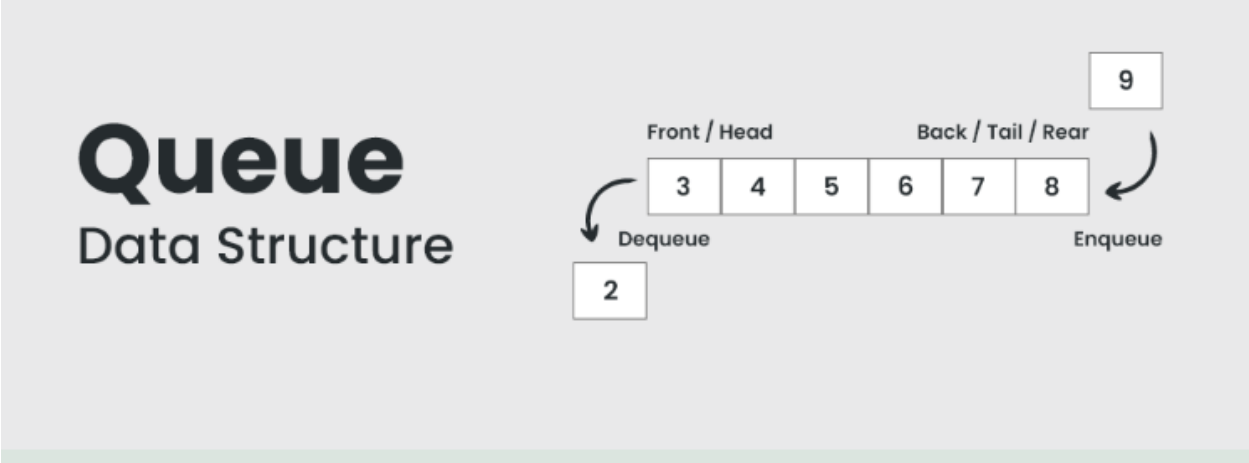
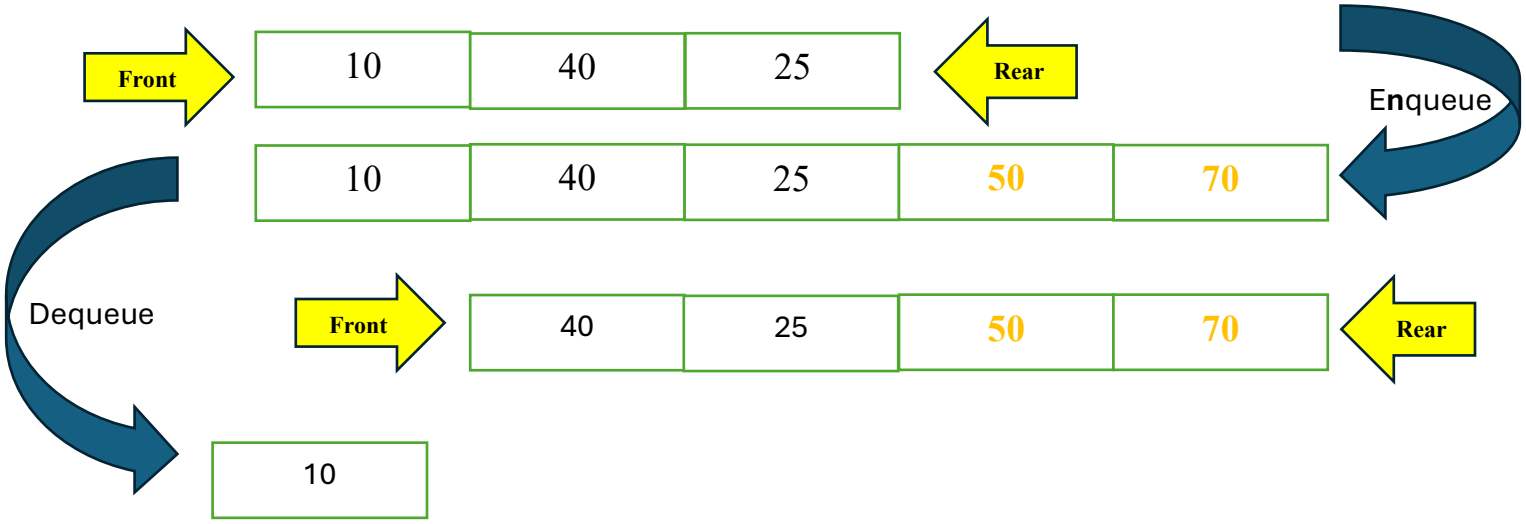
BÀI 2

