Hướng dẫn giải:

Base:

- BT quy về xét 3 trường hợp:

+ Trường hợp 1: (Có nghiệm là cơ số trong đoạn 2 đến $\sqrt{n}$) trong trường hợp này chỉ cần chạy một vòng for đến căn n để tìm kết quả, nếu tồn tại kết quả thì in ra kq, còn không ta sang trường hợp 2.

+ Trường hợp 2: (Có nghiệm trong khoảng $\sqrt{n}$ đến n) gọi x là cơ số được chọn x > $\sqrt{n}$, vậy $\frac{n}{x}$< $\sqrt{n}$ bởi vì $\sqrt{n}$*x<=n. Vậy khi $\frac{n}{x}$< $\sqrt{n}$ thì $\frac{n}{x}$ chắc chắn là phần dư tiếp theo của phép tính nếu đêm chia cho x vì x> $\sqrt{n}$. Vậy ta đặt i là phần dư của các phép tính theo đề ta có $\frac{n-i}{x} = i$ =>$\frac{n-i}{i} = x$ vì để các phép chia lấy dư là bằng nhau với i<= $\sqrt{n}$. Vậy ta chỉ cần chạy một vòng for từ $\sqrt{n}$ về lại 1 nếu như (n-i) chia hết cho i thì nhận kết quả (vì sao phải chia hết vì x là một số nguyên, nên phải thoải mản đẳng thức trên), chú ý loại bỏ trường hợp x=i vì như thế phép chia lấy dư sau sẽ =0 là sai kq. Nếu vẫn k có kq ta tiến hành trường hợp 3

+ Trường hợp 3: nghiệm sẽ là n+1.

Độ phức tạp của thuật toán O($\sqrt{n}$).

Combin:

Bài toán này chỉ là xét chọn các số sao cho thoải mãn từng gt của x(x là số thứ x).

Ta chỉ cần chọn 1 số bắt đầu từ 1 đếm xem từ 1 có thể tạo thành bao nhiêu bộ số khác nhau. Số bộ số khác nhau dc tính theo tổ hợp chập k-số phần tử đã chọn của n-i phần từ với i là số đang được chọn.

Vậy để tính phần này mình cần tạo 1 mảng C(k,n) bằng phương pháp quy hoạch động với công thức array[i][j]=array[i-1][j]+array[i-1][j-1]. Sau đó chỉ cần xét nếu mà c(k,n) cần tính mà nhở hơn x thì kq đã lớn hơn số vừa thử nên trừ cho số bộ số nhỏ ứng với số đó, còn ngược lại tức là kq của mình năm trong số đang xét nhận ngay số đó tính số tiếp theo. Đến khi nào bạn đã chọn dc k số và x=0 thì bạn đã chọn được bộ số cần xét. Khuyên dùng 1 mảng để lưu các số đã chọn.

Độ phức tạp của bài là O(n*n). Vì cách tính là O(n) nhưng các bạn sinh ra mảng C(k,n) độ phức tạp là O(n*n) nên kết quả cuối cùng bài toán độ phức tạp là O(n^2).

FirstNum:

Với bài này các bạn có thể sài cách quy về chuỗi và tìm kiếm sẽ được giải thuật full nhưng theo mình nó chưa tối ưu nên mình xin trình bày cách $n\log_{10}(n)$.

Việc là của các bạn hết sức đơn giản là mỗi lần các bạn chỉ cần sóa số đầu tiền xong cộng với số tiếp theo đằng sau so sánh với số n nếu không bằng nhau thì bạn đếm số lượng tăng lên 1, còn bằng nhau thì kq sẽ là số lượng đã đếm- $n\log_{10}(n)+1$;

Vậy việc chia lấy dư rồi nhân lên cho 1 số có n số thì được xem là $n\log_{10}(n)$vì thế nó sẽ nhanh hơn rất nhiều so với việc bạn chuyển thành chuỗi rồi tìm kiếm.

Độ phức tạp là nlog(n).