Piotr Stańczyk
Treść zadania, Opracowanie

Rafał Rusin, Jakub Radoszewski
Program

OI, Etap I

Bankomat

Bajtocki Bank Bitowy (w skrócie BBB) ma największą w Bajtocji sieć bankomatów. Klienci BBB wypłacają pieniądze z bankomatów na podstawie karty bankomatowej i 4-cyfrowego kodu PIN. Niedawno, w celu zwiększenia bezpieczeństwa swoich klientów, BBB zainstalował przy każdym bankomacie kamerę. Kamery przesyłają rejestrowany obraz do BBB drogą radiową. Niestety, sygnał wysyłany przez kamery jest przechwytywany przez szajkę złodziejaszków komputerowych. Złodziejaszkowie starają się odkryć 4-cyfrowe kody PIN klientów BBB, którym następnie kradną karty bankomatowe. Wiedząc o tym, klienci BBB, wprowadzając PIN i przesuwając palec nad klawiaturą, starają się wykonywać nadmiarowe ruchy. Kamera nie jest w stanie wychwycić naciskania klawiszy, rejestruje jedynie przesunięcia palca. Tak więc, jednoznaczne wyznaczenie wprowadzanego kodu PIN jest zazwyczaj niemożliwe. Przykładowo, klient przesuwający swój palec najpierw nad klawiszem 1, a potem 5, mógł wprowadzić następujące kody PIN: 1111, 1115, 1155, 1555 lub 5555. Zdesperowani złodziejaszkowie gromadzą nagrania z kamer, licząc na to, że być może na podstawie wielu nagrań tego samego klienta będą w stanie wyznaczyć wprowadzany przez niego PIN, lub choćby ograniczyć liczbę możliwych kodów. Zebrawszy sekwencje wprowadzane przez pewnego bogatego klienta BBB, złożyli Ci propozycję „nie do odrzucenia".

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia opis zarejestrowanych sekwencji ruchów palca, jakie klient banku wykonywał wprowadzając swój PIN,*
- *wyznaczy liczbę różnych kodów PIN, jakie może mieć klient (czyli liczbę tych 4-cyfrowych kodów PIN, które mogły być wprowadzone do bankomatu przy wykonywaniu zadanych sekwencji ruchów palca),*
- *wypisze znalezione rozwiązanie na standardowe wyjście.*

Wejście

Pierwszy wiersz wejścia zawiera jedną dodatnią liczbę całkowitą n równą liczbie zarejestrowanych scen wprowadzania kodu PIN przez klienta, $1 \leqslant n \leqslant 1000$. W kolejnych n wierszach znajdują się opisy tych scen, po jednej w wierszu. Opis każdej sceny składa się z dwóch dodatnich liczb całkowitych oddzielonych pojedynczym odstępem. Pierwsza z tych liczb, t, to długość sekwencji ruchów, $1 \leqslant t \leqslant 10\,000$. Druga z tych liczb to t-cyfrowa liczba, której kolejne cyfry reprezentują kolejne klawisze, nad którymi klient przesuwał palec. Łączna długość wszystkich sekwencji nie przekracza 1 000 000.

Wyjście

W pierwszym i jedynym wierszu wyjścia powinna znaleźć się jedna dodatnia liczba całkowita równa liczbie możliwych kodów PIN klienta.

Przykład

Dla danych wejściowych:

```
2
3 123
3 234
```

poprawnym wynikiem jest:

```
5
```

Rozwiązanie

Wprowadzenie

W treści zadania zwraca uwagę stosunkowo małe ograniczenie na długość kodu PIN (jest ono stałe — równe 4). W wielu urządzeniach, z których korzystamy w życiu codziennym (choćby w telefonach komórkowych), kody PIN są właśnie czterocyfrowe. Mają więc tylko 10 000 możliwych wartości, co pozwala na ich sekwencyjną weryfikację.

