Konkursik 21.01.2011 — rozwiązania zadań zadań

Tomasz Kulczyński

22 stycznia 2011

SCI — Miejskie podchody

Przede wszystkim sprowadzamy zadanie do terminów grafowych: mając dany graf z wagami krawędzi, należy w zbiorze wyróżnionych wierzchołków znaleźć dwa sobie najbliższe. Rozmiary danych oraz to, że są wagi krawędzi sugerują jeden algorytm: Dijkstra. Niewątpliwie należy go jednak w jakiś sposób zmodyfikować. Dwa różne sposoby:

1. Uruchamiamy Dijkstrę startującą ze wszystkich wyróżnionych punktów na raz, dla każdego wierzchołka poza minimalną odległością od któregoś wyróżnionego pamiętamy też z którego wyróżnionego wierzchołka jest ta odległość (tzw. „kolor wierzchołka"). Po zakończeniu Dijkstry, wynikiem jest minimum po krawędziach, których końce są różnych kolorów, z sumy długości tej krawędzi i odległości końców od wyróżnionych wierzchołków.

2. Uruchamiamy Dijkstrę, tyle że dla każdego wierzchołka pamiętamy najkrótszą ścieżkę do niego z wyróżnionych oraz z którego wyróżnionego można przyjść tą ścieżką, a także, dodatkowo, drugą najkrótszą ścieżkę z innego wyróżnionego. Dijkstra traktuje taki wierzchołek jak dwa osobne i być może rozważa je w dwóch różnych momentach swego działania. Wynik to minimum po wierzchołkach ich dwóch odległości od dwóch źródeł.

FAC — Silnie

Generalnie interesują nas rozkłady silni na czynniki pierwsze. Będziemy chcieli używać liczb $t(x,p)$ równej największemu takiemu k, że p^k jest dzielnikiem $x!$. Niech M będzie zakresem interesujących nas liczb (konkretnie, $M = 10007$, bo to jest najmniejsza liczba pierwsza większa od 10000) Niech P będzie liczbą liczb pierwszych mniejszych od M.

Są dwa podejścia:

1. Obliczamy i zapamiętujemy całą tablicę $t(x,p)$ rozmiaru $M \times P$, korzystając ze wzoru $t(x,p) = t(x-1,p) + c$, gdzie c obliczamy rozkładając x na czynniki pierwsze. Całość można policzyć w czasie $O(MP)$, a potem mamy $t(x,p)$ w czasie stałym.

2. Można też skorzystać ze wzoru znanego matematykom:

$$t(x,p) = \lfloor \frac{x}{p} \rfloor + \lfloor \frac{x}{p^2} \rfloor + \lfloor \frac{x}{p^3} \rfloor + ...$$

   i nie korzystając z żadnej dodatkowej pamięci, za każdym razem obliczać potrzebne nam akurat $t(x,p)$ w czasie $O(\log_p x)$

Teraz możemy przystąpić do rozwiązywania zadania. Dla każdej liczby pierwszej p obliczmy:

$$c(p) = t(p_1,p) + t(p_2,p) + \ldots + t(p_n,p) - t(q_1,p) - t(q_2,p) - \ldots - t(q_m,p)$$

W takim razie $c(p)$ jest krotnością p w rozkładzie naszego wyniku na czynniki pierwsze. Jeśli któreś $c(p)$ jest ujemne, to wynik nie jest całkowity i wypisujemy -1. W przeciwnym wypadku musimy odtworzyć wynik. W tym celu przebiegamy r-em od M do 2, a dla danego r liczymy

$$s = \min_p c(p)/t(r,p)$$

Jeśli $s > 0$, to na wyjście wypiszemy parę (r,s), a od każdego $c(p)$ należy odjąć $s * t(r,p)$ i postępować tak samo dalej. Całe rozwiązanie działa w czasie $O((n+m+M) * P)$ (ewentualnie trzeba przemnożyć przez $logM$, jeśli używamy drugiej wersji obliczania $t()$, ale w praktyce to ta druga działa szybciej przez cache itd.).

BAS — Basen

Rozwiązanie dynamiczne. Próbujemy budować optymalny rozkład po zamianach, po jednym torze. O dotychczasowo zbudowanym rozwiązaniu pamiętamy:

- ile już mamy torów
- ile już zużyliśmy ludzi
- ilu ludzi pływa na ostatnim z naszych torów

Dla takiego stanu wynikiem jest minimalna liczba zamian, które trzeba wykonać, żeby taki stan osiągnąć. Zauważmy, że tak powstające rozwiązanie można rozszerzyć tylko na co najwyżej trzy sposoby: $(i, s, o) \Rightarrow (i + 1, s + o, o)$ $(i, s, o) \Rightarrow (i + 1, s + o + 1, o + 1)$ $(i, s, o) \Rightarrow (i + 1, s + o - 1, o - 1)$ Dla każdego z tych sposobów musimy umieć szybko policzyć, ile zamian torów pomiędzy torami i-tym, a $i + 1$-szym potrzeba, żeby taki stan osiągnąć, ale jest to zawsze $a_1+a_2+\ldots+a_i-s$, więc wystarczy stablicować sumy prefiksowe początkowego ciągu a_i. Takie rozwiązanie działa w czasie $O(nsx)$, gdzie x jest maksymalną liczbą ludzi, którą można umieścić na jednym torze w optymalnym rozwiązaniu. Ogólnie możemy tylko uznać, że $x \leqslant s$, co daje za wolne rozwiązanie. Możemy jednak zauważyć, że:

- dla $n > 60$ na pewno $x < 60$, czyli w tym przypadku mamy co najwyżej O(n * s * 60) operacji, czyli dość mało
- dla $60 \geqslant n > 10$ na pewno $x < 160$, czyli mamy co najwyżej O(60 * s * 160) operacji, czyli też mało
- dla $10 \geqslant n$ mamy co najwyżej O(10 * s * s) operacji, czyli też mało

Zamiast rozpisywać takie przypadki, można próbować ograniczyć x przez jakieś wyrażenie z literkami n i s, ale jest to dość kłopotliwe, i ciężkie później do zapisania w programie. Nie daje też raczej tak dobrych szacowań na liczbę wykonanych operacji (u nas, co najwyżej kilkadziesiąt milionów), a co najmniej bardzo ciężko tak dobre szacowanie uzyskać.