Data Structures and Algorithms

HUS – HKI, 23 - 24

Assignment 4

Lecturer: Nguyễn Thị Hồng Minh
Trần Bá Tuấn - Đặng Trung Du

§ Stack - Queue §

Phần 1: Mục tiêu

- Sinh viên nắm được định nghĩa, các thao tác và độ phức tạp của các thao tác ứng với cấu trúc dữ liệu ngắn xếp (stack) và hàng đợi (Queue)
- Sinh viên triển khai cấu trúc dữ liệu ngăn xếp, hàng đợi bằng mảng và danh sách liên kết.
- Sinh viên hiểu các thuật toán sử dụng ngăn xếp để giải các bài toán như kiểm tra xâu ngoặc hợp lệ, biến đổi biểu thức trung tố thành tiền tố/hậu tố, tính toán biểu thức tiền tố/hậu tố bằng ngăn xếp.

Phần 2: Tài liệu đọc thêm

- Stack (Ngăn xếp): https://www.geeksforgeeks.org/stack-data-structure/
- Queue (Hàng đợi): https://www.geeksforgeeks.org/queue-data-structure/

Phần 3: Thực hành

(1) Bài tập

1. Stack - Ngăn xếp

Bài tập 1. *Trả lời câu hỏi kèm theo đoạn code sau:*

Cho một stack rỗng s. Giả sử chuỗi họ và tên của bạn viết liền không dấu là hoten. Sau một loạt các thao tác sau, kết quả cuối cùng của stack là bao nhiêu?

```
for(int i = 0; i < hoten.length() ; i++){
    if(i % 2 == 0){
        s.push(hoten.charAt(i));
    }
    if(i % 3 == 0) {
        s.pop();
    }
}
```

Bài tập 2. *Tạo giao diện StackInterface như sau:*

```java
public interface StackInterface<E> extends Iterable<E> {
    public void push(E element);
    public E pop();
    public boolean isEmpty();
    public E top();
}
```

a) *Xây dựng cấu trúc dữ liệu Stack sử dụng mảng, cài đặt giao diện StackInterface đã xây dựng ở trên với gợi ý như trong slide bài giảng (Stack Implementation using array).*

b) *Xây dựng cấu trúc dữ liệu Stack sử dụng danh sách liên kết, cài đặt giao diện StackInterface đã xây dựng ở trên với gợi ý như sau:*

```java
public class LinkedListStack<E> implements StackInterface<E> {
    class Node {
        E element;
        Node next;
    }

    private Node stack = null;

    @Override
    public void push(E element) {

    }

    @Override
    public E pop() {
        return null;
    }

    @Override
    public boolean isEmpty() {
        return false;
    }

    @Override
    public E top() {
        return false;
    }

    @Override
    public Iterator<E> iterator() {
        // TODO Auto-generated method stub
        return new StackIterator();
    }
    class StackIterator implements Iterator<E> {
        private Node currentNode = stack;
        @Override
        public boolean hasNext() {
            // TODO Auto-generated method stub
            return currentNode != null;
        }
        @Override
        public E next() {
            // TODO Auto-generated method stub
            E data = currentNode.element;
            currentNode = currentNode.next;
            return data;
        }
    }
}
```

Bài tập 3. *a) Sử dụng stack viết chương trình xét tính hợp lệ về dấu ngoặc của biểu thức:*

Ví dụ biểu thức hợp lệ về dấu ngoặc
$(a-b)*(c+d)$
$(10+8)/((5-2)*17)$

Ví dụ biểu thức không hợp lệ về dấu ngoặc
$(a+b)*c-d)$
$(10-8/((2+5)*17)$
$)u-v)*(m+n)$

b) Tính giá trị biểu thức nếu hợp lệ về dấu ngoặc

Ví dụ 1:

- *Input:* $(1+((2+3)*(8*5)))$
- *Output:* 201

Ví dụ 2:

