Jakub Radoszewski
Treść zadania

Jakub Pawlewicz, Jakub Radoszewski
Opracowanie, Program

OI, Etap I

Zapytania

Kryptograf Bajtazar pracuje nad złamaniem szyfru ABB (Agencji Bezpieczeństwa Bajtocji). Doszedł już do tego, że przy odszyfrowywaniu wiadomości będzie musiał wielokrotnie odpowiadać na zapytania postaci: „dla danych liczb naturalnych a, b i d, ile jest takich par liczb naturalnych (x,y), że:

- $1 \leqslant x \leqslant a$,
- $1 \leqslant y \leqslant b$,
- $NWD(x,y) = d$, *gdzie* $NWD(x,y)$ *to największy wspólny dzielnik liczb* x *i* y*".*

Bajtazar chciałby zautomatyzować swoją pracę, więc poprosił Cię o pomoc.

Zadanie

Napisz program, który:

- *wyczyta ze standardowego wejścia listę zapytań, na które musi odpowiedzieć Bajtazar,*
- *wyznaczy odpowiedzi na te zapytania,*
- *wypisze wynik na standardowe wyjście.*

Wejście

Pierwszy wiersz wejścia zawiera jedną dodatnią liczbę całkowitą n ($1 \leqslant n \leqslant 50\,000$), oznaczającą liczbę zapytań Bajtazara. Każdy z kolejnych n wierszy zawiera po trzy liczby całkowite a, b i d ($1 \leqslant d \leqslant a,b \leqslant 50\,000$), pooddzielane pojedynczymi odstępami. Każda taka trójka reprezentuje jedno zapytanie.

Wyjście

Twój program powinien wypisać n wierszy. Wiersz i powinien zawierać jedną liczbę całkowitą: odpowiedź na i-te zapytanie z wejścia.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*

`2`
`4 5 2`
`6 4 3`
	`3`
	`2`

Pary uzyskane w pierwszym zapytaniu to: $(2,2)$, $(2,4)$ i $(4,2)$, a w drugim: $(6,3)$ i $(3,3)$.

Rozwiązanie

Wprowadzenie

Oznaczmy przez $Z_d(a,b)$ odpowiedź na zapytanie (a,b,d), czyli liczbę takich par liczb całkowitych (x,y), dla których $1 \leqslant x \leqslant a$, $1 \leqslant y \leqslant b$ i $\mathrm{NWD}(x,y) = d$. Okazuje się, że stosunkowo łatwo można wyeliminować z zapytań parametr d:

Fakt 1

$$Z_d(a,b) = Z_1\left(\left\lfloor \frac{a}{d} \right\rfloor, \left\lfloor \frac{b}{d} \right\rfloor\right). \tag{1}$$

Dowód Wartość $Z_d(a,b)$ jest równa liczbie elementów zbioru $\{(x,y) : 1 \leqslant x \leqslant a, 1 \leqslant y \leqslant b, \mathrm{NWD}(x,y) = d\}$. Dowolną parę (x,y) z tego zbioru możemy przedstawić w postaci $(x'd, y'd)$, gdzie $\mathrm{NWD}(x',y') = 1$ oraz, co więcej, zachodzą ograniczenia: $1 \leqslant x' \leqslant \lfloor \frac{a}{d} \rfloor$ i $1 \leqslant y' \leqslant \lfloor \frac{b}{d} \rfloor$. Ponieważ przekształcenie (x,y) w (x',y') jest różnowartościowe, to dowodzi to, że $Z_d(a,b) \leqslant Z_1(\lfloor a/d \rfloor, \lfloor b/d \rfloor)$.

Możemy też spojrzeć na sytuację z drugiej strony: z każdej pary (x',y') spełniającej nierówności $1 \leqslant x' \leqslant \lfloor \frac{a}{d} \rfloor$, $1 \leqslant y' \leqslant \lfloor \frac{b}{d} \rfloor$ oraz warunek $\mathrm{NWD}(x',y') = 1$, możemy utworzyć parę $x = x'd$, $y = y'd$ spełniającą $1 \leqslant x \leqslant a$, $1 \leqslant y \leqslant b$ i $\mathrm{NWD}(x,y) = d$. Podobnie jak poprzednio, przekształcenie jest różnowartościowe, więc widzimy, że $Z_1(\lfloor a/d \rfloor, \lfloor b/d \rfloor) \leqslant Z_d(a,b)$. ■

Będziemy od tej pory zakładać, że w każdym zapytaniu parametr d jest równy 1 i dla uproszczenia przyjmiemy zapis:

$$Z(a,b) = Z_1(a,b). \tag{2}$$

Niech dalej $Q \subseteq \mathbb{N} \times \mathbb{N}$ oznacza zbiór wszystkich zapytań, na jakie mamy odpowiedzieć. Rozmiar $|Q|$ tego zbioru oznaczmy zgodnie z treścią zadania przez n. Przez m oznaczmy maksimum liczb, które występują w zapytaniach z Q.

Rozwiązanie wzorcowe

W zadaniu musimy odpowiedzieć na wiele pytań $Z(a,b)$. Sprawdźmy, czy nie da się wykorzystać obliczonych wcześniej odpowiedzi, do wyznaczania odpowiedzi na kolejne pytania. Na przykład, zastanówmy się, czy na podstawie $Z(a,b)$ możemy szybko obliczyć $Z(a,b+1)$ lub $Z(a+1,b)$. Dokładniej, ponieważ $Z(a,b) = Z(b,a)$ i różnica między $Z(a,b+1)$ a $Z(a,b)$ jest dokładnie taka, jak między $Z(b+1,a)$ a $Z(b,a)$, to możemy skoncentrować się jedynie na poszukiwaniu sposobu wyznaczania różnic typu $Z(a,b+1) - Z(a,b)$.

