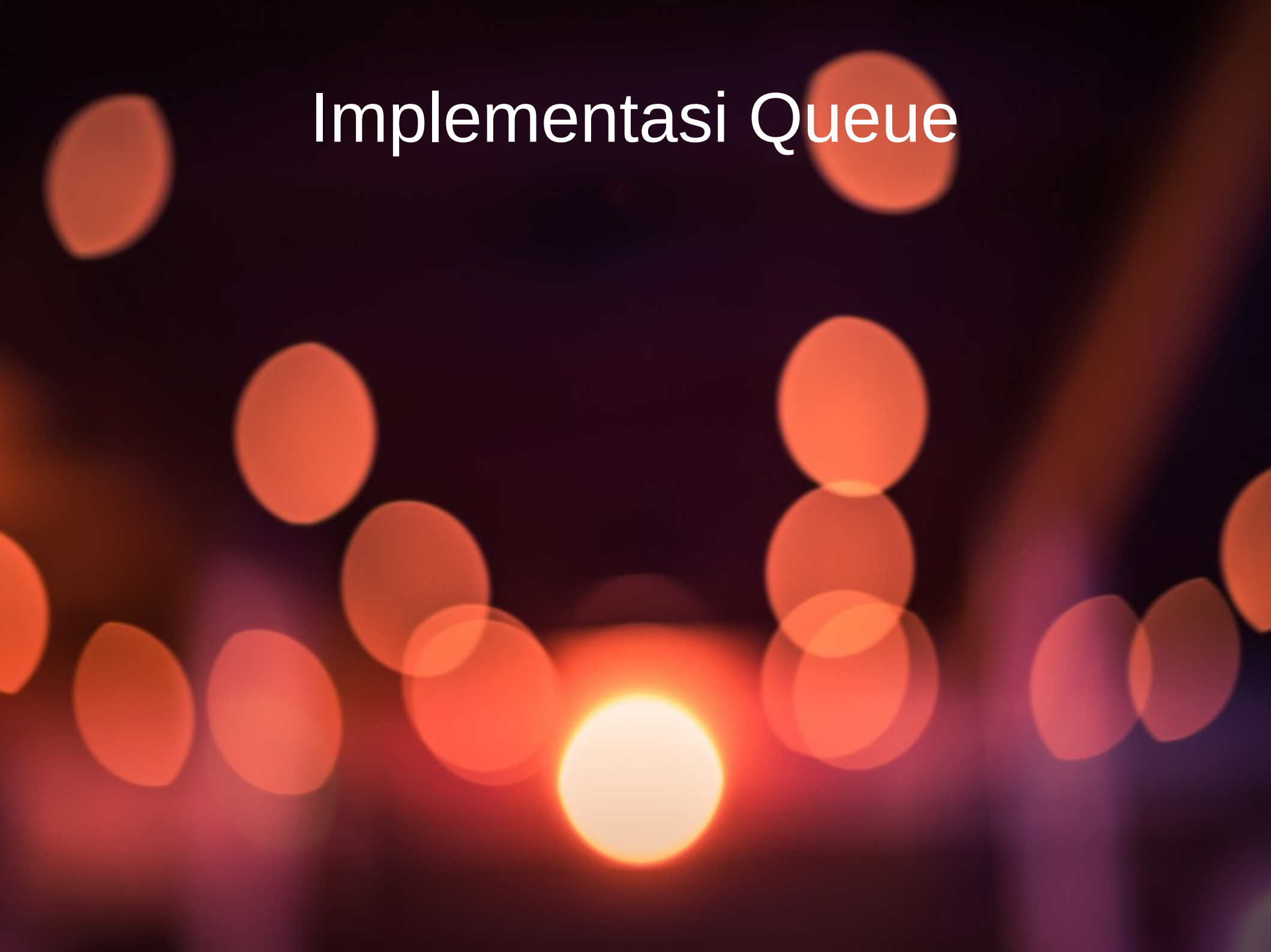
Implementasi Queue



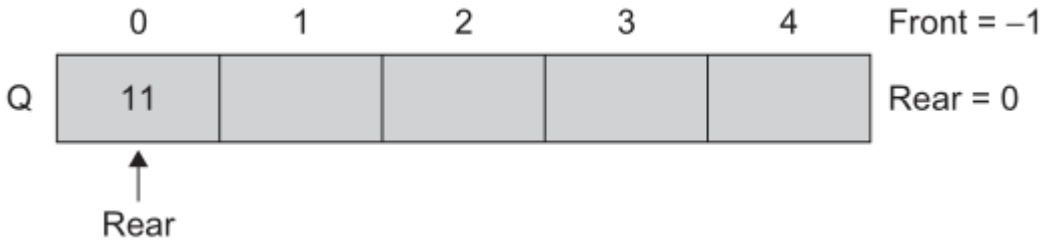
Queue berbasis array

- Struktur untuk menampung data menggunakan tabel / array
- Namun operasi dibatasi sesuai ADT queue:
  - Enqueue
  - Dequeue
- Sesuai karakteristik array, alokasi statis di awal program menyebabkan kapasitas array fix (tetap) sehingga queue punya Maximum daya tampung

