Wojciech Guzicki
Treść zadania, Opracowanie

Wojciech Dudek
Program

Szyfr

Dany jest ciąg dodatnich liczb całkowitych a_i (dla $i = 1, 2, \ldots, n$). Ciąg ten jest używany do szyfrowania n-bitowych wiadomości. Jeśli mamy wiadomość, której kolejne bity tworzą ciąg (t_1, t_2, ..., t_n) ($t_i \in \{0, 1\}$), to po zaszyfrowaniu ma ona postać liczby:

$$S = t_1 a_1 + t_2 a_2 + \cdots + t_n a_n$$

Zadanie

Masz dane zaszyfrowane wiadomości oraz ciągi liczb (a_i), których użyto do ich zaszyfrowania. Twoje zadanie polega na odkodowaniu zaszyfrowanych wiadomości i zapamiętaniu ich w określonych plikach. Nie musisz dostarczać żadnego programu. Wystarczy, że zapiszesz odszyfrowane wiadomości.

Wejście

Dysponujesz kilkoma zestawami danych. Każdy zestaw jest zapisany w osobnym pliku: `szyk.in`*, gdzie k jest numerem zestawu. W pierwszym wierszu każdego z tych plików znajduje się jedna liczba całkowita n, $5 \leqslant n \leqslant 40$. W kolejnych n wierszach zapisany jest ciąg liczb (a_i): w $i+1$-ym wierszu zapisana jest jedna dodatnia liczba całkowita a_i. Suma liczb a_i nie przekracza 2 000 000 000. W $n+2$-im wierszu zapisana jest jedna liczba całkowita S — zaszyfrowana wiadomość, $0 \leqslant S \leqslant a_1 + a_2 + \cdots + a_n$.*

Wyjście

Dla każdego zestawu danych `szy*.in` *powinieneś utworzyć plik* `szy*.out` *zawierający odszyfrowaną wiadomość. W pierwszym wierszu tego pliku należy zapisać kolejne liczby t_i, bez żadnych odstępów między nimi. Dane testowe zostały dobrane tak, że zaszyfrowane wiadomości są określone jednoznacznie.*

Przykład

Dla pliku wejściowego `szy.in`*:*

```
24
19226985
123697
67356296
19721773
1113273
69335448
23680077
```

```
9029881
85168664
93676782
5253843
77616588
78572630
13375812
17199980
101508862
59248276
3505733
35790095
62028546
85726819
56462819
103373994
91757169
667509506
```

poprawną odpowiedzią jest plik wyjściowy `szy.out`*:*

```
110001000101101100010101
```

Rozwiązanie

Problem, który mamy rozwiązać, jest szczególnym przypadkiem tzw. problemu pakowania plecaka. W całej ogólności został on bardzo dobrze opisany w książce M. Sysły ([26] str. 215–228), gdzie w szczególności można znaleźć dość skuteczne algorytmy oparte na metodzie programowania dynamicznego. Działają one dobrze wtedy, gdy suma liczb $W = a_1 + a_2 + \cdots + a_n$ jest dość mała (złożoność wynosi $O(nW)$). W naszym przypadku, gdy liczba W może wynosić około $2 \cdot 10^9$, te algorytmy są nieprzydatne.

Zadanie nie ma dobrego rozwiązania. Mianowicie jeśli dane są dowolne liczby całkowite $a_1, a_2, \ldots, a_n$ i liczba całkowita S, to zadanie polegające na znalezieniu ciągu $(t_1, t_2, \ldots, t_n)$ o wyrazach ze zbioru $\{0,1\}$ takiego, że

$$t_1 a_1 + t_2 a_2 + \cdots + t_n a_n = S,$$

jest przykładem problemu NP-zupełnego. Nie możemy więc spodziewać się rozwiązania działającego w czasie wielomianowym. Jedyne znane rozwiązania tego problemu w całej ogólności działają w czasie wykładniczym. Przyjrzyjmy się zatem takim rozwiązaniom.

Najbardziej narzuca się metoda przeszukania wszystkich ciągów zerojedynkowych; też trzeba to zrobić zręcznie, by wielokrotnie nie dodawać do siebie tych samych liczb. Taki program działa w czasie $O(2^n)$.

