Implementation

In this lesson, we'll implement a queue using an array.

We'll cover the following

Implementation

A queue can be implemented using an array or a doubly-linked list. They will have the same time complexity.

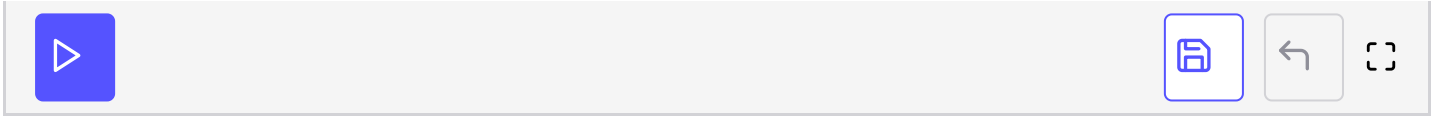
Array

> The limitation in using an array is that the maximum size of the queue is limited.

Keep two pointers `front` and `back`.

```
#include <bits/stdc++.h>
using namespace std;

struct Queue {
  static const int SZ = 8;
  int arr[SZ];
  int front, back;

  Queue() {
    front = back = -1;
  }

  bool isEmpty() {
    return (front < 0);
  }

  void push(int x) {
    if (front == -1) {    // first Node
      front = back = 0;
      arr[back] = x;
```

```
    }
    else {
      arr[++back] = x;
    }
  }

  int pop() {
    if (front > back) {
      cout<<"Queue Underflow";
      return - 1;
    }
    return arr[front++];
  }

  int get_front() {
      if (front > back) {
        cout<<"Empty Queue";
        return -1;
      }
      return arr[front];
  }

  int get_back() {
      if (front > back) {
        cout<<"Empty Queue";
        return -1;
      }
      return arr[back];
  }

  void print_queue() {
    for (int i = front; i <= back; i++)
      cout<<arr[i]<<" <- ";
    cout<<"\n";
  }
};

int main() {
  Queue queue;
  queue.push(1);  queue.print_queue();
  queue.push(2);  queue.print_queue();
  queue.push(3);  queue.print_queue();
  queue.pop();  queue.print_queue();
  queue.pop();  queue.print_queue();
  return 0;
}
```



Linked list

To queue using doubly linked list is very similar and we'll skip the code as an exercise.

Hint: Use `head` as the front and `tail` as the back of the Queue.

Push => Inserts at the end (after tail).

Pop => deletes first node.

In the next lesson, we'll see how to C++ STL queue to solve problem in competitions.