Kondisi Queue kosong

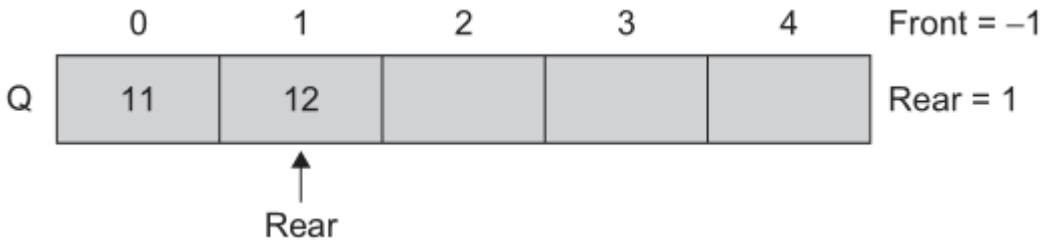
0	1	2	3	4	Front	Rear	Action
					-1	-1	Queue Empty

Mengisi queue 1 elemen



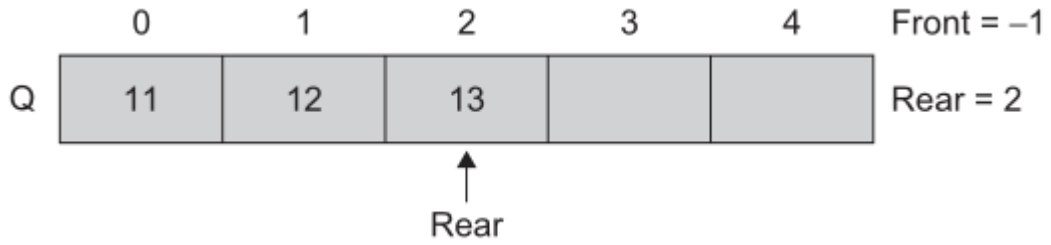
0	1	2	3	4	Front	Rear	Action
11					-1	0	Queue Empty

Mengisi queue 1 elemen



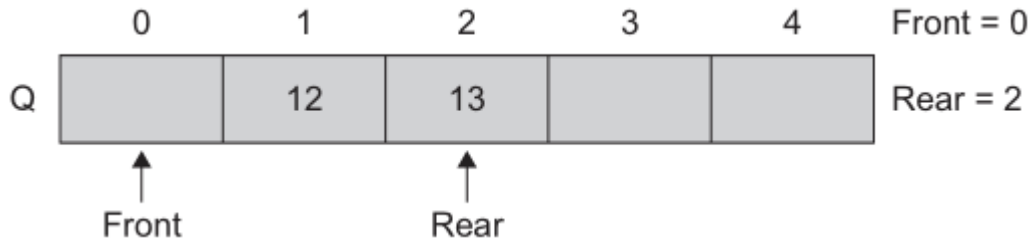
0	1	2	3	4	Front	Rear	Action
11	12				-1	1	Queue Empty