KIỂU DỮ LIỆU HÀNG ĐỢI

1. Cơ chế hoạt động

Hàng đợi hay còn gọi là Queue là cấu trúc dữ liệu tuyến tính tuân theo nguyên tắc FIFO (First In First Out), do đó các phần tử được thêm vào trước hàng đợi cũng sẽ là các phần tử lấy ra đầu tiên.





2. Biểu diễn hàng đợi bằng mảng một chiều

Biểu diễn hàng đợi bằng mảng một chiều là cách tổ chức và quản lý dữ liệu của hàng đợi (queue) bằng cách sử dụng một mảng một chiều. Trong Python, các thao tác cơ bản của danh sách tương tự như các thao tác của hàng đợi. Do đó, hàng đợi được biểu diễn bằng kiểu list của Python.

5	15	40	60	96	

Hàng đợi	**5**	**15**	**40**	**60**	**96**	
Chỉ số mảng	0	1	2	3	4	

3. Các thao tác cơ bản của hàng đợi

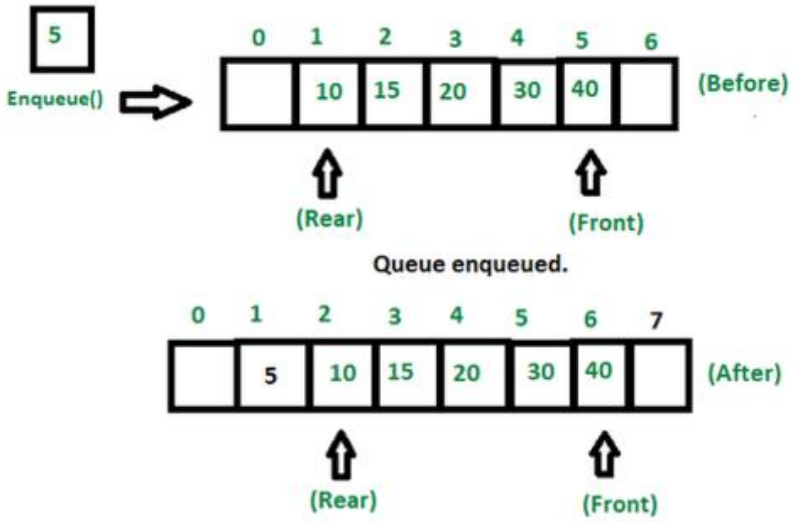
Khi thao tác với hàng đợi, chúng ta thường thực hiện các thao tác nhất định:

- enqueue() : Để chèn một phần tử vào hàng đợi.
- dequeue() : Để xóa một phần tử khỏi hàng đợi.
- front() : Trả về giá trị của phần tử đầu tiên của hàng đợi mà không xóa phần tử đó.
- rear() : Trả về giá trị của phần tử cuối cùng của hàng đợi mà không xóa phần tử đó.
- isEmpty() : Trả về true nếu hàng đợi rỗng, nếu không trả về false.
- isFull() : Trả về true nếu hàng đợi đầy, nếu không trả về false.

Thuật toán enqueue trong cấu trúc dữ liệu hàng đợi

Các bước sau đây cần được thực hiện để thêm dữ liệu vào hàng đợi:

