Jakub Pachocki — Treść zadania, Opracowanie
Jakub Pachocki — Program

Dostępna pamięć: 64 MB. *OI, etap II, dzień pierwszy, 8.02.2012*

Bony

Sklep ze słodyczami, którego właścicielem jest Bajtazar, prowadzi sprzedaż pysznych cukierków karmelowych. Dla każdej liczby całkowitej dodatniej c w sklepie znajduje się dokładnie jedna paczka zawierająca c cukierków (i w chwili obecnej nie są przewidywane kolejne dostawy). Aby zachęcić klientów do kupna łakoci, Bajtazar powrzucał do m paczek bony na roczny zapas czekolady. Upewnił się przy tym, aby nie wrzucić więcej niż jednego bonu do tej samej paczki.

W przyszłym tygodniu w Bajtogrodzie rozpoczynają się obchody karnawału, który potrwa n dni; k-tego dnia karnawału odbędzie się przyjęcie, w którym będzie uczestniczyć a_k osób. Bajtazar jest przekonany, że k-tego dnia rano każdy z uczestników odbywającego się tego dnia przyjęcia kupi w jego sklepie najmniejszą dostępną paczkę cukierków, której zawartość będzie można rozdzielić równo pomiędzy wszystkie zaproszone osoby. Przykładowo, jeśli $n = 2$, $a_1 = 4$, $a_2 = 2$, to pierwszego dnia karnawału zostaną sprzedane kolejno paczki zawierające cztery, osiem, dwanaście i szesnaście cukierków, a drugiego dnia — paczki z dwoma i sześcioma cukierkami.

Bajtazar zastanawia się, którzy klienci kupią paczki z bonami. Poprosił Cię, żebyś napisał program, który pomoże mu to określić.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita m ($1 \leqslant m \leqslant 1\,000\,000$), oznaczająca liczbę bonów. W k-tym z kolejnych m wierszy znajduje się liczba całkowita b_k ($1 \leqslant b_k \leqslant 1\,000\,000$) oznaczająca rozmiar paczki (tj. liczbę cukierków w paczce), w której Bajtazar umieścił k-ty bon. Liczby te są podane w kolejności rosnącej.

W następnym wierszu znajduje się jedna liczba całkowita n ($1 \leqslant n \leqslant 1\,000\,000$), oznaczająca liczbę dni karnawału. W k-tym z kolejnych n wierszy znajduje się liczba całkowita a_k ($1 \leqslant a_k \leqslant 1\,000\,000$), oznaczająca liczbę gości zaproszonych na przyjęcie odbywające się w k-tym dniu.

W testach wartych łącznie przynajmniej 50% punktów żadna z liczb podanych na wejściu nie przekroczy 5 000.

Wyjście

W pierwszym wierszu standardowego wyjścia Twój program powinien wypisać liczbę całkowitą z — liczbę sprzedanych paczek z bonami. W kolejnych z wierszach powinny znaleźć się numery wszystkich klientów, którzy kupili paczkę z bonem, w porządku rosnącym. Klientów numerujemy od 1 w kolejności dokonywania zakupów.

Przykład

Dla danych wejściowych:

```
1
6
8
16
3
4
2
4
```

poprawnym wynikiem jest:

```
3
2
4
6
```

Rozwiązanie

Wprowadzenie

Dla lepszego zrozumienia podanego problemu, warto na początku wyobrazić go sobie w sposób czysto matematyczny. Dane jest n liczb naturalnych $a_1, \ldots, a_n$. Tworzymy ciąg c, złożony z $a_1 + \ldots + a_n$ liczb naturalnych, przez postawienie na początku a_1 najmniejszych liczb naturalnych podzielnych przez a_1 (pierwsze przyjęcie), następnie a_2 najmniejszych niewystępujących dotąd w ciągu liczb naturalnych podzielnych przez a_2 (drugie przyjęcie), itd. Zadanie polega na efektywnym określeniu, na jakich pozycjach w ciągu c znajdują się liczby należące do zadanego zbioru $B = \{b_1, \ldots, b_m\}$.

W przykładzie z treści zadania mamy $a = (4, 2, 4)$ oraz $B = \{1, 6, 8, 16\}$. Ciąg c wyznaczony dla podanego ciągu a ma postać $c = (4, 8, 12, 16, 2, 6, 20, 24, 28, 32)$. Drugi, czwarty i szósty element tego ciągu należą do zbioru B.

Rozwiązanie wzorcowe

Oznaczmy przez M maksimum z wszystkich liczb występujących na wejściu. Rozwiązanie wzorcowe polega na symulowaniu zakupów kolejnych klientów. Należy tylko pamiętać o tym, aby nie rozważać rozmiarów paczek większych niż M (gdyż w nich i tak nie ma żadnych bonów), a w przypadku powtarzających się elementów ciągu a, przy rozpatrywaniu a_i zaczynać przeglądanie paczek od ostatniej paczki rozpatrzonej dla elementu równego a_i.

