CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

NGĂN XẾP - HÀNG ĐỢI

LaTeX, bởi Ngô Hoàng Tùng

I. Mục tiêu bài học

1. Ngăn xếp (Stack)

a. Hiểu được khái niệm ngăn xếp, các thao tác cơ bản trên ngăn xếp.

b. Biết cách cài đặt ngăn xếp.

c. Biết cách sử dụng ngăn xếp trong các bài toán thực tế.

2. Hàng đợi (Queue)

a. Hiểu được khái niệm hàng đợi, các thao tác cơ bản trên hàng đợi.

b. Biết cách cài đặt hàng đợi.

c. Biết cách sử dụng hàng đợi trong các bài toán thực tế.

II. Nội dung bài học

1. Ngăn xếp (Stack)

a. Hiểu được khái niệm ngăn xếp, các thao tác cơ bản trên ngăn xếp.
– KKhái niệm: Ngăn xếp là một cấu trúc dữ liệu theo nguyên tắc LIFO (Last In First Out), nghĩa là phần tử được thêm vào sau cùng sẽ được lấy ra trước tiên.
– Các thao tác cơ bản:
+ Push: Thêm phần tử vào ngăn xếp.
+ Pop: Lấy phần tử trên cùng ra khỏi ngăn xếp và loại bỏ nó.
+ Peek/Top: Lấy giá trị của phần tử trên cùng mà không loại bỏ nó.
+ IsEmpty: Kiểm tra ngăn xếp có rỗng hay không.

b. Biết cách cài đặt ngăn xếp.

```
struct Stack { // CTDL Stack
    int top;
    int capacity;
    int* array;
};
Stack* createStack(int capacity) { // Create a stack of given capacity
    Stack* stack = new Stack;
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = new int[capacity];
    return stack;
}
bool isFull(Stack* stack) { // Check if stack is full
    return stack->top == stack->capacity - 1;
}
```

```
bool isEmpty(Stack* stack) { // Check if stack is empty
    return stack->top == -1;
}
void push(Stack* stack, int item) { // Push an item onto the stack
    if (isFull(stack)) return;
    stack->array[++stack->top] = item;
}
int pop(Stack* stack) { // Pop an item from the stack
    if (isEmpty(stack)) return -1;
    return stack->array[stack->top--];
}
int Top(Stack* stack) { // Get the top item of the stack
    if (isEmpty(stack)) return -1;
    return stack->array[stack->top];
}
void deleteStack(Stack* stack) { // Delete the stack
    delete[] stack->array;
    delete stack;
}
```

2. Hàng đợi (Queue)

a. Hiểu được khái niệm hàng đợi, các thao tác cơ bản trên hàng đợi.
– Khái niệm: Hàng đợi là một cấu trúc dữ liệu theo nguyên tắc FIFO (First In First Out), nghĩa là phần tử được thêm vào trước sẽ được lấy ra trước tiên.
– Các thao tác cơ bản:
+ Enqueue: Thêm phần tử vào cuối hàng đợi.
+ Dequeue: Lấy phần tử đầu tiên ra khỏi hàng đợi và loại bỏ nó.
+ Peek/Front: Lấy giá trị của phần tử đầu tiên mà không loại bỏ nó.
+ IsEmpty: Kiểm tra hàng đợi có rỗng hay không.

b. Biết cách cài đặt hàng đợi.

```
struct Queue { // CTDL Queue
    int front, rear, size;
    int capacity;
    int* array;
};
Queue* createQueue(int capacity) { // Create a queue of given capacity
    Queue* queue = new Queue;
    queue->capacity = capacity;
    queue->front = queue->size = 0;
    queue->rear = capacity - 1; // Rear is initialized to the last
index
    queue->array = new int[capacity];
    return queue;
}
bool isFull(Queue* queue) { // Check if queue is full
    return (queue->size == queue->capacity);
}
bool isEmpty(Queue* queue) { // Check if queue is empty
```

```
        return (queue->size == 0);
    }
    void enqueue(Queue* queue, int item) { // Enqueue an item to the queue
        if (isFull(queue)) return;
        queue->rear = (queue->rear + 1) % queue->capacity; // Circular
increment
        queue->array[queue->rear] = item;
        queue->size++;
    }
    int dequeue(Queue* queue) { // Dequeue an item from the queue
        if (isEmpty(queue)) return -1;
        int item = queue->array[queue->front];
        queue->front = (queue->front + 1) % queue->capacity; // Circular
increment
        queue->size--;
        return item;
    }
    int front(Queue* queue) { // Get the front item of the queue
        if (isEmpty(queue)) return -1;
        return queue->array[queue->front];
    }
    void deleteQueue(Queue* queue) { // Delete the queue
        delete[] queue->array;
        delete queue;
    }
```

*Ngoài lề: Trong C++, ta có thể sử dụng thư viện STL để cài đặt ngăn xếp và hàng đợi một cách dễ dàng hơn. Ví dụ:

```
#include <stack> // STL of stack
#include <queue> // STL of queue

std::stack<int> s; // Create a stack
s.push(1); // push an element onto the stack
int top = s.top(); // Get the top element
s.pop(); // Pop the top element

std::queue<int> q; // Create a queue
q.push(1); // Enqueue an element into the queue
int front = q.front(); // Get the front element
q.pop(); // Dequeue the front element
```

– Nhưng trong chương trình học sẽ không sử dụng thư viện STL, mà sẽ tự cài đặt các cấu trúc dữ liệu này bằng danh sách liên kết để hiểu rõ hơn về cách hoạt động của chúng cũng như phù hợp với môn học.
– Có thể tìm hiểu thêm ở đây: https://wiki.vnoi.info/algo/data-structures/Stack và https://www.geeksforgeeks.org/queue-cpp-stl/

III. Bài tập

1. Ngăn xếp (Stack)

Câu 1:

Cho n là số nguyên dương và n số nguyên dương $a_1, a_2, \ldots, a_n$. Số may mắn của dãy số $a_1, a_2, \ldots, a_k$ (với $k > 1$) là kết quả XOR của số lớn nhất và lớn thứ hai trong dãy số đó (XOR trong C++ là ^). Với mỗi đoạn $[l, r]$ $(1 \leq l \leq r \leq n)$, sẽ có 1 số may mắn, hãy tìm số may mắn có giá trị lớn nhất trong tất cả các đoạn $[l, r]$ của dãy số $a_1, a_2, \ldots, a_n$.