Mengisi queue 3 elemen



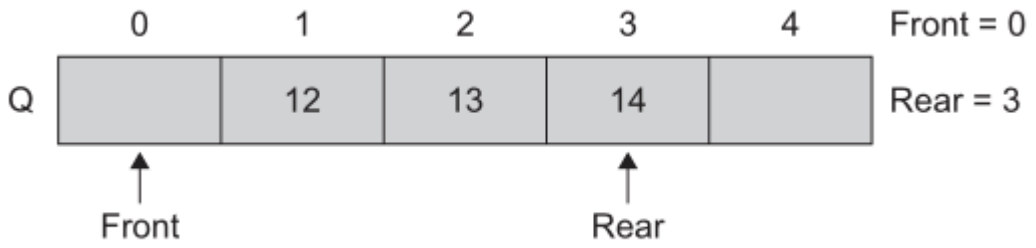
0	1	2	3	4	Front	Rear	Action
11	12	13			-1	2	Queue Empty

Dequeque 1 elemen



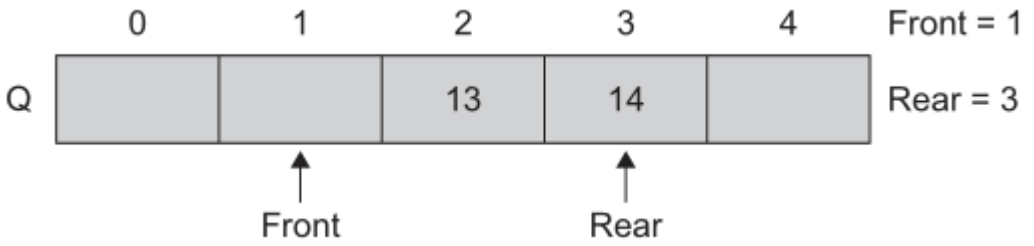
0	1	2	3	4	Front	Rear	Action
	12	13			0	2	Queue Empty

Enqueue 1 elemen



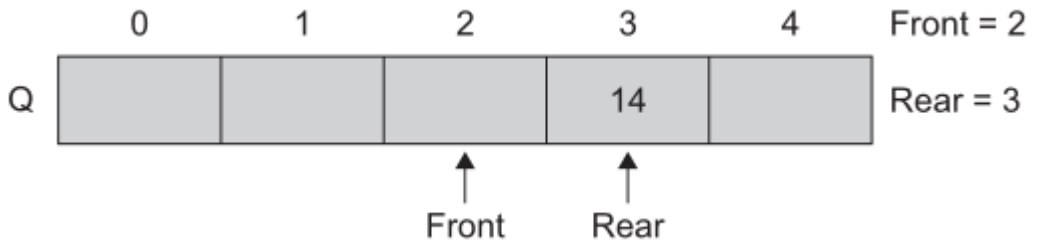
0	1	2	3	4	Front	Rear	Action
	12	13	14		0	3	Queue Empty

Dequeue 1 elemen



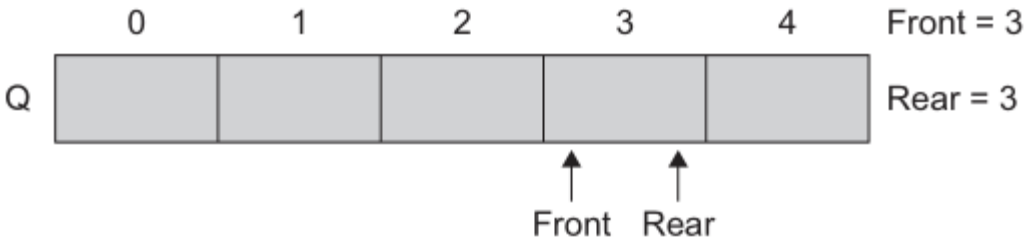
0	1	2	3	4	Front	Rear	Action
		13	14		1	3	Queue Empty

Dequeue 1 elemen



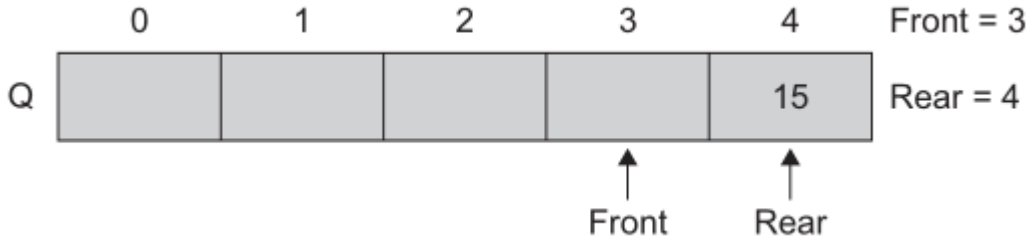
0	1	2	3	4	Front	Rear	Action
			14		2	3	Queue Empty