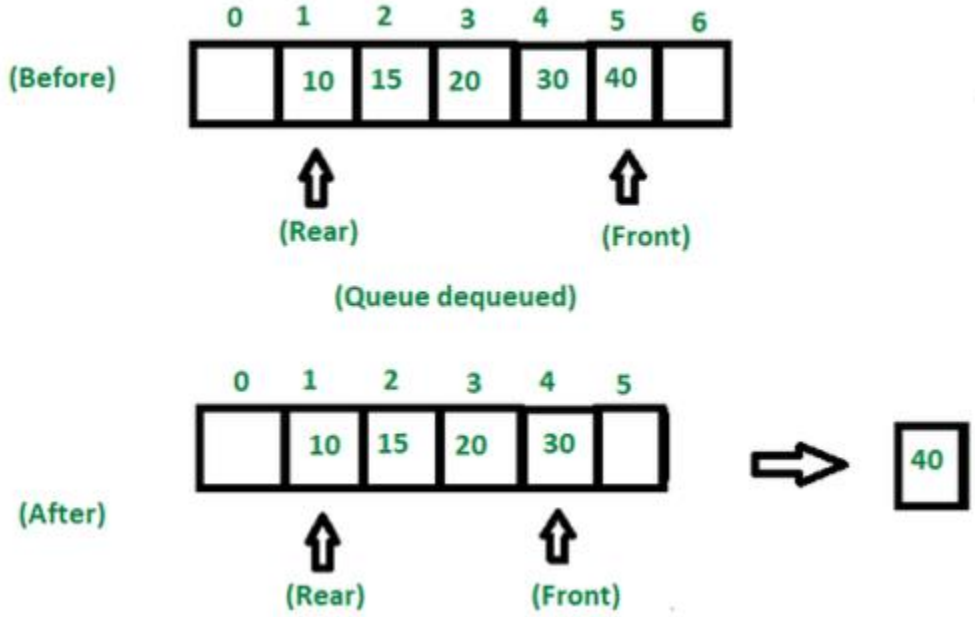
- Kiểm tra xem hàng đợi đã đầy chưa.
- Nếu hàng đợi đầy, hãy trả về lỗi và thoát.
- Nếu hàng đợi chưa đầy, hãy tăng con trỏ phía sau để trỏ đến khoảng trống tiếp theo.
- Thêm phần tử dữ liệu vào vị trí hàng đợi, nơi mà phần tử sau cùng đang trỏ tới.
- Trả về kết quả.



Thuật toán dequeue trong cấu trúc dữ liệu hàng đợi

Các bước sau đây được thực hiện để xóa phần tử của hàng đợi:

- Kiểm tra xem hàng đợi có trống không.
- Nếu hàng đợi trống, hãy trả về lỗi và thoát.
- Nếu hàng đợi không trống, hãy truy cập dữ liệu mà mặt trước đang trỏ tới.
- Tăng con trỏ phía trước để trỏ tới phần tử dữ liệu khả dụng tiếp theo.
- Trả về kết quả.



Thuật toán front trong cấu trúc dữ liệu hàng đợi

- Nếu hàng đợi trống thì trả về giá trị nhỏ nhất.
- Nếu không, trả về giá trị phía trước.

Thuật toán rear trong cấu trúc dữ liệu hàng đợi

- Nếu hàng đợi trống thì trả về giá trị nhỏ nhất.
- Nếu không, trả về giá trị phía sau cùng.

Thuật toán isEmpty trong cấu trúc dữ liệu hàng đợi

- Kiểm tra xem giá trị phía trước có bằng -1 hay không, nếu có thì trả về true nghĩa là hàng đợi trống.
- Nếu không trả về false, nghĩa là hàng đợi không trống.

Thuật toán isFull trong cấu trúc dữ liệu ngăn xếp

- Kiểm tra xem giá trị phía trước có bằng 0 và phía sau có bằng sức chứa của hàng đợi không, nếu có thì trả về true.
- Nếu không trả về false