Definicja 1 Przez $C(a,b)$ oznaczać będziemy liczbę takich wartości całkowitych $1 \leqslant x \leqslant a$, dla których $\mathrm{NWD}(x,b) = 1$.

Zauważmy, że $Z(a,b+1) - Z(a,b) = C(a,b+1)$. Efektywny sposób wyznaczania wartości $C(a,b)$ daje z kolei następujący fakt:

Fakt 2 *Niech $b = p^k q$, $b > 1$, gdzie p jest liczbą pierwszą i p nie dzieli q, a k jest liczbą całkowitą dodatnią. Wówczas*

$$C(a,b) = C(a,q) - C\Big(\Big\lfloor \frac{a}{p} \Big\rfloor, q\Big). \tag{3}$$

Dowód Jeżeli liczba naturalna x, $1 \leqslant x \leqslant a$, jest względnie pierwsza z b, to jest ona również względnie pierwsza z q, a zatem $C(a,b) \leqslant C(a,q)$. Ile dokładnie wynosi różnica $C(a,q) - C(a,b)$? Jest ona równa liczbie tych $1 \leqslant x \leqslant a$, które są względnie pierwsze z q, ale dzielą się przez p, czyli liczbie elementów zbioru $\big\{p \cdot 1, p \cdot 2, \ldots, p \cdot \big\lfloor \frac{a}{p} \big\rfloor\big\}$ względnie pierwszych z q. Ponieważ $\mathrm{NWD}(p,q) = 1$, to $C(a,q) - C(a,b)$ jest równa liczbie elementów zbioru $\big\{1, 2, \ldots, \big\lfloor \frac{a}{p} \big\rfloor\big\}$ względnie pierwszych z q, czyli $C\big(\big\lfloor \frac{a}{p} \big\rfloor, q\big)$. ■

Korzystając z Faktu 2, możemy skonstruować rekurencyjny algorytm wyznaczania $C(a,b)$. Dla parametru $b > 1$ stosujemy w nim wzór (3); dla parametru $b = 1$, mamy $C(x,1) = x$. Zastanówmy się, jaką złożoność czasową ma ten algorytm.

Definicja 2 Niech $\omega(b)$ oznacza liczbę różnych dzielników pierwszych b.

W każdym z wywołań rekurencyjnych, zgodnie ze wzorem (3), b zostaje zastąpione przez q, które ma o jeden dzielnik pierwszy mniej ($\omega(q) = \omega(b) - 1$). Na głębokości rekursji równej $\omega(b)$ otrzymamy zatem $2^{\omega(b)}$ składników postaci $\pm C(x,1)$, jako że każdy poziom rekursji jest dwukrotnie liczniejszy od poprzedniego. Ponieważ łączna liczba wywołań rekurencyjnych w algorytmie jest równa $2^0 + 2^1 + \cdots + 2^{\omega(b)} = 2^{\omega(b)+1} - 1$, więc koszt czasowy wyznaczania $C(a,b)$ wynosi $O(2^{\omega(b)})$. Jedyny problem, jaki dotychczas przemilczeliśmy, to sposób znajdowania dzielnika pierwszego liczby b (wraz z krotnością jego występowania w rozkładzie b na czynniki pierwsze). Rozwiązanie tej kwestii stanowi *sito Eratostenesa*.

Sito Eratostenesa

Sito Eratostenesa to powszechnie znany algorytm, który pozwala efektywnie wyznaczyć wszystkie liczby pierwsze w przedziale $[1,m]$. Przedstawimy jedną z jego implementacji.

W algorytmie będziemy wyznaczać tablicę wartości logicznych *pierwsza*$[2..m]$. Na początku wypełniamy ją w całości wartościami *true* (prawda). Następnie analizujemy kolejno liczby $i = 2, \ldots, m$ i dla każdej z nich, „wykreślamy" z tablicy jej wielokrotności. Wykreślanie można rozpocząć od i^2, gdyż, jak łatwo zauważyć, wszystkie mniejsze wielokrotności i musiały już wcześniej zostać wykreślone. Oto pseudokod sita Eratostenesa:

```
1: for i := 2 to m do pierwsza[i] := true;
2: for i := 2 to m do
3:   if pierwsza[i] then
4:   begin
5:     j := i^2;
6:     while j ⩽ N do
7:     begin
8:       pierwsza[j] := false;
9:       j := j + i;
```

```
10:       end
11:    end
```

Sito Eratostenesa jest algorytmem prostym i krótkim w implementacji, a dodatkowo — efektywnym. Można pokazać, że przedstawiona procedura ma złożoność czasową $O(m \log\log m)$. Dowód tego oszacowania nie jest jednak łatwy, a nam wystarczy, gdy wykażemy, że algorytm działa w czasie $O(m \log m)$ — co jest znacznie prostsze. Zauważmy, że pole j tablicy *pierwsza* odwiedzamy w instrukcji w wierszu 8 co najwyżej tyle razy, ile różnych dzielników pierwszych ma j. Liczbę tych dzielników można ograniczyć przez $O(\log m)$, więc liczbę wszystkich operacji można oszacować przez $O(m \log m)$.