Dequeue 1 elemen



0	1	2	3	4	Front	Rear	Action
					3	3	Queue Empty

Enqueue 1 elemen

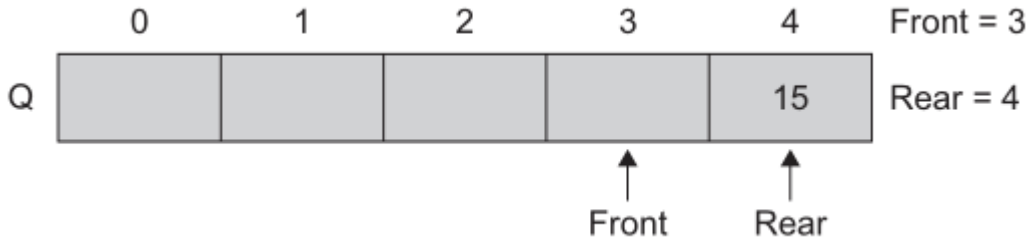


Rear mencapai Batas Maks

0	1	2	3	4	Front	Rear	Action
					3	4	Queue Empty

Masalah yang muncul

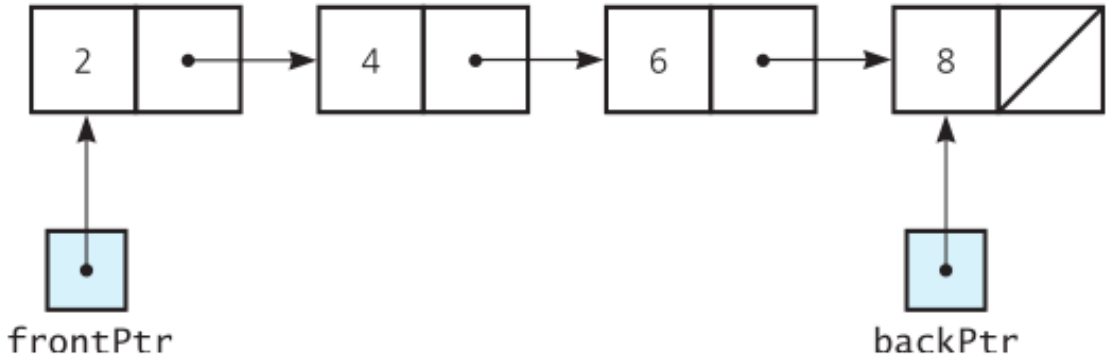
- Queue sebenarnya memiliki banyak ruang kosong didepan
- Namun karena pointer Front sudah bergerak ke belakang dan Rear pada posisi Maks, maka kita tidak bisa menambah elemen lagi



0	1	2	3	4	Front	Rear	Action
					3	4	Queue Empty

Queue dengan List Berkait

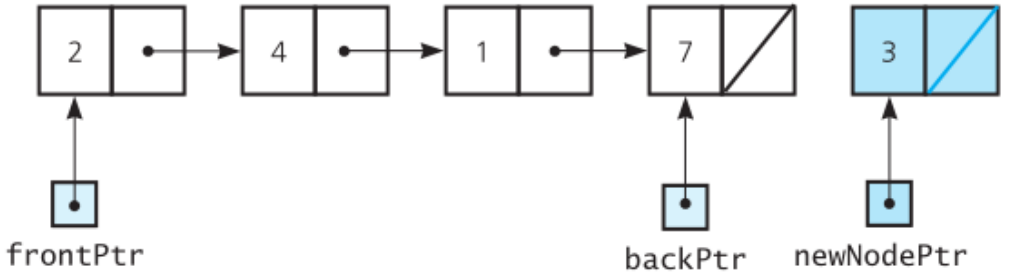
- Untuk menampung data tidak menggunakan array, tapi list berkait (Linked List)
- Keuntungan: dengan alokasi memori dinamis batas maksimum secara eksplisit tidak ada karena selalu bisa meminta memori dari Heap