- *Input:* $(5-(8-4)*(2+3))+(8/4)$
- *Output:* -13

2. Queue - Hàng đợi

Bài tập 4. *Xây dựng giao diện QueueInterface như sau:*

```
public interface QueueInterface<E> extends Iterable<E> {
    public void enqueue(E element);
    public E dequeue();
    public boolean isEmpty();
}
```

a) Xây dựng kiểu dữ liệu Queue sử dụng mảng với gợi ý như sau:

```
public class ArrayQueue<E> implements QueueInterface<E> {

    private E[] queue;
    private int n = 0;
    private int top = 0;
    private int count = 0;
    private int default_size = 100;
    public ArrayQueue(int capacity) {
        n = capacity;
        queue = (E[]) new Object[capacity];
    }
    public ArrayQueue() {
        n = default_size;
        queue = (E[]) new Object[default_size];
    }
    @Override
    public void enqueue(E element) {
        // TODO Auto-generated method stub
    }
    @Override
    public E dequeue() {
        // TODO Auto-generated method stub
        return null;
    }
    @Override
    public boolean isEmpty() {
        // TODO Auto-generated method stub
```

```
28          return false;
29      }
30      @Override
31      public Iterator<E> iterator() {
32          // TODO Auto-generated method stub
33          return new ArrayQueueIterator();
34      }
35
36      class ArrayQueueIterator implements Iterator<T> {
37          private int current = top;
38          private int num = 0;
39          @Override
40          public boolean hasNext() {
41              // TODO Auto-generated method stub
42              return num < count;
43          }
44          @Override
45          public E next() {
46              // TODO Auto-generated method stub
47              E data = queue[(current + num) % n];
48              num++;
49              return data;
50          }
51      }
52  }
```

b) Xây dựng kiểu dữ liệu Queue sử dụng danh sách liên kết.

Bài tập 5. *Sử dụng queue kết hợp với stack đã xây dựng ở trên viết chương trình kiểm tra chuỗi Palindrome.*

(2) Bài tập thêm

Luyện tập 1. *Sinh viên tự chọn 2 bài trong Bài tập từ 13 đến 20.*

https://codelearn.io/learning/cau-truc-du-lieu-va-giai-thuat

1. Luyện tập về Stack - Ngăn xếp

Luyện tập 2. *a) Triển khai stack bằng mảng*

https://practice.geeksforgeeks.org/problems/implement-stack-using-array/1

b) Triển khai stack bằng danh sách liên kết

https://practice.geeksforgeeks.org/problems/implement-stack-using-linked-list/1

Lưu ý: *Sinh viên đã hoàn thiện ở Bài tập 2, đồng thời sinh viên có thể dựa vào các link trên để tham khảo thêm cho bài tập 2.*

Luyện tập 3. *Xóa phần tử giữa của stack*

https://practice.geeksforgeeks.org/problems/delete-middle-element-of-a-stack/1

Gợi ý:

- *Stack chỉ hỗ trợ 2 thao tác chèn và xóa phần tử ở đỉnh stack.*
- *Đề bài đã cho kích thước của stack, thực hiện các thao tác lấy ra các phần tử ở đỉnh stack cho đến khi phần tử ở giữa được lấy ra.*
- *Sau đó, đưa các phần tử đã được lấy ra trở lại stack trừ phần tử ở giữa.*

Luyện tập 4. *Kiểm tra xâu ngoặc hợp lệ (Mô tả chi tiết bài tập 3 ở trên)*

https://leetcode.com/problems/valid-parentheses/

Gợi ý:

- *Nguồn - https://www.geeksforgeeks.org/check-for-balanced-parentheses-in-an-expression/*
- *Dùng một stack chứa các ngoặc mở mà chưa tìm được ngoặc đóng tương ứng.*
- *Duyệt xâu, nếu gặp ngoặc mở thì cho vào stack.*
- *Nếu gặp ngoặc đóng thì kiểm tra tính tương thích giữa ngoặc đóng hiện tại và ngoặc mở đang có ở đỉnh stack. Nếu tương thích thì đó sẽ là ngoặc mở tương ứng của ngoặc đóng đang xét. Loại ngoặc đó ra khỏi stack. Nếu không tương thích hoặc stack đang rỗng thì xâu ngoặc không hợp lệ.*
- *Khi kết thúc duyệt xâu mà stack không rỗng, nghĩa là có nhiều ngoặc mở hơn ngoặc đóng, xâu ngoặc cũng không hợp lệ.*