Input:

Dòng đầu tiên chứa số nguyên dương n $(1 \leq n \leq 1000)$.

Dòng thứ hai chứa n số nguyên dương phân biệt $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$.

Output:

In ra số may mắn lớn nhất trong tất cả các đoạn [l, r] của dãy số $a_1, a_2, \ldots, a_n$.

Input	Output
5 5 2 1 4 3	7

Giải thích:

– Các đoạn [l, r] của dãy số $a_1, a_2, \ldots, a_n$ là: [1, 2], [1, 3], [1, 4], [1, 5], [2, 3], [2, 4], [2, 5], [3, 4], [3, 5], [4, 5].

– Số may mắn của các đoạn này lần lượt là: (5 XOR 2 = 7), (5 XOR 1 = 4), (5 XOR 4 = 1), (5 XOR 3 = 6), (2 XOR 1 = 3), (2 XOR 4 = 6), (2 XOR 3 = 1), (1 XOR 4 = 5), (1 XOR 3 = 2), (4 XOR 3 = 7).

– Số may mắn lớn nhất là 7, tương ứng với đoạn [1, 2] hoặc [4, 5].

Câu 2:

Bạn được cho 2 số nguyên dương n và k, cùng với 1 dãy số nguyên dương $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$. Với mỗi đoạn con độ dài k của mảng, hãy tính độ dài lớn nhất của một greedy subsequence trong đoạn con đó. Một greedy subsequence là dãy thỏa phần tử sau luôn lớn hơn phần tử trước đó và các phần tử phải liên tiếp trong dãy con k phần tử.

Input:

Dòng đầu tiên chứa 2 số nguyên dương n và k $(1 \leq n, k \leq 10^5)$.

Dòng thứ hai chứa n số nguyên dương $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^9)$.

Output:

In ra $n - k + 1$ số nguyên, mỗi số là độ dài lớn nhất của một greedy subsequence trong đoạn con độ dài k của mảng.

Số thứ nhất ứng với đoạn con từ a_1 đến a_k, số thứ hai ứng với đoạn con từ a_2 đến a_{k+1}, ...

Input	Output
6 4 1 5 2 5 3 6	2 2 3

Giải thích:

– Đoạn con đầu tiên là [1, 5, 2, 5], độ dài lớn nhất của greedy subsequence là 2 (có thể là [1, 5] hoặc [2, 5]).

– Đoạn con thứ hai là [5, 2, 5, 3], độ dài lớn nhất của greedy subsequence là 2 (có thể là [5, 5] hoặc [2, 3]).

– Đoạn con thứ ba là [2, 5, 3, 6], độ dài lớn nhất của greedy subsequence là 3 (có thể là [2, 5, 6] hoặc [3, 5, 6]).

2. Hàng đợi (Queue)

Câu 1:

Cho n là số nguyên dương và 2 mảng a, b có n phần tử là hoán vị của các số từ 1 đến n. Bạn sẽ thực hiện 1 trong các thao tác sau:
+ Nếu phần tử đầu tiên của mảng a và b bằng nhau, bạn sẽ loại bỏ phần tử đầu tiên của cả hai mảng.
+ Nếu không bằng nhau thì ta di chuyển phần tử đầu tiên của mảng a vào cuối mảng a.
Mỗi thao tác sẽ tốn 1 giây. Hãy tính thời gian tối thiểu để làm rỗng cả hai mảng.
Input:
Dòng đầu tiên chứa số nguyên dương n ($1 \leq n \leq 1000$).
Dòng thứ hai chứa n số nguyên dương $a_1, a_2, \ldots, a_n$ là hoán vị của các số từ 1 đến n.
Dòng thứ ba chứa n số nguyên dương $b_1, b_2, \ldots, b_n$ là hoán vị của các số từ 1 đến n.
Output:
In ra số nguyên duy nhất là thời gian tối thiểu để làm rỗng cả hai mảng.

Input	Output
3 1 3 2 2 3 1	6

Giải thích:
– Các thao tác sẽ là: + Di chuyển phần tử đầu sang cuối (1 giây). Mảng a: [3, 2, 1], b: [2, 3, 1].
+ Di chuyển phần tử đầu sang cuối (1 giây). Mảng a: [2, 1, 3], b: [2, 3, 1].
+ Vì có 2 phần tử đầu bằng nhau nên ta thưcj hiện xóa phần tử đầu của 2 mảngmảng (1 giây). Mảng a: [1, 3], b: [3, 1].
+ Di chuyển phần tử đầu sang cuối (1 giây). Mảng a: [3, 1], b: [3, 1].
+ Xóa phần tử đầu (1 giây). Mảng a: [1], b: [1].
+ Xóa phần tử đầu (1 giây). Mảng a: [], b: [].
– Tổng thời gian là 6 giây.

IV. Chia sẻ kinh nghiệm

HẾT