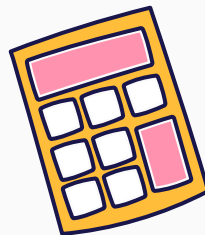


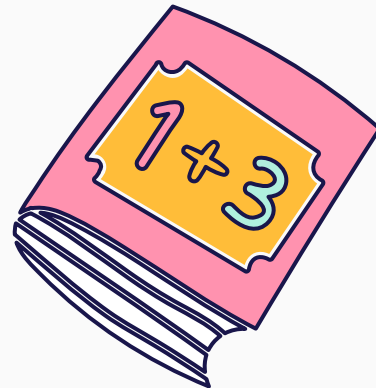
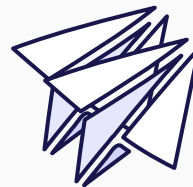
SEGMENT TREE

Chào mừng đến với bài thuyết trình nhóm 12

$$2+2$$



GROUP 12



GROUP 12

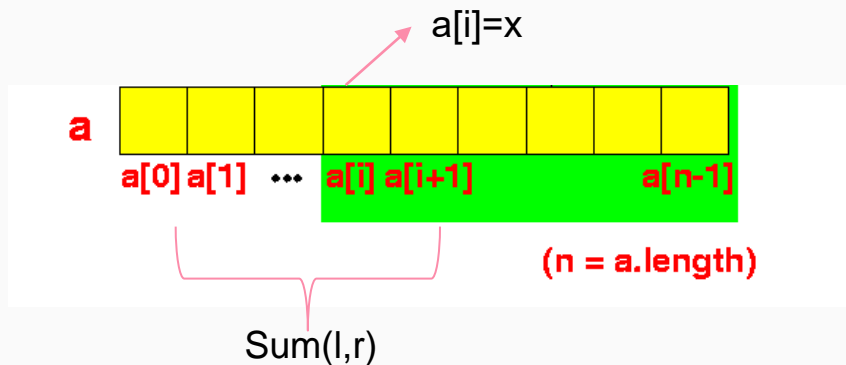
- ❑ Lữ Thị Thúy Quỳnh - 20521826(Nhóm trưởng)
- ❑ Trần Duy Thanh - 20521925
- ❑ Trần Văn Long - 20521576

PROBLEM

Bài toán có một mảng a có n phần tử

Yêu cầu 1: Tìm tổng các phần tử từ vị trí l đến r trong đó $0 \leq l \leq r \leq n-1$

Yêu cầu 2: Thay đổi giá trị của một phần tử được chỉ định thành một giá trị mới x : $a[i] = x$ trong đó $0 \leq i \leq n-1$.



PROBLEM

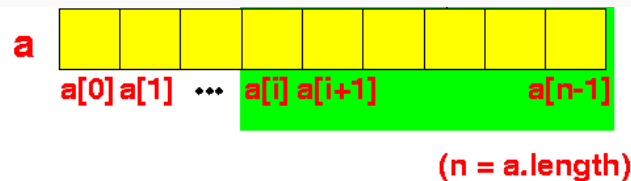
Cách 1: Dùng vòng lặp duyệt qua từng phần tử

Truy vấn tổng: $O(n)$, cập nhật: $O(1)$

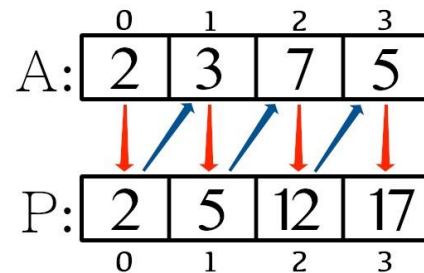
Cách 2: Tạo một mảng khác và lưu trữ tổng từ đầu đến i trong mảng này.

Truy vấn tổng: $O(1)$, cập nhật: $O(n)$

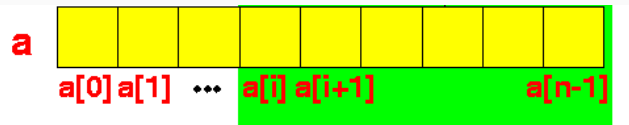
→ Hoạt động tốt khi mà truy vấn nhiều và không cần cập nhật giá trị nhiều



Prefix Sum
P of A



PROBLEM



(n = a.length)

Nếu chúng ta phải thực hiện truy vấn và update nhiều như nhau ?

