1A – Gra Mobilna

Autor zadania: Adam Gąsienica-Samek

Autorzy omówienia: Eryk Kopczyński, Paweł Parys

Treść

Mamy daną listę n wydarzeń, opisaną przez pary liczb (a_i, b_i), przy czym $a_i \geq 0$ oraz $b_i \geq 1$. Gdy przed i-tym wydarzeniem mamy x wojowników, to mamy do wyboru: dodać a_i do x albo pomnożyć x przez b_i; to daje nam liczbę wojowników po tym wydarzeniu. Danych jest również q zapytań o trójki (x, l, r). Zapytanie takie opisuje sytuację, w której zaczynamy tuż przed wydarzeniem $l+1$ mając x wojowników; należy wypisać maksymalną liczbę wojowników możliwych do uzyskania po wydarzeniu r. Wypisujemy resztę z dzielenia tak obliczonej liczby przez $p = 10^9 + 7$.

Rozwiązanie

Pierwsza obserwacja: zawsze warto wziąć tę z dwóch możliwości, która daje lepszy natychmiastowy zysk.

Niech m będzie ograniczeniem górnym na liczby a_i, b_i; zgodnie z treścią zadania $m = p - 1$. Druga obserwacja: jeśli mamy więcej niż m żołnierzy (sytuacja „big"), to wiadomo, co robić: jeśli $b_i > 1$ („wydarzenie mnożące") to zawsze opłaca się mnożyć, a jeśli nie („wydarzenie niemnożące"), to zawsze opłaca się dodać. Te ruchy nazwijmy *big-optymalnymi*.

Trzecia obserwacja: po $\log m$ wydarzeniach mnożących na pewno mamy big.

Czwarta obserwacja: złożenie dowolnej liczby operacji big-optymalnych można reprezentować za pomocą pary liczb (c, d) mówiących, że najpierw mnożymy przez c, a potem dodajemy d. Ponadto pary takie możemy łatwo ze sobą składać. Faktycznie, pojedyncza operacja big-optymalna jest tej postaci: $(b_i, 0)$ gdy $b_i > 1$ oraz $(1, a_i)$ gdy $b_i = 1$. Natomiast zastosowanie kolejno dwóch takich par (c, d) oraz (c', d') do x daje wynik $(x \cdot c + d) \cdot c' + d' = x \cdot (c \cdot c') + (d \cdot c' + d)$, taki sam jak para $(c \cdot c', d \cdot c' + d')$. Oczywiście wszystkie liczby w tak uzyskiwanych parach będziemy trzymać modulo p (to nie przeszkadza w ich składaniu).

Umożliwia to uzyskanie struktury danych, która dla danego przedziału będzie szybko zwracać złożenie operacji big-optymalnych na tym przedziale. Mamy tu dwie równie dobre opcje. Pierwsza możliwość to użycie drzewa przedziałowego, gdzie w wierzchołkach odpowiadającym przedziałom trzymamy złożenie operacji big-optymalnych na tych przedziałach. Liczymy to drzewo przedziałowe raz, a potem zapytania o przedziały realizujemy w $O(\log n)$. Druga możliwość, to skorzystanie z odwracalności operacji big-optymalnych: jeśli złożenie znanego (c, d) oraz nieznanego (c', d') daje znane (s, t), to możemy obliczyć $c' = s/c$ oraz $d' = t - d \cdot c'$. Występuje tutaj dzielenie modulo p, które jest równoważne mnożeniu przez d^{p-2}, ponieważ z małego twierdzenia Fermata mamy $d^{p-1} \equiv 1 \pmod{p}$. Możemy więc stablicować złożenie operacji big-optymalnych na każdym prefiksie i potem użyć powyższego sposobu na uzyskanie wyniku dla dowolnego przedziału z wyniku dla dwóch prefiksów; realizujemy to w czasie $O(\log p)$ ze względu na konieczność potęgowania.

Piata obserwacja: bloki wydarzeń niemnożących można skompresować do jednej dodawanej liczby (i również używać sum prefiksowych, gdy wartości l lub r są gdzieś pośrodku takiego bloku).

Podsumujmy opisując całość rozwiązania. Najpierw obliczamy złożenia operacji big-optymalnych oraz bloków wydarzeń niemnożących w czasie $O(n)$. Następnie, dla każdego zapytania, najpierw przetwarzamy co najwyżej $\log m$ pierwszych wydarzeń mnożących i bloków wydarzeń niemnożących, aż do uzyskania liczby większej niż m, a następnie aplikujemy złożenie operacji big-optymalnych. Daje nam to czas $O(n + q(\log m + \log p))$.