

Спасение Полинезии

В данной задаче требовалось построить минимальное остовное дерево на графе специального вида: в нем может быть слишком большое количество неявно заданных ребер, однако эти ребра следуют определенным правилам из условия: из вершины $n + i$ ребра одинакового веса ведут в вершины с l_i по r_i .

Поскольку на таком графе может быть порядка nq ребер, просто построить этот граф в явном виде не получится. Но посмотрим, как будет вести себя алгоритм Краскала, если такой граф все же построить:

- ребра будут рассматриваться в порядке возрастания c_i ;
- то есть ребра, ведущие куда-то из $n + i$, для каждого фиксированного i , будут рассмотрены подряд;
- и все эти ребра, соединяющие вершины из разных компонент связности, будут взяты в MST.

Поэтому сначала отсортируем все данные в условия описания ребер по возрастанию c_i . А затем будем выполнять шаги из алгоритма Краскала, но оптимизированным образом.

1. Будем рассматривать ребра $(n + i, l_i)$, $(n + i, l_i + 1)$, \dots , $(n + i, r_i)$ именно в таком порядке (порядок ребер одного веса на корректность алгоритма Краскала не влияет).
2. Заметим, что вершина $n + i$ пока что ни с чем не соединена, поэтому первое такое ребро (между $n + i$ и l_i) гарантированно будет добавлено в MST.
3. Для каждого следующего ребра заметим, что $n + i$ и $j + 1$ находятся в разных компонентах связности тогда и только тогда, когда j и $j + 1$ находятся в разных компонентах связности: действительно, если мы рассматриваем ребра по возрастанию их конца, то к моменту рассмотрения $j + 1$ вершины $n + i$ и j уже связаны ребром или путем.

Более того, итоговый граф будет связан тогда и только тогда, когда для любого j от 1 до $n - 1$ вершины j и $j + 1$ находятся в одной компоненте связности. Поэтому мы свели всю задачу к проверке связности соседних из первых n вершин.

А для этого достаточно поддерживать множество всех несвязных соседних вершин, и при обработке всех требуемых ребер с помощью бинарного поиска в этом множестве находить все несвязанные пары из отрезка $[l_i, r_i]$ и удалять их. В сумме такое решение работает за $O((n + q) \log n)$.