→ Segment tree là cấu trúc dữ liệu giúp chúng ta thực hiện cả truy vấn và update trong thời gian $O(\log(n))$


$$1 + 3$$

$$2 + 2$$



TABLE OF CONTENTS

01

Giới thiệu

03

Ứng dụng

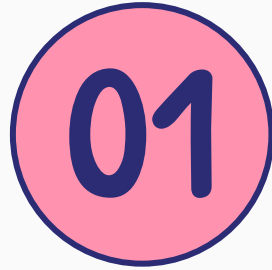

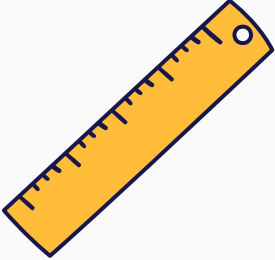
02

Xây dựng Segment tree
& truy vấn và cập nhật

04

Quiz





01

GIỚI THIỆU





$2+2$

2

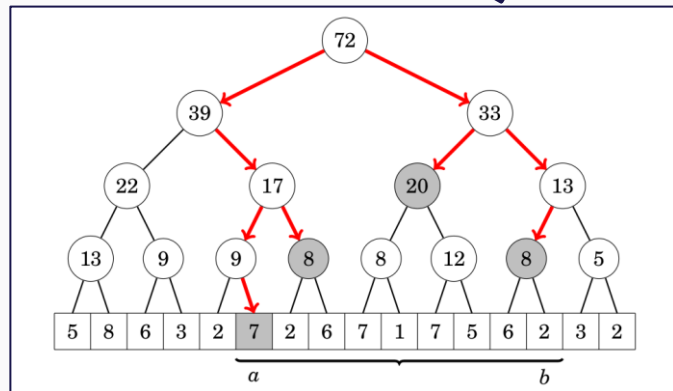
ĐỊNH NGHĨA

Cây phân đoạn là một cấu trúc dữ liệu giúp thực hiện các truy vấn trong một phạm vi của một mảng một cách hiệu quả, trong khi vẫn rất linh hoạt để cho phép sửa đổi mảng (với độ phức tạp $O(\log(n))$)

Các phép toán có thể thực hiện trên segment tree.

Vd: sum, min, max, xor, BCNN, UCLN,...

→ Có tính kết hợp: $(a \circ b) \circ c = a \circ (b \circ c)$. (\circ là phép toán)

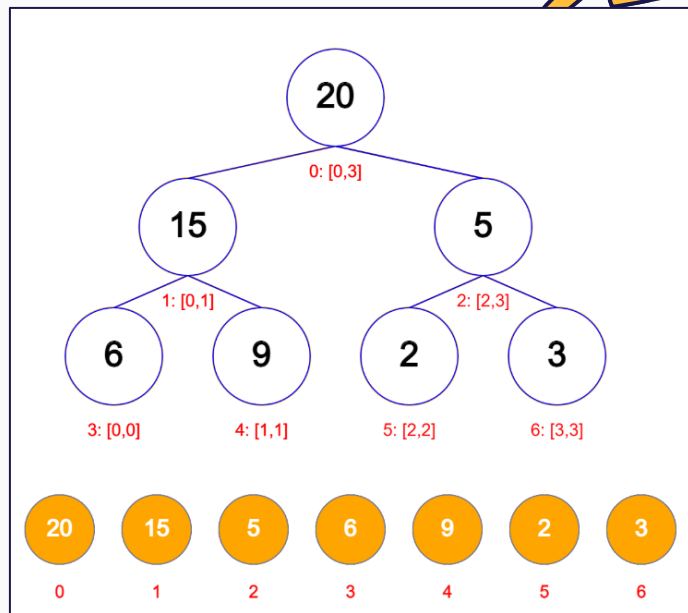


2

CẤU TRÚC SEGMENT TREE

Các nút lá là các phần tử của mảng ban đầu.
Các node cha là một vài kết hợp của các nút lá.

