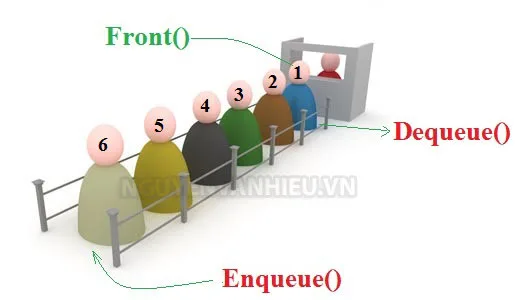
Queue-Deque-Priority Queue

1. Queue.
2. Khái niệm: H**àng đợi** là một cấu trúc dữ liệu biểu diễn một danh sách các phần tử đứng trong "hàng chờ" được xử lý.
3. Nguyên tắc hoạt động: FIFO(First In First Out) :Bổ sung phần tử sẽ thực hiên ở cuối danh sách. Loại bỏ phần tử thực hiện ở đầu danh sách (Tương tự 1 hàng người đợi mua vé, ai vào trước thì có vé trước,ai ở sau sẽ phải chờ.



1. Khai báo: queue< kiểu dữ liệu > q;
2. Các thao tác:

-Tên\_queue.front():Trả về phần tử đầu tiên của hàng đợi .O(1)

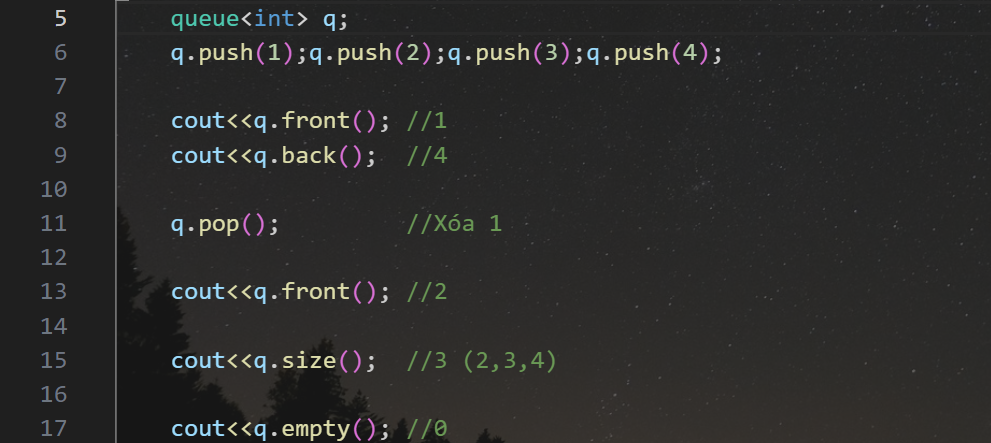
-Tên\_queue.back():Trả về phần tử đầu tiên của hàng đợi .O(1)

-Tên\_queue.push():Thêm phần tử vào cuối hàng đợi .O(1)

