Podzielność iloczynu II

Kod zadania: **ilo**

Olimpiada Informatyczna Juniorów

Zadanie

Dany jest ciąg N liczb całkowitych $A_1, \ldots, A_N$. W jednym ruchu można **zwiększyć lub zmniejszyć** dowolny element ciągu o 1. Należy:

- wyznaczyć **minimalną liczbę ruchów**, po których wykonaniu iloczyn

$$P = A_1 \cdot A_2 \cdots A_N$$

  nie będzie podzielny przez 6,
- podać **przykładowy ciąg** uzyskany po wykonaniu minimalnej liczby ruchów.

Rozwiązanie

Iloczyn liczb jest podzielny przez 6 wtedy i tylko wtedy, gdy jest podzielny jednocześnie przez 2 i przez 3:

$$6 \mid P \iff (2 \mid P) \text{ oraz } (3 \mid P).$$

(gdzie $a \mid b$ czytamy jako „a dzieli b").

Z kolei:

- 2 dzieli iloczyn liczb wtedy i tylko wtedy, gdy co najmniej jedna z tych liczb jest parzysta,
- 3 dzieli iloczyn liczb wtedy i tylko wtedy, gdy co najmniej jedna z tych liczb jest podzielna przez 3.

Aby iloczyn P **nie był** podzielny przez 6, wystarczy więc spełnić *jeden* z warunków:

(A) żadna liczba w ciągu nie jest parzysta lub (B) żadna liczba w ciągu nie jest podzielna przez 3.

Strategia A: eliminacja liczb parzystych. Chcemy, żeby wszystkie liczby były nieparzyste. Każdą liczbę parzystą trzeba zmienić o co najmniej 1, aby stała się nieparzystą; liczby nieparzyste mogą pozostać bez zmian. Koszt tej strategii to:

$$\text{koszt}_2 = \text{liczba liczb parzystych}.$$

Przykładowa konstrukcja: każdą liczbę parzystą zwiększamy o 1.

Strategia B: eliminacja liczb podzielnych przez 3**.** Chcemy, by żadna liczba nie była podzielna przez 3. Każdą liczbę podzielną przez 3 trzeba zmienić o co najmniej 1, aby przestała być podzielna przez 3; liczby niepodzielne przez 3 mogą pozostać bez zmian. Koszt tej strategii to:

$$\text{koszt}_3 = \text{liczba liczb podzielnych przez 3}.$$

Przykładowa konstrukcja: każdą liczbę podzielną przez 3 zwiększamy o 1.

Odpowiedź. Minimalna liczba ruchów to:

$$\min(\text{koszt}_2,\ \text{koszt}_3).$$

Jeżeli $\text{koszt}_2 \leq \text{koszt}_3$, wybieramy strategię A (zmieniamy wszystkie liczby na nieparzyste); w przeciwnym razie wybieramy strategię B (eliminujemy wszystkie liczby podzielne przez 3). Ciąg po zmianach otrzymujemy, stosując powyższą regułę dla każdej liczby niezależnie.

Złożoność obliczeniowa: liniowa $O(N)$ – jedno przejście po ciągu, aby zliczyć liczby podzielne przez 2 i 3, a potem drugie do zbudowania przykładowego wyniku.

Olimpiada finansowana jest ze środków Ministra Edukacji w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2025/2026, 2026/2027, 2027/2028".

Spóźnienia

Kod zadania: **spo**

Olimpiada Informatyczna Juniorów

Zadanie

Odbywa się spotkanie, na którym N rycerzy $R_1, \ldots, R_N$ zasiada przy okrągłym stole. Rycerz R_i przemawia przez A_i minut. Spotkanie rozpoczyna pewien rycerz R_i, następnie przemawia R_{i+1}, po nim R_{i+2} i tak dalej, aż do R_{i-1}.

Gdyby wszystko przebiegało bez zakłóceń, czas całego spotkania byłby stały i wynosiłby $A_1 + A_2 + \ldots + A_N$. Rycerze jednak spóźniają się na spotkanie.

Rycerz R_i przybywa w minucie B_i. Jeśli kolej rycerza na przemówienie nadejdzie, zanim dotrze on na miejsce, wszyscy obecni na niego czekają. Należy obliczyć całkowity czas spotkania dla każdego z rycerzy, przy założeniu, że to on przemawia jako pierwszy.

Przykład

Dla $N = 3$ rycerzy oraz tablic $A = [1, 2, 1]$ i $B = [3, 2, 7]$, jeśli spotkanie rozpocznie rycerz R_1:

- Rycerz R_1 przybywa w minucie 3 i kończy przemawiać w minucie $3 + 1 = 4$.
- Rycerz R_2 przybywa w minucie 2, czeka do minuty 4 i kończy przemawiać w minucie $4 + 2 = 6$.
- Rycerz R_3 przybywa w minucie 7, nie musi czekać (rozpoczyna w minucie 7) i kończy przemawiać w minucie $7+1 = 8$.