Przedstawiony algorytm można uzupełnić tak, by pozwalał znajdować rozkład na czynniki pierwsze liczb z przedziału $[1,m]$. Wykorzystamy do tego celu dodatkową tablicę *dzielnik*, w której dla każdego elementu zapiszemy jego dzielnik, który znajdziemy jako pierwszy. Na początku algorytmu wypełnimy wszystkie komórki tablicy *dzielnik* wartościami -1, natomiast w momencie modyfikacji tablicy *pierwsza* (wiersz 8) ustawiamy $dzielnik[j] := i$, o ile tylko element $dzielnik[j]$ nie było jeszcze ustawiony na dodatnią wartość. W ten sposób dla każdej liczby złożonej w odpowiedniej komórce tablicy *dzielnik* otrzymamy jej *najmniejszy* dzielnik pierwszy. Zauważmy, że opisana modyfikacja nie powoduje wzrostu złożoności czasowej całego algorytmu.

```
1:  for i := 2 to m do dzielnik[i] := -1;
2:  for i := 2 to m do pierwsza[i] := true;
3:  for i := 2 to m do
4:     if pierwsza[i] then
5:     begin
6:        j := i^2;
7:        while j <= N do
8:        begin
9:           pierwsza[j] := false;
10:          if dzielnik[j] < 0 then dzielnik[j] := i;
11:          j := j + i;
12:       end
13:    end
```

Zawartość tablicy *dzielnik* możemy wykorzystać do rozkładu liczby na czynniki pierwsze. Dla liczby j jej pierwszy dzielnik odczytujemy z pola $dzielnik[j]$, po czym przechodzimy do poszukiwania rozkładu liczby $j/dzielnik[j]$ itd. Dla dowolnej liczby spośród $2,\ldots,m$ złożoność czasowa tego procesu to $O(\log m)$.

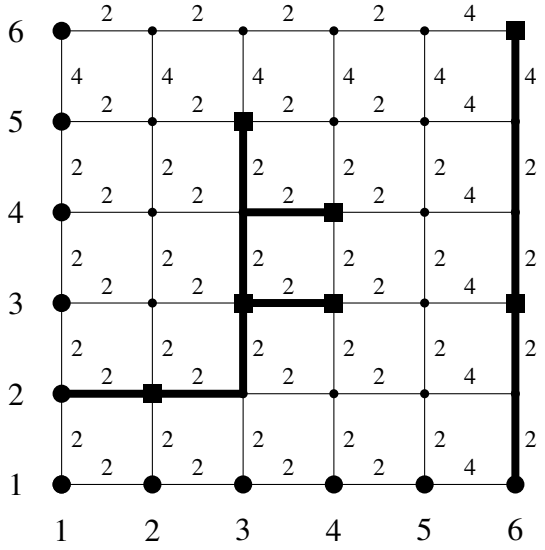
Interpretacje problemu

Potrafimy już efektywnie wyznaczać wartości $C(a,b)$. Spróbujmy zastosować tę umiejętność do znalezienia wartości $Z(a,b)$ dla wszystkich pytań ze zbioru Q. W tym celu przedstawimy dwie interpretacje zagadnienia: geometryczną i grafową — obie przydadzą się nam w dalszym toku opisu.

Krata. Zapytania możemy wyobrazić sobie jako punkty kraty $[1,m] \times [1,m]$ (o całkowitych współrzędnych) na płaszczyźnie. Wiemy, że znając wartość funkcji Z dla punktu $(a,b-1)$, umiemy policzyć tę wartość dla punktu (a,b) w czasie $O(2^{\omega(b)})$. W takim samym czasie umiemy na podstawie $Z(b-1,a)$ wyznaczyć $Z(b,a)$.

Graf. Problem możemy przedstawić także jako graf nieskierowany G, którego wierzchołki odpowiadają punktom kraty, a krawędzie łączą wszystkie pary wierzchołków, które odpowiadają punktom sąsiadującym w kracie (czyli różniącym się na jednej współrzędnej o 1). Krawędziom łączącym wierzchołki $(a,b-1)$ oraz (a,b) przypisujemy wagę $2^{\omega(b)}$. Taką samą wagę przypisujemy krawędziom $((b,a),(b-1,a))$.

Wartościami funkcji Z, które potrafimy wyznaczyć natychmiast, są wartości w punktach ze zbioru $X = \{(x,1) \mid 1 \leqslant x \leqslant m\} \cup \{(1,x) \mid 1 \leqslant x \leqslant m\}$ — dla każdej liczby naturalnej x zachodzi $Z(x,1) = Z(1,x) = x$. W takim razie problem wyznaczenia wszystkich potrzebnych nam wartości funkcji Z możemy sprowadzić do problemu konstrukcji najlżejszego lasu rozpinającego w G, łączącego wszystkie punkty ze zbioru Q z punktami ze zbioru X. Na rys. 1 jest przedstawione najlżejsze drzewo rozpinające dla przykładowego zbioru Q w przykładowym grafie G.



Rys. 1: Grafowa interpretacja problemu dla $m = 6$. Przy krawędziach są zapisane wagi, równe 2 lub 4 (gdyż $\omega(2) = \omega(3) = \omega(4) = \omega(5) = 1$, a $\omega(6) = 2$). Dużymi kółkami oznaczono punkty ze zbioru X, natomiast kwadraty reprezentują punkty zbioru $Q = \{(2,2),(3,3),(3,5),(4,3),(4,4),(6,3),(6,6)\}$. Pogrubionymi kreskami wyróżniono krawędzie najlżejszego drzewa rozpinającego (dowolnie wybranego, bo istnieje ich więcej). Ma ono wagę 26.

Powstaje pytanie, w jaki sposób wyznaczyć taki las. Niestety, graf G ma $\Theta(m^2)$ wierzchołków oraz krawędzi, co nie nastraja optymistycznie (oczywiście nie oznacza to, że problemu nie da się rozwiązać, tylko nie widać sposobu opartego na klasycznych technikach). W rozwiązaniu wzorcowym zadowolimy się więc algorytmem pozwalającym znaleźć potencjalnie trochę gorszy sposób połączenia punktów ze zbioru Q z punktami ze zbioru X.