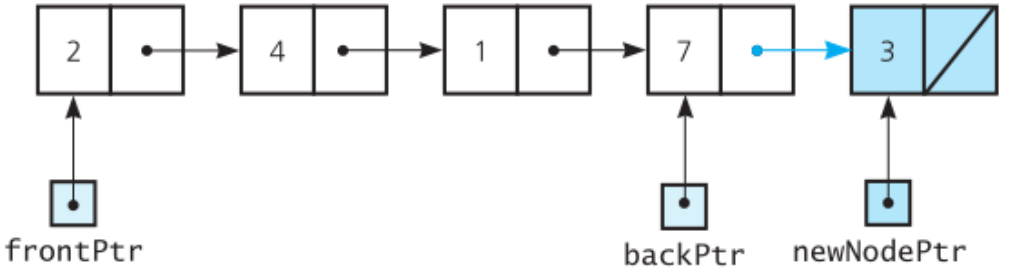
- Penunjuk posisi depan dan belakang menggunakan pointer

Operasi Enqueue

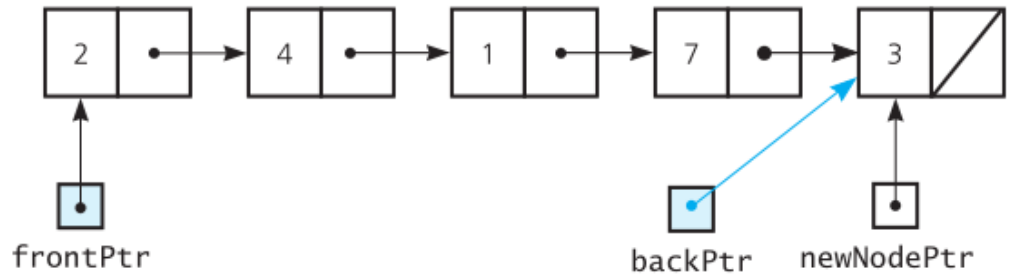
- Menambah elemen pada bagian belakang




```
1. newNodePtr->setNext(nullptr);
```



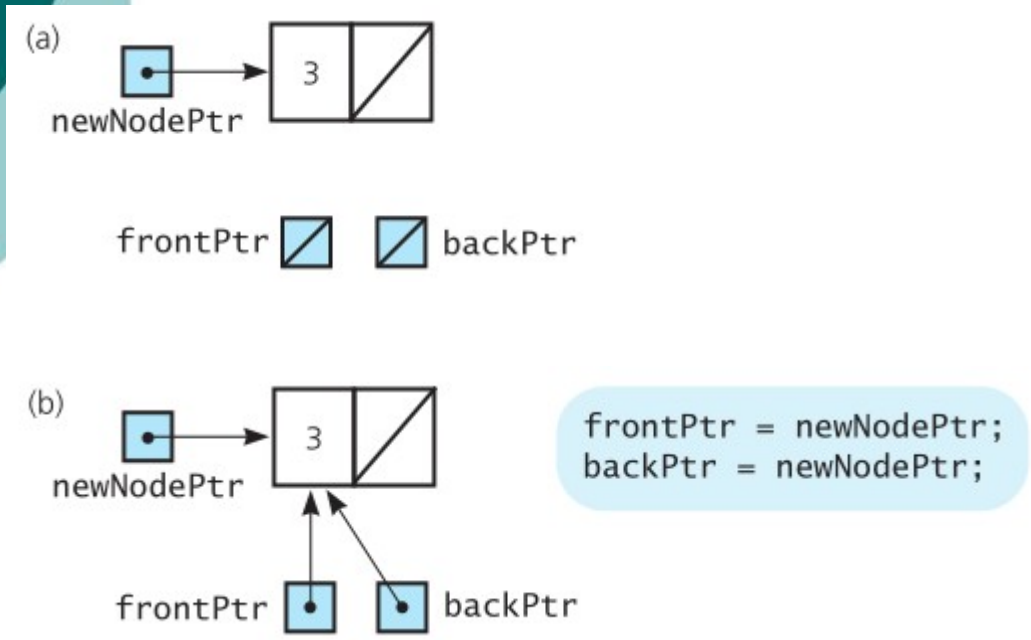

```
2. backPtr->setNext(newNodePtr);
```




```
3. backPtr = newNodePtr;
```

Enqueue (2)