Zatem odpowiedź to 8.

Rozwiązanie

Operacje na indeksach wykonujemy modulo N.

Symulacja $O(N^2)$

Możemy sprawdzić każdego rycerza jako rozpoczynającego. Dla ustalonego rycerza początkowego R_i symulujemy spotkanie krok po kroku, śledząc aktualny czas. Kiedy przychodzi kolej na rycerza R_j, zaczyna on przemawiać w czasie będącym maksimum z czasu zakończenia przemowy poprzedniego rycerza i czasu jego przybycia B_j. Do tej wartości dodajemy czas trwania jego przemowy A_j, otrzymując czas jej zakończenia. Powtarzamy to dla wszystkich N rycerzy w kolejności. Ponieważ jest N możliwych rycerzy rozpoczynających, a symulacja dla każdego z nich trwa $O(N)$, całkowita złożoność tego podejścia wynosi $O(N^2)$.

Rozwiązanie $O(N)$

Niech $T(i)$ oznacza czas trwania spotkania, jeśli rozpoczyna je rycerz R_i. Całkowity czas spotkania to suma czasów wszystkich przemówień $S = A_0 + \cdots + A_{N-1}$ powiększona o sumę wszystkich okresów oczekiwania. Kluczowe jest zrozumienie, co determinuje czas zakończenia przemowy przez ostatniego z rycerzy.

Można udowodnić, że całkowity czas spotkania $T(i)$, gdy jako pierwszy przemawia rycerz R_i, wyraża się wzorem:

$$T(i) = S + \max_{j=i}^{i+N-1} \{B_j - (A_i + A_{i+1} + \ldots + A_{j-1})\},$$

gdzie:

- S: suma czasów wszystkich przemówień.

Olimpiada finansowana jest ze środków Ministra Edukacji w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2025/2026, 2026/2027, 2027/2028".

- $A_i + A_{i+1} + \ldots + A_{j-1}$: łączny czas przemówień rycerzy od R_i do R_{j-1}. Oznaczmy tę sumę jako $A[i..j-1]$.
- $B_j - A[i..j-1]$: różnica ta porównuje czas przybycia rycerza R_j z czasem, jaki upłynął na przemówieniach wszystkich rycerzy przed nim (pamiętając, że R_i przemawia jako pierwszy).
- $\max_{j=i}^{i+N-1}\{B_j - A[i..j-1]\}$: maksimum z tak zdefiniowanych różnic.

Dlaczego interesuje nas maksimum z tych wartości? Rozważmy ostatniego rycerza R_j, który zaczął przemawiać dokładnie w chwili swojego przybycia na spotkanie, tj. w czasie B_j (taki rycerz musi istnieć, ponieważ R_i z definicji posiada tę własność).

Jego moment przybycia i zarazem rozpoczęcia przemowy to z założenia B_j. Ile czasu uczestnicy spędzili do tej pory na milczącym oczekiwaniu? Wiemy, że rycerze od R_i do R_{j-1} zdążyli już zakończyć swoje przemówienia, a gdy R_j zaczyna mówić, mamy chwilę B_j. Odejmując od czasu B_j łączny czas ich przemówień, dowiadujemy się, jak długo trwała cisza (czas oczekiwania) do momentu B_j. Z drugiej strony, ponieważ R_j to *ostatni* taki rycerz, każdy przemawiający po nim rycerz był już obecny na spotkaniu, gdy nadeszła jego kolej. Oznacza to, że po rycerzu R_j nie było już żadnych opóźnień!

Wobec tego całe spotkanie trwa tyle, co suma czasów wszystkich przemówień (S) plus czas spędzony w ciszy do chwili B_j, czyli $B_j - A[i..j-1]$.

Wystarczy zatem udowodnić, że dla tego rycerza R_j osiągane jest wspomniane maksimum:

1. Dla $k > j$ wiemy, że moment rozpoczęcia przemowy przez rycerza R_k to $B_j + A[j..k-1]$ (z powodu braku opóźnień). Ponadto $B_k < B_j + A[j..k-1]$, ponieważ R_k nie zaczął mówić od razu w chwili swojego przybycia na spotkanie. Zatem:
$$B_k - A[i..k-1] < B_j + A[j..k-1] - A[i..k-1] = B_j - A[i..j-1],$$
co należało pokazać.
2. Dla $k < j$ oznaczmy czas rozpoczęcia przemowy przez R_k jako C_k. Oczywiście $C_k \geq B_k$ (rycerz musi być już na spotkaniu, aby zacząć mówić). Dodatkowo, od momentu rozpoczęcia przemowy przez R_k do momentu rozpoczęcia przemowy przez R_j musiało minąć co najmniej $A[k..j-1]$ czasu. Wynika stąd, że $A[k..j-1] \leq B_j - C_k \iff C_k \leq B_j - A[k..j-1]$. Otrzymujemy zatem:
$$B_k - A[i..k-1] \leq C_k - A[i..k-1] \leq B_j - A[k..j-1] - A[i..k-1] = B_j - A[i..j-1],$$
co kończy dowód.

