KỸ NĂNG QHD TRÊN DÃY SỐ

I/ Phân loại:

Các loại bài tập liên quan đến dãy số chia làm 2 loại:

- **Loại 1:** Thêm, sửa, xóa. Loại này hướng giải phải dùng cây.
- **Loại 2:** Dãy con. Trong đó dãy con chia làm 3 loại:
  + **Loại A:** Dãy con không liên tiếp có số lượng tùy ý thỏa mãn yêu cầu đề bài.
  + **Loại B:** Dãy con liên tiếp có số lượng tùy ý thỏa mãn điều kiện đề bài.
  + **Loại C:** Dãy con Không liên tiếp có đúng K phần tử thỏa mãn yêu cầu của đề.

Với phần này mình chỉ tìm hiểu Loại 2 với phương pháp QHD.

II/ Chi tiết:

1/ Loại A:

- Với dạng bài này thì không nhất thiết phải QHD mà chỉ cần xét tường phần tử của đề bài có thỏa mãn yêu cầu không, nếu thỏa mãn thì cho vào tập con. Việc này rất dễ dàng với độ phức tạp là O(n).

VD: Tìm dãy con có tổng lớn nhất.

Input	Output
5 1 -2 1 -3 1	3
5 -1 -2 -3 -4 -5	-1

Với bài này ta thấy ngay chỉ cần chọn các phần tử >=0 cho vào dãy con là được nếu không có thì chọn số âm lớn nhất là dãy con, việc chọn này cần xét từng phần tử rồi bỏ vào nên độ phức tạp là O(n) mà chúng ta không hề sử dụng QHD chi cả.

- **Bài tập VD Del(Đề không chuyên 2012).**

2/ Loại B:

- Với dạng bài này cần phải đưa về QHD để giải quyết với độ phức tạp là **O(n).**
- Phương pháp giải của dạng bài này tùy thuộc vào yêu cầu bài toán mà tư duy sẽ khác nhau, nhưng cuối cùng **F(n)** luôn được tính dựa trên các **F(x)** với **x<n**. Việc cần làm là tìm cách tính **F(n)** thông qua **F(x).**

VD1: Tìm dãy con có tổng lớn nhất, với các dãy con liên tiếp nhau.

Input	Output
16 2 -4 5 -8 4 -1 -1 1 1 1 -2 2 4 -6 9 -4	12

Với bài này các dãy con liên tiếp nhau và không có thêm điều kiện gì nên **F(n)** sẽ đươc thông qua **F(n-1).** Bởi vì nếu đã có **n-1** phần tử liên tiếp câu hỏi được đặt ra là thêm 1 phần tử nữa vào tập **n-1** phần tử trước đó có thỏa mãn yêu cầu bài không. Vậy ta tư duy theo hướng này. Ta xét vị trí thứ **i<=n** ta sẽ tìm được **i** dãy con có kết thúc tại **i** từ **i** dãy con đó ta chọn được dãy con lớn nhất kết thúc tại **i**. Vậy nếu theo cách nghĩ như vậy chúng ta phải tốn 2 vòng for để tính được hết **n** phần tử này.(Đây là bước mở màng cho QHD nếu suy nghĩ được đến đây và vét được theo giải thuật **$O(N^2)$** thì mọi người nên làm). Vậy bây giờ ta cần khử 1 vòng for đi và tư duy theo QHD, rõ ràng nếu đã có **i-1** phần tử nếu bỏ phần tử thứ **i** vào thì một là làm dãy con đó tang tổng lên hai là giảm. Như thế ta chỉ cần nếu tăng lên thì bỏ vào được còn giảm thì xem nó có giảm xuống nhỏ hơn giá trị **i** hay không, nếu không vẫn bỏ vào được vì đó là dãy con lớn nhất kết thúc tại **i**. Vậy ta sẽ cần 1 mảng maxsum để lưu lại max của từng vị trí **i**.

Với **maxsum[i]=max(maxsum[i-1]+Array[i],Array[i]).**

Vậy khởi đầu **maxsum[1]=Array[1].**

Độ phức tạp sẽ là **O(n).**

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array[i]	2	-4	5	-8	4	-1	-1	1	1	1	-2	2	4	-6	9	-4
Maxsum[i]	2	-2	5	-3	4	3	2	3	4	5	3	5	9	3	12	8

Vậy max của cả mảng maxsum là 12 là kết quả của bài toán.

