Capítulo 1

Estructuras de Datos

1.1. Pilas y Colas

Se da por conocido el concepto de pilas y colas. En este texto, se usará notación de programación orientada objetos. En concreto, dado un objeto o que contiene un método m, se lanza la ejecución del método m con la sentencia $o.m()$. Dada una pila p que contiene elementos de tipo T se asume la existencia de los siguientes métodos:

- void push(T arg): se insertar el elemento arg en la pila.
- T top(): devuelve el elemento que está en la parte superior de la pila.
- void pop(): elimina el elemento que está en la parte superior de la pila.

Dada una cola q se asume la existencia de los siguientes métodos:

- void push(arg): se insertar el elemento arg al final de la cola.
- T front(): devuelve el elemento que está en la primera posición de la cola.
- void pop(): elimina el elemento que está al frente de la cola.

1.1.1. Pila con Devolución de Mínimo

En ocasiones nos puede interesar desarrollar una pila que además de los métodos anteriores, incluya un método *min* que devuelva el mínimo valor presente actualmente en la pila. Para hacer esto de forma eficiente, cada posición de la pila puede mantener parejas de números (valor, minimo), en las que el primer término es el valor insertado y el segundo termino contiene el mínimo valor desde dicha posición hasta el elemento más profundo de la pila. Así, si se insertan los valores 5, 3, 8 y 4, la pila iría evolucionando como se aprecia en la Figura 1.1. Para obtener el mínimo en cualquier instante, basta con devolver el segundo valor de la pareja presente en la cima de la pila.

Un posible código en c++ es el siguiente:

```
class StackMin {
        public:
```

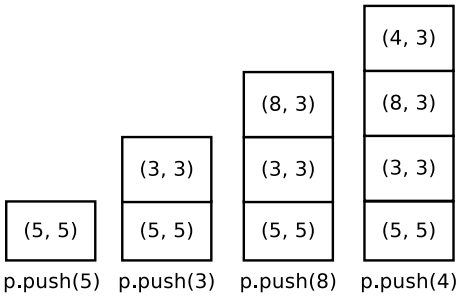

FIGURA 1.1: Pila con mantenimiento del mínimo valor

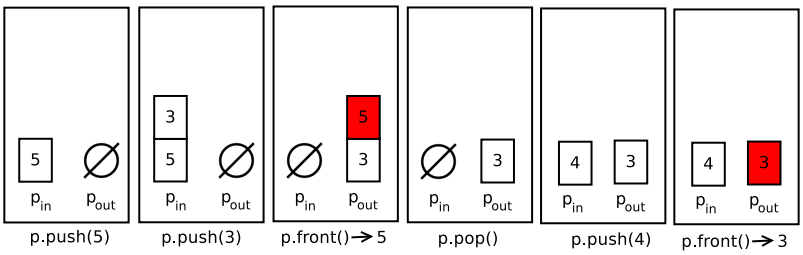

FIGURA 1.2: Generación de cola a partir de dos pilas

```
        StackMin(){ st.push(make_pair(INT_MAX, INT_MAX)); }
        void push(int v){ st.push(make_pair(v, min(v, st.top().second))); }
        int top(){ return st.top().first; }
        void pop(){ st.pop(); }
        int minV(){ return st.top().second; }
        int size(){ return st.size() - 1; }
        bool empty() { return size() == 0; }
    private:
        stack< pair<int,int> > st;
};
```

1.1.2. Creación de Cola con Dos Pilas

A partir del tipo pila, podemos generar una cola. En concreto necesitamos dos pilas: p_{in} y p_{out}. Al insertar nuevos elementos, lo hacemos en p_{in}. Para retirar elementos, checamos si p_{out} está vacío. Si es así, vamos sacando los elementos de p_{in} uno por uno, e insertándolos en p_{out}. Nótese que esto tiene el efecto de invertir los elementos, es decir, los elementos más antiguos quedarán en posiciones superiores de la pila p_{out}, por lo que al retirar ahora un elemento de p_{out}, será el elemento más antiguo, tal y como se espera en una cola. Cuando intentemos retirar elementos y p_{out} no esté vacío, directamente retiramos de p_{out}. La Fig 1.2 ilustra el compartamiento de la estructura ante diferentes operaciones.

Un posible código en c++ es el siguiente:

```
class QueueWithStacks{
    public:
        void push(int v){ p_in.push(v); }
        int front(){ transfer(); return p_out.top(); }
        void pop(){ transfer(); p_out.pop(); }
        int size() { return p_in.size() + p_out.size(); }
        bool empty() { return size() == 0; }

    private:
        void transfer(){
            if (p_out.empty()){
                while(!p_in.empty()){
                    p_out.push(p_in.top());
                    p_in.pop();
                }
            }
        }
        stack<int> p_in;
        stack<int> p_out;
};
```

1.1.3. Cola con Devolución de Mínimos

Si pensamos de forma directa en desarrollar una cola capaz de devolver el mínimo valor actual, parece más problemático que en el caso de la pila, debido a que al salir un valor, la cola pasa a estar en un estado diferente por el que no había pasado nunca, al contrario de lo que pasaba en la pila que volvía al estado previo a haber insertado dicho valor. En cualquier caso, esto se puede resolver de varias formas, y una de ellas consiste en unir los dos conceptos visto en las secciones anteriores. De esta forma, si en las

pilas p_{in} y p_{out} almacenamos el mínimo, ya podríamos también devolver el mínimo de la cola, siendo este el mínimo valor entre el contenido en p_{in} y p_{out}. Un posible código es el siguiente:

```
class QueueMin {
        public:
                void push(int v){ p_in.push(v); }
                int front(){ transfer(); return p_out.top(); }
                void pop(){ transfer(); p_out.pop(); }
                int size() { return p_in.size() + p_out.size(); }
                int minV() { return min(p_in.minV(), p_out.minV()); }
                bool empty() { return size() == 0; }

        private:
                void transfer(){
                        if (p_out.empty()){
                                while(!p_in.empty()){
                                        p_out.push(p_in.top());
                                        p_in.pop();
                                }
                        }
                }
                StackMin p_in;
                StackMin p_out;
};
```

1.1.4. Problemas

- Dado un arreglo A de tamaño N, y un valor $M \leq N$, debemos calcular la suma del valor mínimo de todos los subarreglos de A que tienen tamaño exactamente M con complejidad O(N) . Juez: `http://www.hackerrank.com/`.