

Snowy Roads

(Решение)

Това беше друга от сравнително лесните задачи (планирана за трета по леснота, след Choso и EGN). При нея имаме няколко начина, по които можем да подходим.

По-дългият, но по-стандартен такъв, е да ползваме двоично търсене по отговора. На всяка негова стъпка строим граф, съдържащ само първите mid ребра (където mid е текущата стойност от двоичното търсене). В този граф ползваме или BFS или DFS за да проверим свързаността. Очевидно това би работело, тъй като ако графът е свързан с mid ребра, то той ще е свързан и с $mid+1$ ребра. Обратно, ако не е свързан с mid ребра, няма да бъде свързан и с $mid-1$ ребра. Следователно функцията е монотонна и двоично търсене може да бъде приложено.

Така сложността би била $O(\log(M) * (N+M))$ на тест. Тъй като и Binary Search и BFS/DFS са доста стандартни алгоритми, предполагам повечето по-неопитни състезатели ще се спрат на това решение.

Забележете, че ако при горното решение не ползваме двоично търсене, а пускаме BFS/DFS след всяка стъпка, то решението ни би Time Limit-нало. С малко наблюдение, обаче, можем да забележим, че всъщност не ни трябва да чистим масива с посетените върхове на всяка стъпка - ако те са били посетени при предходната, то ще бъдат посетени и при текущата! Следователно можем да решим задачата и в $O(N+M)$ на тест с инкрементално пускане на BFS/DFS след всяко добавено ребро. Това решение е не само по-бързо, но и по-кратко, тъй като липсва кода за двоично търсене. За сметка на това трябва да внимаваме да обикаляме само непосетени до сега върхове, а също така дали новодобавеното ребро свързва вече посетени, все още непосетени или един посетен и един непосетен връх.

Всъщност съществува и друг алгоритъм, който е още по-кратък и предполагам повечето от добрите състезатели ще предпочетат. Тъй като ни интересува само кога графът става с една единствена свързана компонента, то можем да ползваме структурата disjoint-set forest (union-find) за да пазим свързаността на графа. Добавяйки ребро, или не правим нищо (ако двата свързани върха са вече в една компонента), или ги обединяваме в структурата данни. Този алгоритъм също е (съвсем малко по-бавен) от $O(N+M)$.