- Menambah elemen jika queue kosong




```
class NodeInt  //node menyimpan info integer
{
private:
  int info;
  NodeInt* next;
public:
  NodeInt(const int i)
  {
    info = i;
  }
  void setNext(NodeInt* nextNode)
  {
    next = nextNode;
  }
};

class LinkedQueue  //queue disimpan sebagai list
{
private:
  NodeInt* backPtr;
  NodeInt* frontPtr;
public:
  LinkedQueue(); //constructor
  bool isEmpty() const;
  void enqueue(const int& item);
};
```

Enqueue(3)

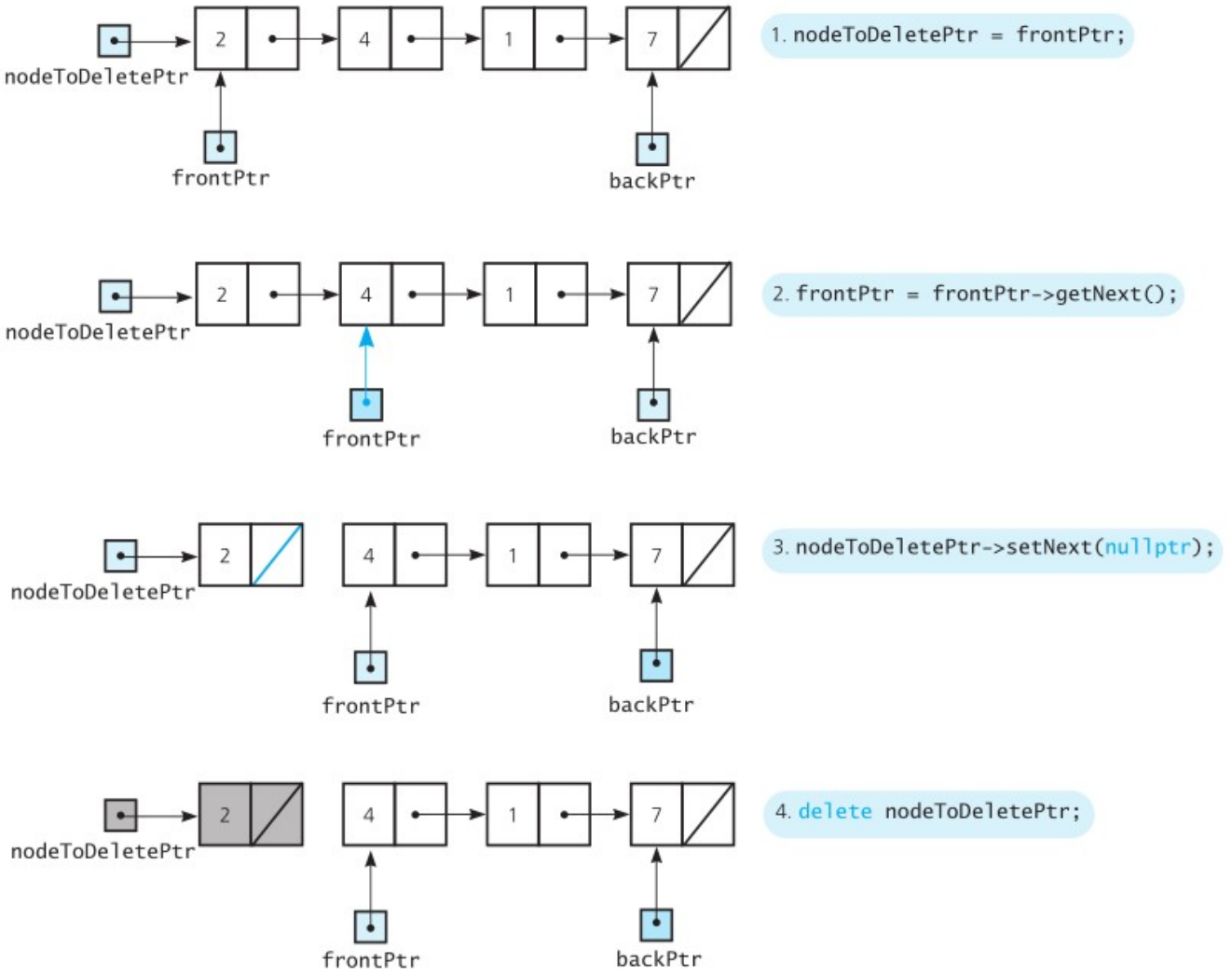
- Implementasi enqueue

```
class LinkedQueue  //queue disimpan sebagai list
{
private:
  NodeInt* backPtr;
  NodeInt* frontPtr;
public:
  LinkedQueue(); //constructor
  bool isEmpty() const;
  void enqueue(const int& item);
};

void LinkedQueue::enqueue(const int& item)
{
  NodeInt* newNodePtr = new NodeInt(item);
  if (isEmpty())
    frontPtr = newNodePtr;
  else
    backPtr->setNext(newNodePtr);
  backPtr = newNodePtr;
}
```

