Karol Pokorski
Treść zadania, Opracowanie

Michał Zgliczyński
Program

Dostępna pamięć: 64 MB. *OI, etap II, dzień drugi, 9.02.2012*

Rozkład Fibonacciego

Ciąg liczb Fibonacciego to ciąg liczb całkowitych zdefiniowany rekurencyjnie w następujący sposób:

$$Fib_0 = 0, \quad Fib_1 = 1, \quad Fib_n = Fib_{n-2} + Fib_{n-1} \text{ dla } n > 1.$$

Początkowe elementy tego ciągu to: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Bajtazar bada, w jaki sposób różne liczby można przedstawić jako sumy lub różnice liczb Fibonacciego. Aktualnie zastanawia się, jak daną dodatnią liczbę całkowitą k przedstawić jako sumę lub różnicę jak najmniejszej liczby (niekoniecznie różnych) liczb Fibonacciego. Na przykład, liczby 10, 19, 17 i 1070 można przedstawić minimalnie, odpowiednio, za pomocą 2, 2, 3 oraz 4 liczb Fibonacciego:

$$\begin{aligned} 10 &= 5 + 5 \\ 19 &= 21 - 2 \\ 17 &= 13 + 5 - 1 \\ 1070 &= 987 + 89 - 5 - 1 \end{aligned}$$

Pomóż Bajtazarowi! Napisz program, który dla danej dodatniej liczby całkowitej k wyznaczy minimalną liczbę liczb Fibonacciego potrzebnych do przedstawienia liczby k jako ich sumy lub różnicy.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita p ($1 \leqslant p \leqslant 10$) oznaczająca liczbę zapytań. W kolejnych p wierszach znajduje się po jednej dodatniej liczbie całkowitej k ($1 \leqslant k \leqslant 4 \cdot 10^{17}$).

Wyjście

Twój program powinien wypisać na standardowe wyjście dla każdego zapytania minimalną liczbę liczb Fibonacciego potrzebnych do przedstawienia liczby k jako ich sumy lub różnicy.

Przykład

Dla danych wejściowych:

```
1
1070
```

poprawnym wynikiem jest:

```
4
```

Rozwiązanie

Wprowadzenie

Dana jest dodatnia liczba całkowita k. *Rozkładem* liczby k nazwiemy dowolną parę multizbiorów (P_+, P_-) spełniającą warunek:

$$\sum_{i \in P_+} Fib_i - \sum_{i \in P_-} Fib_i = k.$$

Rozmiarem rozkładu nazwiemy sumę rozmiarów multizbiorów P_+ i P_-. Rozkład nazwiemy *optymalnym*, jeśli nie istnieje żaden inny rozkład liczby k o mniejszym rozmiarze.

Pierwsze rozwiązania

Do uzyskania poprawnego, ale powolnego rozwiązania wystarczy tylko kilka dosyć intuicyjnych obserwacji. Przede wszystkim możemy bez straty ogólności przyjąć, że wszystkie elementy multizbiorów P_+ i P_- (czyli indeksy liczb Fibonacciego w rozkładzie) są nie mniejsze niż 2. Dalej:

Obserwacja 1. Jeśli w rozkładzie liczby istnieje indeks i, taki że $i \in P_+$ oraz $i \in P_-$, to rozkład ten nie jest optymalny.

Faktycznie, rozkład opisany w powyższej obserwacji można polepszyć, usuwając z obu multizbiorów po jednym wystąpieniu indeksu i.

Chociaż warunki zadania dopuszczają wykorzystanie jednej liczby Fibonacciego więcej niż raz, wydaje się naturalne, że w ten sposób nie otrzyma się rozkładu optymalnego. Przykład z treści zadania $(10 = 5 + 5)$ zdaje się temu przeczyć, jednak tę samą liczbę możemy także przedstawić inaczej, tym razem już bez powtórzeń: $10 = 2 + 8$.

Obserwacja 2. Rozkład, w którym jakaś liczba Fibonacciego występuje co najmniej trzykrotnie, nie jest optymalny. Co więcej, dla każdej liczby istnieje rozkład optymalny, w którym każda liczba Fibonacciego występuje co najwyżej raz.

