Анализ на задача coloring

Тагове: обхождане на граф, двуделно оцветяване

Ако граф може да се оцвети, ще го наричаме *двуделен*.

Нулева подзадача – 0 точки

Тази задача е оставена за комуникация със системата – ако състезателят забелязва, че решението му връща верен отговор на примерните тестове на машината му, но системата твърди, че решението връща грешен отговор на тези тестове, можем да се уверим, че грешката не е в решението, а нещо механично – примерно грешно въведен вход, работа с невалидни индекси на масиви или подобни.

Решения на подзадачи 1, 2, 4 – 20 точки

Първите няколко задачи бяха предоставени на състезателите, които тепърва се срещат с идеята за оцветяване на граф. Целта беше да се създаде възможност тези състезатели да изградят интуиция относно идеята граф да има цветове по върхове.

За първата подзадача трябва да се разглеждат два типа компоненти: едните са с единствен връх, който винаги може да бъде оцветен в червено, а другите бяха два върха, свързани с ребро, съответно от тях един трябва да е червен, другият син – този с по-голяма стойност ще оцветим червен.

Втората подзадача леко надграждаше миналата, защото вече са позволени по-дълги "верижки": трябва да се закачим от едниния край на верижката и да започнем да я обхождаве с примерно `while`-цикъл и да изберем по-доброто от двете: четните или нечетните в реда на посещаване върхове да бъдат червени.

Четвъртата подзадача се различава от предходните, но отново допуска решение без да се правят стандартните алгоритми за обхождане на граф: за всеки връх с повече от един съсед имаме избор – или самия връх ще е червен, или всичките му съседи.

Решение на подзадача 5, 6 – 26 точки

Допълнителното условие на подзадачата дефинира всяка компонента в графа като дърво – общо можем да кажем, че графът е гора. Дърветата винаги са двуделни и това улеснява значително задачата в този случай – харесваме си връх, който да кажем оцветяваме в червено; всеки негов съсед трябва да е син, а всички съседи на съседите (без вече оцветеният първоначален връх) ще трябва отново да са оцветени в червено. По този начин повтаряме "вълнообразно" оцветяването, докато има неоцветени върхове. Понеже този процес обработва всеки връх точно веднъж (има единствен прост път от връх до всеки друг), двуделното оцветяване винаги е възможно.

Единственият казус, който остава да разрешим е дали оптимално сме оцветили компонентата. Дори това да не е така, единственият начин да преоцветим компонентата е всеки син връх да стане червен и всеки червен да стане син. Така имаме два избора за кои точно да бъдат червените върхове – избираме този, който дава по-добър резултат.

Решение на подзадача 3 – 16 точки

Тук компонентите са от три типа: самотни върхове, пътечки от върхове и цикли от върхове.

Това е подзадачата, където опцията да можем да оцветим графът почва да играе съществена роля – бързо разписване на малки случаи ще покаже, че единствено циклите с нечетна дължина не могат да се оцветят според условието. Това всъщност и напълно описва графите, които могат да се оцветят: граф е двуделен, тогава и само тогава когато съществува нечетен цикъл в него. Осъзнаването на това ограничение е добра стъпка в цялото решение на задачата, въпреки че не е напълно задължително, за да се достигне пълно решение.

Решение на цялата задача – 100 точки

За да решим задача с граф, все в някакъв момент ще трябва да го обходим. ☺️

Ще опишем двете идеи за решение на задачата: едната се основава на `BFS`, другата на `DFS`.

Когато пускаме `BFS` по граф, съзнателно или не, винаги разслояваме графа на "слоеве", съответно по минималната дистанция между началния връх и всеки друг връх. Върху така разслоен граф, можем да приложим идея подобна на тази, с която оцветявахме дърветата – първият връх (този, от който сме започнали обхождането) ще е червен, всички негови съседи (които са на първия слой) ще са сини, техните съседи (втори слой) ще са червени и т.н и т.н. Можем още със самото обхождане да проверяваме дали случайно не се опитваме да "преоцветим" вече оцветен връх, като в този случай ще знаем, че не е възможно да се оцвети графа въобще. Това е идеята, имплементирана в `bfs.cpp`. Друга опция е след като извършим оцветяването да проверим всяко ребро дали двата върха, закачени на него са разноцветни.

Ще спомена едно характеризиращо свойство на разслояването на графа, което правим с обхождането в ширина: при него, всяко ребро или свързва два върха от съседни слоеве, или свързва два върха от един слой. За да се убедим, че това е вярно, ще предположим противното – че в графа има ребро (u, v), което свързва два върха на слоеве, между които има поне един друг слой. Да кажем, пуснали сме `BFS`-а така, пресметнали сме стандартния `dist[u]` масив. Тогава излиза, че `dist[u] < dist[v] - 2`, но както казахме между тези два върха има ребро: няма как разликата в най-късите пътища помежду им да е по-голяма от 1. Достигнахме до противоречие, което може единствено да следствие от изначалното ни предположение.

Горният абзац ни убеждава в следното: проблем при оцветяването може да се получи единствено при ребра, които свързват два върха от един и същи слой (или формално, ребро (u, v), за което `dist[u] == dist[v]`).

Второто решение, описано в `dfs.cpp`, е както следва:

Фиксирайки цветът на един от върховете, ние всъщност фиксираме цветът на всички върхове в графа – процесът е подобен на този, който описахме в подзадачата, където графът е гора. Можем би за да се убедим в това, единственото трябва да осъзнаем, че повечето ребра, които графът сега има представляват повече проблеми, които могат да се създадат (причини графът да не е двуделен), а не някакви опции, за които да избираме. Независимо как почнем да оцветяваме графа (по кое ребро решим да тръгнем), ако има нечетен цикъл в графа, обхождането ни ще се завърти по него и ще се "счупи" опитът ни да оцветим върховете. Ако пък това не се случи, значи сме открили успешно оцветяване – по никакъв начин не сме се излъгали.

Разсъждения като горните би трябвало да са достатъчни, за да мотивират идеята да пуснем едно обхождане в дълбочина, което да оцвети графа – независимо по какъв път тръгне то, ако има отговор, ще го намери, ако не, пак ще се сблъска с този проблем.

Автор: Иван Лупов