Đối với mỗi nút ở chỉ mục i , nút con bên trái ở chỉ số $2 * i + 1$, nút con bên phải ở $2 * i + 2$ và nút cha ở $\lfloor (j - 1) / 2 \rfloor$. (j là index node con)

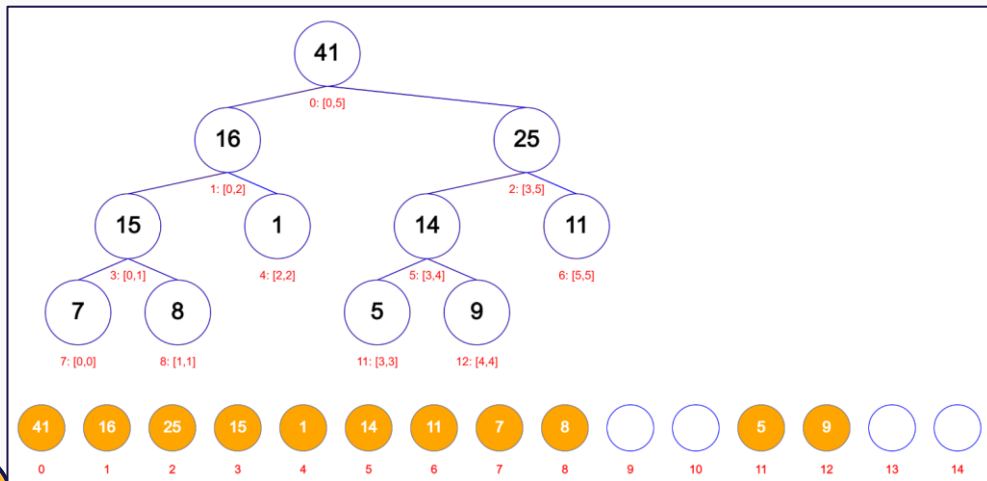


2

Segment tree trong bộ nhớ

Giống như Heap, cây phân đoạn được cài đặt trên mảng. Khác ở đây cây phân đoạn là cây nhị phân đầy đủ

→ có thể tối ưu hóa sự lãng phí này bằng cách sử dụng một số triển khai thông minh, nhưng code cho tính tổng và cập nhật trở nên phức tạp hơn.

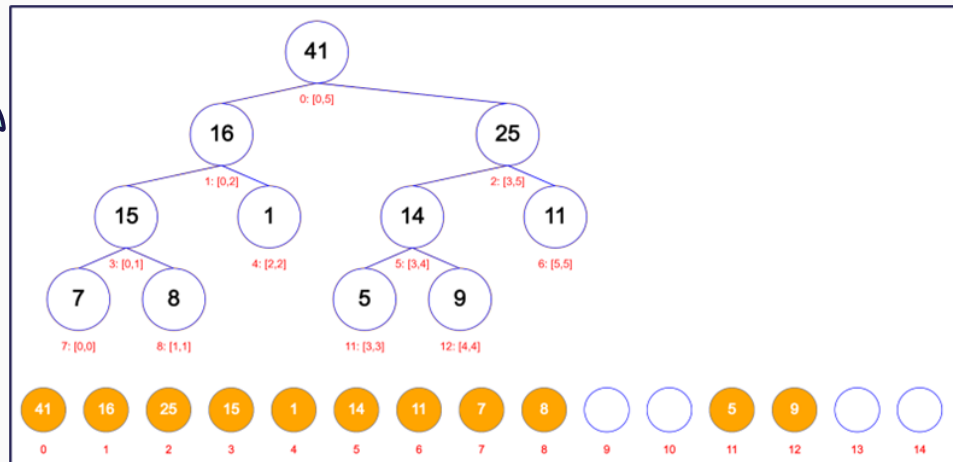


$A = \{7, 8, 1, 5, 9, 11\}$

KÍCH THƯỚC CỦA MẢNG CÀI ĐẶT

Chiều cao của cây là $\log_2 n$.