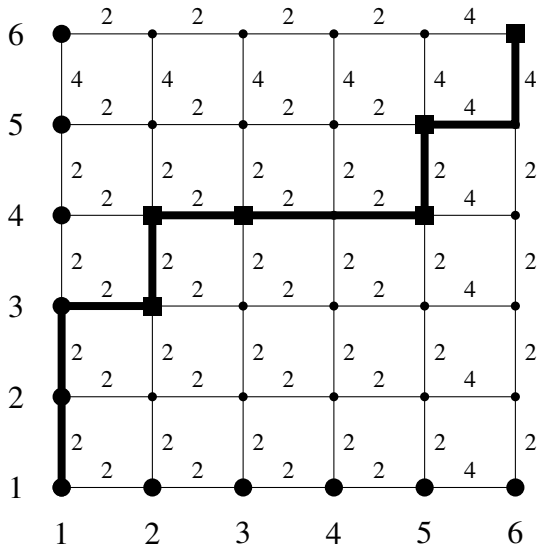
Rozwiązanie części problemu

Skoro zrezygnowaliśmy z konstrukcji optymalnego lasu połączeń, spróbujmy wybrać pewną podgrupę powiązanych ze sobą zapytań, na które będziemy w stanie stosunkowo szybko odpowiedzieć. Wybierzmy ze zbioru Q taki ciąg k punktów $(x_1,y_1),\ldots,(x_k,y_k)$, że ciągi $x_1,\ldots,x_k$ i $y_1,\ldots,y_k$ są monotoniczne. Możemy to zrobić następująco:

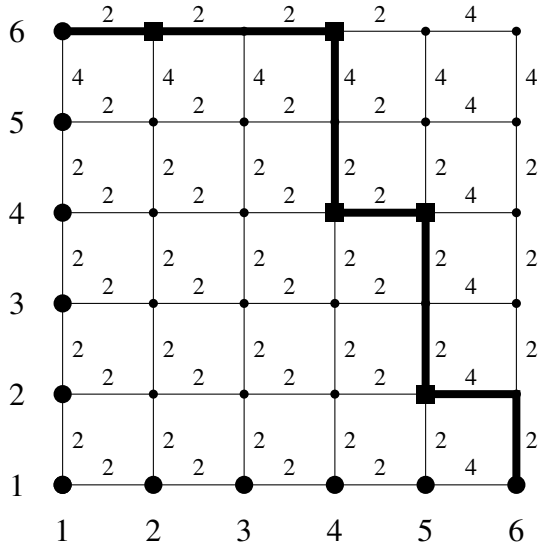
- sortujemy punkty ze zbioru Q niemalejąco po pierwszej współrzędnej (na przykład metodą sortowania przez scalanie w czasie $O(n\log n)$);
- w ciągu drugich współrzędnych znajdujemy najdłuższy podciąg niemalejący i najdłuższy podciąg nierosnący (potrafimy to zrobić w czasie $O(n\log n)$, patrz, na przykład, opracowanie zadania *Egzamin na Prawo Jazdy* w niniejszej książeczce);
- wybieramy punkty, których drugie współrzędne tworzą dłuższy z powyższych podciągów (w czasie $O(n)$).

Wartości Z dla wszystkich punktów $(x_1,y_1),\ldots,(x_k,y_k)$ możemy wyznaczyć, przemieszczając się w grafie po ścieżce długości $2m-2$:

- jeżeli ciąg y_i jest niemalejący, to jest to ścieżka od punktu $(1,1)$ do (m,m) (patrz rys. 2);
- jeżeli zaś ciąg y_i jest nierosnący, to ścieżka prowadzi od punktu $(1,m)$ do $(m,1)$ (patrz rys. 3).



Rys. 2: Ścieżka biegnąca z punktu $(1,1)$ do (m,m) przez punkty $(2,3)$, $(2,4)$, $(3,4)$, $(5,4)$, $(5,5)$, $(6,6)$; w tym przykładzie mamy $m=6$ oraz $k=6$.



Rys. 3: Ścieżka biegnąca z punktu $(1,m)$ do $(m,1)$ przez punkty $(2,6)$, $(4,6)$, $(4,4)$, $(5,4)$, $(5,2)$; w tym przykładzie mamy $m=6$ oraz $k=5$.

W obu przypadkach ścieżka zawiera dokładnie $m-1$ krawędzi poziomych oraz $m-1$ krawędzi pionowych. Suma wag zarówno poziomych, jak i pionowych krawędzi, wyraża się wzorem $\sum_{i=2}^{m} 2^{\omega(i)}$. Okazuje się, że możemy oszacować rząd wielkości tej sumy[1]:

$$\sum_{i=2}^{m} 2^{\omega(i)} = \frac{6m\ln m}{\pi^2} + O(m) = \theta(m\log m). \tag{4}$$

To oznacza, że obliczenie wszystkich wartości $C(a,b)$ odpowiadających krawędziom ścieżki oraz wartości $Z(a,b)$ odpowiadających punktom ścieżki, można wykonać w czasie $O(m\log m)$.

Udało się nam wybrać grupę k zapytań i odpowiedzieć na nie w czasie $O(n\log n + m\log m)$. Możemy więc usunąć je ze zbioru Q i kontynuować takie samo postępowanie, aż do momentu znalezienia odpowiedzi na wszystkie zapytania z Q. Pozostaje pytanie, ile tego typu rund musimy wykonać, aby rozwiązać zadanie.

Ile części ma problem?

Wykażmy poniższy fakt, który pomoże nam znaleźć odpowiedź na nurtujące nas pytanie:

Fakt 3 *Niech dany będzie ciąg liczb naturalnych $a=(a_1,\ldots,a_{k^2+1})$ długości k^2+1. W ciągu tym istnieje podciąg niemalejący długości $k+1$ lub podciąg nierosnący długości $k+1$.*

Dowód Uzasadnienie faktu, wykorzystujące klasyczne twierdzenie Dilwortha, można znaleźć w książce [34]. Tutaj przedstawimy inny dowód, oparty jedynie na prostych sztuczkach kombinatorycznych.

