Omówienie zadań z turnieju drużynowego nr 33

Rafał Mańczyk, Wojciech Rybak

10 maja 2025

1 Crafting

Najłatwiejszym sposobem na zaimplementowanie tego zadania jest wczytanie stanu *crafting table* jako ciągu znaków oraz zapisanie w kodzie ciągów znaków reprezentujących poszczególne receptury i porównanie ich.

2 Cykl

Chcemy znaleźć taką parę liczb a, b, że mamy krawędzie $(1, a)$, $(1, b)$, (a, b). W tym celu przeiterujemy się po wszystkich krawędziach z wejścia (a, b), takich, że a i b są różne od 1, i sprawdzimy, czy a i b są połączone z 1.

3 Dzrzewo

Użyjemy centroidów. Dla aktualnego centroidu policzymy dla wierzchołków z jego części drzewa dwie wartości: $d(v)$ — odległość od centroidu oraz $mx(v)$ — maksymalną wartość na ścieżce od centroidu do v.

Szukamy takich par (u, v), że $\max(mx(u), mx(v)) - d(u) - d(v) \geqslant k$ oraz u i v nie należą do tego samego poddrzewa, gdy drzewo jest ukorzenione w centroidzie.

Posortujmy wszystkie wierzchołki rosnąco względem wartości $mx(v)$ i w takiej kolejności przeiterujmy się po nich. Dla danego wierzchołka v wiemy, że wcześniej rozpatrzone wierzchołki u spełniają $mx(u) \leqslant mx(v)$, więc

$$\max(mx(u), mx(v)) - d(u) - d(v) = mx(v) - d(u) - d(v).$$

Czyli dla ustalonego v szukamy takich u, że $d(u) \leqslant mx(v) - d(v) - k$. Możemy znaleźć liczbę takich u za pomocą drzewa przedziałowego.

W ten sposób policzymy pary spełniające nierówność, ale nie mamy gwarancji, że u i v należą do różnych poddrzew. Dlatego musimy użyć tej samej metody dla każdego poddrzewa osobno, aby policzyć i odjąć te pary, które pochodzą z tego samego poddrzewa.

Złożoność takiego algorytmu to $O(n \log^2 n)$.

4 Dzielniki

Niech $g = \text{nwd}(a, b)$. Jeśli $\text{nwd}(x, a) = g$, to w szczególności g musi być dzielnikiem x. Rozwiązanie będzie polegać na iterowaniu się po wielokrotnościach g z przedziału (l, r), tak długo, jak nie trafimy na x spełniające założenie zadania albo nie wyjdziemy poza przedział. Zapiszmy $x = X \cdot g$, teraz nasza pętla iteruje się po kolejnych X. Podobnie $a = A \cdot g$ i $b = B \cdot g$.

Teraz warunek z zadania wygląda tak: $\text{nwd}(X \cdot g, A \cdot g) = g$, $\text{nwd}(X \cdot g, B \cdot g) = g$, co jest równoważne warunkom $\text{nwd}(X, A) = 1$ oraz $\text{nwd}(X, B) = 1$. Zauważmy, że ponieważ iterujemy się po kolejnych X, to średnio co $O(\log n)$ iteracji trafimy na liczbę pierwszą. Wszystkie liczby pierwsze, poza tymi które występują w rozkładzie A lub B, spełniają warunki zadania. Liczb pierwszych w rozkładzie A lub B jest co najwyżej $O(\log n)$, więc wykonując zaledwie $O(\log^2 n)$ iteracji, powinniśmy znaleźć jakiś dobry X.

5 Pac-Man

Możemy policzyć odległości od każdego duszka do wolnych pól na planszy algorytmem BFS w czasie $O(k * n^2)$ dla każdego testu. Następnie dla każdego pola wystarczy policzyć czas, w którym dotrze do niego pierwszy duszek wzorem z treści zadania.

