Tips to Code

[Tìm tổ hợp nCr: 1](#_Toc53694660)

[Mod rule: 1](#_Toc53694661)

[Two Pointer Technique 1](#_Toc53694662)

[Window Sliding Technique 1](#_Toc53694663)

[Tìm tổng dãy con lớn nhất: 2](#_Toc53694664)

[Binary Search: 2](#_Toc53694665)

[binary\_search(startaddress, endaddress, valuetofind) 2](#_Toc53694666)

# Tìm tổ hợp nCr:

|  |  |
| --- | --- |
| nCr=(n-1)C(r-1)+(n-1)Cr | nC0=nCn=1 |

Vị trí của tam giác Pascal Là C(n,r)

# Mod rule:

|  |
| --- |
| (a+b)%m=(a%m+b%m)%m  (a\*b)%m=(a%m\*b%m)%m |

# Two Pointer Technique

Dùng để tìm cặp trong một mảng đã sắp xếp. 1 3 5 6 7 || x=9

Ví dụ: Cho mảng có n phần tử, tìm 2 phần tử có tổng bằng x.

Lấy 2 con trỏ ở đầu và cuối mảng, cộng các giá trị ở giữa hai con trỏ.

* Nếu sum<x, dịch con trỏ trái sang phải.
* Nếu sum>x, dịch con trỏ phải sang trái.

# Window Sliding Technique

Cách chuyển một vòng lặp lồng nhau thành một vòng lặp đơn để giảm thiểu độ phức tạp.

* Cộng k phần tử đầu tiên lưu vào window\_sum.
* Tiếp tuck duyệt hết mảng tìm tổng lớn nhất.
* Để lấy được tổng k số hiện tại chỉ cần loại bỏ phần tử đầu tiên của dãy trước và thêm phần tử cuối cùng của dãy hiện tại

# Tìm tổng dãy con lớn nhất:

* Idea: Tìm kiếm tất cả các đoạn liên tiếp dương của mảng(max\_here). Và theo dõi max của các đoạn con đó(max\_sum). Mỗi lần lấy được giá trị dương thì so sánh với max\_sum.
  + Max\_here=max\_here + a[i].
  + Nếu(max\_sum<max\_here) => max\_sum=max\_here.
  + Nếu (max\_ending\_here<0) max\_here=0.
* Return max\_sum;

<https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/>

# Binary Search:

## binary\_search(startaddress, endaddress, valuetofind)

return 1 if found, else return 0;

# Map

Gồm 2 phần tử:Key value và mapped value,trong đó:

* Các key mang giá trị khác nhau.
* Key dùng để xác định giá trị phần tử, mapped là giá trị của key.
* Các phần tử được sắp xếp.

Initialize: map<kieu\_du\_lieu\_key,kieu\_du\_lieu\_mapped>

* unordered\_map: Các phần tử ko đc sắp xếp , độ phức tạp lớn hơn