[1] Źródło: `http://www.research.att.com/~njas/sequences/A064608`, *The On-Line Encyclopedia of Integer Sequences*, A064608

Zdefiniujmy ciąg $b_1, \ldots, b_{k^2+1}$, gdzie b_i jest długością najdłuższego podciągu niemalejącego ciągu a kończącego się elementem a_i. Następnie rozważmy dwa przypadki:

(1) Jeżeli w ciągu b istnieje wyraz większy niż k, to znaczy, że w ciągu a istnieje podciąg niemalejący długości $k+1$, co kończy dowód w tym przypadku.

(2) Niech wszystkie wyrazy ciągu b będą mniejsze niż $k+1$, czyli w ciągu b występuje najwyżej k różnych wartości. Wówczas, na mocy zasady szufladkowej Dirichleta, wiemy, że istnieje wartość (oznaczmy ją x) występująca w tym ciągu $k+1$ razy. Pokażmy, że elementy ciągu a, którym w ciągu b odpowiadają (czyli występują na tych samych pozycjach) wartości x, tworzą ciąg malejący. Załóżmy, że $b_i = b_j = x$ dla $i < j$. Gdyby $a_i \leqslant a_j$, to moglibyśmy ciąg kończący się na a_i o długości x przedłużyć o a_j. Tym samym $b_j \geqslant x+1$, co jest sprzeczne z założeniem $b_i = b_j$.

■

Z Faktu 3 widać natychmiast, że jeżeli zbiór Q ma n elementów, to długość k skonstruowanego w jednej rundzie algorytmu ciągu zapytań jest nie mniejsza niż $\lceil\sqrt{n}\rceil$. Na pierwszy rzut oka może się więc wydawać, że po $O(\sqrt{n})$ rundach rozwiążemy całe zadanie. Nie jest to jednak oczywiste, ponieważ w kolejnych rundach rozmiar $|Q|$ zmniejsza się i znajdujemy odpowiedzi na mniej niż $\lceil\sqrt{n}\rceil$ pytań. Niemniej jednak pierwsze wrażenie okazuje się słuszne, co możemy udowodnić.

Fakt 4 *Liczba rund jest rzędu* $O(\sqrt{n})$.

Dowód Podzielmy zbiór liczb naturalnych na parami rozłączne przedziały $[i^2+1,(i+1)^2]$, dla $i \geqslant 1$. Rozważmy liczbę naturalną $n \in [i^2+1,(i+1)^2]$. Pokażmy, że jeśli byłaby ona długością ciągu a w naszym problemie, to po co najwyżej 3 rundach algorytmu długość ciągu znajdzie się w przedziale $[(i-1)^2+1,i^2]$. Wynika to stąd, że wartość $n > i^2$ musi na mocy Faktu 3 w jednej rundzie zmaleć co najmniej o i. Po maksymalnie 3 takich redukcjach, długość ciągu osiągnie wartość $n-3i \leqslant (i+1)^2-3i = i^2+2i-1-3i \leqslant i^2$, czyli spadnie do przedziału $[(i-1)^2+1,i^2]$, o ile nie znalazła się tam wcześniej.

Ponieważ wartość i jest równa $\lceil\sqrt{n}\rceil - 1$, to po co najwyżej $3(\lceil\sqrt{n}\rceil - 1) = O(\sqrt{n})$ rundach cały zbiór Q zostanie przetworzony. ■

Ostatecznie pokazaliśmy, że złożoność czasowa zaproponowanego algorytmu to

$$O\big(\sqrt{n}(n\log n + m\log m)\big).$$

Możemy się jeszcze zastanowić, ile tracimy w porównaniu z metodą opartą na wyznaczaniu najlżejszego lasu rozpinającego w grafie G. Okazuje się, że w pesymistycznym przypadku waga takiego lasu może wynosić

$$\Omega\Big(\sqrt{n}\sum_{i=2}^{m} 2^{\omega(i)}\Big),$$

gdzie — jak już wspominaliśmy — sumę występującą w tym wyrażeniu można oszacować wyrażeniem:

$$\sum_{i=2}^{m} 2^{\omega(i)} = \theta(m\log m).$$

Pesymistyczny przypadek występuje, na przykład, dla punktów Q równomiernie rozłożonych na kracie:

$$Q = \left\{ \left(\left\lfloor \frac{im}{\sqrt{n}} \right\rfloor, \left\lfloor \frac{jm}{\sqrt{n}} \right\rfloor \right) \mid i, j = 1, \ldots, \lfloor \sqrt{n} \rfloor \right\}.$$

Zatem w najgorszym razie, nawet gdybyśmy pominęli koszt znajdowania najlżejszego lasu rozpinającego w G, to i tak na wszystkie pozostałe obliczenia musielibyśmy poświęcić czas $\Omega(\sqrt{nm}\log m)$. Widzimy więc, że zastępując najlżejsze drzewo rozpinające układem ścieżek o monotonicznie posortowanych współrzędnych punktów, nie pogarszamy złożoności obliczeniowej rozwiązania.

Implementacja opisanego w tym rozdziale algorytmu znajduje się w plikach `zap.cpp` i `zap0.pas`.

Rozwiązanie alternatywne

W rozwiązaniu wzorcowym w istotny sposób korzystaliśmy z tego, że w zadaniu mamy udzielić odpowiedzi na dużą liczbę zapytań. Okazuje się jednak, że istnieje rozwiązanie, które pozwala na efektywne odpowiadanie także na pojedyncze zapytania. Pomysł ten został zaimplementowany przez zdecydowaną większość zawodników, którzy zdobyli za zadanie maksymalną liczbę punktów. Ponieważ w rozwiązaniu tym są wykorzystywane zaawansowane techniki matematyczne, więc prezentujemy je jako drugie w kolejności.