Zauważmy także, że na obrazie zarejestrowanym przez kamerę widać jedynie przesuwanie palca nad klawisze, nie widać natomiast, czy klient naciska klawisz i ile razy. Tak więc scena przedstawiająca przesuwanie palca kolejno nad klawisze 1, 2, 0 i 3 może oznaczać zarówno, że klient wpisał PIN 2233, jak i 1203 (oraz wiele innych).

Oznaczenia

Niech $s_1, s_2, \ldots, s_n$ będą sekwencjami wprowadzania kodu przez jednego klienta zarejestrowanymi przez kamerę. Oznaczmy przez w dowolny czterocyfrowy kod PIN, a przez $P(w)$ — jego *postać bez powtórzeń*, czyli sekwencję uzyskaną z kodu w przez usunięcie sąsiednich powtórzeń cyfr (przykładowo $P(1211) = 121$, $P(4333) = 43$, $P(1873) = 1873$).

Aby znaleźć wszystkie kody PIN, które mogą być kodem obserwowanego klienta, musimy dla każdej zarejestrowanej sekwencji s i każdego potencjalnego kodu PIN w sprawdzić, czy $P(w)$ jest podciągiem s. Szybkie wykonywanie tego sprawdzenia jest więc kluczowe dla efektywności algorytmu. Okazuje się, że można je wykonać w czasie liniowo zależnym od długości testowanego kodu PIN (a w naszym przypadku wynosi ona tylko 4).

Procedurę sprawdzenia, czy dla słowa a (potencjalnego kodu) jego postać bez powtórzeń $P(a)$ jest podciągiem słowa $b = b_0 b_1 \ldots b_{m-1}$ (zarejestrowanej sekwencji), rozpoczniemy od przetworzenia słowa b. Zdefiniujmy tablicę $B[x][y]$, dla $x \in \{0 \ldots m-1\}$ oraz $y \in \{0 \ldots 9\}$, w następujący sposób:

$$B[x][y] = \begin{cases} x & \text{dla } y = b_x \text{ oraz } x \leqslant m-1 \\ -1 & \text{dla } y \neq b_x \text{ oraz } x = m-1 \\ B[x+1][y] & \text{dla } y \neq b_x \text{ oraz } x \leqslant m-2 \end{cases}$$

Zawartość tablicy B można obliczyć w czasie $O(10 \cdot m)$ korzystając bezpośrednio z powyższej definicji. Z kolei mając już tablicę B można łatwo przeprowadzić test, o którym była mowa wcześniej.

```
1: procedure wB(a)
2:   pos = 0;
3:   for(int k = 0; k ⩽ 3 && pos ≠ −1; k++)
4:     pos = B[pos][a[k]];
5:   return (pos ≠ −1);
```

Sposób działania powyższego algorytmu jest prosty. Wystarczy zauważyć, że $B[x][y]$ to najmniejsza pozycja $z \geqslant x$ w słowie b, na której występuje cyfra y (lub -1, gdy cyfra y nie występuje na pozycjach $z \geqslant y$). Poszukiwania rozpoczynamy od cyfry $a[0]$ i $pos = 0$, znajdując za pomocą tablicy B w czasie stałym pierwsze wystąpienie $a[0]$ w b. Potem poszukujemy $a[1]$ począwszy od pozycji, na której znaleźliśmy $a[0]$. Analogicznie postępujemy dla $a[2]$ i $a[3]$. Dodatkowo, w przypadku serii jednakowych cyfr w słowie a, nie zmieniamy wartości pos, gdyż $B[pos][a[k]] = pos$ dla $a[k] = b_{pos}$. W ten sposób poszukujemy w rzeczywistości wystąpienia $P(a)$, a nie słowa a.

Ostatecznie procedura zwraca wartość *true* wtedy i tylko wtedy, gdy $P(a)$ występuje w b jako podciąg, a więc słowo a jest kodem PIN, którego wprowadzanie mogło zostać zarejestrowane, jako sekwencja b.

