Анализ на задача `rover`

Тагове: умна симулация, префиксен масив, показалки

Решение за 9 точки (13%)

Като използваме ограничението, че $K \leq 10^3$, то е достатъчно наивно с два вложени цикъла да симулираме тръгването на марсохода от всяка възможна инструкция и да следим за първия момент, когато ще излезем от таблицата или ще стигнем непроходима клетка.

Сложност: $O(K^2)$.

Решение за от 29 до 50 точки (от 38% до 67%)

Ще разгледаме започване от всяка възможна инструкция и ще намерим първата инструкция, в която излизаме от таблицата или стигаме непроходима клетка, по бърз начин. В някои случаи бихме могли да приложим двоично търсене за намирането, но тук не можем. Нямаме бърз начин да проверим дали някакъв набор от инструкции е валиден. Трябва да се възползваме от допълнителното ограничение за малкия брой непроходими клетки.

Нека да анализираме по-внимателно ситуацията за фиксирана непроходима клетка - (x_p, y_p). Започваме от дадена позиция в инструкциите, която съответства на началната позиция на марсохода (x_{st}, y_{st}) и искаме да характеризираме по-добре момента, когато той ще отиде на координати (x_p, y_p). Ако смятаме префиксните $(\Delta x, \Delta y)$ от изпълняването на инструкции, то искаме да направим прехода $(x_{curr}, y_{curr}) \rightarrow (x_{next}, y_{next})$, което да съответства на $(x_{st}, y_{st}) \rightarrow (x_p, y_p)$. Това става точно когато разликите по x и по y са равни или $x_{next} - x_{curr} = x_p - x_{st}$ и аналогично за y. Обръщаме внимание, че x_{curr} е фиксирано, както и знаем x_p и x_{st}. Тогава $x_{next} = x_{curr} + (x_p - x_{st})$, но ние ще използваме алтернативен начин на характеризация: $x_{next} - x_p = x_{curr} - x_{st}$.

Сега вече решението е следното - обхождаме инструкциите отзад-напред и разглеждаме всеки момент като възможно стъпване в непроходима клетка. Нека сега сме на инструкция с номер i. Записваме в някаква структура данни, че ако имаме $(x_{curr} - x_p, y_{curr} - y_p)$, то на тази позиция i ще достигнем непроходима клетка. Тук (x_{curr}, y_{curr}) са префиксните $(\Delta x, \Delta y)$ до инструкция i, а (x_p, y_p) са координатите на съответната непроходима клетка. Когато разглеждаме започването на набора инструкции от дадена позиция, то просто поглеждаме за първият индекс, на който ще достигнем непроходима клетка за $(x_{curr} - x_{st}, y_{curr} - y_{st})$ (ако има такъв). Това означава, че всички набори, започващи от текущата инструкция до преди тази ще са валидни и трябва да ги добавим към отговора. За да е пълна проверката за валидност трябва да обработваме и случая с излизането от таблицата. Един начин е да добавим ограждане от непроходими клетки, но това е тромаво и ще забави решението. Затова можем по подобен начин да следим изравнени разлики, но само по едната координата, защото ние просто искаме координатите по x и y да не стават 0 и N. Това може да става просто с масив.

Относно структурата от данни, която да ползваме за наредените двойки разлики. Един неприятен детайл е, че тези разлики могат да стават доста големи, защото x_{curr} и y_{curr} са в интервала от $[-K, K]$. Затова можем да използваме *map*, в който да записваме на наредените двойки разлики съответните индекси. Заради време и памет, по-добре е да се използва *unordered_map*. Той няма вградено поддържане на наредени двойки и един вариант е сами да добавим хеширане за тях, но най-просто е да ги кодираме по някакъв начин в едно число, което все пак ще се побира в *long long int*.

Сложност: $O(KM)$ с по-голяма константа, където M е броят непроходими клетки.

Пълно решение - 75 точки (100%)

Понеже няма подзадачи и тестовете не са от най-силните можем да пробваме да изчийтим тези последни точки като направим малко по-голям двумерен масив вместо някакъв *map* с надеждата да не ни трябват много крайните клетки, които няма да поддържаме. Силното ограничение по памет е сложено точно, за да може такова решение да не изкарва много повече точки от наивното решение.

Намирането на първите моменти, когато излизаме от таблицата е достатъчно добро, така че ще оптимизираме намирането на първите моменти, когато стъпваме в непроходима клетка. Няма да използваме сложни структури данни, защото реално погледнато ние просто търсим наредени двойки. Нека да сортираме всички възможни разлики $(x_{curr} - x_{st}, y_{curr} - y_{st})$ лексикографски. Тогава ако фиксираме непроходима клетка (x_p, y_p) и позиция в инструкциите, искаме да намерим бързо от сортираните разлики наредените двойки $(x_{curr} - x_p, y_{curr} - y_p)$. Ако предварително сортираме лексикографски и префиксите (x_{curr}, y_{curr}), то $(x_{curr} - x_p, y_{curr} - y_p)$ също ще са сортирани лексикографски за фиксирана непроходима клетка. Така всъщност можем да приложим метода на показалките, като просто поддържаме показалка при разликите, за да можем бързо да намираме съответните наредени двойки. Този алгоритъм всъщност е точно този, който се използва при сливане на два сортирани масива в нов сортиран масив за линейно време. Един важен детайл е, че търсим в разликите само наредени двойки, които са получени за по-ранни инструкции от съответната инструкция. Затова всъщност трябва да сортираме наредени тройки, като третата компонента е съответният индекс.

Сложност: $O(K \log_2 K + KM)$, където M е броят непроходими клетки.

Автор: Илиян Йорданов (zaeta)