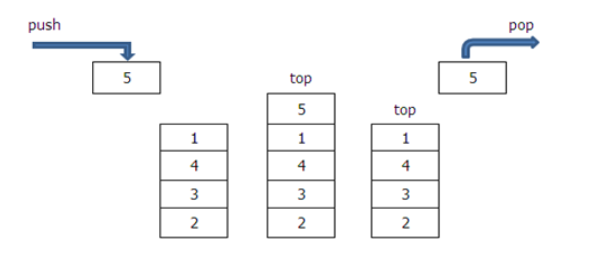
**STACK & BÀI TOÁN TÌM MAX, MIN TRÊN ĐOẠN TỊNH TIẾN**

1. **STACK**
2. **KHÁI NIỆM**

* Stack (Ngăn xếp) là một cấu trúc dữ liệu tuyến tính tuân theo nguyên tắc Last in first out (LIFO) . Điều này có nghĩa là phần tử cuối cùng được chèn vào bên trong ngăn xếp sẽ bị loại bỏ đầu tiên.
* Trong Stack, các đối tượng được nạp vào(push) và lấy ra (pop) chỉ từ 1 đầu được gọi là đỉnh (top) của ngăn xếp.



1. **CÁCH HOẠT ĐỘNG CỦA CẤU TRÚC DỮ LIỆU NGĂN XẾP**



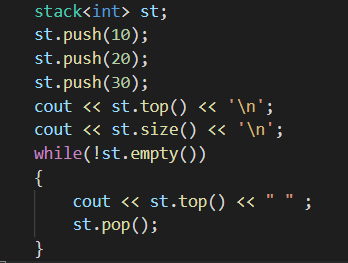
1. Một con trỏ được gọi là TOP được sử dụng để theo dõi phần tử trên cùng trong ngăn xếp.
2. Khi khởi tạo ngăn xếp, chúng ta đặt giá trị của nó thành -1 để có thể kiểm tra xem ngăn xếp có trống không bằng cách so sánh TOP == -1.
3. Khi push một phần tử, chúng ta tăng giá trị của TOP và đặt phần tử mới vào vị trí được trỏ tới TOP.
4. Khi pop một phần tử, chúng ta return phần tử được trỏ đến bởi TOP và giảm giá trị của nó.
5. Trước khi push, kiểm tra xem ngăn xếp đã đầy chưa. nếu ngăn xếp là đầy, tiến trình bị lỗi và thoát ra.
6. Trước khi pop, kiểm tra xem ngăn xếp đã trống chưa. Nếu ngăn xếp là trống, tiến trình bị lỗi và thoát ra.
7. **CÁC THAO TÁC TRÊN STACK**
8. **Các hàm cơ bản**

* **push()**: Thêm một phần tử vào đỉnh của ngăn xếp, số phần tử của ngăn xếp tăng lên 1. Nếu ngăn sắp xếp đã đầy thì nó được cho là điều kiện tràn
* **pop()**: Xóa bỏ phần tử đầu tiên ở đỉnh của ngăn xếp, số phần tử của ngăn xếp giảm đi 1.
* **top()**: Trả về phần tử trên cùng của ngăn xếp.
* **empty()**: Kiểm tra ngăn xếp trống hay không.

1. **Cài đặt ngăn xếp**

Để sử dụng Stack, chúng ta có thể sử dụng các thư viện hỗ trợ có sẵn của C++(#include<stack>) hoặc bằng array, struct

* Cú pháp stack<kiểu dữ liệu> (Tên stack);



* Cài đặt stack sử dụng mảng

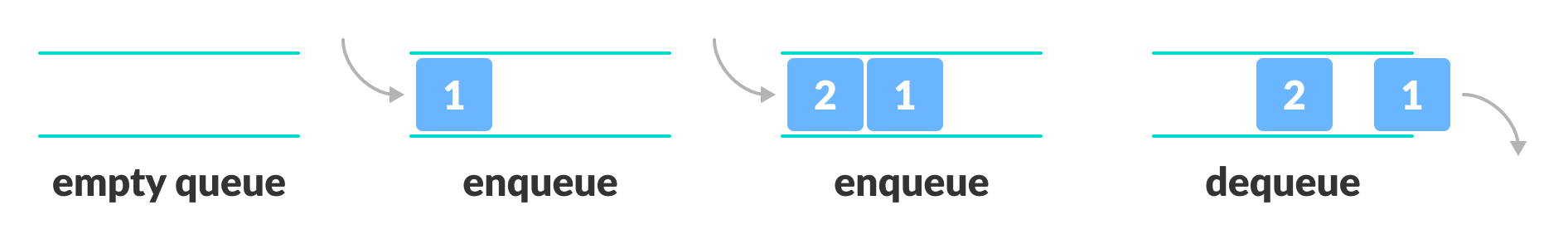
[**https://ideone.com/7rLRe1**](https://ideone.com/7rLRe1)

1. **Độ phức tạp của các hàm trong Stack**

* push(), pop(), empty() và top() đều mất O(1) thời gian. Chúng ta không chạy bất kỳ vòng lặp nào trong bất kỳ hàm nào trong số này.