4. Cài đặt hàng đợi bằng ngôn ngữ lập trình Python

```
queue = []
def enqueue(queue, value):
    queue.append(value)
    print(f"Đã thêm {value} vào hàng đợi.")
def dequeue(queue):
    if is_empty(queue):
        print("Hàng đợi rỗng! Không thể xóa phần tử.")
        return None
    value = queue.pop(0)
    print(f"Đã xóa {value} khỏi hàng đợi.")
    return value
def front(queue):
    if is_empty(queue):
        print("Hàng đợi rỗng! Không có phần tử để xem.")
        return None
    print(f"Phần tử đầu tiên là: {queue[0]}")
    return queue[0]
```

```
def rear(queue):
    if is_empty(queue):
        print("Hàng đợi rỗng! Không có phần tử để xem.")
        return None
    print(f"Phần tử cuối cùng là: {queue[-1]}")
    return queue[-1]
def is_empty(queue):
    return len(queue) == 0

def print_queue(queue):
    if is_empty(queue):
        print("Hàng đợi rỗng.")
    else:
        print("Các phần tử trong hàng đợi:", " ".join(map(str, queue)))
def main():
    print("Chương trình quản lý hàng đợi.")
    print("Chọn một tùy chọn:")
    print("1: Thêm giá trị vào hàng đợi (enqueue).")
    print("2: Xóa phần tử ở đầu hàng đợi (dequeue).")
    print("3: Xem phần tử đầu hàng đợi (front).")
    print("4: Xem phần tử cuối hàng đợi (rear).")
    print("5: In ra các phần tử trong hàng đợi (print queue).")
    print("6: Thoát chương trình (exit).")
    while True:
        try:
            choice = int(input("\nNhập lựa chọn (1-6): ").strip())
            if choice == 1:
                value = input("Nhập giá trị:").strip()
                enqueue(queue, value)
            elif choice == 2:
                dequeue(queue)
            elif choice == 3:
                peek(queue)
            elif choice == 4:
                rear(queue)
            elif choice == 5:
                print_queue(queue)
```

```
            elif choice == 6:
                print("Thoát chương trình.")
                break
            else:
                print("Lựa chọn không đúng! Vui lòng nhập số từ 1 đến 6.")
        except ValueError:
            print("Vui lòng nhập một số hợp lệ từ 1 đến 6!")
if __name__ == "__main__":
    main()
```

5. Một số ví dụ mẫu

Bài 1. Viết chương trình nhập mảng từ bàn phím và in ra các số chia hết cho 3 và 5, yêu cầu sử dụng hàng đợi để lưu trữ các số chia hết cho 3 và 5, sau đó in ra các số này.

```
queue = []
def enqueue(queue, value):
    queue.append(value)
def dequeue(queue):
    if is_empty(queue):
        return None
    return queue.pop(0)
def is_empty(queue):
    return len(queue) == 0
def main():
    array = input("Nhập mảng các số, cách nhau bởi dấu cách: ").strip().split()
    array = [int(x) for x in array]  # Chuyển đổi mảng sang số nguyên
    print("\nĐang xử lý các số chia hết cho 3 và 5...")
    for num in array:
        if num % 3 == 0 and num % 5 == 0:
            enqueue(queue, num)
    print("\nCác số chia hết cho 3 và 5 là:")
    while not is_empty(queue):
        print(dequeue(queue), end=" ")
    print()
```

```
if __name__ == "__main__":
    main()
```

Bài 2. Viết chương trình sử dụng hàng đợi để in ra các số chính phương , phần tử đầu tiên , phần tử cuối cùng của hàng đợi với đầu vào là các phần tử từ hai mảng nhập từ bàn phím

```
import math
queue = []
def enqueue(queue, value):
    queue.append(value)
def is_empty(queue):
        return len(queue) == 0
def is_perfect_square(n):
    if n < 0:
        return False
    sqrt_n = int(math.sqrt(n))
    return sqrt_n * sqrt_n == n
def front(queue):
        if is_empty(queue):
        return None
    return queue[0]
def rear(queue):
        if is_empty(queue):
        return None
    return queue[-1]
def main():
    array1 = input("Nhập mảng 1 (các số cách nhau bởi dấu cách):
").strip().split()
    array2 = input("Nhập mảng 2 (các số cách nhau bởi dấu cách):
").strip().split()
    # Chuyển đổi các phần tử trong mảng thành số nguyên
    array1 = [int(x) for x in array1]
    array2 = [int(x) for x in array2]
    # Kiểm tra và thêm các số chính phương từ hai mảng vào hàng đợi
    for num in array1 + array2:
```

```
        if is_perfect_square(num):
            enqueue(queue, num)
    # In các số chính phương từ hàng đợi
    print("\nCác số chính phương được thêm vào hàng đợi:")
    if not is_empty(queue):
        print(" ".join(map(str, queue)))
    else:
        print("Không có số chính phương nào trong hai mảng.")
    # In phần tử đầu tiên và phần tử cuối cùng
    print("\nPhần tử đầu tiên trong hàng đợi:", peek(queue))
    print("Phần tử cuối cùng trong hàng đợi:", rear(queue))
if __name__ == "__main__":
    main()
```