Dowód: Aby uzasadnić powyższe, wystarczą proste rachunki z wykorzystaniem definicji rekurencyjnej ciągu Fibonacciego:

$$\begin{aligned} 3 \cdot Fib_i &= (Fib_{i-2} + Fib_{i-1}) + Fib_i + Fib_i = Fib_{i-2} + (Fib_{i-1} + Fib_i) + Fib_i \qquad (1) \\ &= Fib_{i-2} + (Fib_{i+1} + Fib_i) = Fib_{i-2} + Fib_{i+2}; \\ 2 \cdot Fib_i &= (Fib_{i-2} + Fib_{i-1}) + Fib_i = Fib_{i-2} + (Fib_{i-1} + Fib_i) = Fib_{i-2} + Fib_{i+1}. \qquad (2) \end{aligned}$$

Za pomocą operacji (1) każde potrójne wystąpienie liczby Fibonacciego można zastąpić wystąpieniem tylko dwóch liczb Fibonacciego, co oczywiście zmniejsza rozmiar rozkładu. Taka operacja nie może więc być wykonalna w żadnym rozkładzie optymalnym. Natomiast za pomocą operacji (2) możemy sukcesywnie eliminować wszystkie podwójne wystąpienia liczb Fibonacciego z rozkładu, nie pogarszając rozmiaru rozkładu. Należy tu jednak być ostrożniejszym, gdyż po wykonaniu operacji (2) z kolei liczby Fib_{i-2} i Fib_{i+1} mogą występować w rozkładzie wielokrotnie, więc potencjalnie moglibyśmy uzyskać nieskończoną sekwencję operacji usuwania podwójnych wystąpień. Warto jednak zauważyć, że każda taka operacja zmniejsza o jeden *sumę indeksów* liczb Fibonacciego występujących w rozkładzie. To pozwala upewnić się, że operacji drugiego typu wykonamy zawsze tylko skończenie wiele i na końcu każda liczba Fibonacciego wystąpi w rozkładzie co najwyżej raz. (Dodajmy jeszcze, że jeśli po wykonaniu którejś operacji to liczba Fib_2 występuje dwukrotnie, możemy od razu zamienić oba jej wystąpienia na Fib_3 i orzec, że rozkład nie był optymalny). ■

Z dotychczasowych rozważań wynika, że możemy skupić się na poszukiwaniu *zbiorów* (a nie multizbiorów) P_+ i P_-, w których łącznie każdej liczby Fibonacciego używamy co najwyżej raz. W dalszej części opisu będą nas interesować tylko takie rozkłady.

Dalej, nietrudno uwierzyć, że w rozkładzie optymalnym liczby nie opłaca nam się używać liczb Fibonacciego istotnie większych od tej liczby. To stwierdzenie jest sprecyzowane w poniższej obserwacji. Dowód obserwacji pomijamy — wyniknie on z dowodu poprawności rozwiązania wzorcowego.

Obserwacja 3. Jeśli $Fib_m \leqslant k < Fib_{m+1}$, to istnieje rozkład optymalny liczby k, w którym indeksy liczb Fibonacciego nie przekraczają $m+1$.

Możemy już teraz zaproponować pierwsze rozwiązanie. Wystarczy iterować po kolejnych indeksach $i = 2, 3, \ldots, m+1$ dla m określonego jak w obserwacji 3 i dla każdego z nich rozpatrzyć trzy możliwości: albo $i \in P_+$, albo $i \in P_-$, albo $i \notin P_+$ oraz $i \notin P_-$. Po każdej sekwencji wyborów sprawdzamy, czy otrzymaliśmy rozkład liczby k, a jeśli tak, czy jest to rozkład zawierający mniej liczb Fibonacciego niż najlepszy dotychczas znaleziony. Jest to, oczywiście, rozwiązanie wykładnicze, ale wykładnicze względem m; w każdym kroku mamy trzy możliwe wybory, więc złożoność czasowa tego rozwiązania to $O(3^m)$.

Aby przekonać się, jaka jest złożoność czasowa tego rozwiązania względem k, warto przypomnieć sobie znany wzór Bineta na n-tą liczbę Fibonacciego:

$$Fib_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Pierwsze z wyrażeń w nawiasie, równe w przybliżeniu 1,618, nazywa się zwyczajowo *złotym podziałem* i oznacza grecką literą ϕ. Z kolei drugie z wyrażeń jest równe $1-\phi \approx -0{,}618$, więc dowolne jego potęgi o naturalnym wykładniku są zawsze mniejsze co do wartości bezwzględnej niż 1. Widzimy więc, że n-ta liczba Fibonacciego jest równa, z dokładnością do stałego czynnika, ϕ^n. Mamy stąd $k \approx \phi^m$, czyli $m \approx \log_\phi k$, przy czym ta ostatnia równość zachodzi z dokładnością do stałej. Ostatecznie:

$$3^m = O\big(3^{\log_\phi k}\big) = O\big(3^{\log_3 k / \log_3 \phi}\big) = O\big((3^{\log_3 k})^{\log_\phi 3}\big) = O\big(k^{\log_\phi 3}\big).$$

Widzimy zatem, że nasze rozwiązanie jest w rzeczywistości wielomianowe względem k, przy czym wykładnik wielomianu znajduje się gdzieś pomiędzy 2 a 3 (ponieważ, jak łatwo sprawdzić z użyciem kalkulatora, $1{,}618^2 \approx 2{,}618 < 3$ i $1{,}618^3 \approx 4{,}236 > 3$). Dokładniejsze obliczenia pokazują, że $\log_\phi 3 \approx 2{,}283$.