Skuteczniejsza jest metoda przeszukiwania z nawrotami. Ten algorytm polega na przeglądaniu wszystkich przypadków, które mogą prowadzić do rozwiązania. Przypuśćmy, że w danym momencie zdecydowaliśmy się wybrać liczby $t_1, t_2, \ldots, t_k$, gdzie $k < n$. Niech

$$Suma = t_1 a_1 + t_2 a_2 + \cdots + t_k a_k.$$

Mamy teraz następujące możliwości:

1. Jeśli $Suma > S$, to dalsze poszukiwania nie mają sensu i musimy cofnąć się do ciągu liczb $t_1, t_2, \ldots, t_{k-1}$.

2. Jeśli $Suma + a_{k+1} + \cdots + a_n < S$, to też musimy zakończyć poszukiwania i cofnąć się do ciągu liczb $t_1, t_2, \ldots, t_{k-1}$.

3. Jeśli nie zachodzi żaden z powyższych przykładów, to rozważamy dwa przypadki: $t_{k+1} = 0$ i $t_{k+1} = 1$.

Aby lepiej wykorzystać możliwość przerywania poszukiwań (i cofania się do poprzedniego ciągu liczb t_i), sortujemy na początku liczby $a_1, a_2, \ldots, a_n$ od największej do najmniejszej. Wtedy przypadki, w których mamy się cofnąć, wystąpią wcześniej i będziemy musieli sprawdzić mniej przypadków. Ten algorytm działa zupełnie dobrze wtedy, gdy wśród liczb $a_1, a_2, \ldots, a_n$ występują zarówno liczby małe, jak i duże. Na przykład dla ciągów „prawie geometrycznych", tzn. takich, że $a_i \approx c^i$ dla pewnej stałej c, działa on bardzo szybko. Natomiast jeśli wszystkie liczby a_i są podobnego rzędu wielkości, ten algorytm też będzie działał w czasie $O(2^n)$.

Można to zadanie rozwiązać szybciej. Zapisujemy wszystkie możliwe sumy $\sum_{i \leqslant n/2} t_i a_i$ oraz $\sum_{i > n/2} t_i a_i$ w dwóch tablicach P i Q. Niech tablica Q ma długość k. Zapisujemy też odpowiednie ciągi t_i (wystarczą do tego trzy bajty). Następnie sortujemy obie tablice. Daną sumę identyfikujemy za pomocą następującej procedury:

```
1: i := 1;
2: j := k;
3: OK := false;
4: while not OK do
5:    if P[i] + Q[j] < S then i := i + 1
6:    else if P[i] + Q[j] > S then j := j - 1
7:    else OK := true;
```

Szukany ciąg liczb t_i znajdujemy teraz łatwo z ciągów odpowiadających sumom $P[i]$ i $Q[j]$.

Ten program działa w czasie rzędu $2^{n/2}$, wymaga jednak bardzo dużej pamięci. To ograniczenie pamięci spowodowało ograniczenie górnej wartości n do 40. Interesujące jest jednak to, że jest to najszybszy algorytm rozwiązujący ten problem w całej ogólności. Program wzorcowy wykorzystuje właśnie ten algorytm.

Problem postawiony w tym zadaniu wziął się z zaproponowanej w 1978 roku przez Merklego i Hellmana metody szyfrowania. Opiszemy teraz krótko tę metodę.

Najpierw wybieramy ciąg liczb $b_1, b_2, \ldots, b_n$, dla których problem pakowania plecaka można rozwiązać bardzo łatwo. Na przykład wybieramy tzw. ciąg superrosnący, tzn. taki, że każda liczba b_k jest większa od sumy liczb poprzednich:

$$b_k > b_1 + b_2 + \cdots + b_{k-1}.$$

Znalezienie prostego algorytmu, za pomocą którego możemy w czasie liniowym wyznaczyć liczby $t_1, t_2, \ldots, t_n$ takie, że

$$t_1 b_1 + t_2 b_2 + \cdots + t_n b_n = S,$$

pozostawimy Czytelnikowi jako ćwiczenie.