Operasi dequeue

- Update pointer depan



Dequeue(2)

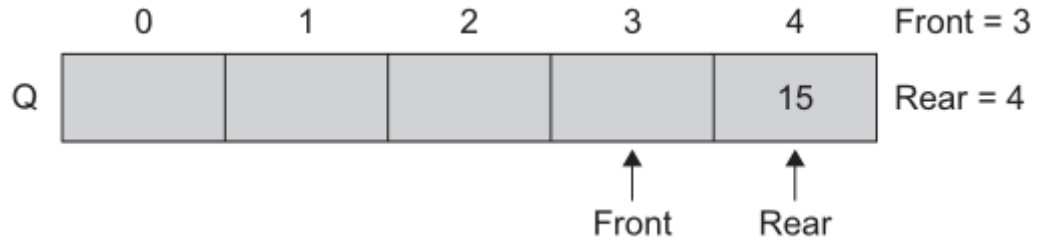
- Implementasi dequeue

```
class NodeInt  //node menyimpan info integer
{
private:
  int info;
  NodeInt* next;
public:
  NodeInt(const int i)
  {
    info = i;
  }
  void setNext(NodeInt* nextNode)
  {
    next = nextNode;
  }
  NodeInt* getNext()
  {
    return next;
  }
};
```

```
void LinkedQueue::dequeue()
{
  if( !isEmpty() ) //dequeue: queue tdk boleh kosong
    {
      NodeInt* nodeToDelete = frontPtr; //yg akan dihapus
      if( frontPtr == backPtr ) //queue 1 elemen
        {
          frontPtr = 0;
          backPtr = 0;
        }
      else
        frontPtr = frontPtr->getNext();
      nodeToDelete->setNext(0);
      delete nodeToDelete;
      nodeToDelete = 0;
    }
}
```

Circular Queue

- Kembali pada masalah implementasi queue dengan array

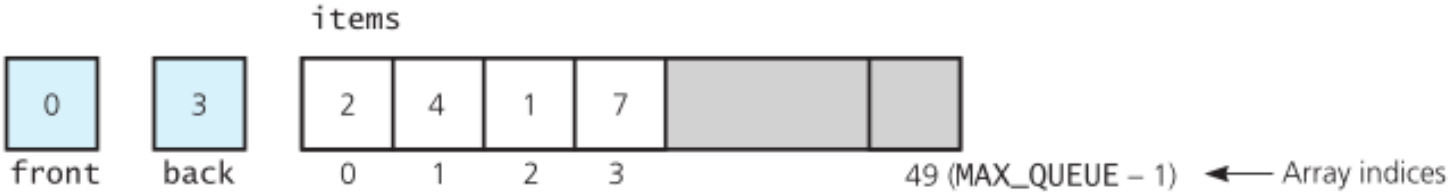


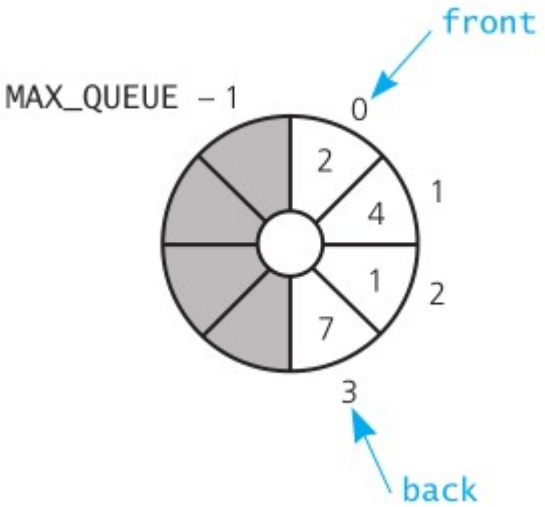
0	1	2	3	4	Front	Rear	Action
					3	4	Queue Empty