Bezpośrednia implementacja tego rozwiązania (zawarta w plikach `rozs1.cpp` i `rozs2.cpp`) pozwalała uzyskać 12%–24% maksymalnej punktacji.

Usprawnienia

Podane rozwiązanie można znacząco usprawnić, używając pewnych klasycznych technik algorytmicznych. Techniki te podajemy raczej w charakterze dygresji, gdyż nie były one potrzebne w rozwiązaniu wzorcowym.

Jednym z pomysłów jest zastosowanie metody *meet in the middle*. W naszym przypadku polega ona na tym, aby zbiór indeksów dostępnych liczb Fibonacciego, tj. $\{2, 3, \ldots, m+1\}$, podzielić na dwa rozłączne zbiory o rozmiarze jak najbliższym $m/2$. Dla każdego z tych zbiorów możemy wygenerować $O(3^{m/2})$ możliwych kombinacji wyborów odpowiednich liczb, tworząc listę par uporządkowanych (s, r), gdzie s jest sumą, którą można uzyskać, dodając/odejmując pewne r różnych liczb Fibonacciego z danego zbioru. Możemy oczywiście dla każdego s zapamiętać za każdym razem tylko jedną parę — tę z minimalną wartością parametru r. Teraz wystarczy złożyć w całość wyniki dla obu zbiorów. W tym celu można posortować pary uzyskane dla każdego ze zbiorów i dla każdej pary (s_1, r_1) z pierwszego zbioru próbować wyszukać odpowiadającą jej parę postaci $(k - s_1, r_2)$ z drugiego zbioru. W ten sposób otrzymujemy rozkład zawierający $r_1 + r_2$ liczb Fibonacciego. Poszukiwanie pary $(k - s_1, r_2)$ można, na przykład, zrealizować wyszukiwaniem binarnym, co prowadzi do rozwiązania o złożoności czasowej $O(3^{m/2} m/2) = O(k^{(\log_\phi 3)/2} \log k)$, czyli $O(k^{1{,}142} \log k)$. Alternatywnie, odpowiadające sobie wzajemnie pary można zidentyfikować w jednym sprytnym przejściu po dwóch posortowanych listach. Takie rozwiązanie zostało zaimplementowane w pliku `rozs3.cpp`. Uzyskiwało ono 36% maksymalnej punktacji.

Inne podejście polega na dostrzeżeniu analogii między naszym problemem a problemem wydawania reszty. Liczba k jest tu kwotą do wydania, a liczby Fibonacciego (oraz liczby przeciwne do nich) są nominałami, którymi dysponujemy. Chcemy wydać k, używając jak najmniej spośród $2m$ dostępnych nominałów, których suma wartości bezwzględnych jest rzędu $O(k)$. Prosty algorytm oparty na programowaniu dynamicznym rozwiązuje ten problem w czasie $O(km)$, czyli, w naszym przypadku, $O(k \log k)$. Umiejętna implementacja tego algorytmu (patrz np. plik `rozs4.cpp`) pozwalała uzyskać 48% punktów.

Rozwiązanie wzorcowe

Z pomocą różnych sztuczek algorytmicznych udało nam się otrzymać rozwiązania działające w czasie zbliżonym do liniowego względem rozkładanej liczby k. Aby uzyskać coś istotnie szybszego, o czasie działania logarytmicznym (bądź polilogarytmicznym) względem k, potrzebujemy jeszcze kilku spostrzeżeń. Zacznijmy od następującej, bardzo prostej obserwacji, która posłuży nam do dowodu ważnej własności rozkładów optymalnych (twierdzenie 1).

Obserwacja 4. Niech (P_+, P_-) będzie rozkładem liczby k. Jeśli dla pewnego i zachodzi:

(a) $i \in P_+$ oraz $i+1 \in P_+$ lub

(b) $i \in P_-$ oraz $i+1 \in P_-$ lub

(c) $i \in P_-$ oraz $i+1 \in P_+$ lub

(d) $i \in P_-$ oraz $i+2 \in P_+$

to rozkład (P_+, P_-) nie jest rozkładem optymalnym.