Rozwiązanie wzorcowe

Rozwiązanie wzorcowe rozpoczyna się od wczytania wszystkich sekwencji $s_1, s_2, \ldots, s_n$ i stworzenia dla nich tablic $B_1, B_2, \ldots, B_n$ zgodnie ze schematem opisanym wcześniej. Po zakończeniu tych przygotowań przeglądane są kolejno wszystkie kody PIN ze zbioru $\{0000, \ldots, 9999\}$. Dla każdego z nich sprawdzamy, czy procedura `wB` wykonana dla tablicy B_i pozwoli odnaleźć jego postać bez powtórzeń w sekwencji s_i. Końcowy wynik obliczamy zliczając kody występujące we wszystkich wczytanych sekwencjach.

Budowa jednej tablicy B_i odbywa się w czasie liniowym ze względu na jej wielkość, zatem sumaryczny czas potrzebny na skonstruowanie wszystkich tablic wynosi $O(10N_s) = O(N_s)$, gdzie $N_s = \sum_{i=1}^{n} |s_i|$. Wykonanie pojedynczego testu wymaga czasu stałego, zatem skonfrontowanie jednego kodu PIN ze wszystkimi sekwencjami $s_1, s_2, \ldots, s_n$ zajmuje czas $O(n)$. Takie sprawdzenie musimy wykonać dla każdego możliwego kodu PIN — jest ich $r = 10\,000$. Cały algorytm działa więc w czasie $O(N_s + n \cdot r)$.

Opisane rozwiązanie ma złożoność pamięciową $O(N_s)$, ponieważ wszystkie tablice B_i są konstruowane na początku i przechowywane w pamięci. Mieści się to w ograniczeniach podanych dla zadania. Można jednak nieco zmienić algorytm istotnie redukując jego zapotrzebowanie na pamięć. W tym celu wystarczy rozważać sekwencje $s_1, s_2, \ldots, s_n$ kolejno, dla każdej z nich budować tablicę B i konfrontować wszystkie możliwe kody PIN z tą sekwencją. Do zapisania wyników testów jest wówczas potrzebna dodatkowa tablica o rozmiarze $O(r)$, zawierająca dla każdego kodu liczbę sekwencji, które mogą oznaczać

jego wprowadzenie. W ten sposób złożoność pamięciowa algorytmu zostaje zredukowana do $O(r+m)$, gdzie m jest ograniczeniem długości jednej sekwencji, przy zachowaniu bez zmian złożoności czasowej.

Inne rozwiązania

Zadanie można rozwiązać wykonując zwykłe wyszukiwanie kodu PIN (a dokładniej, jego postaci bez powtórzeń) w sekwencji w czasie liniowo zależnym od sumy długości kodu i sekwencji. Daje to w rezultacie program działający w czasie $O(N_s \cdot r)$.

Testy

Zadanie było sprawdzane na 15 testach.

Pierwsze trzy testy (*ban1.in* – *ban3.in*) to małe testy poprawnościowe. Test *ban4.in* to duży test poprawnościowy o specyficznej strukturze. Okazał się on najtrudniejszy — na nim najczęściej programy zawodników zwracały złe wyniki. Reszta testów (*ban5.in* – *ban15.in*) została wygenerowana w sposób pseudo-losowy.

Poniżej znajduje się krótki opis każdego z testów, zawierający liczbę zarejestrowanych scen wprowadzania kodu PIN n oraz sumaryczną długość sekwencji reprezentujących zarejestrowane sceny N_s.

Nazwa	n	N_s
ban1.in	2	6
ban2.in	1	1
ban3.in	5	195
ban4.in	100	15 125
ban5.in	100	807 101
ban6.in	100	973 642
ban7.in	1 000	973 718
ban8.in	1 000	974 747

Nazwa	n	N_s
ban9.in	1 000	974 369
ban10.in	1 000	974 127
ban11.in	500	987 664
ban12.in	500	987 696
ban13.in	1 000	974 364
ban14.in	1 000	974 459
ban15.in	1 000	974 731