- Tidak bisa elemen menambah meskipun masih ada ruang

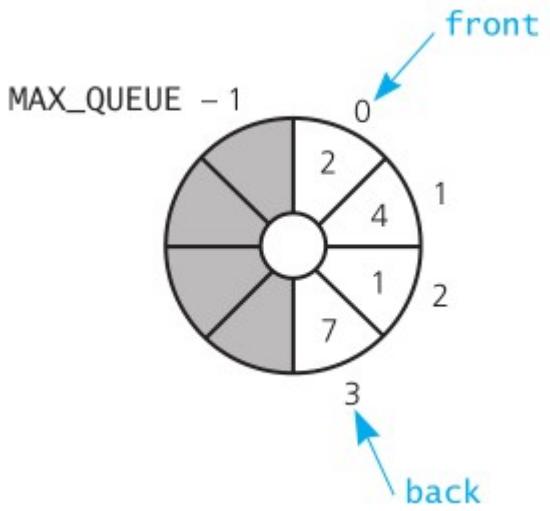
Circular Queue (2)

- Salah satu solusi dengan melihat array sebagai circular (berputar dari belakang ke depan)
- Misal array size 50, telah diisi (enqueue) 4 elemen

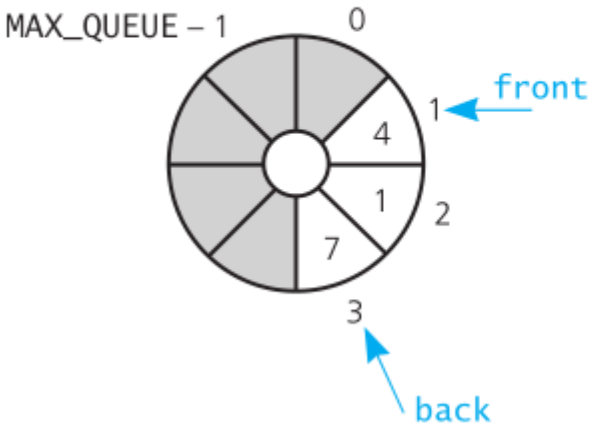




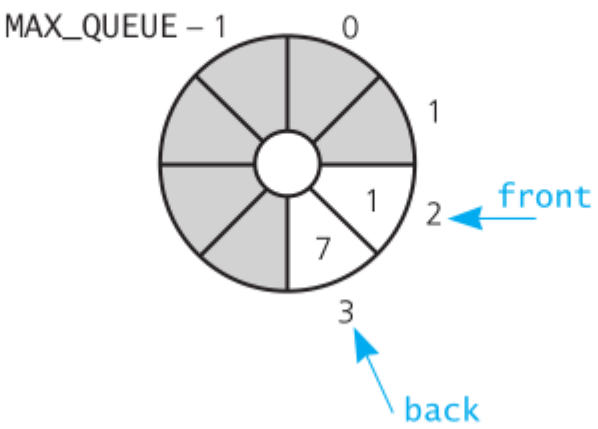
Circular Queue (3)



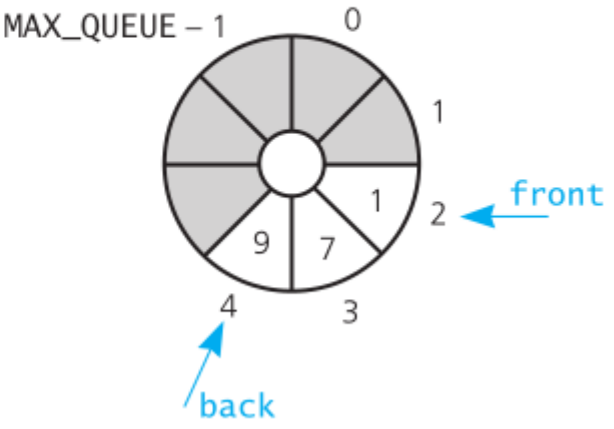
dequeue();



dequeue();



enqueue(9);



Circular Queue (4)