Innymi słowy, dla każdej liczby naturalnej $p \in [1, M]$ pamiętamy wartość *ostatnia*$[p]$, oznaczającą ostatnią wielokrotność liczby p zużytą w trakcie rozważania jakiegoś $a_i = p$. Jeśli *ostatnia*$[p] > M$, możemy wówczas przerwać rozpatrywanie tego a_i. Poza tym utrzymujemy tablicę wartości logicznych *paczka*$[1..M]$, wskazującą, które paczki już zostały wykupione, a także podobną tablicę *bon*$[1..M]$, informującą, w których paczkach znajdują się bony. Poniżej znajduje się pseudokod algorytmu symulującego zakupy kolejnych klientów przy użyciu podanych tablic.

```
1:  program bony
2:  begin
3:    for i := 1 to M do begin
4:      ostatnia[i] := 0;
5:      paczka[i] := false;
6:      bon[i] := false;
7:    end
8:    for i := 1 to m do
9:      bon[b[i]] := true;
10:   liczba_klientów := 0;
11:   for i := 1 to n do begin
12:     p := a[i];
13:     akt := ostatnia[p] + p;
14:     uczestnik := 1;
15:     while (akt ⩽ M) and (uczestnik ⩽ p) do begin
16:       if not paczka[akt] then begin
17:         paczka[akt] := true;
18:         if bon[akt] then
19:           wypisz(liczba_klientów + uczestnik);
20:         uczestnik := uczestnik + 1;
21:       end
22:       if uczestnik ⩽ p then akt := akt + p;
23:     end
24:     ostatnia[p] := akt;
25:     liczba_klientów := liczba_klientów + p;
26:   end
27: end
```

Powyższe rozwiązanie może na pierwszy rzut oka wydawać się niezbyt efektywne. Występująca w nim pętla **while**, zagnieżdżona w pętli **for**, może za każdym razem wykonywać wiele obrotów, w tym dla rozmiarów paczki *akt*, które zostały już wykupione (tj. gdy *paczka*[*akt*] = **true**). Zauważmy jednak, że dla każdej możliwej wartości p, jaką może przyjąć a_i, każdą z $\left\lfloor \frac{M}{p} \right\rfloor$ wielokrotności p rozpatrzymy co najwyżej raz (dzięki wykorzystaniu tablicy *ostatnia*). To daje nam następujące ograniczenie górne na łączną liczbę obrotów pętli **while**:

$$\sum_{p=1}^{M} \frac{M}{p} = M \cdot H_M = O(M \log M).$$

W powyższym wzorze H_M to M-ta liczba harmoniczna, $H_M = \frac{1}{1} + \frac{1}{2} + \ldots + \frac{1}{M}$, którą to liczbę możemy oszacować z góry przez $O(\log M)$. To oszacowanie pojawiało się już w zadaniach olimpijskich (np. w zadaniu *Korale* z XVII Olimpiady Informatycznej [17]), można o nim także poczytać w książce [25].

Całe rozwiązanie działa więc w czasie $O(M \log M)$. Jego implementacje można znaleźć w plikach `bon.cpp` i `bon1.pas`.

Inne rozwiązania

Gdyby w powyższym algorytmie dla każdego a_i przeglądać jego wielokrotności od początku, bądź też iterować po wielokrotnościach a_i przekraczających M, otrzyma się rozwiązanie o złożoności czasowej $O(M^2)$. Implementacje rozwiązania wykazującego pierwszą z wymienionych usterek można znaleźć w plikach `bons1.cpp` i `bons2.pas`. Zgodnie z informacją podaną w treści zadania, takie rozwiązania zdobywały na zawodach około 50 punktów.

Możliwym błędem w implementacji rozwiązania wzorcowego było niezauważenie, że wyniki (czyli numery klientów, którzy zakupią paczki z bonami) mogą nie mieścić się w 32-bitowym typie całkowitym (patrz pliki `bonb1.cpp` i `bonb2.pas`). Taki błąd kosztował na zawodach utratę 40 punktów.

Testy

Rozwiązania zawodników były sprawdzane za pomocą 10 zestawów danych testowych. Wszystkie testy były generowane w sposób losowy, przy czym w testach *3b*, *4c*, *5c*, *6b*, *7bcd*, *8bcd*, *9cd* oraz *10cd* liczby gości zaproszonych na poszczególne przyjęcia były w znacznej części niewielkimi liczbami całkowitymi z wybranego zakresu. W poniższej tabeli m oznacza liczbę bonów, natomiast n — liczbę przyjęć.

Nazwa	m	n
bon1a.in	5	15
bon1b.in	5	15
bon1c.in	5	15
bon2a.in	20	50
bon2b.in	1	1
bon2c.in	17	100
bon3a.in	300	700
bon3b.in	300	700
bon4a.in	2 000	2 000
bon4b.in	2 000	2 000
bon4c.in	2 000	2 000
bon5a.in	3 000	5 000
bon5b.in	5 000	5 000
bon5c.in	5 000	5 000
bon6a.in	12 345	32 000
bon6b.in	30 000	40 000

Nazwa	m	n
bon7a.in	120 345	320 000
bon7b.in	200 000	300 000
bon7c.in	200 000	300 000
bon7d.in	200 000	300 000
bon8a.in	400 000	400 000
bon8b.in	370 000	400 000
bon8c.in	370 000	400 000
bon8d.in	400 000	400 000
bon9a.in	800 000	900 000
bon9b.in	80	900 000
bon9c.in	370 000	800 000
bon9d.in	370 000	400 000
bon10a.in	1 000 000	1 000 000
bon10b.in	500 000	1 000 000
bon10c.in	1 000	1 000 000
bon10d.in	1 000	1 000 000