- **Bài tập VD Bot(đề chuyên, không chuyên 2015).**

VD2: Tìm dãy con liên tiếp có tổng lớn nhất sau cho có chỉ số bắt đầu là **i** chỉ số kết thúc là **j** mà **j-i+1** phải chia hết cho **3**.

Input	**Output**
16 2 -4 5 -8 4 -1 -1 1 1 1 -2 2 4 -6 9 -4	9

Dễ dàng thấy được vd này giống như bài khác nhưng có kẹp theo một điều kiện, chính điều kiện này đã là cách tính QHD của mình thay đổi **F(n)** phải được tính thông qua **F(n-3).** Cụ thể

Maxsum[i]=max(Array[i]+Array[i-1]+Array[i-2]+maxsum[i-3], Array[i]+Array[i-1]+Array[i-2])

Với khởi đầu **maxsum[1]=maxsum[2]=0** vì không tồn tại dãy con có độ dài chia hết cho 3 kết thúc tại 2 vị trí đầu tiên. Tại sao lại có **Array[i]+Array[i-1]+Array[i-2]** vì bây giờ dãy của chúng ta buộc phải chia hết cho 3 nên phải gọp 3 phần từ liên tiếp tính 1 lần.

Độ phức tạp sẽ là **O(n).**

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array[i]	2	-4	5	-8	4	-1	-1	1	1	1	-2	2	4	-6	9	-4
Maxsum[i]	0	0	3	-7	1	-2	2	0	1	5	0	2	9	0	9	8

Vậy max của dãy maxsum là 9.

- Vậy tùy thuộc vào yêu cầu bài mà sẽ có cách tính khác một chút nhưng xét cho cùng cách giải điều giống nhau.

3/ Loại 3:

- Loại này độ phức tạp phụ thuộc vào số phần tử chọn là **K** là **N**. Độ phức tạp sẽ là **O(K*N).**
- Trong đề **K** sẽ sẽ thường là nhỏ nên mọi người yên tâm.
- Phương pháp giải: ta chọn ngay **K** phần tử đầu tiên của dãy là gọi nó là dãy đáp án. Sau đó xét lần lượt các phần tử thứ **i (k<i<=n)** thử xem có bỏ được vào trong tập kết quả không, nếu có **F(i)** sẽ có kq mới và **K** số mới dc chọn còn không **F(i)** sẽ nhận kq từ **F(i-1).** Vậy **F(n)** sẽ được tính thông qua **F(n-1).**
- Tư duy của bài toán: đã chọn được **K** phần tử rồi bỏ thêm 1 phần tử mới vào thì có làm thay đổi tập kết quả hay k? nếu không sẽ nhận kq từ vị trí đứng đằng trước mới bỏ vào. Vậy việc thay đổi **k** lần để tính kq ứng với mỗi **i** sẽ đưa bài toán về độ phức tạp O**(K*N).** (đó là trường hợp xấu nhất).

VD: Tìm tổng lớn nhất của **Ai+2Aj+3Ak** sao cho **i<j<k.**

Input	**Output**
16 2 -4 5 -8 4 -1 -1 1 1 1 -2 2 4 -6 9 -4	9

Với bài này ta tiếp tục tạo một **maxsum** để lưu các kết quả đã tính và **x y z** là bộ số được chọn.

Ta có **maxsum[i]=max(maxsum[i-1], x+2y+3A[i], x+2z+3A[i], y+2z+3A[i])**.

Với khởi đầu **Maxsum[1]=Maxsum[2]=âm vô cùng**; nếu tính min thì là dương vô cùng.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array[i]	2	-4	5	-8	4	-1	-1	1	1	1	-2	2	4	-6	9	-4
Maxsum[i]	-	-	9	9	24	24	24	24	24	24	24	24	25	25	40	40
X Y Z			2 -4 5	2 -4 5	2 5 4	2 5 4	2 5 4	2 5 4	2 5 4	2 5 4	2 5 4	2 5 4	5 4 4	5 4 4	5 4 9	5 4 9

Vậy max của maxsum là 40 với bộ số chọn là 5 4 9.

Với bài này chỉ **K=3** nên mình đươc về được còn khi **K** lớn buộc phải thêm 1 vòng for cho **K** để thay thế từng phần tử. Vậy với bài này độ phức tạp vẫn là **O(n)** do **K bé.**

- **Bài tập VD SEQ(đề chuyên 2009).**