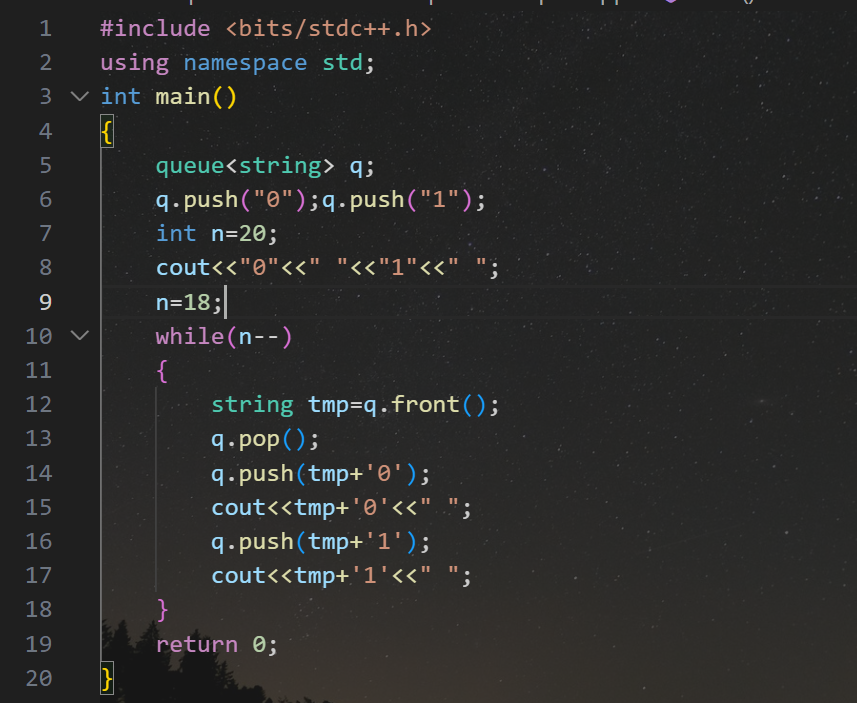
-Tên\_queue.pop():Xóa phần tử Front() của hàng đợi .O(1)

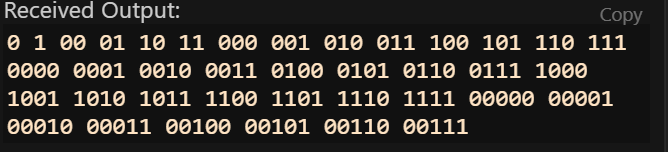
-Tên\_queue.size():Trả về số lượng phần tử trong hàng đợi .O(1)

-Tên\_queue.empty():Kiểm tra xem hàng đợi có rỗng hay ko.O(1)



VD:In ra N số nhị phân đầu tiên:





1. Deque.

Khác với hàng đợi là nó có thể thực hiện thao tác Push-Pop ở 2 đầu.

Các thao tác:

