4B – Skrzyżowania

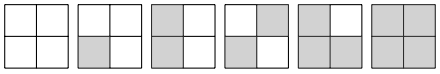
autorzy zadania: Bartosz Kostka i Mateusz Radecki

Treść

Dana jest prostokątna plansza $n \times m$ placów, gdzie każda kolumna placów oraz każdy rząd placów są oddzielone ulicą. Dla każdej czwórki placów tworzących kwadrat 2×2 znamy cykl świateł dla pieszych – dany jest ciąg znaków mówiący dla każdej sekundy czy w danym momencie na zielono świecą się światła przecinające poziomą czy pionową ulicę. Wszystkie cykle są dość małej długości. Mamy dane dużo zapytań o pieszego wyruszającego z jakiegoś placu w jakimś momencie czasu oraz informację o jego docelowym placu. Dla każdego zapytania należy powiedzieć, kiedy najwcześniej pieszy może dotrzeć do swojego celu.

Rozwiązanie

Oczywiście możemy zinterpretować place jako wierzchołki grafu i przejścia dla pieszych jako krawędzie między nimi. Dla ustalonego momentu w czasie zastanówmy się, które place są z których osiągalne. Rozważmy zbiór placów osiągalny z pewnego ustalone placu. Spójrzmy w szczególności na jakieś cztery place układające się w kwadrat 2×2. Z dokładnością co do obrotów, osiągalne place (zaznaczone ciemniejszym kolorem) w tym kwadracie mogą w teorii wyglądać na jeden z następujących sposobów:



Pierwszy, trzeci i szósty układ od lewej są w porządku i mogą wystąpić na planszy, jednak dla drugiego, czwartego i piątego układu nie istnieje stan świateł na środku rysunku, który prowadziłby do właśnie takiej kombinacji osiągalnych placów. Z tego możemy zatem wywnioskować, że każda spójna składowa w grafie jest prostokątem, którego albo zarówno górny bok sięga do samej góry planszy, a dolny bok do dołu planszy, albo lewy i prawy bok sięgają aż do odpowiednio lewego i prawego boku planszy.

Może zdarzyć się tak, że wszystkie światła przecinające jakąś ulicę świecą się w danym momencie na czerwono. Wtedy w żaden sposób nie można przedostać się z jej jedną stronę na drugą. Takie zjawisko, które jest jedynym sposobem na oddzielenie od siebie dwóch spójnych składowych grafu, będziemy określać jako *barierę*.

Oczywiście w żadnym momencie nie może istnieć zarówno pionowa i pozioma bariera. Korzystając z tego faktu możemy wnioskować dalej, że mając przejść z placu (x_1, y_1) do placu (x_2, y_2), kompletnie niezależnymi problemami jest pokonać tę trasę w pionie oraz w poziomie. Wynika to z tego, że jeśli w momencie startu istnieje jakakolwiek bariera oddzielająca pole startowe od pola końcowego, to na pewno nie istnieje na planszy żadna bariera o przeciwnej orientacji, zatem możemy momentalnie zrównać odpowiednią współrzędną pieszego z tą współrzędną pola końcowego. Od tej pory będziemy rozwiązywać problem niezależnie dla poziomych oraz pionowych barier. Dodatkowo, możemy dla każdego wymiaru rozważyć dwa przypadki, zależnie od tego czy pieszy chce przemieścić się w stronę rosnących czy malejących współrzędnych. Dla ustalenia uwagi załóżmy, że chce on przemieścić się w stronę wierszy o większych numerach, tzn. chce przejść z wiersza ℓ do wiersza r, gdzie zachodzi $\ell < r$.

Zauważmy teraz, że ponieważ długości wszystkich cykli nie przekraczają 8 (oznaczmy tę stałą jako c), to cykl tego jak wygląda cała plansza nie przekracza najmniejszej wspólnej wielokrotności liczb od 2 do c, która to jest równa 840. Oznaczmy tę stałą przez k.