Twierdzenie 1. *Jeśli $Fib_m \leqslant k < Fib_{m+1}$, to istnieje optymalny rozkład liczby k, w którym $m \in P_+$ lub $m+1 \in P_+$.*

Dowód: Dla $k \leqslant 2$ teza twierdzenia w oczywisty sposób zachodzi. Załóżmy zatem, że twierdzenie nie jest prawdziwe dla pewnej liczby $k \geqslant 3$, i rozważmy jakikolwiek rozkład optymalny (P_+, P_-) liczby k (w którym P_+ i P_- są zbiorami). Zapytajmy, ile wynosi $z = \max P_+$. Wiemy, że $z \neq m$ i $z \neq m+1$.

Załóżmy, że $z \leqslant m-1$. Na mocy obserwacji 4a, największą sumą rozkładu, jaką możemy wówczas uzyskać, jest:

$$Fib_{m-1} + Fib_{m-3} + Fib_{m-5} + \ldots$$

Powyższa suma kończy się składnikiem Fib_2 lub Fib_3, w zależności od parzystości m, i jest równa $Fib_m - Fib_1$ lub, odpowiednio, $Fib_m - Fib_2$ (czyli w obu przypadkach tyle samo). Ponieważ $k \geqslant Fib_m$, więc ten przypadek nie może zachodzić.

Załóżmy teraz, że $z \geqslant m+2$. Podobnie jak poprzednio, rozkład optymalny o najmniejszej sumie, jaki można uzyskać przy tym założeniu, to:

$$Fib_{m+2} - Fib_{m-1} - Fib_{m-3} - \ldots > Fib_{m+1}.$$

Tym razem skorzystaliśmy z obserwacji 4bcd. Ponieważ $k < Fib_{m+1}$, więc ten przypadek również nie może zachodzić.

Wykazaliśmy zatem, że żaden z przypadków $z \notin \{m, m+1\}$ nie jest możliwy. Otrzymana sprzeczność dowodzi tezy twierdzenia. ■

Powyższe twierdzenie istotnie zawęża klasę rozkładów, jakie musimy rozważać. Ale nie tylko. Jeśli założenia twierdzenia są spełnione, wówczas każda z liczb $k - Fib_m$, $Fib_{m+1} - k$ jest nie większa niż Fib_{m-1} (ponieważ $Fib_{m+1} - Fib_m = Fib_{m-1}$), więc twierdzenie natychmiast implikuje podaną wcześniej bez dowodu obserwację 3. Z tego samego spostrzeżenia wynika również algorytm o złożoności $O(2^{m/2})$, w którym w każdym kroku rozważamy dwie możliwości: albo do rozkładu wybieramy Fib_m, albo Fib_{m+1}. Tego typu rozwiązania (patrz plik `rozs5.cpp`) uzyskiwały na zawodach 87%–100% punktów.

Twierdzenie 1 wciąż dopuszcza pewną niejednoznaczność. Okazuje się, że w zależności od tego, która z liczb Fib_m, Fib_{m+1} jest bliższa k, możemy dokładnie określić, którą z liczb m, $m+1$ warto umieścić w zbiorze P_+. Prawdziwe są bowiem dwa kolejne, „siostrzane” twierdzenia.

Twierdzenie 2. *Załóżmy, że $Fib_m \leqslant k < Fib_{m+1}$. Jeśli $k - Fib_m \leqslant Fib_{m+1} - k$, to istnieje rozkład optymalny, w którym $m \in P_+$.*

Dowód: W dowodzie wystarczy ograniczyć się do przypadku, gdy $k \geqslant 3$. Oznaczmy $a = k - Fib_m$, $b = Fib_{m+1} - k$. Zauważmy, że jeśli w rozkładzie liczby k użyjemy składnika Fib_m, to pozostanie nam do rozłożenia liczba a, a jeśli użyjemy Fib_{m+1}, pozostanie nam do rozłożenia b (lub $-b$, ale to na to samo wychodzi). Należy więc wykazać, że w przypadku, gdy $a \leqslant b$, liczba składników w optymalnym rozkładzie liczby a jest nie większa niż liczba składników niezbędnych do rozłożenia liczby b. Zauważmy, że $a + b = Fib_{m-1}$. Rozważmy dwa przypadki:

- $a < Fib_{m-3}$, wtedy $b > Fib_{m-2}$. Na mocy twierdzenia 1, próbując rozłożyć liczbę b, powinniśmy użyć składnika Fib_{m-1} lub Fib_{m-2}. W pierwszym z tych przypadków pozostajemy z liczbą a do rozłożenia (wykorzystawszy dwa składniki, co jest nieopłacalne, bowiem do takiej samej sytuacji mogliśmy dojść w jednym ruchu). Jeśli zaś użyjemy składnika Fib_{m-2}, będziemy mieć dalej do rozłożenia liczbę $b - Fib_{m-2}$, jednak zauważmy, że $b - Fib_{m-2} = Fib_{m-1} - a - Fib_{m-2} = Fib_{m-3} - a$, a taki ruch mielibyśmy do wykonania także z a. W obu przypadkach nie opłaca nam się rozkładać liczby b.
- $a \geqslant Fib_{m-3}$, wtedy $Fib_{m-3} \leqslant a \leqslant b \leqslant Fib_{m-2}$. Na mocy twierdzenia 1 mamy do rozważenia dwie możliwości: następnym składnikiem rozkładu (zarówno w przypadku, gdy próbujemy dalej rozkładać a, jak i b) może być albo Fib_{m-3}, albo Fib_{m-2}. Jednakże $a - Fib_{m-3} = Fib_{m-2} - b$ oraz $Fib_{m-2} - a = b - Fib_{m-3}$. Stąd, zarówno z a, jak i z b mamy takie same możliwości dalszego rozkładu.

Ostateczna konkluzja jest następująca: rozłożenie liczby a będzie wymagało zawsze nie więcej operacji niż rozłożenie b. Można zatem bezpiecznie dodać m do zbioru P_+ i rozłożyć optymalnie to, co pozostało. ■

Twierdzenie 3. *Załóżmy, że $Fib_m \leqslant k < Fib_{m+1}$. Jeśli $k - Fib_m > Fib_{m+1} - k$, to istnieje rozkład optymalny, w którym $m + 1 \in P_+$.*

Dowód: Analogiczny do dowodu twierdzenia 2. ■

Dwa ostatnie twierdzenia podpowiadają dla każdego k zachłanny ruch, jaki należy wykonać. Jeśli przed wykonaniem tego ruchu zachodzi $Fib_m \leqslant k < Fib_{m+1}$ (i, powiedzmy, $m > 4$), to pozostała do rozłożenia liczba k' po wykonaniu ruchu nie przekracza $\max(k - Fib_m, Fib_{m+1} - k)$, czyli:

$$k' \leqslant \frac{Fib_{m+1} - Fib_m}{2} = \frac{Fib_{m-1}}{2} = \frac{Fib_{m-2} + Fib_{m-3}}{2} < \frac{2Fib_{m-2}}{2} = Fib_{m-2}.$$

To pokazuje, że po mniej więcej $m/3$ ruchach otrzymamy pełny rozkład liczby k.

W zależności od tego, jak efektywnie zaimplementujemy wykonywanie pojedynczego ruchu, otrzymamy rozwiązanie o złożoności czasowej $O(m^2) = O(\log^2 k)$ lub $O(m) = O(\log k)$. Każde z tych rozwiązań uzyskuje, oczywiście, maksymalną punktację. Stosowne implementacje można znaleźć w plikach `roz.cpp`, `roz1.pas`, `roz2.cpp` i `roz3.pas`. Poniżej znajduje się pseudokod rozwiązania działającego w czasie $O(\log k)$.

```
function rozkład(k)
begin
  x := 1;  y := 1;
  while (y < k) do begin
    z := x + y;
    x := y;  y := z;
  end
  r := 0;
  while (k > 0) do begin
    { Niezmiennik: x ⩽ k < y, x i y to dwie kolejne liczby Fibonacciego }
    if (k − x ⩽ y − k) then k := k − x else k := y − k;
    r := r + 1;
    while (x ⩾ k) do begin
      z := y − x;
      y := x;  x := z;
    end
  end
  return r;
end
```

Testy

Rozwiązania zawodników były sprawdzane za pomocą 8 testów. Poniższa tabela zawiera charakterystykę poszczególnych testów: p oznacza liczbę zapytań w danym teście, a K — maksymalną liczbę występującą w zapytaniu.

Nazwa	p	K
roz1.in	10	75
roz2.in	10	500
roz3.in	6	34 130
roz4.in	7	800 000

Nazwa	p	K
roz5.in	10	1 000 000 000
roz6.in	10	1 000 000 000 000
roz7.in	10	100 000 000 000 000
roz8.in	10	400 000 000 000 000 000

Zawody III stopnia

opracowania zadań