****push\_front****: Thêm phần tử vào đầu deque

****push\_back****: Thêm phần tử vào cuối deque

****pop\_front****: Loại bỏ phần tử ở đầu deque

****pop\_back****: Loại bỏ phần tử ở cuối deque

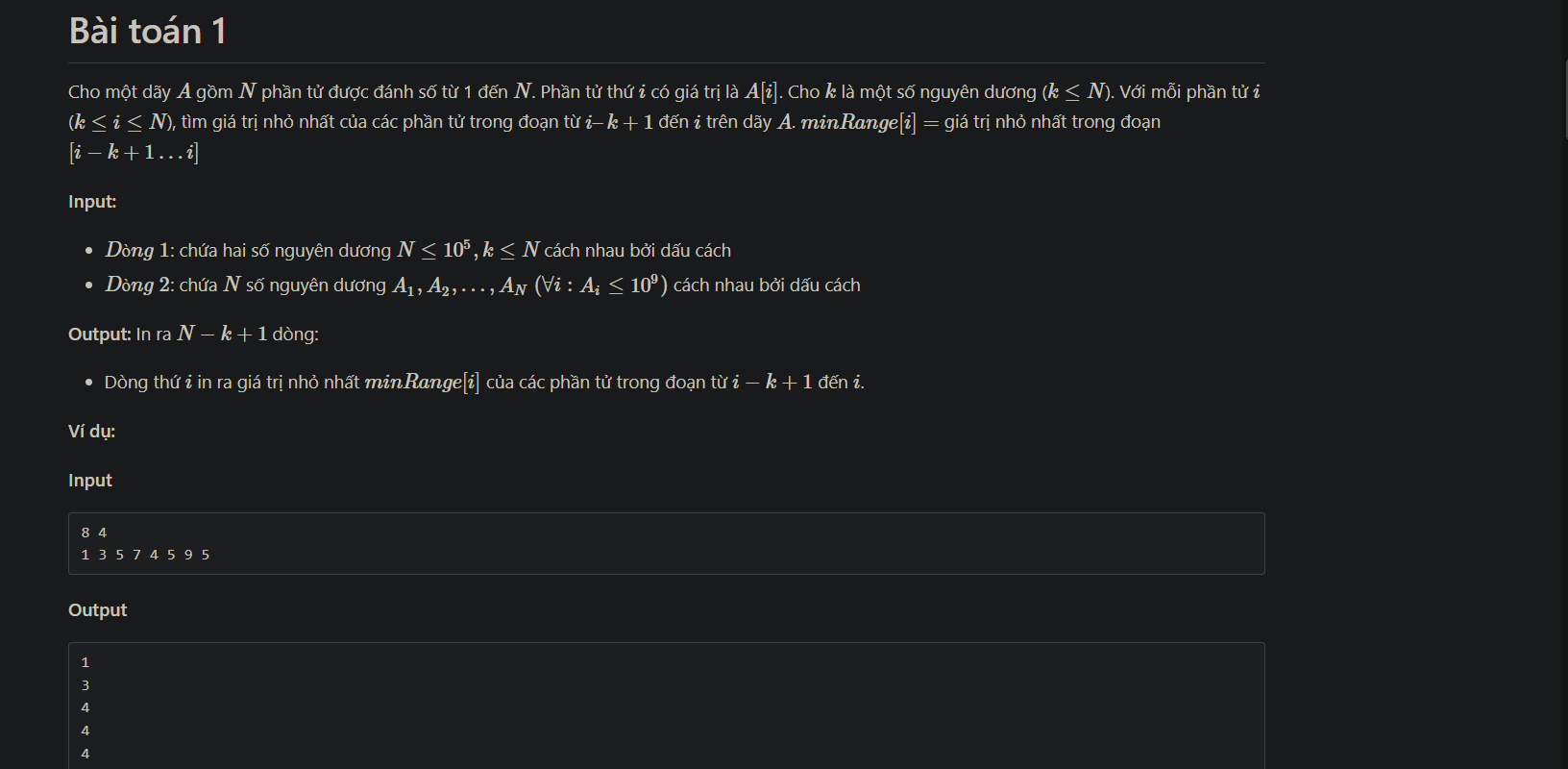
****front****: Trả về giá trị là phần tử đầu trong deque

****back****: Trả về giá trị là phần tử cuối trong deque

****size****: Trả về giá trị nguyên là kích thước của deque.



1. Tìm Min Max trên đoạn tịnh tiến.



C1-chạy cửa sổ tượt :Khá tốn time( O(k\*(n-k)))

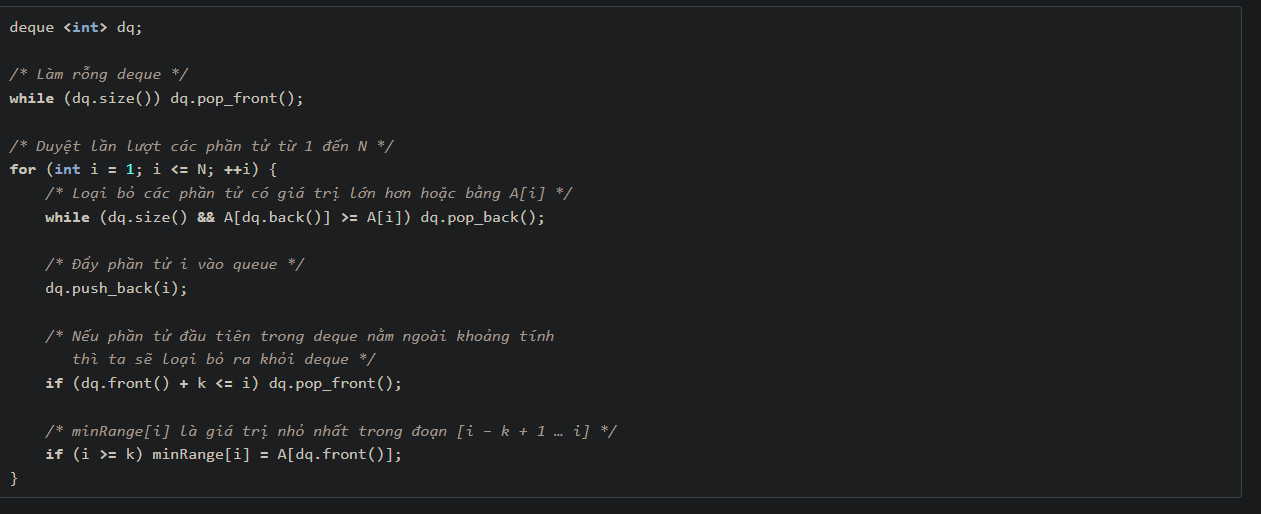
C2-Sử dụng Deque:

-Ý tưởng :luôn ưu tiên dãy trong deque là dãy tăng đơn điệu.

+ Nếu ptu đang xét < phần tử cuối deque ==> Nó ko cần nữa (Vì đang xét pt min mà)=> pop\_back

+Nếu ptu đang xét >= pt cuối deque => Cho nó vào

=> Dãy trogn deque luôn là tăng dần =>Ptu đầu là nhỏ nhất .Lúc này chỉ cần xét xem phần tủ đầu có thuộc dãy: I-k+1 ->I hay ko thôi,có thì lấy,ko thì pop();



1. Hàng đợi ưu tiên.

-Là cấu trúc dữ liệu giúp ta lưu các phần tử vào 1 danh sách mà nó luôn được sắp xếp.

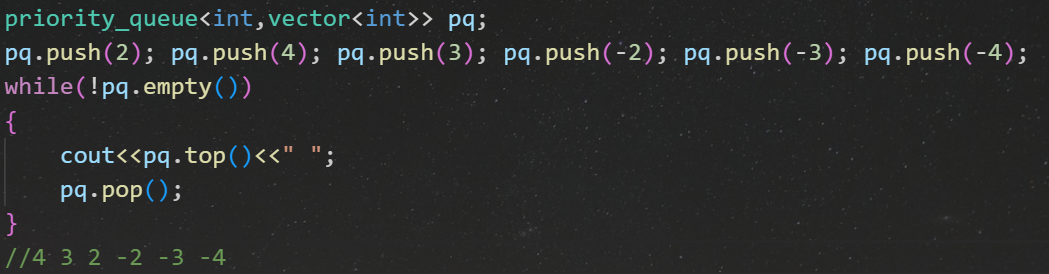
-Cách hoạt động giống queue,chỉ khác là nó sẽ đẩy các phần tử có độ ưu tiên cao hơn lên trên.

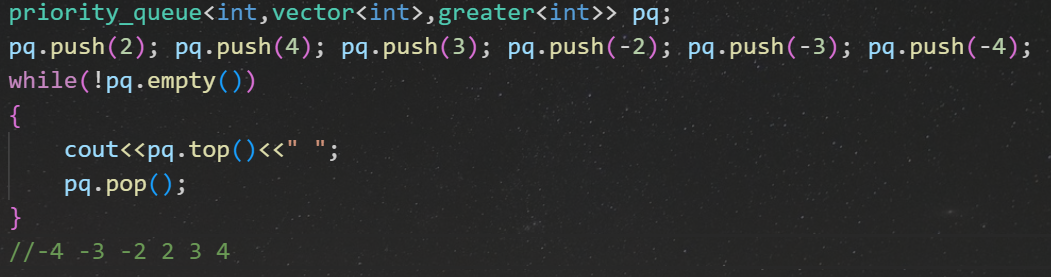
1. Khai báo :

-priority < datatype , vector<datatype> ,cmp> pq;

-priority < datatype , vector<datatype>> pq: pq sẽ ưu tiên phần tử lớn hơn ở trên cùng