Luyện tập 5. *Tính toán biểu thức hậu tố*

https://leetcode.com/problems/evaluate-reverse-polish-notation/
hoặc: https://practice.geeksforgeeks.org/problems/evaluation-of-postfix-expression1735/1

Nguồn: *https://www.geeksforgeeks.org/stack-set-4-evaluation-postfix-expression/*
(Biểu thức trung tố thành hậu tố) *https://practice.geeksforgeeks.org/problems/infix-to-postfix-1587115620/1*

2. Luyện tập về Queue - Hàng đợi

Luyện tập 6. *Triển khai Queue bằng danh sách liên kết*

https://practice.geeksforgeeks.org/problems/implement-queue-using-linked-list/1

- *Nguồn: https://www.geeksforgeeks.org/queue-linked-list-implementation/*

Lưu ý: *Sinh viên đã hoàn thiện ở Bài tập 4, đồng thời sinh viên có thể dựa vào các link trên để tham khảo thêm cho bài tập 4.*

Luyện tập 7. *Triển khai queue bằng 2 stacks*

https://practice.geeksforgeeks.org/problems/queue-using-two-stacks/1

- *Mục đích là kiểm tra kỹ năng triển khai các thao tác push/pop của stack thông qua các thao tác của hàng đợi.*
- *Độ phức tạp của các thao tác stack trong trường hợp này không là $O(1)$ cho cả hai phép toán mà là $O(N)$ và $O(1)$.*

Luyện tập 8. *Số cuộc gọi gần đây*

https://leetcode.com/problems/number-of-recent-calls/

Gợi ý:

- *Trả về số cuộc gọi trong 3001s tính cả thời điểm hiện tại.*
- *Khi thực hiện thao tác ping, thêm giá trị t vào cuối danh sách.*
- *Trả về số phần tử trong danh sách có giá trị nằm trong khoảng $[t - 3000, t]$.*
- *Sử dụng cấu trúc dữ liệu queue để thực hiện các thao tác trên: ping - thêm 1 phần tử t vào queue, sau đó xóa các phần tử khỏi queue nếu giá trị của nó nhỏ hơn $t - 3000$. Kích thước của queue sau khi thực hiện phép xóa chính là đáp án cần tìm.*

Luyện tập 9. *Triển khai queue vòng*

https://leetcode.com/problems/design-circular-queue/

- *Nguồn: https://www.geeksforgeeks.org/circular-queue-set-1-introduction-array-implementation/*

- *Nguồn khác: https://www.geeksforgeeks.org/circular-queue-set-2-circular-linked-list-implementation/?ref=rp*

Luyện tập 10. *Triển khai deque vòng*

https://leetcode.com/problems/design-circular-deque/

- *Nguồn: https://www.geeksforgeeks.org/implementation-deque-using-circular-array/*
- *Nguồn tham khảo thêm: https://www.geeksforgeeks.org/implementation-deque-using-doubly-linked-list/*

(3) Quy cách nộp bài

- Mỗi sinh viên hoàn thành bài tập trong package có tên Hw4_<idSinhvien>_<Hovaten>
- Trong đó, <idSinhvien> là mã sinh viên.
- Sinh viên nộp bài làm trên tài khoản của mình bao gồm:
  1. File nén .zip của thư mục chứa package (Hw4_<idSinhvien>_<Hovaten>.zip)
  2. Tất cả các file nguồn *.java.
- Khi hết hạn nộp bài, các bài làm sẽ công bố để thực hiện chấm chéo bài tập.
- Sinh viên không nộp bài sẽ nhận điểm 0 bài tập tuần.
- Sinh viên CÓ GIAN LẬN trong nộp bài tập sẽ bị ĐÌNH CHỈ môn học (điểm 0 cho tất cả các điểm thành phần).

Lưu ý:

- **Sinh viên phải làm toàn bộ phần bài tập (từ bài tập 1 đến bài tập 5).**
- **Trong phần Bài tập thêm, sinh viên thực hiện yêu cầu của Luyện tập 1, tự chọn 1 bài trong 3 bài (Luyện tập 3 đến 5) và tự chọn 1 bài trong 3 bài (Luyện tập 7 đến 10).**