Kích thước của mảng cài đặt cây phân đoạn là $2 * 2^{\text{height}} - 1$



VD: $n=6$, $\text{height}=3$

Kích thước của mảng cài đặt: $2 * 8 - 1 = 15$

→ Trừ những mảng có số phần tử là lũy thừa của 2 nếu không thì việc lãng phí bộ nhớ luôn xảy ra



02

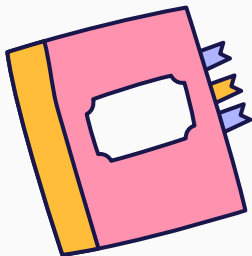
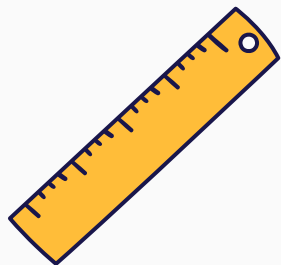
XÂY DỰNG CÂY PHÂN ĐOẠN & Truy vấn và Cập nhật

$$2+2$$



$$2+2$$

XÂY DỰNG CÂY PHÂN ĐOẠN



Ý TƯỞNG BUILD

Bắt đầu với một đoạn $\text{arr}[0 \dots n-1]$. Liên tục chia đoạn hiện tại thành hai nửa (nếu nó chưa trở thành đoạn có độ dài 1), rồi đệ quy trên cả hai nửa, và đối với mỗi đoạn như vậy, chúng ta lưu trữ tổng trong nút tương ứng.

```
def Build(arr, l, r, st, si) :  
    #Độ dài của đoạn bằng 1  
    if (l == r) :  
        st[si] = arr[l];  
        return arr[l];  
    #Nếu không thì tiếp tục chia đôi đoạn đó  
    mid = l + (r - l) // 2  
    st[si] = Build(arr, l, mid, st, si * 2 + 1) + Build(arr, mid + 1, r, st, si * 2 + 2);  
    return st[si];
```

GIẢI THÍCH

A=

1	2	3
0	1	2

st=

6	3	3	1	2		
0	1	2	3	4	5	6

Build(A,0,2,st,0)



Build(A,0,1,st,1)



Build(A,2,2,st,2)



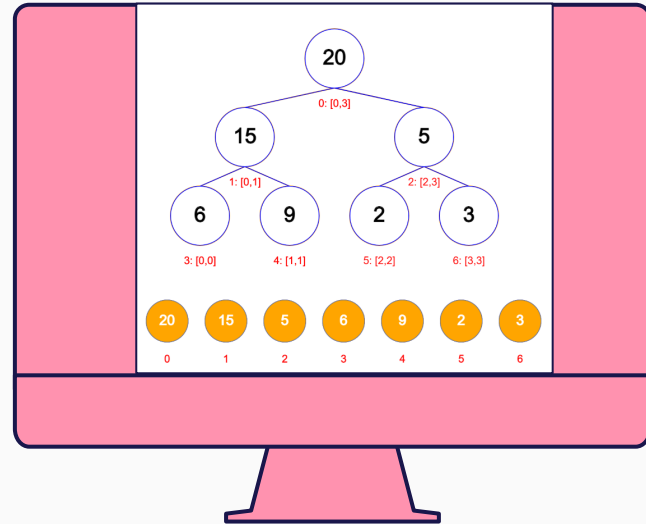
Build(A,0,0,st,3)



Build(A,1,1,st,4)

```
def Build(arr, l, r, st, si) :  
    #Độ dài của đoạn bằng 1  
    if (l == r) :  
        st[si] = arr[l];  
        return arr[l];  
    #Nếu không thì tiếp tục chia đôi đoạn đó  
    mid = l + (r - l) // 2  
    st[si] = Build(arr, l, mid, st, si * 2 + 1)  
    +Build(arr, mid + 1, r, st, si * 2 + 2);  
    return st[si];
```

DEMO BUILD




$$2+2$$


TRUY VẤN CÂY PHÂN ĐOẠN



QUERY

A=

1	2	3
---	---	---

st=

6	3	3	1	2		
0	1	2	3	4	5	6



Tìm được giá trị tổng của đoạn $[0,1]$?