Chcielibyśmy zatem dla każdej poziomej ulicy obliczyć maskę bitową rozmiaru k, która mówi w jakich momentach w czasie jest ona barierą. Nie warto ryzykować zrobieniem tego za wolno, więc zastanówmy się jak szybko możemy to osiągnąć. Po pierwsze, dla ustalonej ulicy i dla każdej liczby i od 2 do c, chcielibyśmy obliczyć w jakich momentach (modulo i) wszystkie skrzyżowania na tej ulicy o długości cyklu i nie pozwalają przez nią przejść. Ponieważ wszystkie skrzyżowania na tej ulicy muszą nie pozwalać przez nią przejść, to interesuje nas bitowy `AND` wszystkich masek dla tych skrzyżowań. Łatwo obliczyć go liniowo względem ich liczby. Następnie, mając maski dla każdej długości i od 2 do c, możemy przeiterować się po reszcie z dzielenia momentu w czasie przez k i dla każdego z nich, w złożoności $\mathcal{O}(c)$ sprawdzić, czy rzeczywiście bariera wtedy istnieje. Sprawi to, że w sumarycznej złożoności $\mathcal{O}(nm + (n+m)ck)$, dowiemy się w których momentach w czasie która bariera istnieje. Jeśli zależy nam na jeszcze większym przyspieszeniu, możemy skorzystać z preprocessingu w złożoności $\mathcal{O}(2^c k)$, który pozwoli nam policzyć to samo w złożoności $\mathcal{O}(2^c k + nm + (n+m)c\frac{k}{64})$, jednak nie jest to kluczowe przyspieszenie i nie poświęcimy mu szczególnej uwagi.

Zastanówmy się zatem, w jaki sposób brutalnie moglibyśmy próbować przejść z placu w wierszu ℓ do placu w wierszu r, znając startowy czas t. Oczywiście musimy poczekać w wierszu ℓ tak długo, aż bariera oddzielająca wiersz ℓ od wiersza $\ell+1$ zniknie. Wtedy opłaca się nam zachłannie przejść do wiersza $\ell+1$ i kontynuować trasę. Wyobraźmy sobie zatem prostokątną planszę rozmiaru $(n+1) \times k$. Dla każdego pola (i, j) (gdzie $0 \leq i \leq n$ oraz $0 \leq j < k$) wiemy, czy musimy przejść z niego do pola $(i, (j+1) \mod k)$, czy do pola $(i+1, j)$. Możemy zatem wyobrazić sobie, że z każdego pola (poza tymi w ostatnim w rzędzie) wychodzi jedna krawędź skierowana, o wadze 1 lub 0, zależnie od tego, czy owa krawędź oznacza poczekanie w tym samym rzędzie, czy ruszenie się o jeden rząd do przodu.

Dodatkowo, dla każdej bariery musi istnieć moment, gdy ona nie istnieje, ponieważ zgodnie z treścią zadania każde światło kiedyś zmienia swój stan. Możemy zatem wywnioskować, że jeśli potraktujemy opisane pola jako graf, to jest on ukorzenionym lasem, gdzie pola w ostatnim rzędzie są korzeniami.

Dla każdego zapytania (ℓ, r, t) interesuje nas odległość między polem $(\ell, t \mod k)$, a najbliższym mu polem w rzędzie r. Zamiast iść od pola startowego do końcowego rzędu, możemy użyć algorytmu DFS na każdym z korzeni. Owe przeszukiwanie w głąb powinno pamiętać aktualny wierzchołek oraz jego odległość do korzenia. Dodatkowo, przyda się nam globalna tablica rozmiaru $n+1$, nazwijmy ją $WHEN[i]$. Za każdym razem, gdy w przeszukiwaniu w głąb będziemy przechodzić z rzędu i do rzędu $i-1$, zapiszemy w polu $WHEN[i]$ odległość od pola, z którego właśnie wyszliśmy, do korzenia. Jeśli dla każdego pola stworzymy listę zapytań, które zaczynają właśnie w tym polu, to odwiedzając je w trakcie algorytmu DFS będziemy znali zarówno odległość od tego pola do korzenia, jak i odległość od odpowiedniego pola w docelowym rzędzie do korzenia. Czas, jakiego pieszy potrzebuje na pokonanie trasy, jest oczywiście równy różnicy tych dwóch wartości.

Ze wszystkimi zebranymi obserwacjami i pomysłami, jesteśmy w stanie rozwiązać zadanie w złożoności $\mathcal{O}(2^c k + nm + (n+m)c\frac{k}{64} + (n+m)k + q)$.