~~Bài 1 : Ý tưởng là đưa về bài toán :~~

~~Cho 2 dãy A, B dãy C có C[k] = max(A[i] + B[j] với mọi i + j = k)~~

~~Ta tọa độ hóa mảng A thành các điểm (i, A[i])~~

~~B thành các điểm (j, A[j])~~

~~Trường hợp đẹp nhất là cả tập A và tập B đều là convex hull -> C chính là tổng minkowski của A và B~~

~~Trường hợp một trong 2 không là convex hull : lấy convex hull của một tập xong lấy nó ghép với một convex hull của tập khác sau đó xóa các điểm thuộc convex hull đi (khó sinh test tle :>)~~

Bài 2 : Ý tưởng là với mỗi phần tử i ta đếm xem có bao nhiêu ước của a[i] đã xuất hiện -> xuất hiện một cặp 2 phần tử, ta cũng đếm xem có bao nhiêu cặp 2 phần tử đã xuất hiện mà một ước của a[i] là thằng lớn hơn trong cặp đó

Đpt(O(nsqrt(n))), có thể tối ưu xuống O(n \* số ước lớn nhất) nhưng không cần thiết.

Bài 3 : Ý tưởng giống mọi bài tìm thứ tự từ điển nhỏ nhất khác, ta sẽ tối ưu lựa chọn ở các vị trí trước rồi kiểm tra nếu lựa chọn như thế thì có cách nào tạo ra một kết quả đủ số hay không.

Trong bài này ta biết độ dài của dãy con sẽ là số lượng phần tử phân biệt trong dãy (đặt là cnt), như vậy nếu chọn một phần tử i là phần tử đầu tiên của dãy thì ta phải kiểm tra xem sau i có đủ cnt – 1 phần tử phân biệt hay không (không tính a[i]).

Với mỗi “mốc” ví dụ như mốc phần tử thứ cnt hay phần tử thứ cnt – 1…, ta sẽ có một đoạn liên tiếp các “ứng viên” có thể là phần tử của mốc hiện tại trong dãy kết quả (giả sử là phần tử thứ x), các ứng viên này phải thỏa mãn là sau nó có đủ x – 1 phần tử (không tính các phần tử đã chọn) hay không.

Trong các ứng viên này, ta sẽ chọn ứng viên có giá trị lớn nhất, nếu có nhiều ứng viên cùng có giá trị lớn nhất thì ta chọn ứng viên xuất hiện đầu tiên, bởi vì các ứng viên hiện tại đáp ứng đủ x – 1 phần tử phân biệt ở sau nó (không tính các số đã chọn), nên khi chọn thêm một số, thì các ứng viên này mất đi nhiều nhất là một phần tử phân biệt sau nó -> nó vẫn là ứng viên cho mốc x – 2 -> thuật toán chỉ cần mở rộng danh sách ứng viên. Ta có thể gọi hàm đệ quy Select(đầu trái của tập ứng viên, đầu phải của tập ứng viên, số lượng phần tử cần lấy thêm (hay còn gọi là “mốc), phần tử cần chọn ở vị trí chẵn hay lẻ), duy trì thứ tự các phần tử ứng viên bằng priority\_queue và khi thêm một phần tử vào dãy kết quả (phần tử x chẳng hạn) thì các ứng viên xuất hiện trước vị trí xuất hiện cuối cùng của x sẽ bị bớt đi 1 phần tử phân biệt -> ta có thể dùng các dạng cấu trúc dữ liệu như segment tree hoặc BIT để duy trì xem một ứng viên đã bị bớt đi bao nhiêu phần tử phân biệt.

Đpt(O(nlogn))