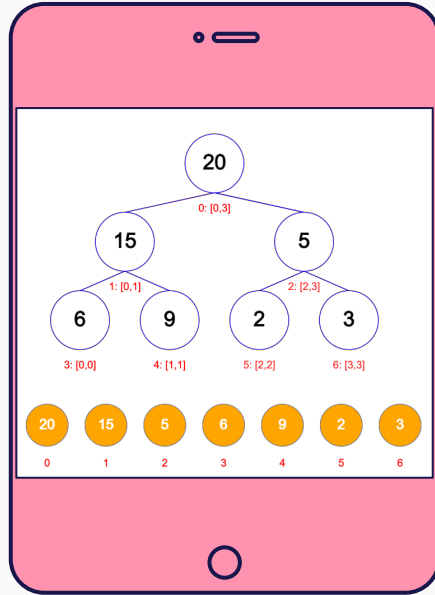
B1: Bắt đầu từ phần tử quản lý đoạn $[0,2]$.

B2: Chia đôi đoạn thành hai nửa $[0,1]$, $[2,2]$

B3: Trả về giá trị tổng của đoạn $[0,1]$

Ý TƯỞNG QUERY

```
def Query_Sum(st, l, r, qs, qe, si) :  
  
    # Nếu đoạn của phần tử nằm trong đoạn cần truy vấn  
    # trả về phần tử quản lý của đoạn đó(tổng cả đoạn đó)  
    if (qs <= l and qe >= r) :  
        return st[si];  
  
    # Nếu đoạn của phần tử nằm ngoài đoạn truy vấn  
    # Thì trả về 0  
    if (r < qs or l > qe) :  
        return 0;  
  
    # Nếu đoạn của phần tử nằm trong đoạn mà có cả đoạn cần truy vấn và không cần truy vấn  
    # Tiếp tục chia đôi đoạn đó để tìm ra phần tử thuộc đoạn truy vấn  
    mid = l + (r - l) // 2  
  
    return Query_Sum(st, l, mid, qs, qe, 2 * si + 1)  
    + Query_Sum(st, mid + 1, r, qs, qe, 2 * si + 2);
```



DEMO QUERY


$$2+2$$


CẬP NHẬT CÂY PHÂN ĐOẠN



Ý TƯỞNG UPDATE

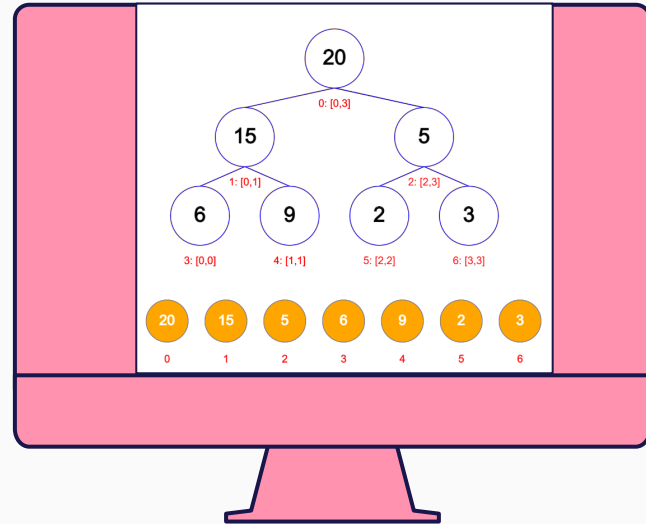
Input: index_update, update_value

Output: mảng ban đầu và mảng lưu cây phân đoạn đã được cập nhật

- Cập nhật giá trị của phần tử index_update trong cây phân đoạn và trong mảng.
- Cập nhật lại các phần tử liên quan trong đoạn có phần tử vừa update. Bằng cách cộng dồn độ lớn chênh lệch giữa giá trị gốc và giá trị update.

```
def updateValueUtil(st, l, r, index, diff, si) :  
  
    # Nếu vị trí muốn update nằm ngoài đoạn của cây phân đoạn  
    if (index < l or index > r) :  
        return;  
    # Tính độ lớn chênh lệch giữa giá trị cũ và giá trị update  
    diff = new_val - arr[i];  
  
    # Update lại giá trị của phần tử có index đó  
    arr[i] = new_val;  
    # Nếu vị trí cần update nằm trong đoạn này thì  
    # Update lại hết các phần tử nằm trong đoạn đó  
    st[si] = st[si] + diff;  
  
    if (se != ss) :  
  
        mid = l + (r - l) // 2;  
        updateValueUtil(st, l, mid, index,  
                        diff, 2 * si + 1);  
        updateValueUtil(st, mid + 1, r, index,  
                        diff, 2 * si + 2);
```

DEMO UPDATE



ỨNG DỤNG



ỨNG DỤNG

Sử dụng phổ biến trong các bài toán xử lý trên dãy số:

Tìm tổng trên đoạn / tìm max/ min
trên đoạn v.v.





