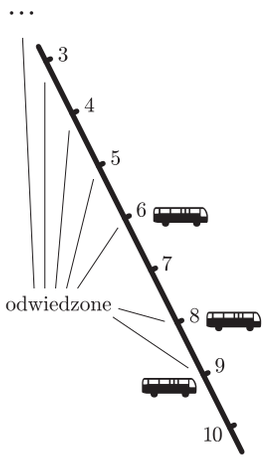

Informatyczny kącik olimpijski (17): Rozkłady jazdy

Wyobraźmy sobie miasto z jedną, długą drogą, wzdłuż której ustawione są (co kilometr) przystanki autobusowe – w sumie jest ich N. Rozkłady jazdy w tym dziwnym mieście często się zmieniają, ale zawsze zachowują pewne cechy. Wzdłuż drogi kursuje K autobusów, z których każdy rano zaczyna z jednego z pierwszych K przystanków (każdy z innego), a wieczorem kończy na jednym z ostatnich K (również każdy na innym). W ciągu dnia autobusy poruszają się tylko w jednym kierunku. Na każdym z przystanków zatrzymuje się dokładnie jeden autobus; co więcej, pomiędzy każdymi dwoma kolejnymi przystankami każdego z autobusów jest nie więcej niż L kilometrów (L jest niewielką liczbą, rzędu 10).

Czy takich rozkładów jazdy może być wiele? Oto właśnie zadanie – obliczenie ich liczby, modulo pewna, zadana liczba M. Z tym właśnie zadaniem zmierzyli się uczestnicy *Google Code Jam* 2008.

Zauważmy, że nie możemy układać rozkładu osobno dla poszczególnych autobusów, gdyż wówczas musielibyśmy przez cały czas pamiętać wszystkie już odwiedzone przystanki (gdyby nie było ograniczenia na odległości między kolejnymi przystankami na trasie autobusu, to można by się jednak obejść bez tego). Z tego względu musimy układać rozkłady dla wszystkich autobusów naraz. Jak to robić? Zacznijmy od sytuacji, w której każdy autobus stoi na swoim początkowym przystanku. Teraz w każdym kroku wybierzmy jeden autobus i ustalmy, jaki jest jego następny przystanek – aż każdy autobus znajdzie się na jednym z ostatnich K przystanków. Taki „niedokończony” rozkład nazwiemy *prefiksem* rozkładu jazdy.

Wygodne spostrzeżenie: jeśli w każdym kroku będziemy wybierać ostatni autobus (tj. ten, który stoi na przystanku najbliższym początkowi drogi) i zawsze przestawiać go tak, aby żaden przystanek nie został definitywnie opuszczony, to:

- w każdym tak otrzymanym prefiksie rozkładu wszystkie autobusy będą w odległości co najwyżej $L-1$ km od ostatniego;
- nie musimy pamiętać, które przystanki zostały już odwiedzone, ponieważ są to wszystkie za ostatnim autobusem oraz te, na których aktualnie stoi któryś autobus (patrz rysunek).

Intuicja podpowiada, że do rozwiązania moglibyśmy użyć techniki programowania dynamicznego. Zauważmy podobieństwo naszego zadania do problemu plecakowego – prefiks rozkładu jazdy jest analogią do częściowo spakowanego plecaka. W przypadku plecaka nie pamiętamy dokładnie, co do niego spakowaliśmy, a jedynie ile to wszystko razem waży (lub zajmuje objętości, zależnie od sformułowania problemu). Jest to pełna wiedza, która nam wystarcza do dokończenia pakowania – dzięki niej możemy obliczyć, ile jeszcze możemy zmieścić. Jaka jest pełna wiedza, która by nam wystarczyła do dokończenia prefiksu p rozkładu jazdy? Musimy znać ostatni(p) – pozycję ostatniego autobusu, oraz pozycje(p), czyli zbiór $K-1$ różnych wartości pomiędzy 1 a $L-1$ – odległości pozostałych autobusów od ostatniego. Oczywiście, jest tylko $\binom{L-1}{K-1}$ różnych zbiorów pozycje(p). Zbiór pozycje(p) możemy reprezentować jako liczbę całkowitą – ma ona ustawiony i-ty bit wtedy i tylko wtedy, gdy $i+1 \in$ pozycje(p). Sortujemy te liczby i przez f(pozycje(p)) oznaczamy indeks w posortowanym ciągu liczby odpowiadającej zbiorowi pozycje(p). Inaczej – funkcja f jest „ponumerowaniem” wszystkich możliwych zbiorów pozycje(p).

Weźmy teraz zbiór następników p (oznaczmy go przez następne(p)), czyli zbiór prefiksów, które mogą powstać z p w wyniku wykonania pojedynczego kroku. Zauważmy, że jeśli $q \in$ następne(p), to ostatni(q) = = ostatni(p) + 1, ponieważ każdy przystanek musi zostać odwiedzony. Co więcej, weźmy takie p i q, że pozycje(p) = pozycje(q), ale niekoniecznie ostatni(p) = = ostatni(q). Jeśli wtedy znamy następne(p), to z tego możemy otrzymać następne(q): wystarczy, że zmienimy wartość ostatni($\cdot$) dla wszystkich prefiksów należących do tego zbioru. Innymi słowy, zbiór następników nie zależy od „bezwzględnej” pozycji autobusów, a jedynie od ich wzajemnego położenia.

Możemy więc utworzyć taką macierz P, że:

- $P_{i,j} = 1$, gdy istnieją takie prefiksy p i q, że f(pozycje(p)) = i, f(pozycje(q)) = j i $q \in$ następne(p) (co jest równoważne z „gdy dla każdego p spełniającego f(pozycje(p)) = i istnieje q takie, że f(pozycje(q)) = j i $q \in$ następne(p)”);
- $P_{i,j} = 0$ w przeciwnym przypadku.

Oznaczmy przez $T_l[i]$ liczbę różnych prefiksów p rozkładu jazdy takich, że ostatni(p) = l i f(pozycje(p)) = i. Obliczenie wektora T_1 jest oczywiste. Dla dowolnego $l \geqslant 2$ zachodzi

$$T_l = T_{l-1} \cdot P$$

W ten sposób możemy obliczyć T_{N-K+1} (wektor ten jest równy $T_1 \cdot P^{N-K}$) i stamtąd odczytać wynik (jak?).

Implementacja tego rozwiązania z wielokrotnym mnożeniem T_1 przez macierz P prowadziłaby do algorytmu o złożoności $O(NR^2)$, gdzie R jest wymiarem macierzy P, czyli $\binom{L-1}{K-1}$. Dla dużych N możemy za to wykorzystać szybkie (binarne) potęgowanie macierzy – złożoność takiego rozwiązania wyniosłaby $O(R^3 \log N)$. Na koniec dodajmy, że wszystkie obliczenia w tym zadaniu prowadzimy modulo M – złożoności rozwiązania, na szczęście, nie muszą pogarszać obliczenia na ogromnych liczbach.

Filip WOLSKI