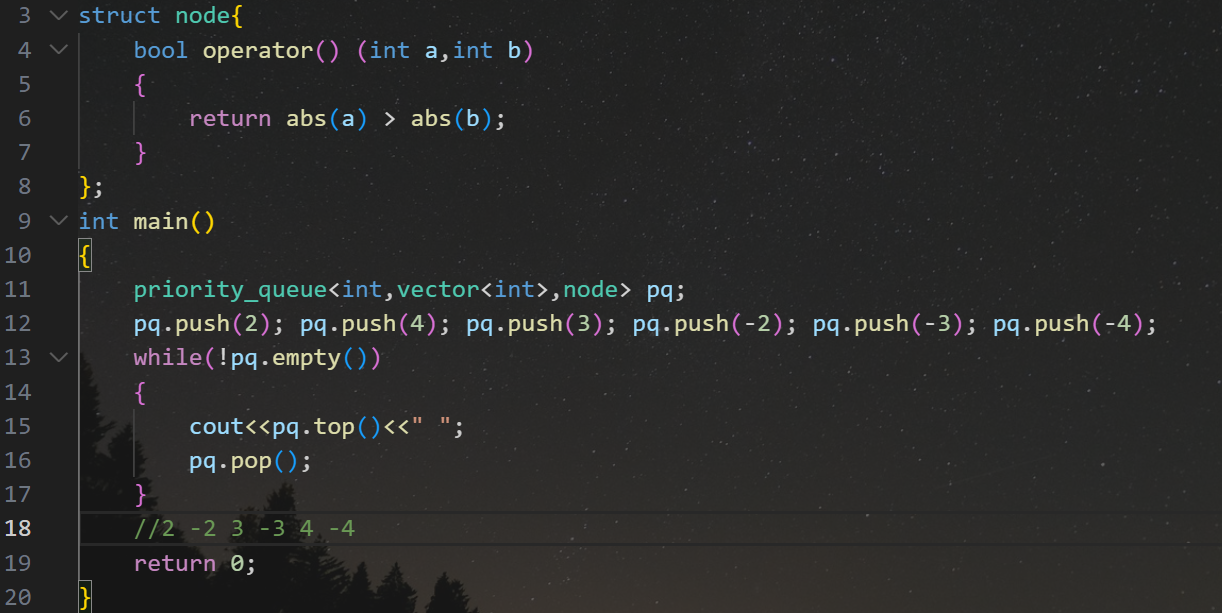
-priority < datatype , vector<datatype> ,greater> pq: pq sẽ ưu tiên phần tử nhỏ hơn trên cùng





-Nếu muốn thay đổi cmp theo ý mình muốn thì:

+Tạo 1 struct có 1 toán tử operater trả về đúng-sai . => Dùng để sắp xếp trong 1 danh sách



2.Thao tác:

Pq.push() : Thêm 1 phần tử vào pq

Pq.pop() : Xóa phần tử đầu pq

Pq.top(): Trả về phần tử đầu pq

Pq.size(): Trả về số lượng phần tử của pq

3.Ví dụ :

-Cho 1 xâu ký tự S. Gọi Giá trị của xâu là tổng bình phương số lần xuất hiện các kí tự khác nhau trong xâu.

-Cho số K,hỏi khi xóa đi K kí tự, giá trị lớn nhất của xâu S là bao nhiêu?

VD: AAABBC ,K=3 => AABBC =>ABBC ==> ABC

-Ý tưởng:

+ Duyệt qua mảng s,sử dụng map để lưu số lần xuất hiện của các kí tự khác nhau trong s.

+ Sử dụng 1 priority queue để lưu số lần xuất hiện.

-Chạy for(1->K) .

+Tạo tmp=pq.top(); tmp--; pq.push(tmp);

//Cứ mỗi lần ta lại xóa 1 ký tự có nhiều lần xuất hiện nhất ở xâu .

=>Duyệt qua pq và tính kết quả.