ỨNG DỤNG

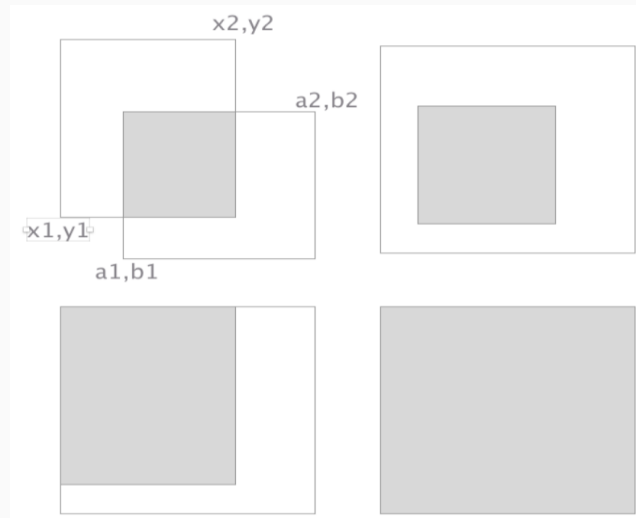
Trong hình học

-Liệt kê tất cả các cặp hình chữ nhật giao nhau.

-Báo cáo:

+Danh sách tất cả các đoạn thẳng tuyến tính trong mặt phẳng.

+Chu vi của một tập các hình chữ nhật trong mp.

