**TABLE**

Input: N, K và 1 xâu S độ dài N

Có 3 TH:

**TH1:** Có 1 số X nào đó chia hết cho K xuất hiện **hoàn toàn** trong S.

Ví dụ K = 3, X = 21 xuất hiện hoàn toàn trong S = “8 |21| 24273”. Khi biết vị trí xuất hiện của X rồi (ở ví dụ này X = 21 xuất hiện bắt đầu ở ký tự thứ 2 của xâu S), dễ dàng kiểm tra được các phần còn lại của xâu có “phù hợp” hay ko. Bởi vì mình biết được các số trước X là X - K, X - 2K, X - 3K, …; các số sau X là X + K, X + 2K, X + 3K, …. Do đó mình sẽ suy ra được xâu S phải “trông” như thế nào.

Ở ví dụ trên, xâu S phù hợp với giả thiết X = 21. Bởi vì phần xâu trước “21” là 8 khớp với 21 - k = 21 - 3 = 18; phần xâu sau “21” là “24273” khớp với 21 + k = 24, 21 + 2k = 27, 21 + 3k = 30. Nhưng nếu S = “8 |21| 24272” thì S ko phù hợp với X = 21.

Ở TH này, ta duyệt mọi trường hợp có thể của X (tức duyệt mọi substring của S), rồi kiểm tra xem S có phù hợp với X giả thiết hay ko. Nếu có, vì đã biết X rồi, nên sẽ dễ dàng biết được xâu S xuất hiện ở giây thứ bao nhiêu (học sinh tự nghĩ).

**TH2:** Chỉ có **duy nhất** 1 số X **xuất hiện** trong xâu S.

Từ “xuất hiện” ở đâu không mang ý nghĩa xuất hiện hoàn toàn như trong TH 1, mà có thể chỉ có 1 phần chữ số của số X xuất hiện trong S thôi.

Ví dụ, K = 3, S = “56” chẳng hạn. X sẽ có dạng là …???56???... trong đó ? là 1 chữ số chưa biết, có thể nhận giá trị từ 0 tới 9. Muốn thời điểm xâu S xuất hiện là nhỏ nhất thì X phải nhỏ nhất.

Ta biết là 1 <= X <= 10^15 và X phải chia hết cho k. Vì số chữ số của X nhỏ ( <= 16), ta sẽ duyệt hết mọi trường hợp: phần của X ko xuất hiện trong S và nằm bên trái S là  L có LL chữ số, phần của X ko xuất hiện trong S và nằm bên phải S là R có RR chữ số (như thế X có LL + n + RR chữ số).

Ví dụ, X = 2 |56| 5 (S = “56”, k = 3), ta có L = 2, LL = 1, R = 5, RR = 1.

Điều kiện: 10^(LL - 1) <= L <= 10^LL - 1, 10^(RR - 1) <= RR <= 10^RR - 1.

Ta chỉ quan tâm X **nhỏ nhất** và chia hết cho K, do đó, ta chỉ cần quan tâm K giá trị nhỏ nhất có thể của L và R. Nói cách khác, chỉ cần xét 10^(LL - 1) <= L < 10^(LL - 1) + K và 10^(RR - 1) <= R < 10^(RR - 1) + K là đủ. Do K nhỏ, duyệt mọi giá trị có thể của L và R, kiểm tra xem khi “lắp” L, R vào X thì  có chia hết cho K hay không.

**TH3:** Có **đúng** 2 số **xuất hiện** trong S.

Từ “xuất hiện” ở đâu không mang ý nghĩa xuất hiện hoàn toàn như trong TH 1, mà có thể chỉ có 1 phần chữ số xuất hiện trong S thôi.

Ví dụ S = “13|26”. A và B = A + K là 2 số xuất hiện trong S. A có 2 chữ số cuối là AA = 13 và B có chữ số đầu là BB = 26.

Tương tự như 2 TH trên, ta cần tìm A nhỏ nhất sao cho A và B = A + K khớp với S.

Gọi L là số chữ số của B.

    ? ? ? ? ? ? **1 3**

+               **1 2 3 4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2 6** ? ? ? ? ? ?

Hình vẽ trên ví dụ L = 8, K = 1234.

Thực hiện cộng từ phải qua trái. Gọi 0 <= c1 <= 1 là nhớ khi vào đến BB. Nghĩa là, theo ví dụ trên, khi cộng xong 6 cột ở bên phải cùng thì sẽ có nhớ là c1.

Khi biết được c1 rồi thì ta sẽ biết được phần đầu của A là bao nhiêu để nó “khớp” với B.

Ví dụ:

khi c1 = 0 thì ta suy ra được:

**2 6** ? ? ? ? **1 3**

+               **1 2 3 4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2 6** ? ? ? ? ? ?

khi c1 = 1 thì ta suy ra được:

**2 5** ? ? ? ? **1 3**

+               **1 2 3 4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2 6** ? ? ? ? ? ?

Gọi c2 là nhớ khi cộng xong các cột ứng với AA. c2 ta tự tính được.

Ví dụ:

**x x** ? ? ? ? **1 3   (xx = 25 hoặc xx = 26, tùy vào c1 = 1 hay 0)**

+               **1 2 3 4**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2 6** ? ? ? ? **4 7**

**1 + 3 = 4 < 10,** vậy c2 = 0.

Do đó chỉ còn 4 dấu ? của số A là ta chưa biết.

Viết lại phép cộng:

              ? ? ? ?

+                   **1 2 (nhớ c2)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(nhớ c1) ? ? ? ?

Ta muốn A nhỏ nhất, vậy 4 dấu hỏi chưa biết của A phải nhỏ nhất (vì các thành phần còn lại của A là biết rồi). Gọi M là số ứng với 4 dấu hỏi chưa biết của A.

Ta biết: 1000 <= M <= 9999.

Nếu c1 = 0 thì M + 12 + c2 <= 9999 => M <= 9999 - 12 - c2

Nếu c1 = 1 thì M + 12 + c2 >= 10000 => M >= 10000 - 12 - c2

Dù c1 = bao nhiêu thì M phải nằm trong 1 khoảng U <= M <= V nào đấy. Lập luận giống TH2, do A cần chia hết L và **nhỏ nhất**, nên chỉ cần duyệt K giá trị nhỏ nhất của M là U <= M < U + k là đủ.

Với mỗi M, khi lắp vào A xem A có chia hết K hay ko.

Duyệt mọi TH có thể của L, c1 và c2, rồi kiểm tra như đã nói ở trên.

**BEADS (APIO2014)**

Tham khảo lời giải ở đây http://codeforces.com/blog/entry/11543