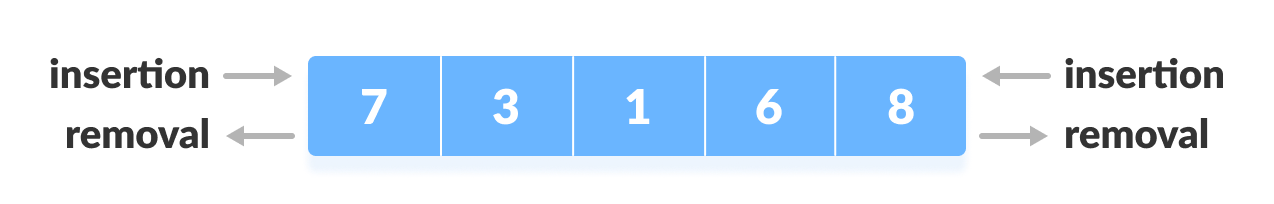
1. **BÀI TOÁN TÌM MAX, MIN TRÊN ĐOẠN TỊNH TIẾN**
2. **DEQUE (Hàng đợi 2 đầu)**
3. **Queue là gì**

* Tương tự như Stack, Queue (hàng đợi) cũng là một cấu trúc dữ liệu. Queue tuân theo quy tắc First in first out(nhập trước xuất trước) - FIFO:



1. **Deque**

- Deque hoặc Double Ended Queue là một loại hàng đợi trong đó việc chèn và loại bỏ các phần tử có thể được thực hiện từ phía trước hoặc phía sau. Do đó, nó không tuân theo quy tắc FIFO (First In First Out)



1. **Các thao tác trên Deque**

- push\_front(x): Đẩy x vào đầu deque

- push\_back(x): Đẩy x vào cuối deque

- pop\_front(): Loại bỏ phần tử ở đầu deque

- pop\_back(): Loại bỏ phần tử ở cuối deque

- empty(): Kiểm tra Deque có rỗng không

- size(): Trả về số phần tử đang có trong deque

**4. Cài đặt**

- Để sử dụng Deque, chúng ta có thể sử dụng các thư viện hỗ trợ có sẵn của C++ (#include<deque> )

- Cú pháp

deque <object\_type> deque\_name;

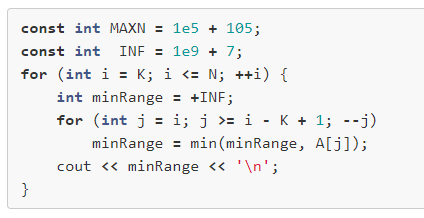
object\_type: kiểu dữ liệu (int, float, struct, …)

deque\_name: tên biến deque

1. **BÀI TOÁN**

Cho một dãy A gồm N phần tử được đánh số từ 1 đến N. Phần tử thứ i có giá trị là A[i]. Cho k là một số nguyên dương (k≤N). Với mỗi phần tử i (k≤i≤N), tìm giá trị nhỏ nhất của các phần tử trong đoạn từ i–k+1 đến i trên dãy A. minRange[i]= giá trị nhỏ nhất trong đoạn [i−k+1…i]

1. **Cách 1:** Duyệt tất cả các đoạn gồm k phần tử liên tiếp trong mảng A để tìm giá trị nhỏ nhất.