Rozważmy wszystkie pary liczb naturalnych, z których pierwsza należy do zbioru $\{1,\ldots,a\}$, a druga — do zbioru $\{1,\ldots,b\}$. Tych par jest oczywiście $a \cdot b$. Możemy je podzielić na rozłączne zbiory ze względu na wartość $NWD(a,b)$, czyli:

$$a \cdot b = Z_1(a,b) + Z_2(a,b) + Z_3(a,b) + \ldots \tag{5}$$

Zdefiniujmy następujące funkcje, określone dla par nieujemnych liczb rzeczywistych:

- $g(x,y) = \lfloor x \rfloor \cdot \lfloor y \rfloor$,
- $f(x,y) = Z(\lfloor x \rfloor, \lfloor y \rfloor)$.

Na podstawie równości (1) oraz (5) możemy zapisać następującą zależność między funkcjami f oraz g:

$$g(x,y) = \sum_{d \geqslant 1} f\left(\frac{x}{d}, \frac{y}{d}\right).$$

Zależność nie wydaje się być przydatna, bo interesują nas wartości funkcji f, a nie g, jednak okazuje się, że możemy z niej wyznaczyć wzór na funkcję f! W tym celu można zastosować *metodę odwracania z użyciem funkcji Möbiusa* (patrz, na przykład, rozdział 4.9 w [30]). W rozważanym przypadku da nam ona następujący wzór:

$$f(x,y) = \sum_{d \geqslant 1} \mu(d) g\left(\frac{x}{d}, \frac{y}{d}\right), \tag{6}$$

gdzie μ jest tak zwaną funkcją Möbiusa. Powyższy wzór za chwilę udowodnimy, jednak najpierw musimy poznać lepiej funkcję μ Möbiusa. Ma ona dwie równoważne definicje (dowód ich równoważności można znaleźć także w [30]).

Def. 1: Pierwsza z nich jest definicją typu rekurencyjnego:

$$\sum_{d|m}\mu(d)=\begin{cases}1 & \text{dla } m=1\\ 0 & \text{dla } m>1.\end{cases}$$

Z tej definicji łatwo wnioskujemy, że $\mu(1)=1$, dalej $\mu(2)=0-\mu(1)=-1$, podobnie $\mu(3)=-1$, wreszcie $\mu(4)=0-\mu(2)-\mu(1)=0$ itd. Kilka początkowych wartości funkcji μ prezentujemy w poniższej tabelce:

i	1	2	3	4	5	6	7	8	9	10
$\mu(i)$	1	-1	-1	0	-1	1	-1	0	0	1

Def. 2: Według drugiej definicji $\mu(m)=0$, jeżeli m dzieli się przez p^2 dla pewnej liczby pierwszej p. W przeciwnym przypadku $\mu(m)=(-1)^r$, gdzie r jest liczbą różnych dzielników pierwszych m.

Poznawszy funkcję Möbiusa, możemy udowodnić równość (6):

Fakt 5 *Jeżeli funkcje f oraz g są związane zależnością*

$$g(x,y)=\sum_{d\geqslant 1} f\Big(\frac{x}{d},\frac{y}{d}\Big)$$

oraz $\sum_{k,d\geqslant 1}|f(x/(kd))|<\infty$, to

$$f(x,y)=\sum_{d\geqslant 1}\mu(d)g\Big(\frac{x}{d},\frac{y}{d}\Big).$$

Dowód Przyjrzyjmy się następującemu ciągowi przekształceń:

$$\begin{aligned}\sum_{d\geqslant 1}\mu(d)g\Big(\frac{x}{d},\frac{y}{d}\Big) &= \sum_{d\geqslant 1}\mu(d)\sum_{k\geqslant 1}f\Big(\frac{x}{kd},\frac{y}{kd}\Big)\\ &= \sum_{m\geqslant 1}f\Big(\frac{x}{m},\frac{y}{m}\Big)\sum_{d,k\geqslant 1,m=kd}\mu(d)\\ &= \sum_{m\geqslant 1}f\Big(\frac{x}{m},\frac{y}{m}\Big)\sum_{d|m}\mu(d).\end{aligned}$$

Zauważmy, że na podstawie pierwszej definicji funkcji Möbiusa wewnętrzna suma $\sum_{d|m}\mu(d)$ jest niezerowa (a dokładniej równa 1) tylko dla $m=1$. To pokazuje, że jedynym niezerowym składnikiem całej sumy będzie $f(x/1,y/1)=f(x,y)$, co kończy dowód. ■

Z definicji funkcji f i g oraz udowodnionego faktu otrzymujemy równość, która pomoże nam efektywnie wyznaczać odpowiedzi na zapytania:

$$Z(a,b)=\sum_{d\geqslant 1}\mu(d)\Big\lfloor\frac{a}{d}\Big\rfloor\cdot\Big\lfloor\frac{b}{d}\Big\rfloor. \tag{7}$$

Od wzoru do algorytmu

Rozpocznijmy od uproszczenia dopiero co wyprowadzonego wzoru (7). Przede wszystkim zauważmy, że sumowanie możemy ograniczyć do $d \leqslant \max(a,b)$, gdyż dla pozostałych wartości d wyrażenie wewnątrz sumy i tak będzie równe zeru (zamiast $\max(a,b)$ można by użyć nawet $\min(a,b)$, ale nam będzie wygodniej pozostać przy wersji z maksimum). Możemy też bez straty ogólności założyć, że $a \geqslant b$, co pozwala zapisać wzór (7) w postaci:

$$Z(a,b) = \sum_{d=1}^{a} \mu(d) \left\lfloor \frac{a}{d} \right\rfloor \cdot \left\lfloor \frac{b}{d} \right\rfloor. \tag{8}$$