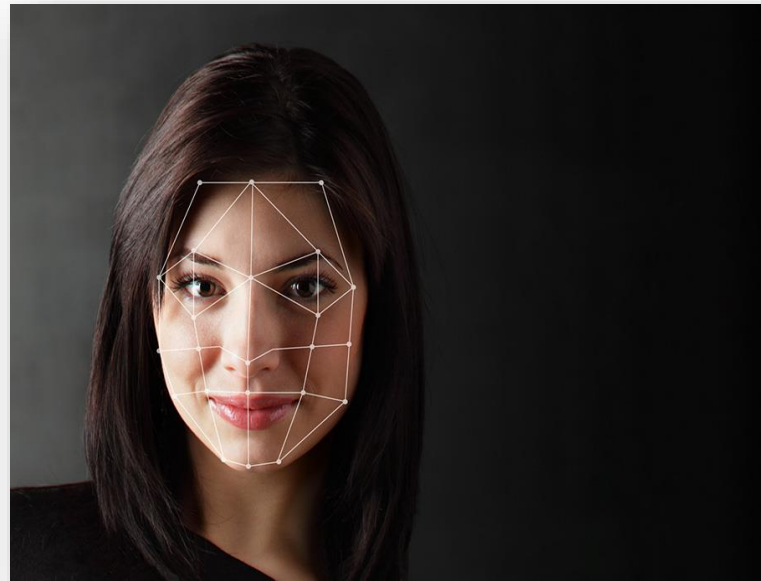


ỨNG DỤNG



Trong Khoa học máy tính

Ứng dụng trong nhận diện
mẫu và xử lí ảnh

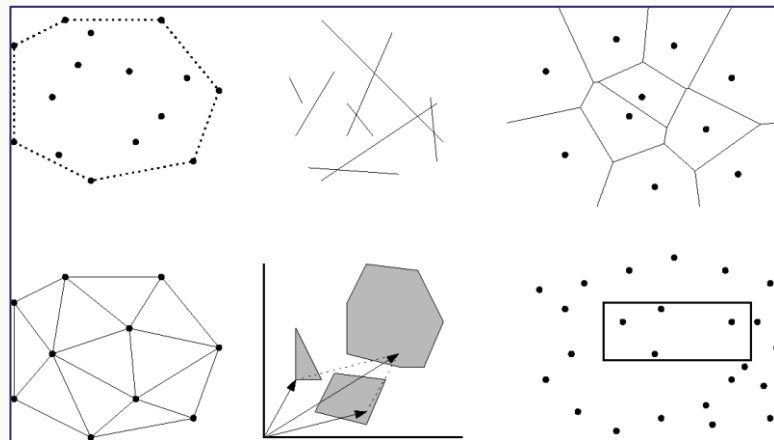


ỨNG DỤNG



Trong Khoa học máy tính

Ứng dụng trong Computational geometry



ỨNG DỤNG



Trong Địa lý

Hệ thống thông tin địa lý GIS

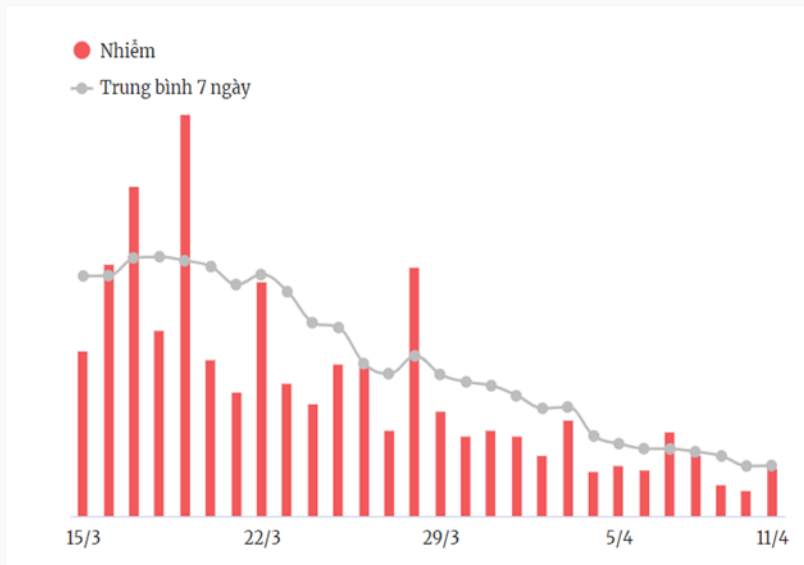


ỨNG DỤNG



Trong đời sống

Ứng dụng vào nhiều bài toán thực hiện truy vấn trên đoạn và cập nhật dữ liệu

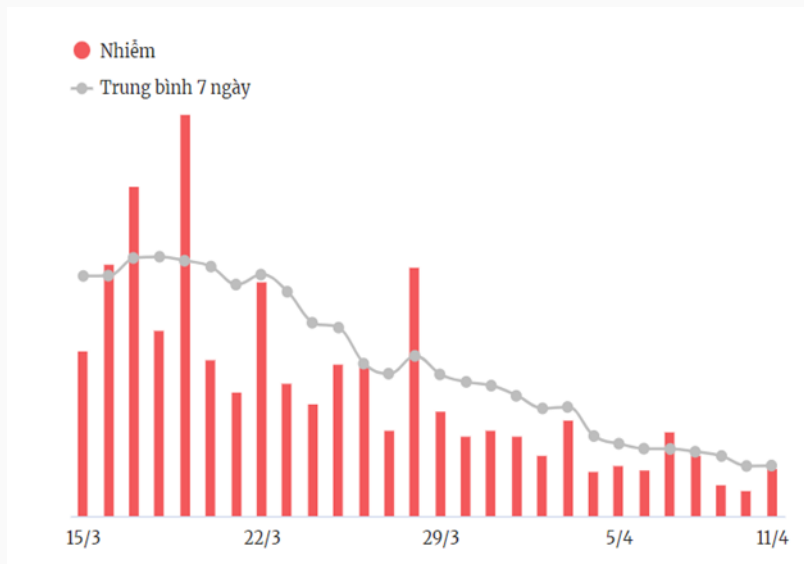


ỨNG DỤNG



Trong đời sống

Tính tổng số ca nhiễm
covid 19



ANY QUESTIONS ?



QUIZ