Dowodzi to, że wzór na $T(i)$ jest prawidłowy. Przekształćmy go nieco, korzystając z sum prefiksowych $P[i] = A_0 + A_1 + \ldots + A_{i-1}$ oraz $P[0] = 0$. Otrzymujemy:

$$T(i) = S + \max_{j=i}^{i+N-1}\{B_j - (A_i + A_{i+1} + \ldots + A_{j-1})\} = S + \max_{j=i}^{i+N-1}\{B_j - (P[j] - P[i])\} = S + P[i] + \max_{j=i}^{i+N-1}\{B_j - P[j]\}.$$

Wyrażenie pod operatorem max zależy teraz wyłącznie od indeksu j, a pozostałą część, czyli $S + P[i]$, potrafimy szybko obliczyć. Chcemy wyznaczyć to „maksimum na przedziale" $[i..i+N-1]$ dla wszystkich możliwych i. Jest to klasyczny problem, który można rozwiązać dla wszystkich rycerzy w czasie $O(N \log N)$ na wiele sposobów, na przykład przy użyciu struktury `std::multiset` lub drzewa przedziałowego.

Istnieje jednak eleganckie rozwiązanie w czasie liniowym $O(N)$, wykorzystujące tak zwaną *kolejkę monotoniczną*. Rozwiązanie to jest dostępne w plikach `spo.cpp` oraz `spo.py`.

Olimpiada finansowana jest ze środków Ministra Edukacji
w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad
i turniejów w latach szkolnych 2025/2026, 2026/2027, 2027/2028".

Wycieczka górska

Kod zadania: **wyc**

Olimpiada Informatyczna Juniorów

Zadanie

Dana jest mapa o wymiarach $n \times m$, w której każda komórka zawiera liczbę całkowitą – wysokość terenu. Należy znaleźć długość najdłuższej ścieżki na mapie, składającej się z pól sąsiadujących bokami (ruch o 1 w górę, w dół, w lewo lub w prawo), wzdłuż której wysokości odwiedzanych punktów tworzą ciąg **ściśle rosnący**.

Rozwiązanie

Oznaczmy przez $dp[i][j]$ długość najdłuższej rosnącej ścieżki zaczynającej się w komórce (i, j).
Niech $N(i, j)$ oznacza zbiór pól sąsiadujących z polem (i, j). Prawdziwa jest następująca zależność:

$$dp[i][j] = 1 + \max_{\substack{(i',j') \in N(i,j) \\ h[i'][j'] > h[i][j]}} dp[i'][j'].$$

(przyjmujemy, że maksimum po pustym zbiorze wynosi 0).

Innymi słowy, najdłuższa ścieżka zaczynająca się w (i, j) ma długość o jeden większą niż najdłuższa ścieżka zaczynająca się w sąsiednim polu o większej wysokości. Istotnie – każda trasa startująca w (i, j) musi w pierwszym kroku przejść do pola ze zbioru $N(i, j)$. Aby kontynuować ścieżkę ściśle rosnącą, możemy przejść tylko do sąsiadów o większej wysokości.

Aby obliczyć wartości $dp[i][j]$ dla wszystkich pól, możemy posortować wszystkie pola (i, j) malejąco względem ich wysokości $h[i][j]$. Następnie iterujemy po posortowanych polach. Dla danego pola (i, j) wartość $dp[i][j]$ możemy obliczyć, korzystając z powyższego wzoru, ponieważ wszystkie sąsiednie pola (i', j') o większej wysokości $h[i'][j']$ zostały już przetworzone ze względu na kolejność sortowania. Ten sposób gwarantuje, że wartości dp dla potrzebnych sąsiadów są już wyznaczone. Daje nam to rozwiązanie o złożoności czasowej $O(nm \log(nm))$ ze względu na sortowanie.

Alternatywnie można wyznaczyć te wartości rekurencyjnie dla każdego pola, korzystając z techniki zwanej *memoizacją*. W tym podejściu funkcja obliczająca $dp[i][j]$ najpierw sprawdza, czy wartość ta została już obliczona. Jeśli tak, zwraca zapamiętaną wartość. W przeciwnym razie oblicza wartość zgodnie ze wzorem, wywołując się rekurencyjnie, zapamiętuje ją i zwraca. To podejście daje złożoność $O(nm)$, ponieważ każda wartość $dp[i][j]$ jest obliczana tylko raz.

Ostateczny wynik to $\max_{i,j} dp[i][j]$.

Olimpiada finansowana jest ze środków Ministra Edukacji
w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad
i turniejów w latach szkolnych 2025/2026, 2026/2027, 2027/2028".