Obliczenia możemy rozpocząć od wyznaczenia wszystkich potrzebnych wartości funkcji Möbiusa $\mu(1),\ldots,\mu(m)$. W tym celu, za pomocą sita Eratostenesa, rozkładamy wszystkie liczby od 1 do m na czynniki pierwsze i wyznaczamy wartość μ z definicji drugiej. Operacja ta ma złożoność czasową $O(m \log m)$. Znając wartości μ, wartość $Z(a,b)$ potrafimy wyliczyć, korzystając ze wzoru (8) w czasie $O(a)$. Nie jest to jednak metoda wystarczająco szybka. Okazuje się, że można ją przyspieszyć, osiągając złożoność czasową odpowiedzi na jedno zapytanie $O(\sqrt{a})$, a całego rozwiązania: $O(n\sqrt{m} + m \log m)$, gdzie drugi składnik wynika z konieczności wyznaczenia potrzebnych wartości funkcji μ. Zobaczmy, jak się to robi.

Na początek obliczmy sumy prefiksowe funkcji μ, czyli wartości

$$F(k) = \sum_{i=1}^{k} \mu(i) \text{ dla } 1 \leqslant k \leqslant m.$$

Potrafimy to zrobić w czasie $O(m)$. Potem możemy w czasie stałym znajdować wartości: $\mu(a) + \mu(a+1) + \cdots + \mu(b) = F(b) - F(a-1)$.

Następnie wyznaczenie sumy ze wzoru (8) podzielmy na dwie części:

$$Z(a,b) \;=\; \Big(\sum_{d=1}^{\lfloor \sqrt{a} \rfloor} \mu(d) \left\lfloor \frac{a}{d} \right\rfloor \cdot \left\lfloor \frac{b}{d} \right\rfloor \Big) + \Big(\sum_{d=\lfloor \sqrt{a} \rfloor + 1}^{a} \mu(d) \left\lfloor \frac{a}{d} \right\rfloor \cdot \left\lfloor \frac{b}{d} \right\rfloor \Big). \tag{9}$$

Składniki pierwszej sumy wyznaczymy bezpośrednio, jak w metodzie opisanej na początku tego rozdziału. Natomiast przy obliczeniach dla $d > \lfloor \sqrt{a} \rfloor$ posłużymy się sprytniejszym algorytmem. Zauważmy, że dla $d > \sqrt{a}$, zarówno $\lfloor \frac{a}{d} \rfloor < \sqrt{a}$, jak i $\lfloor \frac{b}{d} \rfloor < \sqrt{b} \leqslant \sqrt{a}$. To oznacza, że istnieje tylko $O(\sqrt{a})$ różnych wartości każdego z tych ilorazów. Zbadajmy najpierw $\lfloor \frac{a}{d} \rfloor$ (z drugim ilorazem postąpimy analogicznie). Dla każdej liczby $e = 1,\ldots,\lfloor \sqrt{a} \rfloor$ znajdźmy przedział całkowitych wartości d, dla których $\lfloor \frac{a}{d} \rfloor = e$:

$$e \leqslant \frac{a}{d} < e+1,$$

czyli

$$de \leqslant a < d(e+1).$$

Stąd $d \leqslant \lfloor \frac{a}{e} \rfloor$ oraz $d > \lfloor \frac{a}{e+1} \rfloor$ i ostatecznie otrzymujemy zależność

$$d \in \Big[\Big\lfloor \frac{a}{e+1} \Big\rfloor + 1, \Big\lfloor \frac{a}{e} \Big\rfloor \Big]. \tag{10}$$

Korzystając z (10), możemy podzielić cały przedział $[\lfloor\sqrt{a}\rfloor+1,\ldots,a]$ na $O(\sqrt{a})$ parami rozłącznych podprzedziałów, w których wyrażenie $\lfloor\frac{a}{d}\rfloor$ ma wartość stałą. W podobny sposób możemy podzielić ten przedział na $O(\sqrt{a})$ podprzedziałów, w których z kolei wyrażenie $\lfloor\frac{b}{d}\rfloor$ jest stałe. Następnie możemy połączyć oba te podziały w jeden gęstszy podział, tak aby w poszczególnych przedziałach zarówno $\lfloor\frac{a}{d}\rfloor$, jak i $\lfloor\frac{b}{d}\rfloor$, było stałe.

Przykład 1 Niech $a = 18$, $b = 15$. Wówczas $\lfloor\sqrt{a}\rfloor = 4$. W poniższej tabelce są przedstawione wartości $\lfloor\frac{a}{d}\rfloor$ dla $d \in [1,18]$:

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\lfloor\frac{18}{d}\rfloor$	18	9	6	4	3	3	2	2	2	1	1	1	1	1	1	1	1	1

Na podstawie wartości w tabelce możemy dokonać następującego podziału $[\lfloor\sqrt{a}\rfloor+1,a] = [5,18]$ dla a:

- w przedziale $[5,6]$ zachodzi $\lfloor\frac{a}{d}\rfloor = 3$,
- w przedziale $[7,9]$ zachodzi $\lfloor\frac{a}{d}\rfloor = 2$,
- w przedziale $[10,18]$ zachodzi $\lfloor\frac{a}{d}\rfloor = 1$.

Zauważmy, że podział ten jest identyczny z tym, jaki otrzymalibyśmy za pomocą zależności (10):

- $\lfloor\frac{18}{d}\rfloor = 3$ w przedziale $[\lfloor\frac{18}{4}\rfloor+1,\lfloor\frac{18}{3}\rfloor] = [5,6]$,
- $\lfloor\frac{18}{d}\rfloor = 2$ w przedziale $[\lfloor\frac{18}{3}\rfloor+1,\lfloor\frac{18}{2}\rfloor] = [7,9]$,
- $\lfloor\frac{18}{d}\rfloor = 2$ w przedziale $[\lfloor\frac{18}{2}\rfloor+1,\lfloor\frac{18}{1}\rfloor] = [10,18]$.