****

**Độ phức tạp**: O(k⋅(N−k))

1. **Cách 2**

* **Ý tưởng:** Mỗi lần gán minRange=min(minRange,A[j]) thì mấu chốt là những vị trí mà minRange thay đổi. Vậy nên ta sẽ chỉ lưu lại những vị trí có thể làm thay đổi minRange. Ta thấy rằng các vị trí lưu vào tăng dần về cả giá trị cũng như vị trí.
* Khởi tạo một hàng đợi hai đầu nhằm mục đích lưu vị trí của phần tử có giá trị nhỏ nhất. Ở mọi thời điểm, giá trị A[i] trong deque luôn tăng nghiêm ngặt nên phần tử đầu tiên luôn là phần tử nhỏ nhất. N
* **Thuật toán**
* **B1:** Tạo 1 deque để lưu trữ index của các phần tử.
* **B2:** Chạy một vòng lặp duyệt qua từng phần tử. A[i] là phần tử đang xét. Trong khi hàng đợi khác rỗng và A[dq.back()] >= A[i] thì ta sẽ đẩy hết ra khỏi hàng đợi. Sau đó chèn i vào cuối deque
* **B3:** Nếu vị trị của phần tử đầu tiên trong hàng đợi không nằm trong khoảng đoạn xét thì ta loại bỏ nó.
* **B4:** Đáp án cần tìm chính là giá trị của phần tử đầu tiên trong deque.

***\*Ví dụ:***  N = 8, K = 4, A[] = {8,5,10,7,8,4,15,3}

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A[i] = | 8 | 5 | 10 | 7 | 8 | 4 | 15 | 3 |
| i = | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Đẩy i=1 vào deque, A[dq.back()] = 8

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=1 A[i] = 8 |  |  |  |  |

* i = 2, A[i] = 3 < A[dq.back()] 🡺 loại bỏ A[dq.back()]

🡺Đẩy i = 2 vào queue, A[dq.back()] =A[i] = 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=2 A[i] = 5 |  |  |  |  |

* A[3]> A[dq.back()]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=2 A[i] = 5 | i=3 A[i] = 10 |  |  |  |

* A[4] < A[dq.back()], i=4 >= k

🡺 minRange([1,4]) = A[dq.front()] = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=2 A[i] = 5 | i=4 A[i] = 7 |  |  |  |

* i=5 < A[dq.back()], i=5 >= k🡺 minRange([2,5])= A[dq.front()] = 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=2 A[i] = 5 | i=4 A[i] = 7 | i=5 A[i] = 8 |  |  |

* i = 6 >= dq.front()+k 🡺 loại A[2] ra khỏi deque

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=4 A[i] = 7 | i=5 A[i] = 8 |  |  |  |

* i=6, A[i] = 4 > A[5],A[4] 🡺 Loại bỏ hết ra khỏi queue, đẩy i=6 vào dq

i = 6 >=k 🡺 minRange ([3,6])= A[dq.front()] = 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=6 A[i] = 4 |  |  |  |  |

* i=7, A[i] = 15 > 4 🡺 đẩy i=7 vào dq

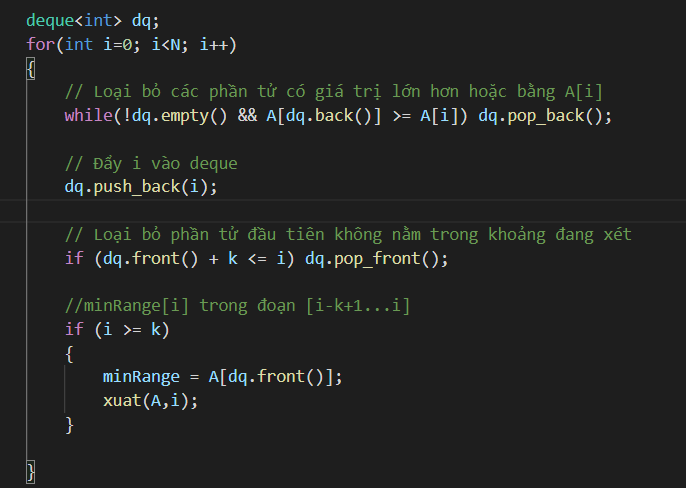
i=7>=k 🡺minRange([4,7]) = A[dq.front()] = 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=6 A[i] = 4 | i=7 A[i] = 15 |  |  |  |

* i = 8, A[i] = 4 <A[6],A[7] 🡺Loại bỏ hết khỏi queue 🡺Đẩy i=8 vào dq
* i=8 >=k => minRange([5,8]) = A[dq.front()] = 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| dq | i=8 A[i] = 3 |  |  |  |  |

***\*CÀI ĐẶT:***



1. **BÀI TẬP**
2. [**https://www.spoj.com/PTIT/problems/BCSTACK/**](https://www.spoj.com/PTIT/problems/BCSTACK/)

**\*Code:** [**https://ideone.com/HfNNMQ**](https://ideone.com/HfNNMQ)

1. [**https://vn.spoj.com/problems/KPLANK/**](https://vn.spoj.com/problems/KPLANK/)

**\*CODE:** [**https://ideone.com/XRn3RP**](https://ideone.com/XRn3RP)