6 Podciągi

Niech $DP(i)$ oznacza liczbę podciągów spełniających założenia zadania, kończących się na i. Dla ustalonego i chcemy przedłużyć podciągi kończące się na j, takim, że $p_i \equiv 1 \pmod{p_j}$. Zauważmy, że jeśli to jest spełnione, to p_j jest dzielnikiem $p_i - 1$. Możemy więc na początku ztablicować dzielniki każdej liczby od 1 do n, a następnie, dla ustalonego i, przeiterować się po dzielnikach $p_i - 1$ i dodać do $DP(i)$ wartość $DP(j)$ dla odpowiadających j.

Dla ustalonego i wykonamy $d(p_i - 1)$ operacji, gdzie $d(x)$ oznacza liczbę dzielników liczby x. Ponieważ wiemy, że $d(1) + d(2) + \cdots + d(n) = O(n \log n)$, mamy rozwiązanie działające dostatecznie szybko.

7 Premia

Można udowodnić, że optymalną strategią pracowników będzie podział na dwie grupy: jedna ustala wartość k, a druga – maksymalną wartość pozwalającą na uzyskanie premii. Dla każdej możliwej liczebności pierwszej grupy tę maksymalną wartość można wyznaczyć wyszukiwaniem binarnym albo – wzorem gdzie x to liczba osób, która finalnie otrzyma premię.

$$\left\lfloor \frac{c\,(n-x)\,k}{100\,n - c\,x} \right\rfloor$$

Złożoność rozwiązania z użyciem wyszukiwania binarnego wynosi $O(n * log(k))$ a bez $O(n)$.

8 Rajdowiec

Ilość okrążeń, które będzie w stanie wykonać rajdowiec jest równa sumie ciągu arytmetycznego $k\%m, k\%m + m, ..., k = ((k\%m + k)/2 * (k/m))$.

9 Skoki

Odpowiedź to Tak, o ile gcd(n, k1, k2)=1, w przeciwnym wypadku odpowiedź to Nie, ponieważ przy dowolnej sekwencji skoków nie zmienimy reszty modulo gcd(n, k1, k2).

10 Podział apartamentowca

Nazwijmy przerwy między mieszkaniami korytarzami.

Jeśli jakiś fragment jest podzielony na co najmniej 2 mieszkania, to istnieje korytarz równoległy do jednego z boków budynku, który biegnie przez całą szerokość lub wysokość fragmentu.

Dowód: Załóżmy nie wprost, że taki korytarz nie istnieje. Wyobraźmy sobie, że znajdujemy się na początku korytarza (czyli przy krawędzi budynku. Taki korytarz musi istnieć, skoro dzielimy budynek na mieszkania) i zaczynamy poruszać się do środka. Poruszamy się według następującego algorytmu: idziemy prosto, aż natrafimy na krawędź mieszkania. W tym momencie, gdy nie możemy iść już dalej na wprost, rozglądamy się w lewo i w prawo. W jednym z tych kierunków na pewno nie dotrzemy do krawędzi budynku, ponieważ byłoby to sprzeczne z założeniem, że nie istnieje korytarz łączący obie przeciwległe krawędzie budynku. Rozpoczynamy ruch w tym właśnie kierunku. Zauważmy, że w pewnym momencie wejdziemy w cykl – od tej pory będziemy przechodzić przez ten sam fragment budynku wielokrotnie. Oznaczałoby to, że wewnątrz tego fragmentu budynku znajduje się zamknięty obszar, co jest niemożliwe, bo mieszkanie wewnątrz nie miałoby okien. Doszliśmy do sprzeczności.

Następnie przystępujemy do rozwiązania zadania za pomocą programowania dynamicznego.

Aktualny stan można opisać wymiarami budynku (a, b) oraz czterema wartościami boole'owskimi, które mówią, po których stronach obecnie rozważanego obszaru mamy krawędź z początkowym budynkiem (dostęp do postawienia okna).

Jeśli wszystkie cztery wartości są fałszywe, zwracamy $-\infty$, ponieważ nie możemy mieć mieszkania w tej części budynku.

W przeciwnym wypadku mamy dwie możliwości:

- Cała ta część budynku zostanie przeznaczona na jedno mieszkanie o ile $a * b \geqslant k$.

- Wybieramy korytarz i dzielimy budynek na dwie części, po czym rekurencyjnie obliczamy wartości programowania dynamicznego dla tych części.