Podobnie możemy skonstruować tabelkę dla b:

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\lfloor\frac{15}{d}\rfloor$	15	7	5	3	3	2	2	1	1	1	1	1	1	1	1

Widoczny w niej podział przedziału $[5,18]$ dla b to:

- w przedziale $[5,5]$ zachodzi $\lfloor\frac{b}{d}\rfloor = 3$
- w przedziale $[6,7]$ zachodzi $\lfloor\frac{b}{d}\rfloor = 2$,
- w przedziale $[8,15]$ zachodzi $\lfloor\frac{b}{d}\rfloor = 1$.

Połączenie podziałów dla a oraz dla b daje podział:

- w przedziale $[5,5]$ mamy $\lfloor\frac{a}{d}\rfloor = 3$ oraz $\lfloor\frac{b}{d}\rfloor = 3$,
- w przedziale $[6,6]$ mamy $\lfloor\frac{a}{d}\rfloor = 3$ oraz $\lfloor\frac{b}{d}\rfloor = 2$,
- w przedziale $[7,7]$ mamy $\lfloor\frac{a}{d}\rfloor = 2$ oraz $\lfloor\frac{b}{d}\rfloor = 2$,

- w przedziale $[8,9]$ mamy $\lfloor \frac{a}{d} \rfloor = 2$ oraz $\lfloor \frac{b}{d} \rfloor = 1$,
- w przedziale $[10,15]$ mamy $\lfloor \frac{a}{d} \rfloor = 1$ oraz $\lfloor \frac{b}{d} \rfloor = 1$,
- w przedziale $[16,18]$ iloczyn $\lfloor \frac{a}{d} \rfloor \cdot \lfloor \frac{b}{d} \rfloor$ ma wartość 0.

■

Liczba podprzedziałów w gęstszym podziale jest również rzędu $O(\sqrt{a})$, gdyż początki i końce podprzedziałów są w nim wyznaczone przez początki i końce podprzedziałów z osobnych podziałów dla a oraz dla b. Znalezienie podziału na podstawie wzorów (10) dla a i b jest dosyć techniczne, lecz niezbyt skomplikowane i może być wykonane w czasie $O(\sqrt{a})$. Oznaczmy znaleziony podział $[l_1, r_1] \cup \cdots \cup [l_s, r_s]$. Teraz możemy wyliczyć drugą sumę z (9). Ponieważ w każdym podprzedziale $[l_i, r_i]$ wyrażenie $\lfloor \frac{a}{d} \rfloor \cdot \lfloor \frac{b}{d} \rfloor$ ma wartość stałą (możemy przyjąć, na przykład, $\lfloor \frac{a}{l_i} \rfloor \cdot \lfloor \frac{b}{l_i} \rfloor$), więc

$$\sum_{d=\lfloor\sqrt{a}\rfloor+1}^{a} \mu(d) \left\lfloor \frac{a}{d} \right\rfloor \cdot \left\lfloor \frac{b}{d} \right\rfloor = \sum_{i=1}^{s} \sum_{d \in [l_i, r_i]} \mu(d) \left\lfloor \frac{a}{d} \right\rfloor \cdot \left\lfloor \frac{b}{d} \right\rfloor = \sum_{i=1}^{s} (F(r_i) - F(l_i - 1)) \cdot \left\lfloor \frac{a}{l_i} \right\rfloor \cdot \left\lfloor \frac{b}{l_i} \right\rfloor.$$

(przypomnijmy, że przez F oznaczyliśmy sumy prefiksowe funkcji Möbiusa). W ten sposób zakończyliśmy konstrukcję algorytmu wyznaczania odpowiedzi na zapytanie $Z(a,b)$ w złożoności czasowej $O(\sqrt{m})$.

Implementacja przedstawionej metody znajduje się w pliku `zap1.cpp`.

Testy

Zadanie było sprawdzane na 15 zestawach danych testowych. Poniżej przedstawiona została tabela z opisami testów, w której n oznacza liczbę zapytań w teście, a parametry m_1 i m_2 wyliczone są według wzorów:

$$\begin{aligned} m_1 &= \max\{\max(a,b) \mid (a,b) \in Q\}, \\ m_2 &= \max\{\min(a,b) \mid (a,b) \in Q\}. \end{aligned}$$

Nazwa	n	m_1	m_2	Opis
zap1.in	10	40	34	test losowy
zap2.in	37	98	63	test losowy
zap3.in	100	200	185	test losowy
zap4.in	12 500	3 125	100	test losowy
zap5.in	49 999	8 333	8304	test losowy
zap6.in	50 000	50 000	100	test losowy
zap7.in	100	49 842	49 768	test losowy
zap8.in	3124	10 000	9 796	test losowy

Nazwa	n	m_1	m_2	Opis
zap9.in	12501	16666	16506	test losowy
zap10.in	50000	50000	49953	test losowy
zap11.in	49729	50000	49778	do pokrycia Q potrzeba $\sqrt{n}$ ścieżek
zap12.in	50000	50000	50000	Q jest zbiorem $\{(50000,x) : 1 \leqslant x \leqslant 50000\}$
zap13.in	49770	49980	49980	liczby występujące w zapytaniach mają dużo dzielników pierwszych
zap14.in	50000	50000	50000	liczby występujące w zapytaniach mają duże największe wspólne dzielniki
zap15.in	50000	50000	43630	liczby występujące w zapytaniach to liczniki i mianowniki pewnych ułamków z ciągu Fareya (więcej o ciągu Fareya można przeczytać np. w [35])

Zawody II stopnia

opracowania zadań